UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

DECORRELACION ESPACIAL RAPIDA Y DE ALTA EFICIENCIA PARA COMPRESION DE IMÁGENES CON PERDIDAS

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires en el área Ciencias de la Computación

Mario Mastriani

Director de tesis: Dra. Marta Mejail

Consejero de estudios: Dra. Marta Mejail

Lugar de trabajo: Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Buenos Aires, 2009.

DECORRELACION ESPACIAL RAPIDA Y DE ALTA EFICIENCIA PARA COMPRESION DE IMÁGENES CON PERDIDAS

Resumen

Se encuentra ampliamente documentado en toda la literatura de procesamiento de imágenes, que la transformada óptima por excelencia por su gran eficiencia para compactar energía de señales e imágenes, es la famosa Transformada Discreta de Karhunen-Loève (TDKL). Siendo la decorrelación de bandas multi e hiperespectrales de origen satelital el ámbito donde más eficiente es. Esto tiene que ver con el hecho de que dichas bandas son imágenes de una misma parcela, pero a diferentes longitudes de onda. Esto implica indirectamente, que la TDKL es más eficiente en el proceso de decorrelación cuando es aplicada a un paquete de imágenes que poseen un valor elevado de un nuevo atributo que denominaremos ***Similitud Morfológica Equi-Posicional (SMEP)****, el cual se formalizará en extenso en el Capítulo 3 (Efecto de utilizar SMEP en la TDKL).*

En cambio, cuando se emplea la TDKL para ser aplicada en casos donde las componentes no poseen el mencionado atributo, el rendimiento de decorrelación de las mismas decrece sensiblemente. Esta es la razón por la cual no se la emplea sola con tanta eficiencia en imágenes monocuadro. La razón fundamental de este problema reside en que para obtener similar rendimiento en la decorrelación de las componentes de una imagen única, hay que segmentar la misma en pequeños bloques (cuantos más pequeños mejor) lo cual implica un gran aumento en el número de dichos bloques, dando lugar a un inaceptable incremento de la complejidad computacional.

No han sido pocos los intentos por solucionar este problema mediante el empleo de sustitutos de la TDKL, teniendo como principal excusa su alto costo computacional, motivado por el cálculo de autovalores y autovectores. En este sentido, se utilizan como sustitutos (y bajo ciertas y particulares circunstancias) fundamentalmente la Transformada Discreta Coseno (TDC) y la Transformada Discreta de Onditas (TDO).

En esta tesis, se estudiará e investigará la aplicación de usar una serie de combinaciones de transformadas a través de apropiados mecanismos de exploración de la imagen por bloques, de manera tal que se comience con una transformada que posicione el problema original en un nuevo ámbito donde la TDKL pueda ser más eficiente al tener el mismo rendimiento que al usar muchos bloques pequeños pero con menos bloques grandes. Se

pondrá de relieve que las transformadas que permiten este aumento en la performance de decorrelación de la TDKL sobre bloques con originalmente bajo SMEP son precisamente las anteriormente mencionadas, es decir, la TDC y la TDO, a las que deberán añadirse la Transformada de Walsh-Hadamard (TWH) y la Transformada Discreta de Hartley (TDH).

Específicamente, el trabajo se enfoca en la aplicación sobre bloques de una imagen, de distintos mecanismos de exploración y barrido de los mismos simultáneamente a la aplicación de las transformadas TDC, TDO, TWH o TDH para la constitución de una matriz tridimensional sobre la cual se emplee la TDKL. Dichas aproximaciones (entre otros) constituyen los aportes.

Por otra parte, estos aportes se compararán con la aplicación de la TDC, TDO y la TDKL por separado y solas. Además, se compararán las técnicas mencionadas con los algoritmos provenientes del Joint Photographic Experts Group (JPEG) y (JPEG2000).

Finalmente, definimos en el Capítulo 1 un conjunto de nuevas métricas para evaluar la eficiencia de decorrelación de la TDKL en base a atributos interbloques más favorables, los cuales formalizaremos en el Capítulo 3.

FAST AND HIGH EFFICIENCY SPATIAL DECORRELATION FOR LOSSY IMAGES COMPRESSION

Abstract

It is widely documented in the image processing literature, that the optimal transform for its high efficiency to compact the energy of signals and images, is the famous Discrete Karhunen-Loève Transform (DKLT), being the decorrelation of satellite multi and hyper-spectral bands the scope where it is more efficient. This has to do with the fact that such bands are images of the same area, but at different wavelengths. This implies indirectly that the DKLT is more efficient in the process of decorrelation when it is applied to a package of images that possesses a high value for a new attribute that we call ***Equi-positional Morphological Similarity (EPMS)****, which will be formalized in full in Chapter 3 (Effect of using EPMS on the DKLT).*

In contrast, when the DKLT is used to be applied in cases where the components do not possess the above-mentioned attribute, their decorrelation performance decreases significantly. This is the reason why they are not used alone so efficiently in monoframe images. The fundamental reason for this problem is that to obtain a similar performance on the decorrelation of the components of a single image, we must divide it into small blocks (how the smaller the better) which results in a large increase in the number of those blocks, giving rise to an unacceptable increase in computational complexity.

Then have been many attempts to solve this problem through the use of substitutes for DKLT, having as a principal excuse their high computational cost, motivated by calculating eigenvalues and eigenvectors. In this sense, primarily Discrete Cosine Transform (DCT) and the Discrete Wavelet Transform (DWT) are used as substitutes (and under certain particular circumstances).

In this thesis, we going to explore and investigate the implementation of the use of a series of combinations of transforms through appropriate exploration mechanisms of the image blocks, starting with a transform that positiones the original problem in a new area where the DKLT can be more efficient having the same performance altairned when using many small blocks but using fewer larger blocks. We will highlight that the changes that allow this increase in performance of DKLT decorrelation of the blocks with low originally EPMS are precisely those listed above, namely the DCT and the DWT, which must be added to transform the

Walsh-Hadamard Transform (WHT) and the Discrete HartleyTransform (DHT).

Specifically, this work focuses on the application of different mechanisms for exploration on the blocks of an image and sweep them simultaneously with the application of the DCT, DWT, WHT or DHT transforms for the formation of a three-dimensional matrix on which we will use the DKLT. Such approaches (among others) are the inputs.

Moreover, these contributions will be compared with the application of the DCT, DWT and DKLT separately and alone. Besides, the techniques above will be compared with algorithms from the Joint Photographic Experts Group (JPEG) and (JPEG2000).

Finally, we define in Chapter 1 a new set of metrics to evaluate the efficiency of DKLT decorrelation based on more favorable attributes inter-block, which are formalized in Chapter 3.

Agradecimientos

Desearía expresar mi agradecimiento y gratitud a la Dra. Marta Mejail por su constante guía, consejo y estímulo desde que nos conocimos.

Quiero agradecer también al Dr. Enrique Segura, por su predisposición y apoyo.

Un particular agradecimiento a los miembros del jurado, Prof. Claudio Delrieux, Prof. Alejandro César Frery y Prof. Nelson D. A. Mascarenhas cuyos aportes, comentarios y observaciones enriquecieron este trabajo.

Un agradecimiento a ANSES, UNTreF y CEMA sin cuyo soporte no podría haber llevado adelante este trabajo a esta altura de mi vida.

Mi agradecimiento y disculpas para Paola y Agustín por sacar de nuestro tiempo en común para hacer este trabajo.

A todos los que de una u otra forma, directa o indirectamente me ayudaron, les doy las gracias.

A El.

Indice General

Página

Indice de Tablas

Indice de Figuras

4.1(g)	AN para las técnicas 1 vs. 8, graficando para los I. 8 y II. 4 primeros sub-bloques.	4.5
4.1(h)	AN para las técnicas 1 vs. 9, graficando para los I. 8 y II. 4 primeros sub-bloques.	4.5
4.1(i)	AN para las técnicas 1 vs. 10, graficando para los I. 8 y II. 4 primeros sub-bloques.	4.6
4.2(a)	ANA para las técnicas 1 vs. 2, graficando para los I. 8 y II. 4 primeros sub-bloques	4.6
4.2(b)	ANA para las técnicas 1 vs. 3, graficando para los I. 8 y II. 4 primeros sub-bloques	4.6
4.2(c)	ANA para las técnicas 1 vs. 4, graficando para los I. 8 y II. 4 primeros sub-bloques	4.7
4.2(d)	ANA para las técnicas 1 vs. 5, graficando para los I. 8 y II. 4 primeros sub-bloques	4.7
4.2(e)	ANA para las técnicas 1 vs. 6, graficando para los I. 8 y II. 4 primeros sub-bloques	4.7
4.2(f)	ANA para las técnicas 1 vs. 7, graficando para los I. 8 y II. 4 primeros sub-bloques	4.8
4.2(g)	ANA para las técnicas 1 vs. 8, graficando para los I. 8 y II. 4 primeros sub-bloques	4.8
4.2(h)	ANA para las técnicas 1 vs. 9, graficando para los I. 8 y II. 4 primeros sub-bloques	4.8
4.2(i)	ANA para las técnicas 1 vs. 10, graficando para los I. 8 y II. 4 primeros sub-bloques	4.9
4.3(a)	Ercp para las técnicas 1 vs. 2, graficando para los I. 8 y II. 4 primeros sub-bloques	4.9
4.3(b)	Ercp para las técnicas 1 vs. 3, graficando para los I. 8 y II. 4 primeros sub-bloques	4.9
4.3(c)	Ercp para las técnicas 1 vs. 4, graficando para los I. 8 y II. 4 primeros sub-bloques	4.10
4.3(d)	Ercp para las técnicas 1 vs. 5, graficando para los I. 8 y II. 4 primeros sub-bloques	4.10
4.3(e)	Ercp para las técnicas 1 vs. 6, graficando para los I. 8 y II. 4 primeros sub-bloques	4.10

Indice de Algoritmos

Indice de Aportes

Capítulo 1

1 Estado del arte

1.1 Introducción

Este capítulo atañe a la Transformada Discreta de Karhunen-Loève (TDKL), sus ventajas y desventajas, así como su aplicación a la compresión de imágenes con pérdidas. Por otra parte, también se analizarán las condiciones bajo las cuales se puede alcanzar un mejor rendimiento de compresión y una mayor rapidez en la obtención de los componentes principales.

1.2 Transformada Discreta de Karhunen-Loève (TDKL) aplicada a una imagen

Explicaremos esta aplicación sobre un ejemplo concreto. Se comienza con una imagen ***I*** como la de la Fig.1.1, y se la divide en bloques o mosaicos [1], como se muestra en la Fig.1.2.

Figura 1.1: Imagen de Lena de 512-por-512 pixeles, con 8 bits-por-pixel (bpp).

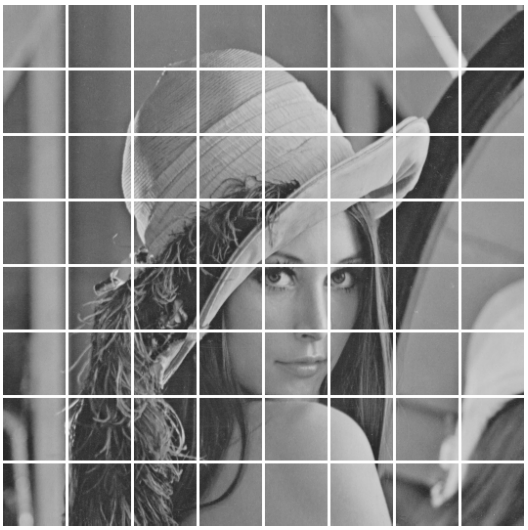


Figura 1.2: Imagen de Lena dividida en bloques de 64-por-64 pixeles.

Para este ejemplo, las dimensiones de la imagen ***I*** son:

*nf**I*** = número de filas de ***I*** = *512*
*nc**I*** = número de columnas de ***I*** = *512*

La Fig.1.3 muestra el detalle de recolección y alineación de los píxeles de idéntica ubicación dentro de cada bloque. Aunque en este caso los bloques son de 4-por-4 pixeles (para simplicidad del dibujo). Dichos bloques pueden estar solapados o no. En este trabajo se trabajará con bloques no solapados.

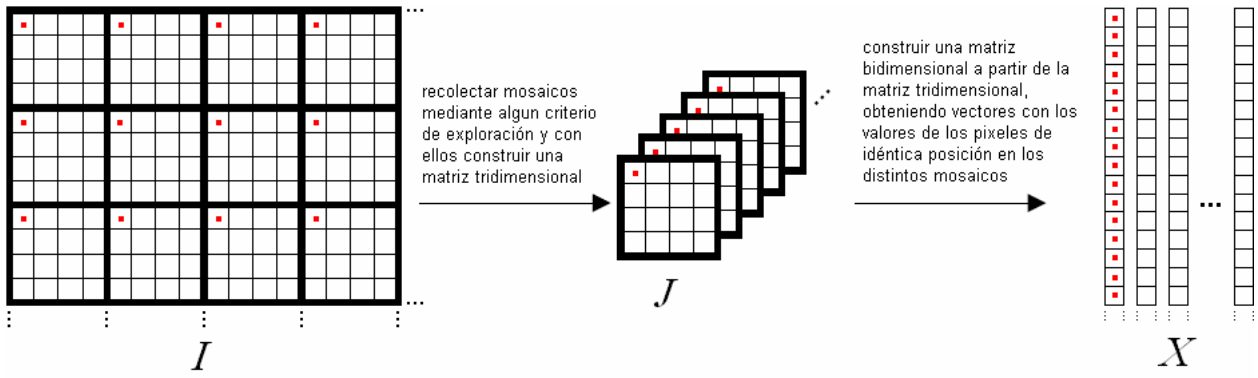


Figura 1.3: Recolección, alineación y vectorización del paquete de bloques.

La mencionada recolección responde a algún criterio de exploración de los bloques, los cuales se analizarán en detalle en el Apéndice A. La etapa intermedia de alineación de los mosaicos permite construir una matriz tridimensional ***J***, en este caso de dimensiones:

nfm = número de filas de los mosaicos = 4
ncm = número de columnas de los mosaicos = 4
nm = número de mosaicos (profundidad) = (*nf**I*** / *nfm*) x (*nc**I*** / *ncm*) = (512/4) x (512/4) = 16384

Posteriormente se exploran por filas los píxeles de cada bloque, constituyendo vectores entre valores correspondientes a posiciones idénticas, lo cual da lugar a una matriz bidimensional ***X*** sobre la cual finalmente se aplicará la TDKL. La dimensión de dicha matriz para este caso es:

*nf**X*** = número de filas de ***X*** = *nm* = 16384
*nc**X*** = número de columnas de ***X*** = *nfm* • *ncm* = 16

1.2.1 Aplicación de la TDKL sobre la matriz bidimensional *X*

El procedimiento de aplicación de la TKDL sobre la matriz bidimensional ***X*** comienza con la creación de la matriz de covarianza de la misma [2-28], la cual resulta según:

$$\boldsymbol{C}_{\mathbf{x}} = E\{(\boldsymbol{X}-\boldsymbol{m}_{\mathbf{x}})(\boldsymbol{X}-\boldsymbol{m}_{\mathbf{x}})^{T}\} \in \Re^{nm \times nm} \qquad (1.1)$$

donde:
$\boldsymbol{X} = [\, x_1\, x_2\, \ldots\, x_{nfm \bullet ncm}\,] \in \Re^{nm \times (nfm * ncm)}$, $x_k \in \Re^{nm \times 1}$ es su *k-ésima* columna, para $k \in [1, nfm \bullet ncm]$
$E\{\bullet\}$ representa la esperanza matemática de "•"
$\{\bullet\}^T$ representa la traspuesta de $\{\bullet\}$
$\boldsymbol{m}_{\mathbf{x}} = E\{\boldsymbol{X}\}$ es el vector valor medio de ***X*** y $\boldsymbol{m}_{\mathbf{x}} \in \Re^{nm \times 1}$

En su apropiada forma matemática, tendremos

$$\boldsymbol{C}_x = \frac{1}{\boldsymbol{nfm}*\boldsymbol{ncm}} \sum_{\boldsymbol{k}=1}^{\boldsymbol{nfm}*\boldsymbol{ncm}} (\boldsymbol{x}_k - \boldsymbol{m}_x)(\boldsymbol{x}_k - \boldsymbol{m}_x)^T \tag{1.2}$$

con:

$$\boldsymbol{m}_x = \frac{1}{\boldsymbol{nfm}*\boldsymbol{ncm}} \sum_{\boldsymbol{k}=1}^{\boldsymbol{nfm}*\boldsymbol{ncm}} \boldsymbol{x}_k \tag{1.3}$$

Por lo tanto, la TDKL resultará

$$y_k = \mathbf{V}^T (x_k - \boldsymbol{m}_\mathbf{x}) \tag{1.4}$$

donde:

$\mathbf{V}^T \in \Re^{\boldsymbol{nm}\times\boldsymbol{nm}}$ y se trata de una matriz unitaria cuyas filas son los autovectores de $\boldsymbol{C}_\mathbf{x}$ y que la diagonaliza convirtiéndola en

$$\boldsymbol{C}_\mathbf{y} = E\{\boldsymbol{Y}\,\boldsymbol{Y}^T\} \in \Re^{\boldsymbol{nm}\times\boldsymbol{nm}} \tag{1.5}$$

siendo:

$$\boldsymbol{Y} = [\, y_1\, y_2\, \ldots\, y_{nfm\,\mathrm{x}\,ncm}\,] \in \Re^{\boldsymbol{nm}\times(\boldsymbol{nfm}*\boldsymbol{ncm})},\ y_k \in \Re^{\boldsymbol{nm}\times 1} \text{ es su } \textit{k-ésima} \text{ columna, para } k \in [1,\ nfm \bullet ncm]$$
$$= \mathbf{V}^T (\boldsymbol{X} - \boldsymbol{m}_\mathbf{x}) \tag{1.6}$$

Reemplazando (1.6) en (1.5) surge

$$\begin{aligned}\boldsymbol{C}_\mathbf{y} &= E\{\, \mathbf{V}^T (\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})\, (\mathbf{V}^T (\boldsymbol{X}-\boldsymbol{m}_\mathbf{x}))^T\} \\ &= E\{\, \mathbf{V}^T (\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})\, (\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})^T\, \mathbf{V}\,\} \\ &= \mathbf{V}^T\, E\{(\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})\, (\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})^T\}\, \mathbf{V} \\ &= \mathbf{V}^T\, E\{(\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})\, (\boldsymbol{X}-\boldsymbol{m}_\mathbf{x})^T\}\, \mathbf{V}\end{aligned} \tag{1.7}$$

mientras que, reemplazando (1.1) en (1.7), tenemos

$$\boldsymbol{C}_\mathbf{y} = \mathbf{V}^T\, \boldsymbol{C}_\mathbf{x}\, \mathbf{V} = \Lambda_\mathbf{y} \tag{1.8}$$

donde $\Lambda_\mathbf{y} \in \Re^{\boldsymbol{nm}\times\boldsymbol{nm}}$ y se trata de una matriz diagonal y unitaria que contiene los autovalores de $\boldsymbol{C}_\mathbf{x}$ en un orden decreciente, es decir, de mayor a menor y monótonamente decreciente, por lo que el primer autovalor será el mayor y el último el menor. Todos los autovalores son reales positivos y cumplirán con $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{\boldsymbol{nm}} \geq 0$ [29, 30].

Finalmente, y dado que todas las entradas de la matriz $\mathbf{V}$ son reales, esta será entonces una matriz ortogonal, constituyendo sus columnas una base ortonormal en $\Re^{\boldsymbol{nm}\times\boldsymbol{nm}}$, al igual que sus filas. Los atributos precedentes establecen que $\boldsymbol{V}^{-1} = \boldsymbol{V}^{\mathrm{T}}$, por lo que la TDKL inversa o TDKL^{-1} será entonces

$$\boldsymbol{X} = \mathbf{V}\boldsymbol{Y} + \boldsymbol{m}_\mathbf{x} \tag{1.9}$$

1.2.2 Propiedades de la TDKL [31, 32]

1.2.2.1 Ventajas

a. Es óptima para decorrelacionar los coeficientes transformados
b. Minimiza la representación total de la Entropía de la secuencia (la Entropía es la función de probabilidad con la que aparecen los diferentes niveles de cuantización de la imagen).
c. Permite una máxima compactación de la energía
d. Minimiza el error de reconstrucción bajo una norma L_2 (útil para compresión).
e. Permite reducir la dimensionalidad del proceso al permitir incluir exclusivamente a los componentes principales asociados a los autovalores más grandes y aun así minimizar el error de reconstrucción resultante bajo una norma L_2 . Este procedimiento de poda es la clave de la compresión de imágenes con pérdidas.

1.2.2.2 Desventajas

a. Depende de la estadística de los bloques
b. No se puede calcular eficientemente en forma distribuida para una imagen separada en bloques
c. La matriz de la transformación no puede ser factorizada en matrices ralas.
d. El cálculo de los autovalores y autovectores se realiza sobre la matriz $\boldsymbol{C}_{\mathbf{x}}$, y las dimensiones de ésta dependen de la cantidad de bloques, es decir, cuanto más bloques se agrupen, mayor es la dimensión de $\boldsymbol{C}_{\mathbf{x}}$ y por ende la complejidad computacional asociada con este procedimiento. Como se verá en la próxima sección, cuanto mas bloques se agrupen menor será el tamaño de los mismos y por ende mejor rendimiento de compactación tendrá la TDKL, es decir, la criticidad del procedimiento se da cuando mejor funciona la transformada.

1.2.3 Compresión de imágenes con pérdidas mediante la TDKL

El método general de aplicación de las transformadas lineales y unitarias para la compresión de imágenes con pérdidas podrá observarse en el Apéndice B, no obstante, en esta sección describiremos en particular el método concerniente a la TDKL.

Basándonos en las Figuras 1.2 y 1.3 el método comienza con la división de la imagen $\boldsymbol{I}$ en mosaicos (o bloques) de igual dimensión y no solapados mediante los cuales se conforma primero la matriz tridimensional $\boldsymbol{J}$ según algún criterio de exploración de los mismos [1], y luego se construyen vectores columnas para la matriz bidimensional $\boldsymbol{X}$ a partir de los píxeles de igual ubicación de todos los mosaicos.

Posteriormente aplicamos a las columnas de la matriz $\boldsymbol{X}$ las Ecuaciones (1.2, 1.3, 1.6 y 1.8). Una graficación típica de la traza de la matriz $\Lambda_{\mathbf{y}}$ puede observarse en la Fig.1.4, donde cuando más rápidamente decrezcan los autovalores a cero, más representará este hecho que los primeros de ellos poseen un mayor porcentaje de la traza, de manera tal que un consecuente proceso de poda en base a este criterio nos indicaría que si un bajo porcentaje en la cantidad considerada de los primeros autovalores acumula un gran porcentaje de la traza, entonces, una pequeña cantidad de los primeros mosaicos acumulan el mayor porcentaje de la energía visual de la imagen original $\boldsymbol{I}$.

La Fig.1.5 esquematiza el proceso de poda [33]. Obsérvese que la superficie gris en las matrices

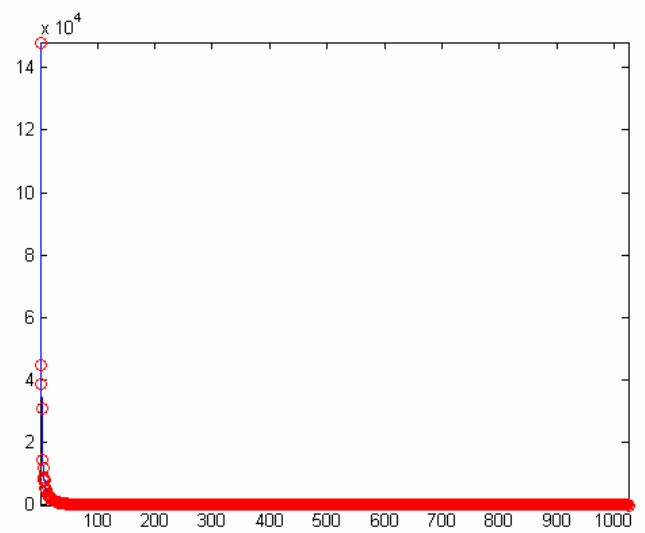


Figura 1.4: Espectro de autovalores: el eje de ordenadas representa el valor de los autovalores, mientras que el eje de absisas representa su orden en la traza de Λ_y o índice.

$\boldsymbol{Y_p}$ y $\boldsymbol{V_p}$ representan los elementos eliminados de ambas matrices en el algoritmo de compresión, mientras que los blancos nos indican los elementos a ser almacenados o transmitidos.

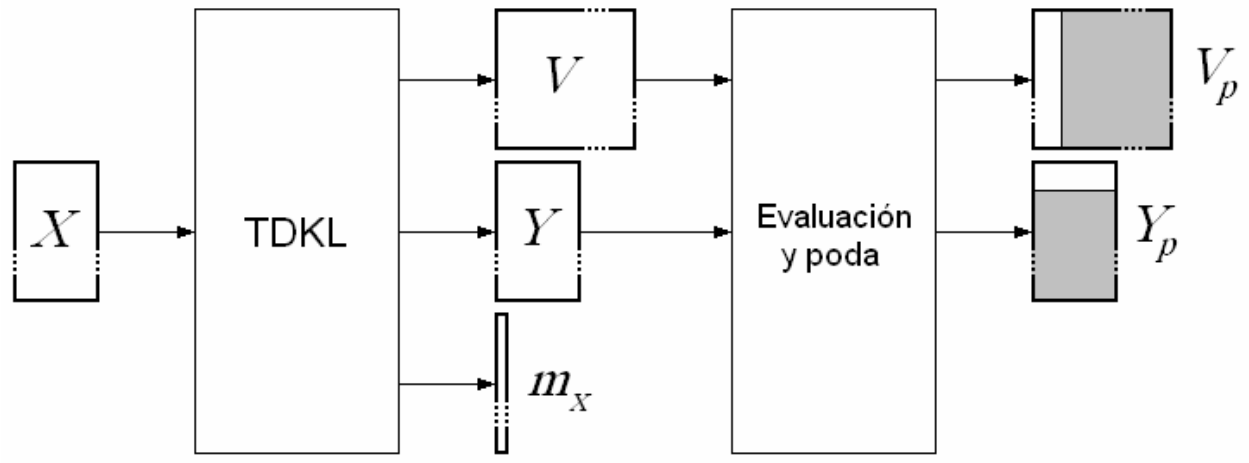


Figura 1.5: Detalle de aplicación de la TDKL en el algoritmo de compresión y posterior poda de las matrices $\boldsymbol{V}$ e $\boldsymbol{Y}$.

En el algoritmo de descompresión, las superficies grises son completadas con ceros antes de aplicar la Ecuación (9) como se muestra en la Fig.1.6.

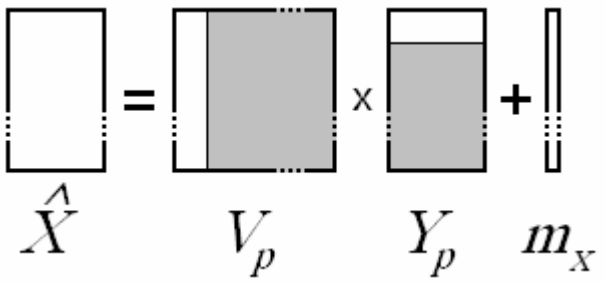


Figura 1.6: TDKL inversa en el algoritmo de descompresión en base a la Ecuación (1.9) y detalle de los sectores completados con ceros en las matrices podadas $\boldsymbol{V_p}$ e $\boldsymbol{Y_p}$.

Donde $\hat{\boldsymbol{x}}$ significa "$\boldsymbol{X}$ estimada".

El resto de los bloques constitutivos de la Fig.1.7, es decir, cuantización y compresión entrópica se desarrollan en detalle en al Apéndice B.

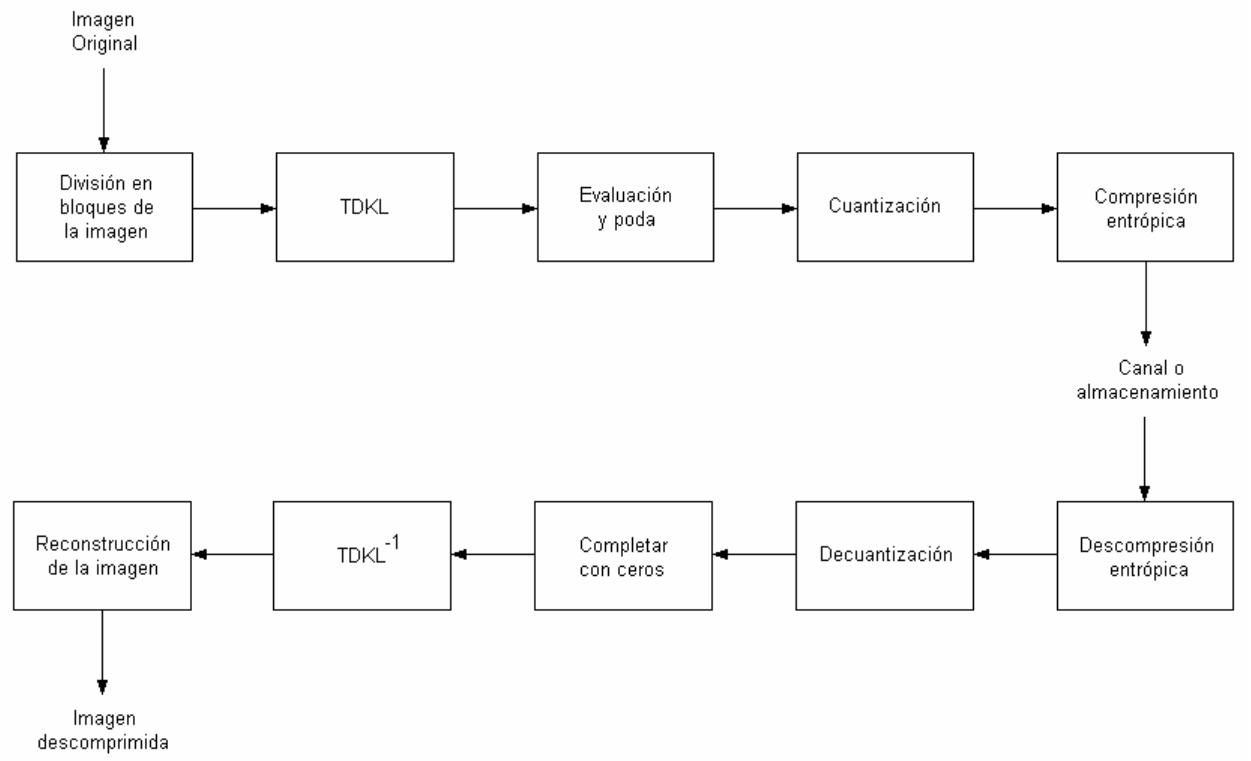


Figura 1.7: Diagrama en bloques de la compresión de imágenes con pérdidas mediante la TDKL.

En síntesis, el Algoritmo 1.1 representa la compresión con pérdidas en base a TDKL, el cual está compuesto por los primeros cinco bloques de la Fig.1.7, es decir,

ALGORITMO 1.1

(a) División en bloques de la imagen
(b) TDKL
(c) Evaluación y poda
(d) Cuantización
(e) Compresión entrópica

Mientras que el Algoritmo 1.2 representa la descompresión con pérdidas en base a la $TDKL^{-1}$, el cual está compuesto por los últimos cinco bloques de la Fig.1.7, es decir,

ALGORITMO 1.2

(a) Descompresión entrópica
(b) Decuantización
(c) Completar con ceros
(d) $TDKL^{-1}$
(e) Reconstrucción de la imagen

1.2.4 Métricas

Existe un gran número de métricas usadas en la medición de la eficiencia de un sistema de compresión [34, 35]. El principal defecto de todas ellas es que dicha medición es siempre relativa a los datos que están siendo comprimidos, por lo que si no se indican cuales fueron, la medida es inútil.

En esta sección se estudian algunas de las métricas más usadas, como también se presenta un conjunto de nuevas, las cuales tratan de establecer otro punto de vista acerca de los sistemas de compresión/descompresión basados en la TDKL, lo que facilita una compresión rápida de su eficiencia.

1.2.4.1. Tasa de Compresión (TC)

La TC, también conocida como poder de compresión, es un término empleado en Ciencias de la Computación para cuantificar la reducción en la dimensión de la representación de los datos producida por un algoritmo de compresión de datos. La TC es análoga a la tasa de compresión física usada para medir la compresión física de las sustancias, y que esta definida en la misma forma, es decir, como la tasa entre la *Dimensión de los Datos Sin Comprimir (DDSC)* y la *Dimensión de los Datos Comprimidos (DDC)* [36]:

$$\boldsymbol{TC} = \frac{\boldsymbol{DDSC}\left[\boldsymbol{bytes}\right]}{\boldsymbol{DDC}\left[\boldsymbol{bytes}\right]} \tag{1.10}$$

Entonces, un algoritmo que comprima un archive de 10MB a 2MB tendrá una TC = 10/2 = 5, frecuentemente expresada como tasa explícita, 5:1 (léase "cinco a uno"), o como tasa implícita, 5X. Es importante señalar que esta formulación se aplica igualmente para compresión, donde la dimensión sin comprimir corresponde a la imagen original; mientras que para la descompresión, la dimensión de los datos comprimidos corresponden a la imagen recuperada [37].

1.2.4.2. Número de bits por pixel (bpp)

El número de bits por pixel o *bpp*, se define como

$$\boldsymbol{bpp} = \frac{\boldsymbol{número\ de\ codificados}}{\boldsymbol{número\ de\ \ pixeles}} \tag{1.11}$$

Esta métrica junto a la TC son las más importantes dentro de la compresión de imágenes [35].

1.2.4.3. Tasa de Poda Teórica de los Bloques (TPTB)

La TPTB, se define como el cociente

$$\boldsymbol{TPTB} = \frac{\boldsymbol{nm}}{\boldsymbol{nmsp}} \tag{1.12}$$

Donde, en base a la poda establecida en la Sección 1.2.3

nmsp = número de mosaicos sobrevivientes de la poda = *nm-nmep*

nmep = número de mosaicos eliminados en la poda

Finalmente esta métrica no puede superar jamás en valor al número de mosaicos

$$\boldsymbol{TPTB} \le \boldsymbol{nm} \tag{1.13}$$

1.2.4.4. Tasa de Poda Real de los Bloques (TPRB)

La TPRB, se define como el cociente entre la *Dimensión de la Imagen*, y la *suma de las dimensiones de los elementos propios de la TDKL* y sobrevivientes del proceso de poda descrito en la Sección 1.2.3, es decir,

$$\boldsymbol{TPRB} = \frac{\boldsymbol{DJ}}{\boldsymbol{Dm_x} + \boldsymbol{DY_p} + \boldsymbol{DV_p}} \tag{1.14}$$

donde:

$\boldsymbol{DJ}$ = Dimensión de la matriz $\boldsymbol{J}$
= *nfm* x *ncm* x *nm*

$\boldsymbol{Dm_x}$ = Dimensión del vector valor medio $\boldsymbol{m_x}$
= *nm* x *1*

$\boldsymbol{DY_p}$ = Dimensión de la matriz $\boldsymbol{Y_p}$ (es decir, $\boldsymbol{Y}$ podada)
= *nfm* x *ncm* x *(nm-nmep)*

$\boldsymbol{DV_p}$ = Dimensión de la matriz $\boldsymbol{V_p}$ (es decir, $\boldsymbol{V}$ podada)
= *(nm-nmep)* x *nm*

Reemplazando apropiadamente las dimensiones definidas en (1.14), finalmente la TPRB quedará

$$\boldsymbol{TPRB} = \frac{\boldsymbol{nfm} \times \boldsymbol{ncm} \times \boldsymbol{nm}}{\boldsymbol{nm} + \boldsymbol{nfm} \times \boldsymbol{ncm} \times (\boldsymbol{nm} - \boldsymbol{nmep}) + (\boldsymbol{nm} - \boldsymbol{nmep}) \times \boldsymbol{nm}} \tag{1.15}$$

La TPRB es la único que nos permite saber que queda y que no antes de la compresión entrópica.

1.2.4.5. Error Cuadrático Medio (ECM)

Mucho mas conocida por sus siglas en inglés, es decir, *Mean Squared Error (MSE)*, se define para dos imágenes monocromáticas $\boldsymbol{I}$ e $\boldsymbol{I_d}$ de dimensiones *nfI* x *ncI*, donde la primera es la imagen original, mientras que la segunda es la resultante del proceso de compresión/descompresión con pérdidas [37]

$$\boldsymbol{MSE} = \frac{1}{\boldsymbol{nfI} \times \boldsymbol{ncI}} \sum_{\boldsymbol{nf}=1}^{\boldsymbol{nfI}} \sum_{\boldsymbol{nc}=1}^{\boldsymbol{ncI}} \left\| \boldsymbol{I}(\boldsymbol{nf}, \boldsymbol{nc}) - \boldsymbol{I_d}(\boldsymbol{nf}, \boldsymbol{nc}) \right\|^2 \tag{1.16}$$

1.2.4.6. Tasa Pico Señal-a-Ruido (TPSR)

Al igual que en el caso anterior, es mucho más popular su expresión en inglés, a saber, *Peak Signal-to-Noise Ratio (PSNR)*, y el cual hace mención a un término de la Ingeniería para la relación entre la potencia máxima posible de una señal y la potencia debida a la corrupción por ruido que afecta la fidelidad de su representación. Dado que muchas señales poseen un muy amplio rango dinámico, la PSNR se expresa usualmente en función de la escala logarítmica en decibeles.

La PSNR es más comúnmente usada como una medida de la calidad en la reconstrucción en la compresión de una imagen, etc. [36]. Se define más fácilmente via el MSE

$$\boldsymbol{PSNR} = 10\log_{10}\left(\frac{\boldsymbol{MAX_I^2}}{\boldsymbol{MSE}}\right) = 20\log_{10}\left(\frac{\boldsymbol{MAX_I}}{\sqrt{\boldsymbol{MSE}}}\right) \tag{1.17}$$

Donde, MAX_I es el valor máximo de un pixel de la imagen. Cuando los píxeles son representados usando 8 bits por pixel, entonces el valor máximo que podrá alcanzar un píxel será de 256. Mas generalmente, cuando las muestras están representadas usando modulación lineal de pulso codificado (por sus famosas siglas en inglés, PCM) con B bits por muestra, el valor máximo posible de MAX_I es 2^B-1.

Para imagines a color con tres rojo-verde-azul (RVA) valores por pixel, la definición de PSNR es la misma excepto que el MSE es la suma sobre todos los valores de las diferencias al cuadrado divididas por la dimensión de la imagen y por tres [36].

Valores típicos para el PSNR en compresión de imagines y video con pérdidas están entre 30 y 50 dB, para lo cual, cuanto más elevado sea el valor, mejor.

1.2.4.7. Energía acumulada total (E_{at})

La Energía acumulada total (E_{at}) se define como la suma de los autovalores de la matriz de covarianza $\boldsymbol{C_x}$, es decir,

$$\boldsymbol{E_{at}} = \sum_{\boldsymbol{k}=1}^{\boldsymbol{nm}} \lambda_{\boldsymbol{k}} \tag{1.18}$$

la misma cobrará particular importancia en la Sección 1.2.5.1 del presente capítulo.

1.2.4.8. Energía en función de los autovalores (E_{fa})

La *Energía en función de los autovalores* (E_{fa}) es complementaria a la graficación de los autovalores de la matriz diagonal $\Lambda_{\mathbf{y}}$ [38], y conjuntamente con la misma permite evaluar la poda considerando la TPTB. La misma se define como:

$$\boldsymbol{E_{fa,n}} = \sum_{\boldsymbol{k}=1}^{\boldsymbol{n}} \lambda_{\boldsymbol{k}}, \quad \boldsymbol{n} \in [1, \boldsymbol{nm}] \tag{1.19}$$

1.2.4.9. Energía porcentual acumulada (E_{pa}) y porcentual disipada (E_{pd})

La *Energía porcentual acumulada* (E_{pa}) permite establecer la energía relativa retenida posterior al proceso de poda, mientras que la *Energía porcentual disipada* (E_{pd}) hace lo propio con la energía relativa desperdiciada al podar los mosaicos asociados a los autovalores más pequeños. Ambas se definen de la siguiente manera:

$$E_{pa} = \frac{\sum_{k=1}^{nmsp} \lambda_k}{E_{at}} \times 100\% \tag{1.20}$$

$$E_{pd} = \left(100 - E_{pa}\right)\% \tag{1.21}$$

Por el *principio de conservación de la energía*, la suma de todas las energías porcentuales (obedeciendo linealmente el principio de superposición) que llamaremos *Energía porcentual total* (E_{pt}) suman un 100 %, es decir,

$$E_{pt} = E_{pa} + E_{pd} = 100\% \tag{1.22}$$

1.2.4.10. Entropía normalizada (H_n)

Si la *energía relativa capturada* por el *k-ésimo* autovalor es

$$E_k = \frac{\lambda_k}{E_{at}} \tag{1.23}$$

entonces la *Entropía normalizada* que permite comparar entre distintos ejemplos será

$$H_n = -\lim_{nm \to \infty} \frac{1}{\ln(nm)} \sum_{k=1}^{nm} E_k \ln(E_k) \tag{1.24}$$

Esta métrica será particularmente importante en la Sección 1.2.5.1 y el límite es teórico [30].

1.2.4.11. Autovalores Normalizados (AN)

Esta métrica es fundamental para todos aquellos casos en los que se quiera comparar el rendimiento de decorrelación de ejemplos para un número similar o distinto de sub-bloques.

Es una expresión objetiva de evaluación de la influencia del número de mosaicos, así como la intervención de alguna otra herramienta de ayuda (como las que se estudiarán en el Capítulo 3) para igual cantidad de mosaicos, con o sin SMEP.

Esta métrica constituye un vector que puede fácilmente llevarse a un plano cartesiano ortogonal, la cual a diferencia de la Fig.1.4 (que representa valores absolutos), nos da una posibilidad mucho más útil al representar valores relativos, ver Fig.1.8(a). Ecuacionalmente responde a la forma:

$$\boldsymbol{AN_k} = \frac{\lambda_k}{\lambda_{\max}} = \frac{\lambda_k}{\lambda_1} \qquad \boldsymbol{con}\ \ \boldsymbol{k} \in [1, \boldsymbol{nm}] \tag{1.25}$$

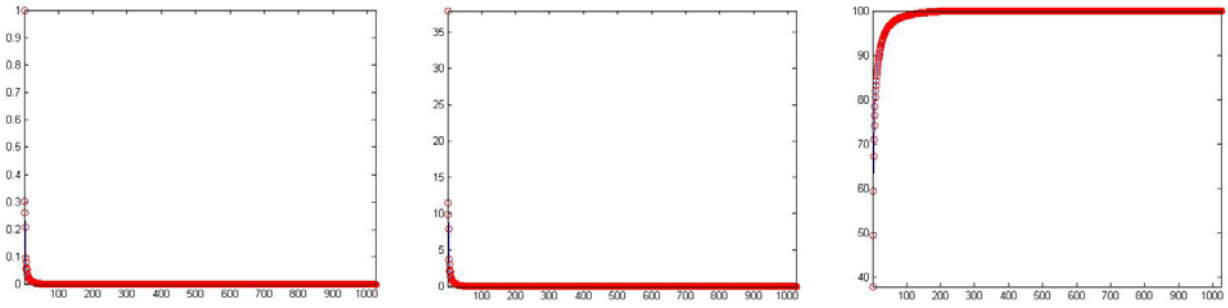


Figura 1.8: El eje de ordenadas representa (a) el valor normalizado de los autovalores, (b) el valor normalizado porcentual, y (c) la E_{rcpa}; mientras que el eje de absisas representa su orden en la traza de Λ_y o índice.

1.2.4.12. Energía Relativa Capturada Porcentual

De la Ecuación (1.23) tomamos la *energía relativa capturada* y la porcentualizamos. La Fig.1.8(b) representa una distribución típica, es decir, que porcentaje de la traza corresponde a cada autovalor en particular.

$$E_{rcp,k} = \frac{\lambda_k}{E_{at}} 100\% \qquad con\ \ k \in [1, nm] \tag{1.26}$$

1.2.4.13. Energía Relativa Capturada Porcentual Acumulativa

Basados en la Ecuación (1.26) acumulamos la misma dando lugar a:

$$\boldsymbol{E_{rcpa,n}} = \sum_{\boldsymbol{k}=1}^{\boldsymbol{n}} \boldsymbol{E_{rcp,k}} \tag{1.27}$$

Esta métrica en particular representa el porcentaje de traza acumulada hasta cada autovalor considerado excluido de la poda, ver Fig.1.8(c).

1.2.4.14. Energía Diferencial

Primero definimos los Autovalores Normalizados Acumulados (ANA) como:

$$\boldsymbol{ANA_n} = \sum_{\boldsymbol{k}=1}^{\boldsymbol{n}} \frac{\lambda_k}{\lambda_1} \tag{1.28}$$

y posteriormente una métrica particularmente útil, la cual será puesta a prueba en el Capítulo 3

$$\boldsymbol{E_d} = \lim_{\boldsymbol{nm}\to\infty} \frac{\boldsymbol{ANA_{nm}} - \boldsymbol{ANA_1}}{\boldsymbol{nm} - 1} \tag{1.29}$$

donde el límite es teórico.

1.2.4.15. Complejidad computacional (CC)

Esta métrica está relacionada con la cantidad de operaciones de adición y multiplicación realizadas sobre los elementos de las matrices y vectores empleadas en el cálculo del algoritmo en cuestión. Se representa mediante el operador "O", y en el caso particular de la TDKL el mismo resulta $O((nf\boldsymbol{I} \times nc\boldsymbol{I})^3)$, léase, la cantidad de operaciones de adición y multiplicación necesarias para el cálculo de la TDKL dependen de la cantidad de píxeles de la imagen $\boldsymbol{I}$ al cubo [15-18].

1.2.4.16. Tiempo empleado por la Unidad Central de Procesamiento (TEUCP)

Este tiempo lo mediremos en segundos, para lo cual emplearemos para todas las simulaciones del presente trabajo a la función *built-in* de MATLAB® cputime [39].

1.2.5 Rendimiento de compactación de la energía mediante la TDKL

En esta sección, se procederá a explicar mediante dos ejemplos como alcanzar un mayor rendimiento de compactación de la energía mediante la TDKL al comprimir imágenes con pérdidas, para finalmente establecer una demostración formal acerca del mismo, la cual estará desarrollada en el Capítulo 3 (Soluciones propuestas).

1.2.5.1 Reducción paulatina en el tamaño de los mosaicos, a TPTB constante [15-18]

En base a la Fig.1.1 y según lo establecido en las Secciones 1.2 y 1.2.1 procederemos a la selección de parámetros para este experimento que consiste en ir reduciendo paulatinamente el tamaño de los bloques tomados sobre la imagen, de manera tal de ir evaluando los cambios en la calidad de la imagen, así como, tomando nota de las métricas mas conspicuas a los efectos de evaluar la performance de compresión/descompresión para una TPTB constante.

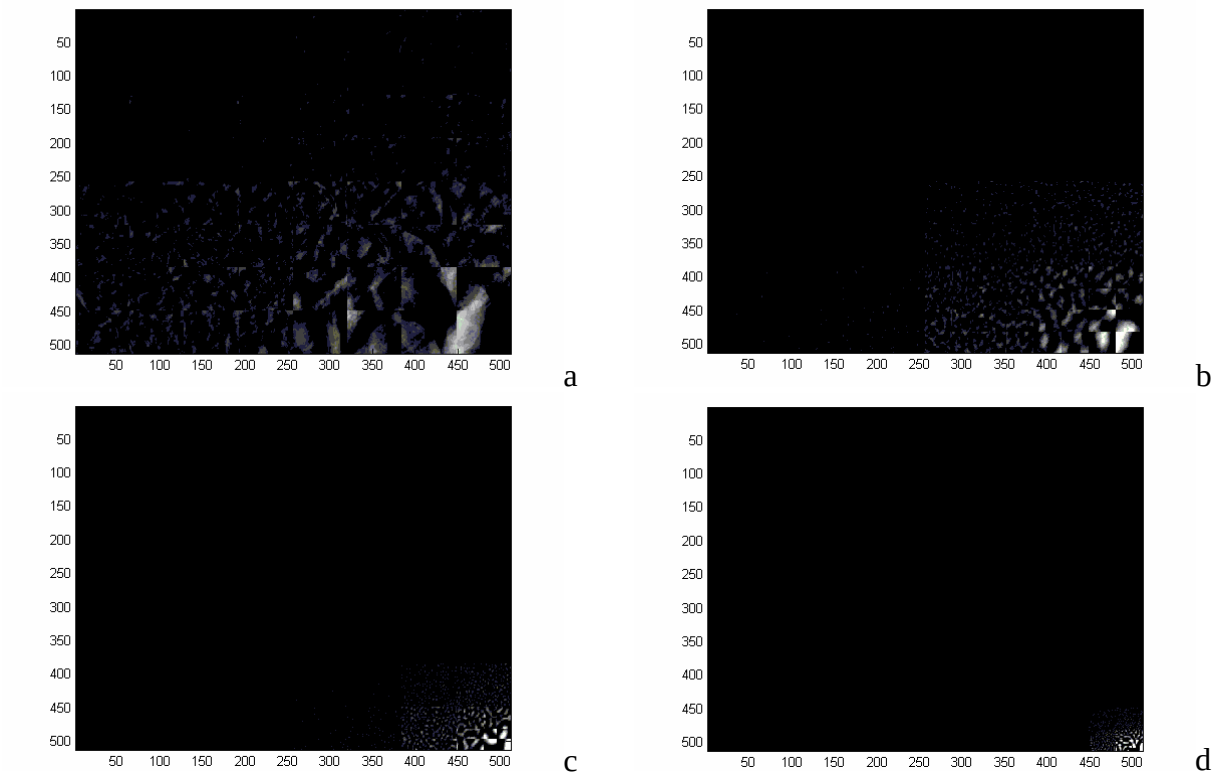


Figura 1.9: Salidas de la TDKL para mosaicos de: a) 64x64, b) 32x32, c) 16x16, y d) 8x8

La anterior, es decir, la Fig.1.9 muestra la salida de la TDKL para cuatro dimensiones distintas de los bloques. A medida que el tamaño de los bloques disminuye se observa que los bloques asociados a los autovalores de mayor tamaño se agrupan en el margen inferior derecho de las imágenes [40, 41]. En la Fig.1.10 se puede observar que a medida que los bloques disminuyen de tamaño, los autovalores caen más rápidamente a cero, y los primeros de estos son los más grandes [42-133]. En la Fig.1.11 la E_{fa} crece más rápidamente con bloques más pequeños. Idénticas consideraciones a las formuladas para la Fig.1.9 se observan en la Fig.1.12 para la salida de la TDKL luego de la poda.

La Fig.1.13 muestra como la calidad de la imagen descomprimida aumenta notablemente con la reducción del tamaño de los bloques. Los mismos comienzan con una dimensión de 64x64 pixeles y luego se va reduciendo la misma, pasando por dimensiones intermedias como 32x32 pixeles y 16x16 pixeles, hasta alcanzar la dimensión mínima considerada de 8x8 pixeles. Por su parte, la Fig.1.14 muestra el error pixel-a-pixel entre la imagen original y la imagen recuperada como resultado de la descompresión. Claramente se puede comprobar por una simple inspección visual de la figura mencionada que para mosaicos más pequeños no se nota este error. En dicha figura se utilizó una paleta de colores en la que el error absoluto positivo posee color rojo, mientras que el error absoluto negativo posee color azul.

Simultáneamente, la Tabla 1.I permite comparar todas las métricas para los cuatro tamaños de mosaicos elegidos. Se puede observar que a medida que el tamaño de estos disminuye también lo hace la H_n . Esto sucede porque a medida que los bloques se hacen cada vez más pequeños tienden al límite en el que son de una dimensión igual a 1x1, en este caso cada bloque será igual a cualquier otro multiplicado por un escalar, es decir, se trata del mismo símbolo multiplicado por un escalar, por eso la entropía baja.

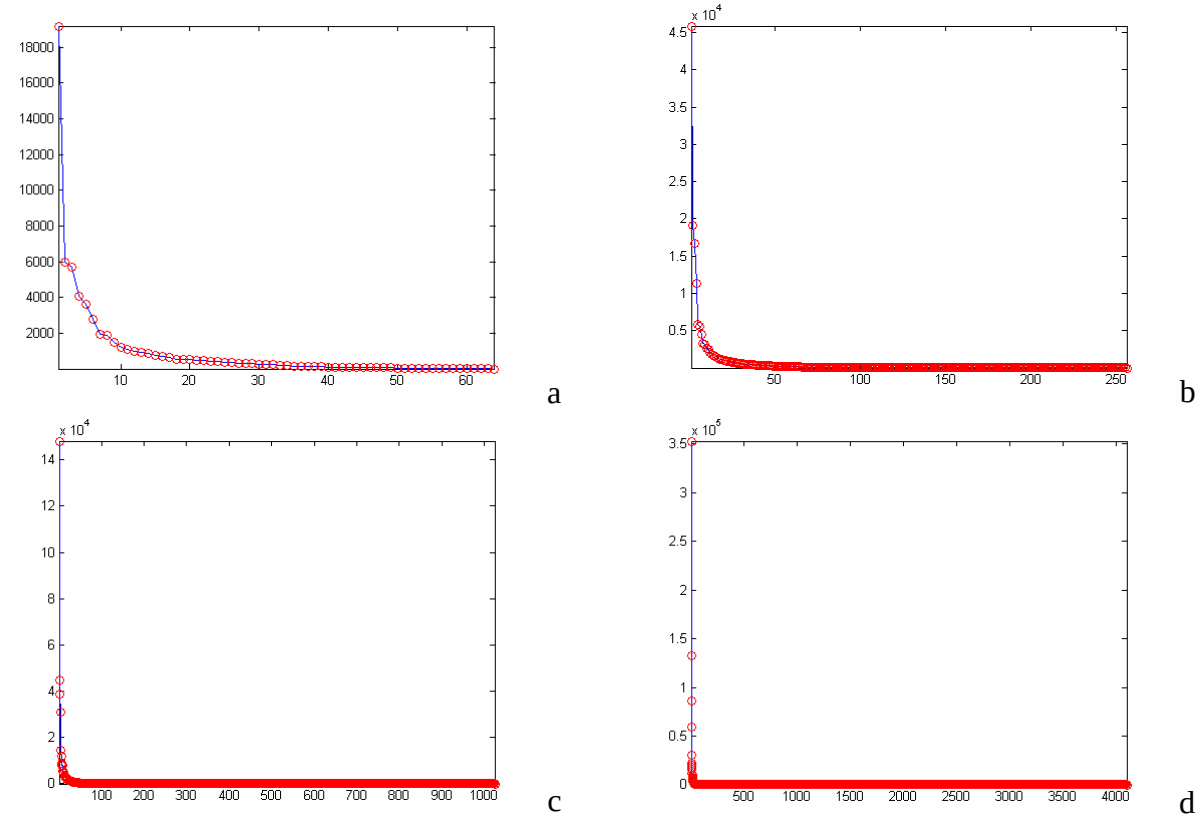


Figura 1.10: Autovalores para mosaicos de: a) 64x64, b) 32x32, c) 16x16, y d) 8x8

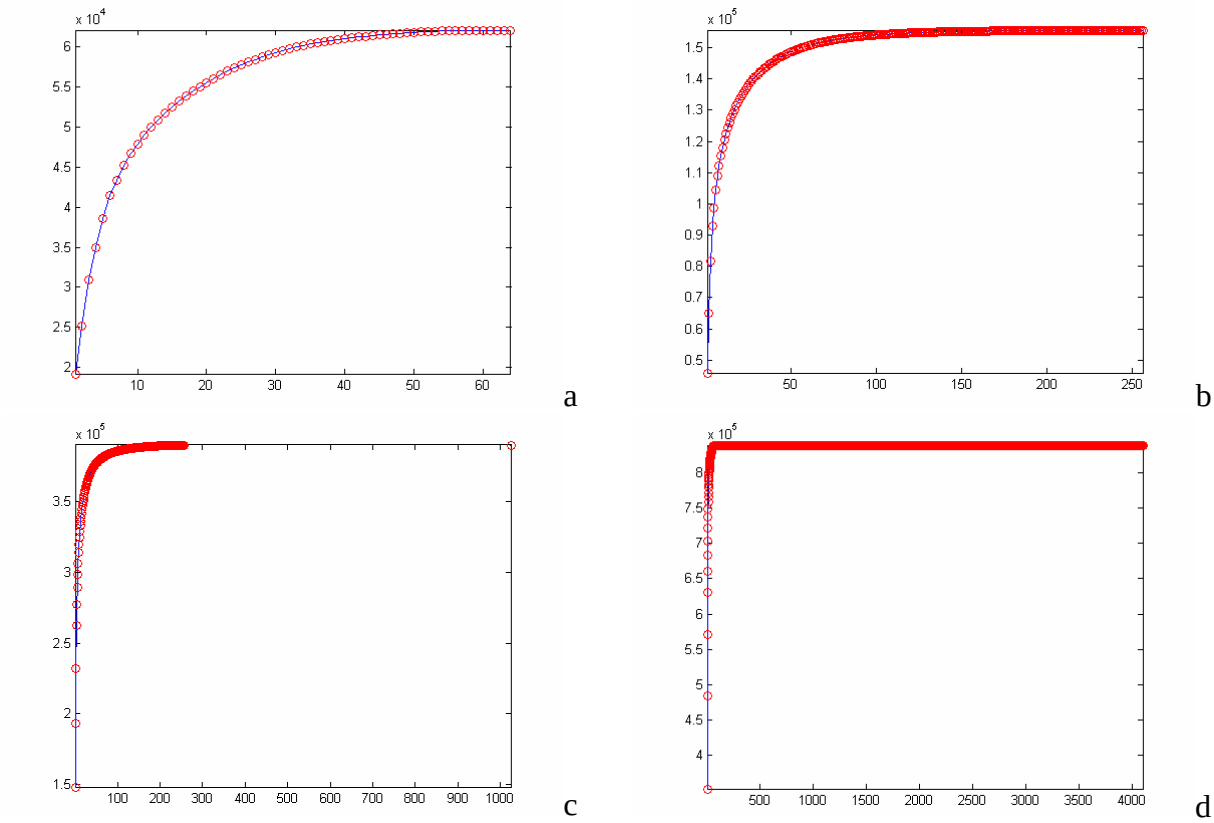


Figura 1.11: E_{fa} para mosaicos de: a) 64x64, b) 32x32, c) 16x16, y d) 8x8

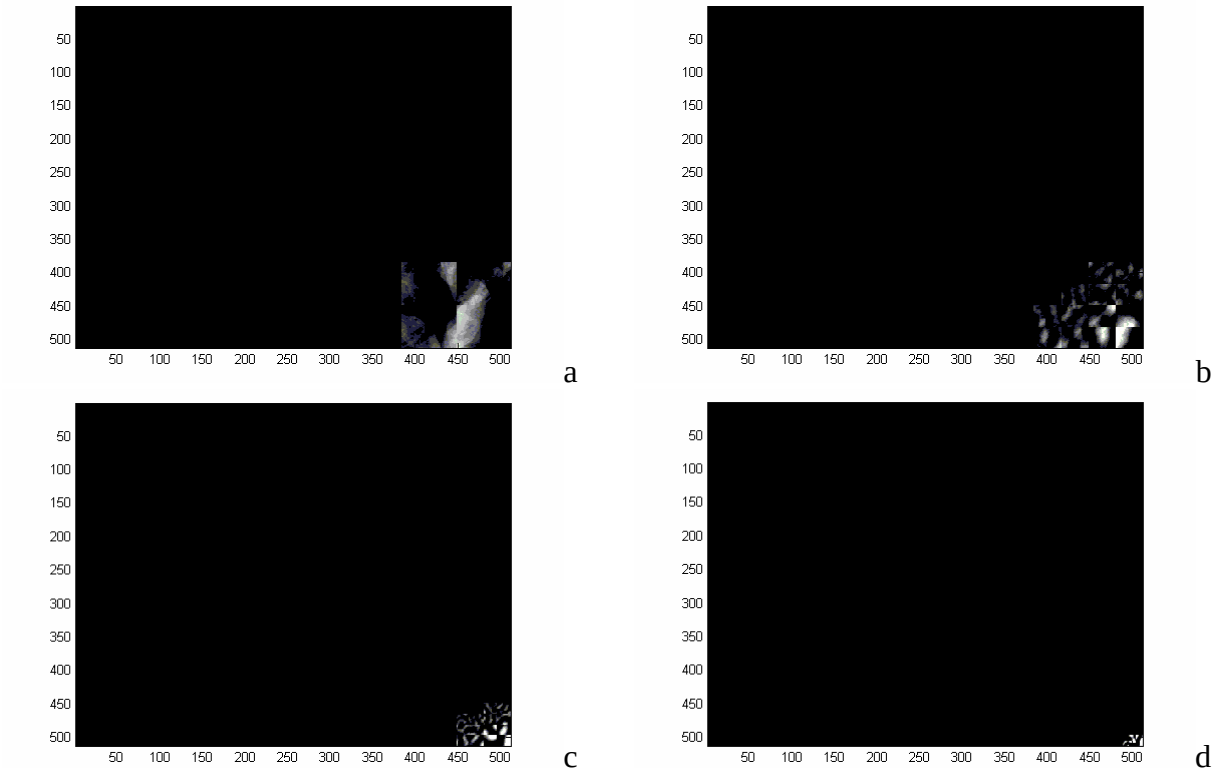


Figura 1.12: Salidas podadas de la TDKL para mosaicos de: a) 64x64, b) 32x32, c) 16x16, y d) 8x8

Figura 1.13: Imagen descomprimida para mosaicos de: a) 64x64, b) 32x32, c) 16x16, y d) 8x8

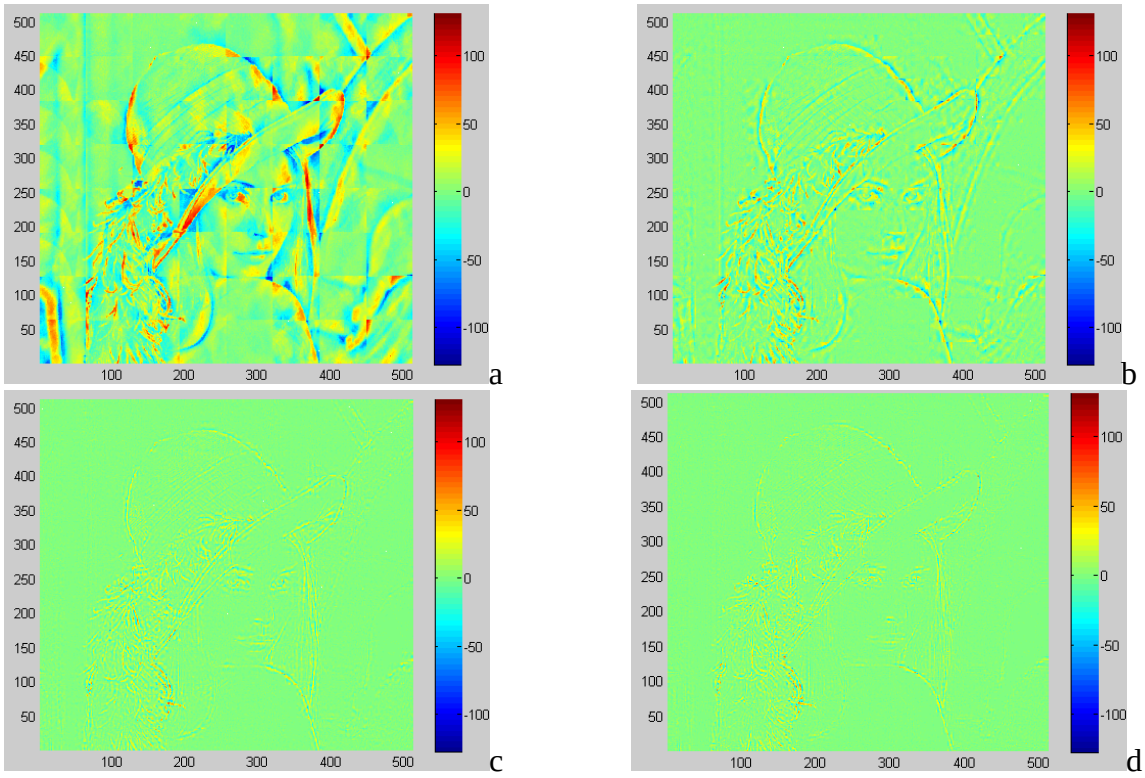


Figura 1.14: Error píxel-a-pixel para mosaicos de: a) 64x64, b) 32x32, c) 16x16, y d) 8x8

Tabla 1.I: Métricas para cuatro tamaños de bloques con TPTB = 16

Métrica	Tamaño de los bloques *nfm* x *ncm*			
	64 x 64	32 x 32	16 x 16	8 x 8
MSE	424.522086666706	122.592655503734	52.7854527500756	43.5184696591238
PSNR	21.8518007060374	27.2461590838327	30.9056610994121	31.7440674601335
TPRB	15.6934865900383	15.5151515151515	13.8378378378378	10.5295629820051
cputime [seg]	45.703125	204.21875 (~3.4 min)	1987.90625 (~33.13 min)	27049.078125 (~7 hor, 51 min)
H_n	0.673251694930656	0.535562327469663	0.381071454326299	0.26563401525018
E_{at}	62129.0433946848	155675.874515534	390195.372283935	839069.187011719
E_{pa}	56.2693837500908	82.8459200673184	97.6562454457456	100
E_{pd}	43.7306162499092	17.1540799326816	2.34375455425445	0

Como se mencionó anteriormente, la Tabla 1.I permite observar cuantitativamente esta disminución de la entropía a medida que disminuye el tamaño de los mosaicos, lo cual es físicamente comprensible dado que si los bloques se hallaran en el límite de su dimensión, es decir, 1x1 entonces cada bloque (pixel) sería igual a cualquier otro de la imagen multiplicado por un escalar

$$\boldsymbol{escalar} \times \boldsymbol{pixel}_i^{1\times1} = \boldsymbol{pixel}_j^{1\times1} \;\forall\; \boldsymbol{i} \neq \boldsymbol{j} \tag{1.30}$$

No obstante, si el bloque es de mayor dimensión, en general, no hay escalar que cumpla con la igualdad anterior, de hecho, la probabilidad de que esto ocurra disminuye con el tamaño.

$$\boldsymbol{escalar} \times \boldsymbol{bloque}_i^{nfm\times ncm} \neq \boldsymbol{bloque}_j^{nfm\times ncm} \;\forall\; \boldsymbol{i} \neq \boldsymbol{j} \wedge \boldsymbol{nfm} > 1 \wedge \boldsymbol{ncm} > 1 \tag{1.31}$$

Las Figuras 1.15 y 1.17 muestran un ejemplo típico de lo mencionado para dos mosaicos de 64x64 cualesquiera de la imagen de la Fig.1.2.

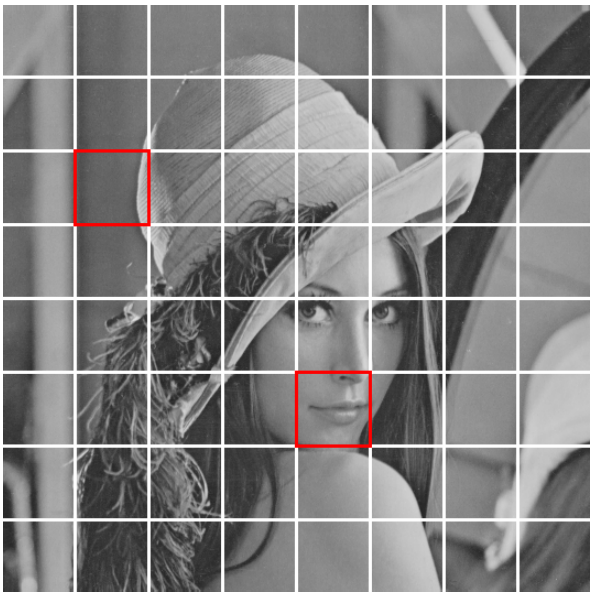


Figura 1.15: Imagen de Lena dividida en bloques de 64-por-64 pixeles, con la selección en rojo de dos mosaicos de la misma

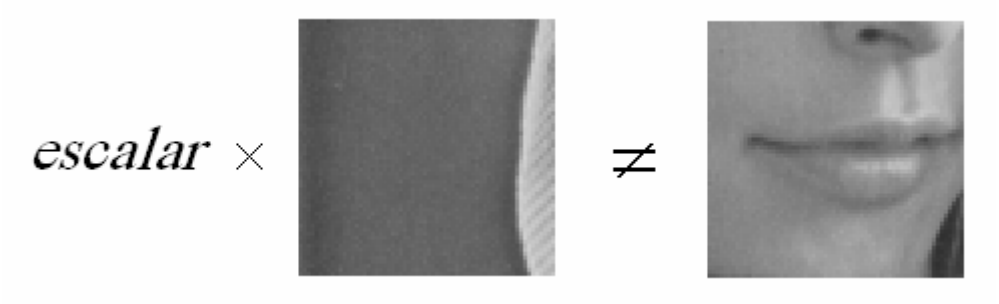


Figura 1.16: Detalle de los mosaicos seleccionados.

1.2.5.2 Métodos rápidos y aproximados de obtención de las componentes principales

Estos métodos aproximados se desarrollarán en detalle en el Apéndice C, no obstante, aquí haremos algunas consideraciones respecto de sus posibles aplicaciones.

Dichas aproximaciones pueden ser:

1. conexionistas como:
    1.1. el Algoritmo de Oja y Karhunen [134-136]
    1.2. el Algoritmo Hebbiano Generalizado (AHG) de Sanger [137]
    1.3. el Algoritmo Hebbiano Kernelizado (AHK) de Kim [138]
    1.4. el AHK con Meta-Descenso Estocástico (MDE) de Günter *et al* [139-140]

y

2. no conexionistas como:
    2.1. el PCA Kernelizado de Schölkopf *et al* [141, 142]
    2.2. el PCA Rápido de Sharma *et al* [143]
    2.3. el PCA Recursivo de Príncipe *et al* [144]
    2.4. la TDKL basado en una red sistólica de Güleryüz *et al* [145, 146]
    2.5. el PCA basado en el filtro de Kalman de Zhong *et al* [147]
    2.6. el Algoritmo de Gradiente Estocástico de Dehaene [148]

Estos métodos tienen los siguientes inconvenientes a los efectos de obtener un buen resultado:

a. Son falsamente rápidos, de hecho tienen una elevada complejidad computacional en pos de converger y minimizar la métrica de la cual derivan [134-140].

b. Son imprecisos, es decir, rara vez llegan a los correctos valores de autovalores y por ende de los autovectores [141-148].

c. En la mayoría de los casos arriban a un espacio lineal equivalente, es decir, aunque decorrelacionan no llegan a los mismos autovalores y autovectores, sino a otros. En el único caso que llegan al mismo autovalor es para un espacio de un único vector.

d. Son de difícil implementación (codificación).

e. Dependen fuertemente de una apropiada elección de las condiciones iniciales.

f. Por lo general y especialmente en el caso de los conexionistas, trabajan con un espacio de dimensión menor al necesario, por lo que para hacerlos trabajar en la dimensión requerida hay que definir elementos ficticios que acompañen a la matriz $\boldsymbol{X}$ a modo de $\boldsymbol{X}+\boldsymbol{\Delta X}$, con $\boldsymbol{\Delta X}$ ficticio.

Los mencionados métodos se tratan en detalle en el Apéndice C.

1.2.5.3 Propuesta basada en atributos interbloques

La Fig.1.17 muestra dos opciones de grilla para la Fig.1.1, ambas conformadas por bloques de 64x64 pixeles cada una. La Fig.1.17(a) es directamente la Fig.1.1 dividida en sub-bloques (o mosaicos) como hemos establecido en la Fig.1.2, mientras que la Fig.1.17(b) es el resultado de aplicar la Transformada de Haar sobre la imagen de la Fig.1.1. mediante un procedimiento recursivo que se detallará en el Capítulo 3 (Efecto de utilizar SMEP en la TDKL).

La Fig.1.18(a) se corresponde con la aplicación de la TDKL a los bloques de la Fig.1.17(a) mientras que la Fig.1.18(b) hace lo propio con los de la Fig.1.17(b). La Fig.1.18 muestra la distribución de autovalores para ambos casos. Claramente puede observarse que la eficiencia de decorrelación de la Fig.1.18(b) es muy superior a la de la Fig.1.18(a). La eficiencia dependerá de la rapidez con la que caen a cero los autovalores de la Fig.1.18, es decir, cuanto más rápido lo hagan, mayor será la eficiencia de decorrelación de los sub-bloques, dado que mayor cantidad de ellos podrán ser eliminados en el proceso de poda con un menor error resultante de este proceso, lo que implica un mejor resultado en todas las métricas vistas.

La Tabla.1.II muestra la comparación de métricas para varios casos. En dicha tabla puede observarse que a igual tamaño de los bloques (64x64) el método propuesto no incrementa la complejidad computacional, pero si mejora dramáticamente a todas las métricas. De hecho, tiene métricas del orden de aquellas para un tamaño de bloques de 16x16 lo cual implica en si mismo una merma sustancial en la complejidad computacional. Esto se debe a que los bloques de la Grilla 2 son más parecidos entre si (aunque provenientes de distintas sub-bandas espectrales resultantes de la aplicación de la Transformada de Haar) que los de la Grilla 1 entre si. No solo son más parecidos morfológicamente sino que la morfología guarda la ubicación posicional, es decir, lo cual insinúa que los vectores columnas de la matriz de representación $\boldsymbol{X}$ deberían ser linealmente dependientes como caso límite de dicha eficiencia.

Similares características se dán en el caso de imágenes satelitales multi e hiperespectrales donde dichas bandas se corresponden con la misma escena a distinta longitud de onda [149-165], razón por la cual la TDKL es tan eficiente en el proceso de decorrelación, mientras que no lo es en el caso de bloques como los de la Grilla1 [69, 149-151, 154, 155, 163-168].

Por lo tanto, un método que, basado en la TDKL, permita ser aplicado a los bloques de la Grilla 1 y así decorrelacionarlos con la eficiencia que el caso mencionado, será muy bienvenido. Dicho método deberá explotar al máximo los atributos interbloques agrupados bajo el nombre de SMEP, El cual junto con el método en si serán formalizados en el Capítulo 3 (Soluciones Propuestas).

El conjunto de figuras de las que se dispone a continuación se completan con la E_{rcap} para el caso

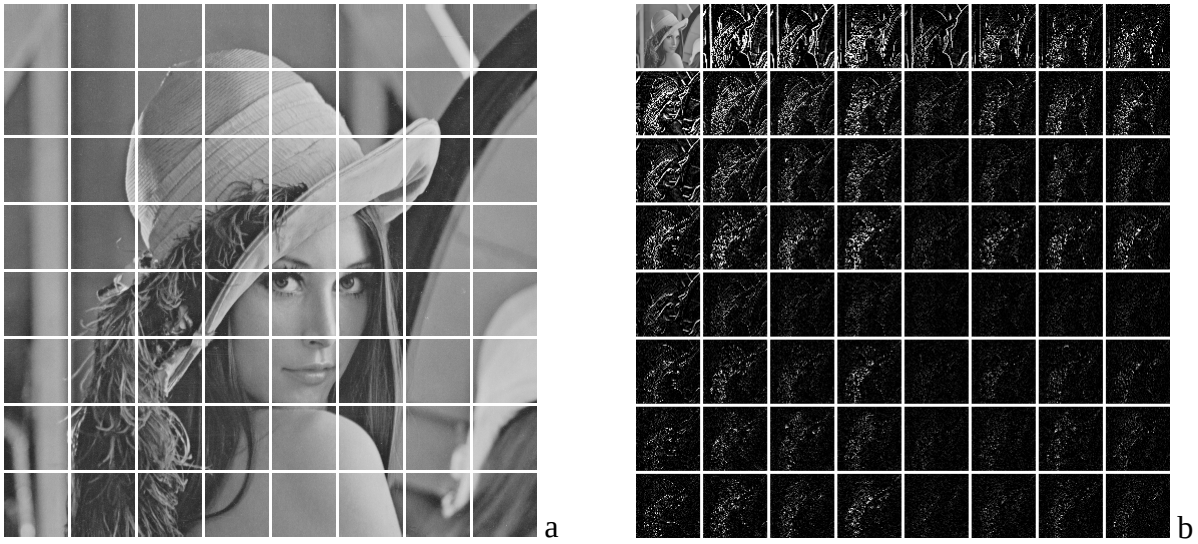


Figura 1.17: Grillas para: a) bloques sin aplicar Haar, y b) bloques aplicando Haar

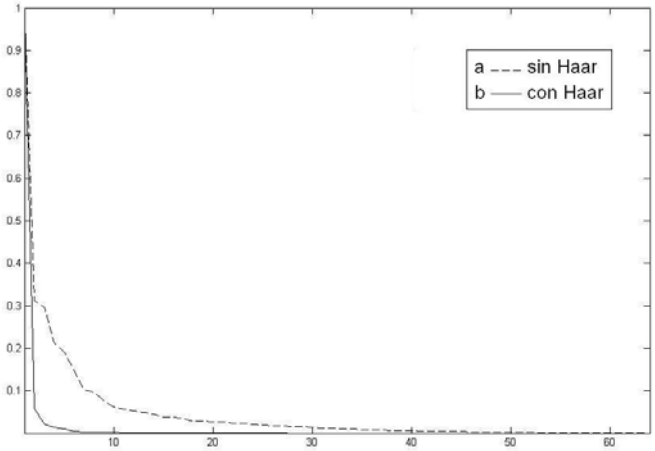


Figura 1.18: Autovalores normalizados contra el primero: a) Grilla 1: sin Haar, y b) Grilla 2: con Haar.

Tabla 1.II: Métricas para las dos grillas con TPTB = 16

Métrica	Grilla 1			Grilla 2
	64 x 64	32 x 32	16 x 16	64 x 64
MSE	424.522086666706	122.592655503734	52.7854527500756	65.2346661268645
PSNR	21.8518007060374	27.2461590838327	30.9056610994121	29.9860191687444
TPRB	15.6934865900383	15.5151515151515	13.8378378378378	15.6934865900383
cputime [seg]	45.703125	204.21875 (~3.4 min)	1987.90625 (~33.13 min)	46.046875
H_n	0.673251694930656	0.535562327469663	0.381071454326299	0.153388064093755
E_{at}	62129.0433946848	155675.874515534	390195.372283935	6867494.71273041
E_{pa}	56.2693837500908	82.8459200673184	97.6562454457456	96.1091896880486
E_{pd}	43.7306162499092	17.1540799326816	2.34375455425445	3.89081031195144

de ambas grillas (Fig.1.19), así como las salidas de la TDKL antes (Fig.1.20) y después de la poda (Fig.1.21), mientras que la Fig.1.22 el resultado de la descompresión para ambas grillas. Finalmente, la Fig.1.23 nos muestra el error *pixel a pixel* para ambas grillas.

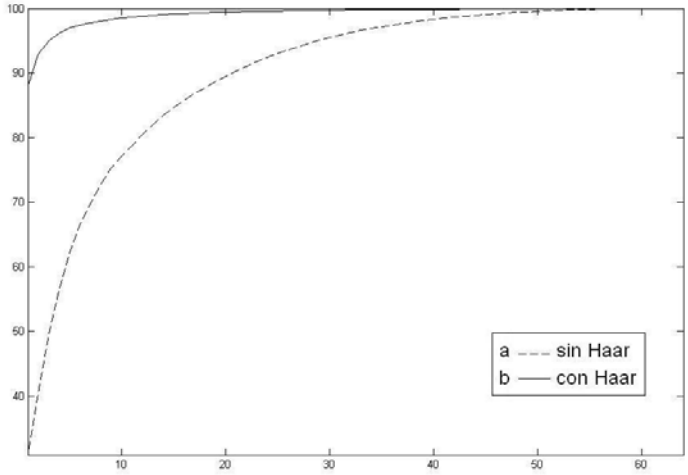
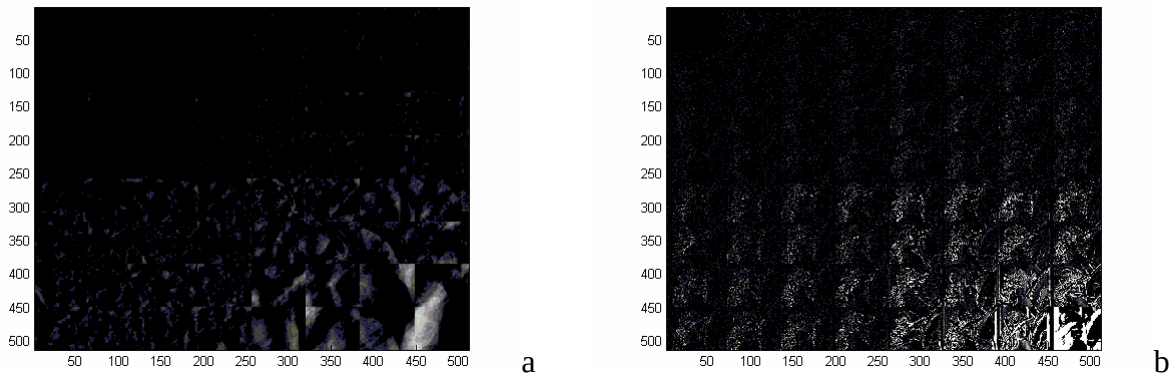
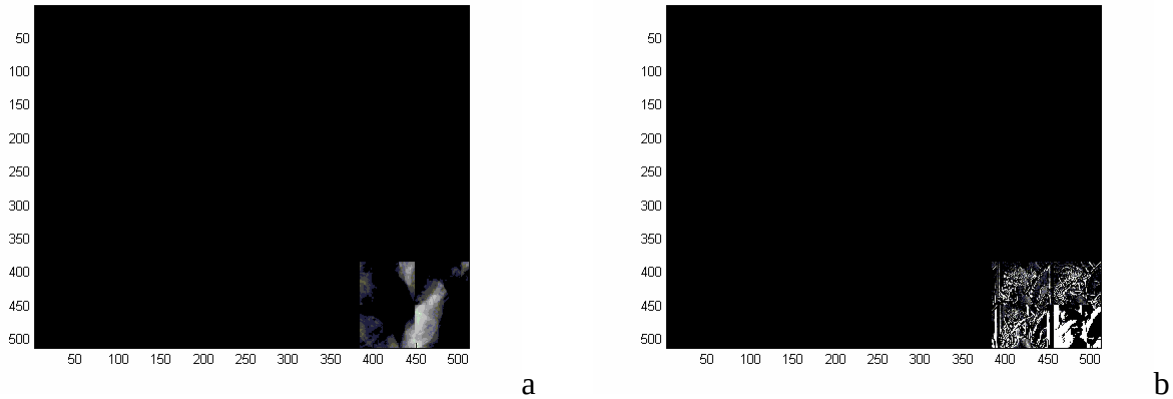


Figura 1.19: E_{rcap} para: a) Grilla 1: sin Haar, y b) Grilla 2: con Haar.



a b

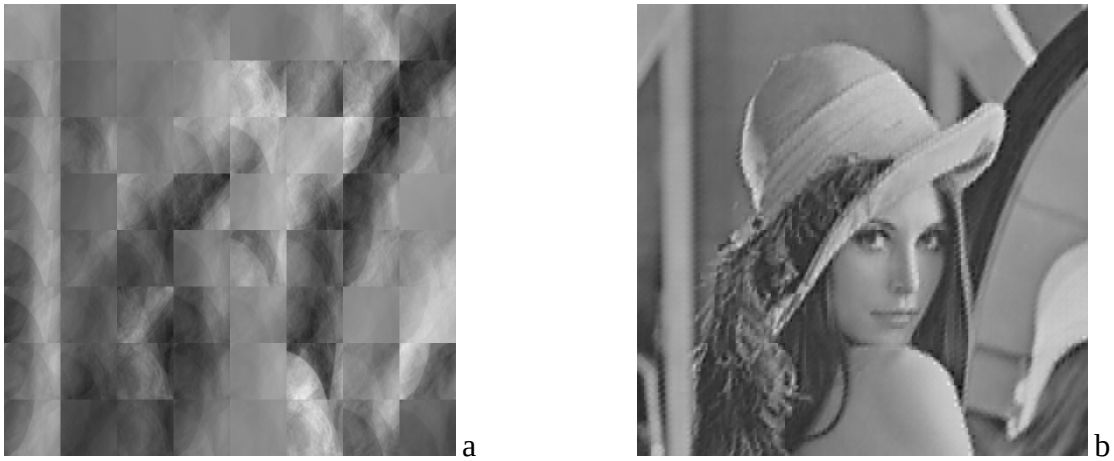
Figura 1.20: Salidas de la TDKL para: a) Grilla 1, y b) Grilla 2



a b

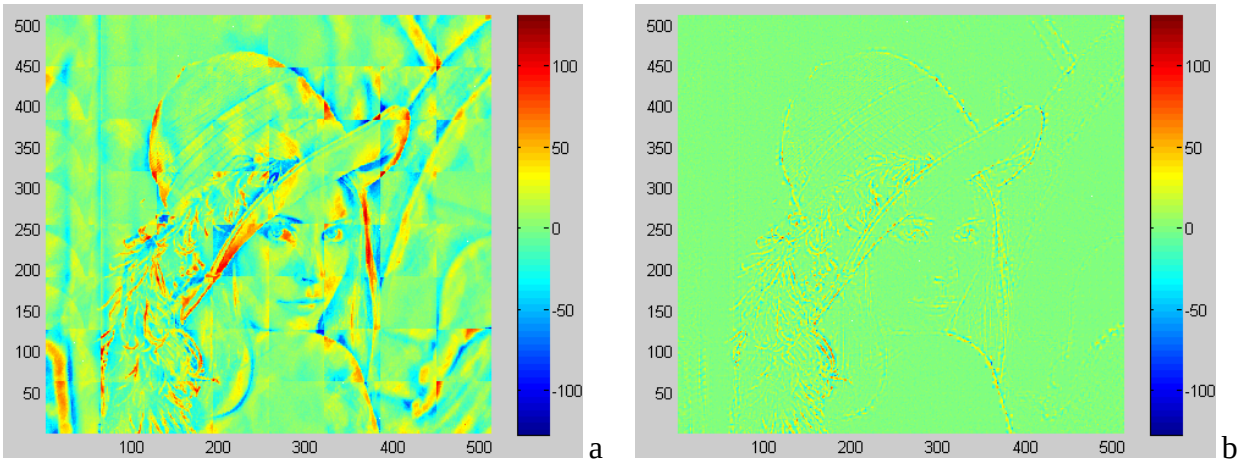
Figura 1.21: Salidas podadas de la TDKL para: a) Grilla 1, y b) Grilla 2

El método propuesto es particularmente interesante en el caso de imágenes satelitales dado que permitiría por primera vez aplicar un procedimiento de decorrelación inter e intrabanda simultáneamente aplicando a tal efecto la TDKL. Esto trae aparejado por primera vez un único procedimiento de implementación con la consiguiente congruencia tecnológica en ambos casos [37]. Todo esto acompañado de una baja en la complejidad computacional con una elevada eficiencia de decorrelación [37].



a b

Figura 1.22: Imagen descomprimida para: a) Grilla 1, y b) Grilla 2



a b

Figura 1.23: Error píxel-a-pixel para: a) Grilla 1, y b) Grilla 2

1.3 Conclusiones del capítulo

En este capítulo se ha desarrollado la aplicación del procedimiento de compresión/descompresión de imágenes monocuadro con pérdidas mediante la aplicación de la TDKL. Además, se han compilado las métricas más conspicuas a los efectos de evaluar la eficiencia de decorrelación de la TDKL y se han propuesto nuevas. Por otra parte, se han estudiado todas las posibilidades de mejora en el rendimiento de decorrelación de la TDKL. A partir de dicho estudio ha surgido una propuesta, la cual a la luz de las simulaciones de la Sección 1.2.5.3, se insinúa como la mejor solución existente y que será formalmente desarrollada en el capítulo siguiente.

Capítulo 2

2. Vehiculizadores de SMEP

2.1. Introducción

En este capítulo se desarrollarán tanto la TDC como la TDO. En este último caso y en particular, la Transformada de Haar [169-186], dado sus valiosos atributos sobre el resto de las bases de onditas [187-284]. No obstante, también se estudiarán la Transformada de Walsh-Hadamard [285] y la Transformada Discreta de Hartley [286], a los efectos de ser empleadas como instrumentos vehiculizadores de SMEP durante el Capítulo 3 (Soluciones Propuestas).

2.2. Transformadas

2.2.1. La Transformada Discreta Coseno (TDC)

Al igual que otras transformadas, la Transformada Discreta Coseno (TDC) intenta descorrelacionar los datos de la imagen. Luego de la transformación, los coeficientes resultantes pueden ser codificados independientemente sin pérdida en la eficiencia de compresión. Esta sección describe a la TDC ya algunas de sus propiedades [287-311].

2.2.1.1. La TDC unidimensional (1D)

La definición más común de la TDC de una secuencia 1D de longitud N es

$$\boldsymbol{C}(\boldsymbol{u}) = \alpha(\boldsymbol{u}) \sum_{\boldsymbol{x}=0}^{\boldsymbol{N}-1} \boldsymbol{f}(\boldsymbol{x}) \cos\left[\frac{\pi(2\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right] \tag{2.1}$$

para $\boldsymbol{u} = 0,1,2,\ldots,\boldsymbol{N}-1$. Similarmente, la transformación inversa se define como

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{\boldsymbol{u}=0}^{\boldsymbol{N}-1} \alpha(\boldsymbol{u})\boldsymbol{C}(\boldsymbol{u}) \cos\left[\frac{\pi(2\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right] \tag{2.2}$$

para $\boldsymbol{x} = 0,1,2,\ldots,\boldsymbol{N}-1$. Para ambas ecuaciones (2.1) y (2.2) $\alpha(\boldsymbol{u})$ se define como

$$\alpha(\boldsymbol{u}) = \begin{cases} \sqrt{\dfrac{1}{\boldsymbol{N}}} & \textbf{\textit{para}}\ \boldsymbol{u} = 0 \\ \sqrt{\dfrac{2}{\boldsymbol{N}}} & \textbf{\textit{para}}\ \boldsymbol{u} \neq 0 \end{cases} \tag{2.3}$$

Esta claro de la Ecuación (2.1) que para $\boldsymbol{u} = 0$, $\boldsymbol{C}(\boldsymbol{u}=0) = \sqrt{\dfrac{1}{\boldsymbol{N}}} \sum_{\boldsymbol{x}=0}^{\boldsymbol{N}-1} \boldsymbol{f}(\boldsymbol{x})$. Entonces, el coeficiente de la primera transformada es el valor promediado de la secuencia de muestras. En la literatura, este valor se refiere al *coeficiente DC (Direct Current, corriente continua)*. Todos los otros coeficientes

de la transformada son llamados los *coeficientes AC (Alternating Current, corriente alterna)*.

Para fijar ideas, ignoremos por un momento las componentes $f(x)$ y $\alpha(u)$ en (2.1). El esquema de $\sum_{x=0}^{N-1}\cos\left[\frac{\pi(2x+1)u}{2N}\right]$ para $N = 8$ y variando los valores de u se muestra en la Fig.2.1.

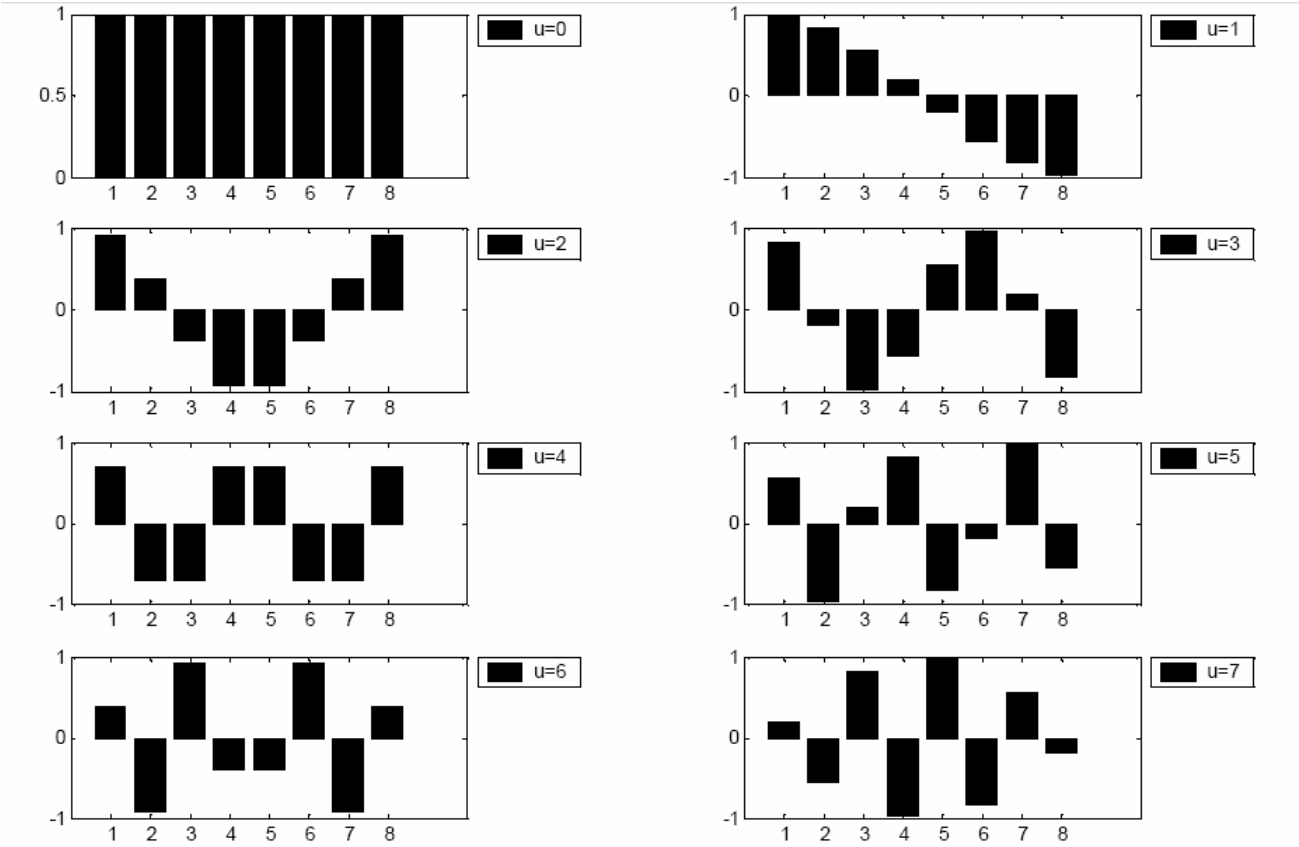


Figura 2.1: Función base coseno uni-dimensional ($N = 8$).

De acuerdo a nuestra observación previa, la primera forma de onda en el margen superior izquierdo de la Fig.2.1 ($u = 0$) hace a un valor constante (DC), mientras que, todas las otras formas de onda ($u = 1, 2, \ldots, 7$) dan formas de ondas en frecuencias progresivamente crecientes [288]. Estas formas de onda reciben el nombre de funciones base coseno. Notemos que estas funciones base son ortogonales. Por lo que, la multiplicación de cualquier forma de onda en la Fig.2.1 con otra forma de onda seguida de una suma sobre todos los puntos de las muestras dará un valor escalar de cero, mientras que la multiplicación de cualquier forma de onda en la Fig.2.1 consigo misma seguida de una suma dará un valor escalar constante. Las formas de onda son independientes, es decir, ninguna de las funciones base puede ser representada como una combinación de otras funciones base [289].

Si la secuencia de entrada tiene más de N puntos de muestras entonces puede ser dividida en sub-secuencias de longitud N y la TDC puede ser aplicada a estos trozos independientemente. Un aspecto a poner de relieve en este punto consiste en notar que en cada cálculo los valores de los puntos de las funciones base no cambiarán. Solo los valores de la $f(x)$ cambiará en cada sub-secuencia. Esta es una propiedad muy importante, Dado que muestra que las funciones base pueden ser pre-calculadas

off-line y entonces multiplicadas por las sub-secuencias. Esto reduce el número de operaciones matemáticas (por ejemplo, multiplicaciones y adiciones) y de esta manera la eficiencia del cálculo.

2.2.1.2. La TDC bidimensional (2D)

El objetivo de esta sección es estudiar la eficacia de la TDC aplicada a imágenes. Esto requiere una extensión de las ideas presentadas en la sección anterior para un espacio bidimensional. La TDC-2D es una extensión directa del cado 1D y esta dada por

$$\boldsymbol{C}(\boldsymbol{u},\boldsymbol{v}) = \alpha(\boldsymbol{u})\alpha(\boldsymbol{v})\sum_{\boldsymbol{x}=0}^{\boldsymbol{N}-1}\sum_{\boldsymbol{y}=0}^{\boldsymbol{N}-1}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})\cos\left[\frac{\pi(2\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right]\cos\left[\frac{\pi(2\boldsymbol{y}+1)\boldsymbol{v}}{2\boldsymbol{N}}\right] \tag{2.4}$$

para $\boldsymbol{u},\boldsymbol{v} = 0,1,2,\ldots,\boldsymbol{N}-1$ y $\alpha(\boldsymbol{u})$ y $\alpha(\boldsymbol{v})$ están definidas en (2.3). La transformada inversa está definida como

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{u}=0}^{\boldsymbol{N}-1}\sum_{\boldsymbol{v}=0}^{\boldsymbol{N}-1}\alpha(\boldsymbol{u})\alpha(\boldsymbol{v})\boldsymbol{C}(\boldsymbol{u},\boldsymbol{v})\cos\left[\frac{\pi(2\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right]\cos\left[\frac{\pi(2\boldsymbol{y}+1)\boldsymbol{v}}{2\boldsymbol{N}}\right] \tag{2.5}$$

para $\boldsymbol{x},\boldsymbol{y} = 0,1,2,\ldots,\boldsymbol{N}-1$. Las funciones base 2D pueden ser generadas al multiplicar las funciones base 1D orientadas horizontalmente (las cuales se muestran en la Fig.2.1) con el grupo de las orientadas verticalmente de las mismas funciones [288]. Las funciones base para $N = 8$ son mostradas en dicha figura. Nuevamente, podemos notar que las funciones base exhiben un incremento progresivo en frecuencia en ambos casos, es decir, en las direcciones vertical y horizontal. La función base en el margen superior izquierdo de la Fig.2.2 resulta de la multiplicación de la componente DC con su transpuesta. Por lo tanto, esta función asume un valor constante y esta referenciada como el coeficiente DC.

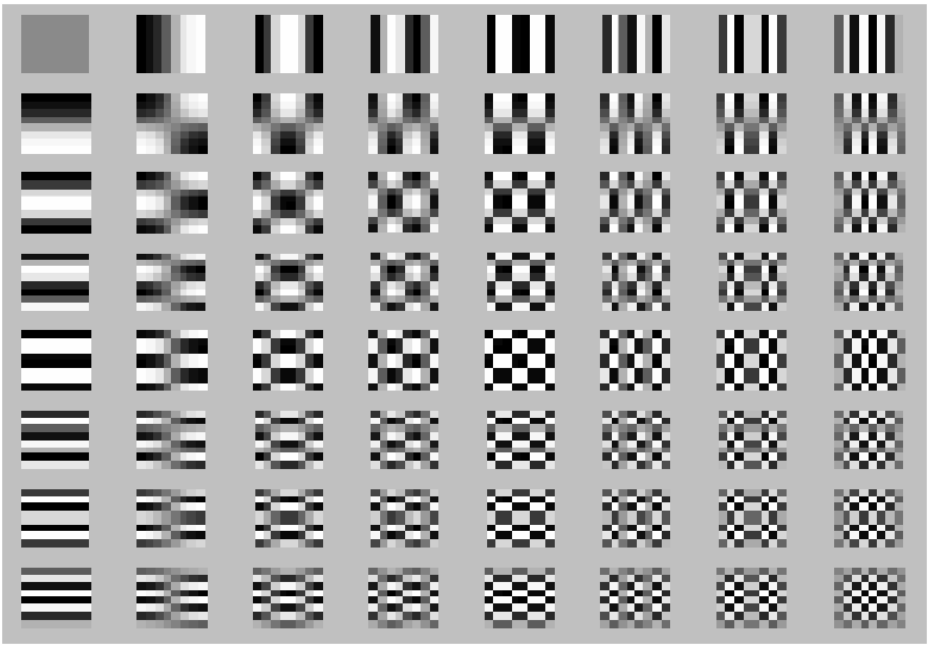


Figura 2.2: Funciones base TDC-2D ($N = 8$). Gris neutral representa cero, blanco representa amplitud positiva, y negro representa amplitud negativa.

2.2.1.3. Propiedades de la TDC

Las discusiones en las secciones precedentes han desarrollado los fundamentos matemáticos para la TDC. No obstante, el conocimiento intuitivo en su aplicación al de procesamiento de imágenes no ha sido presentado. En esta sección se describen (con ejemplos) algunas propiedades de la TDC que son de particular importancia para aplicaciones en procesamiento de imágenes.

2.2.1.3.1. Descorrelación

Como se discute en el Apéndice B (secciones B.2.1.1 y B.2.3), la ventaja principal de la transformación de la imagen es la de remover la redundancia entre pixeles vecinos. Esto lleva a coeficientes transformados decorrelacionados que se pueden codificar en forma independiente. Tomemos el ejemplo de la Fig.2.3 para describir las características de decorrelación de la TDC-2D. La autocorrelación normalizada de las imágenes antes y después de TDC se muestra en la Fig.2.4.

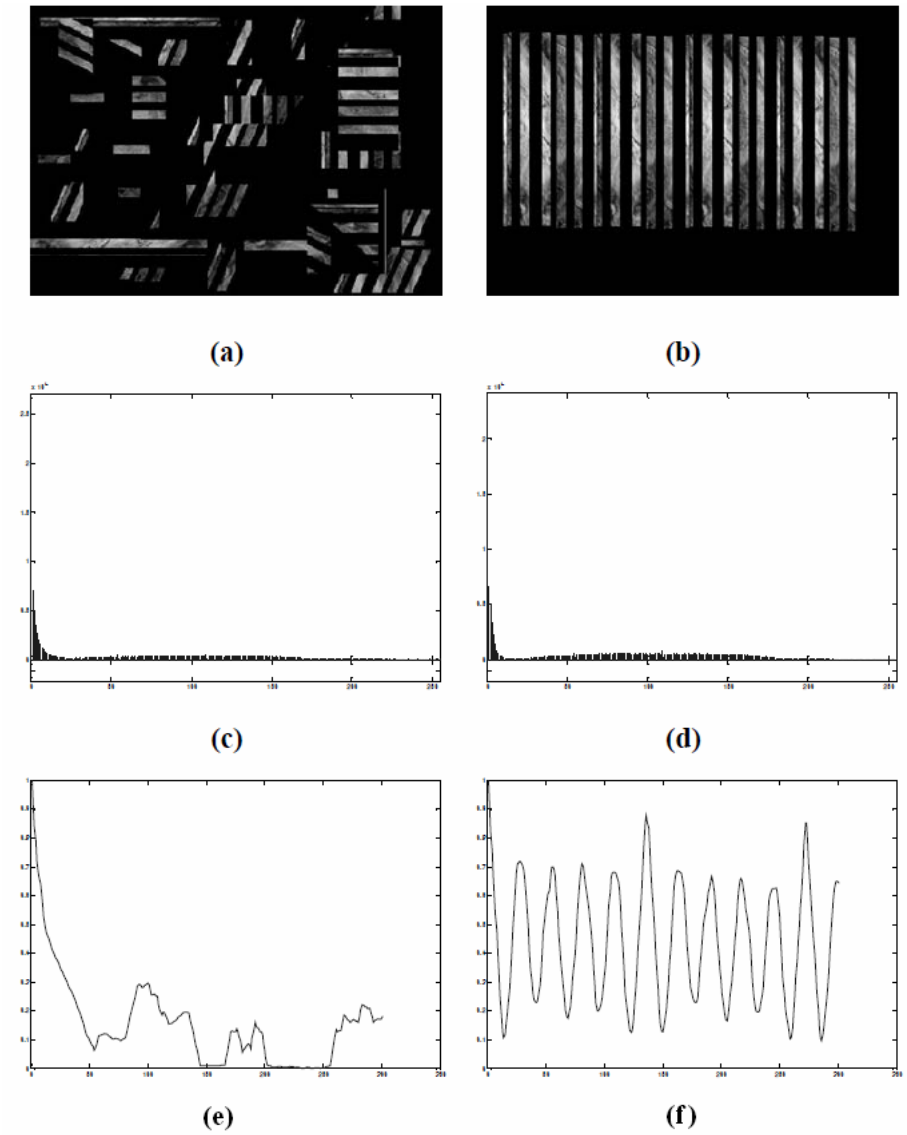


Figura 2.3: (a) Primera imagen, (b) segunda imagen, (c) histograma de la primera imagen, (d) histograma de la segunda imagen, (e) autocorrelación normalizada de una línea de la primera imagen, (f) autocorrelación normalizada de una línea de la segunda imagen.

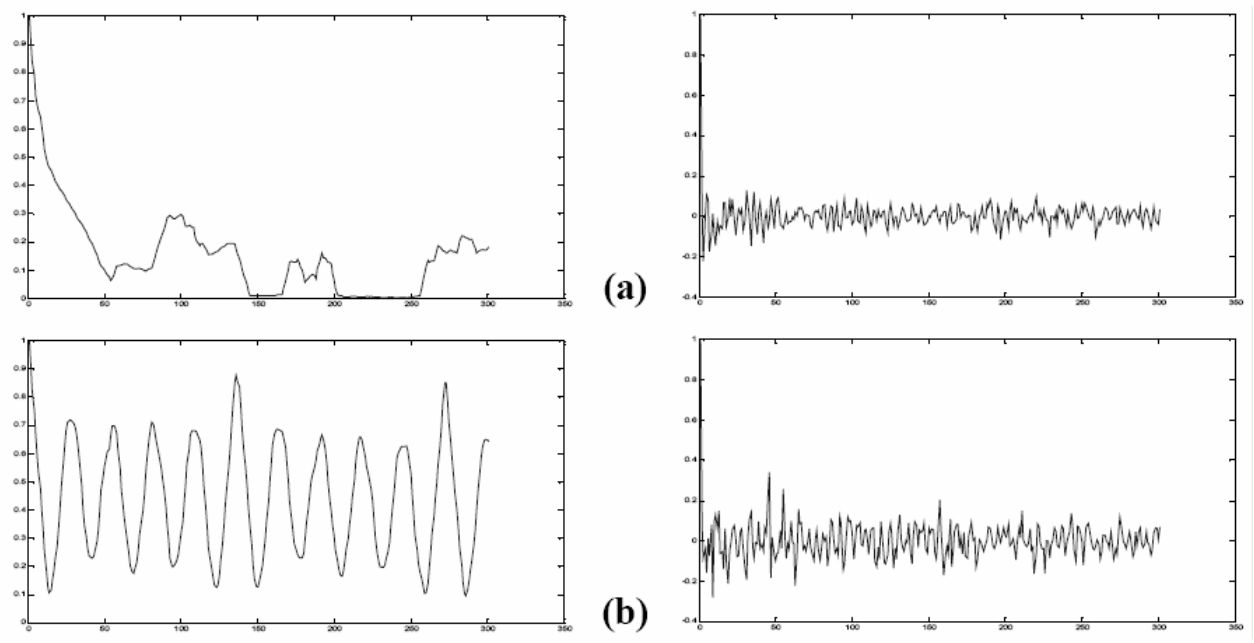


Figura 2.4: (a) Autocorrelación normalizada de una imagen decorrelacionada antes y después de TDC. (b) Autocorrelación normalizada de una imagen correlacionada antes y después de TDC.

Claramente, la amplitud de la autocorrelación después de la operación de aplicar la TDC es muy pequeña en todos los grupos de acción local. Por lo tanto, se puede inferir que la TDC exhibe excelentes propiedades de descorrelación.

2.2.1.3.2. Compactación de energía

La eficacia del esquema de transformación puede ser directamente medida por su habilidad para compactar los datos de entrada en la menor cantidad de coeficientes posible. Esto permite al cuantizador descartar coeficientes de relativamente poca amplitud sin introducir distorsión visual en la imagen reconstruida. La TDC exhibe excelente compactación de energía para imágenes altamente correlacionadas.

Vamos a examinar nuevamente las dos imágenes de la Fig.2.3(a) y (b). En adición a sus respectivas propiedades de correlación discutida en la sección precedente, la imagen descorrelacionada posee mayor cantidad de variaciones de intensidad fuerte que el caso de la imagen correlacionada. Por lo tanto, la primera tiene más contenido de alta frecuencia que la segunda. La Fig.2.5 muestra la TDC de ambas imágenes. Evidentemente, la imagen descorrelacionada tiene su energía bien distribuida, mientras que la energía de la imagen correlacionada está concentrada en la región de baja frecuencia (es decir, en la margen superior izquierdo).

Otros ejemplos de la propiedad de compactación de la energía por parte de la TDC con respecto a las mismas imágenes estándar están presentes en la Fig.2.6.

Una mirada más cercana a la Fig.2.6 muestra que la misma se compone de cuatro grandes clases de imagen. Las Fig.2.6(a) y (b) contienen grandes áreas de intensidades que varían lentamente. Estas imágenes pueden ser clasificadas como imágenes de baja frecuencia con detalles espaciales bajos. Una operación de TDC sobre estas imágenes provee muy buena compactación de energía en la región de baja frecuencia de la imagen transformada. La Fig.2.6(c) contiene un número de

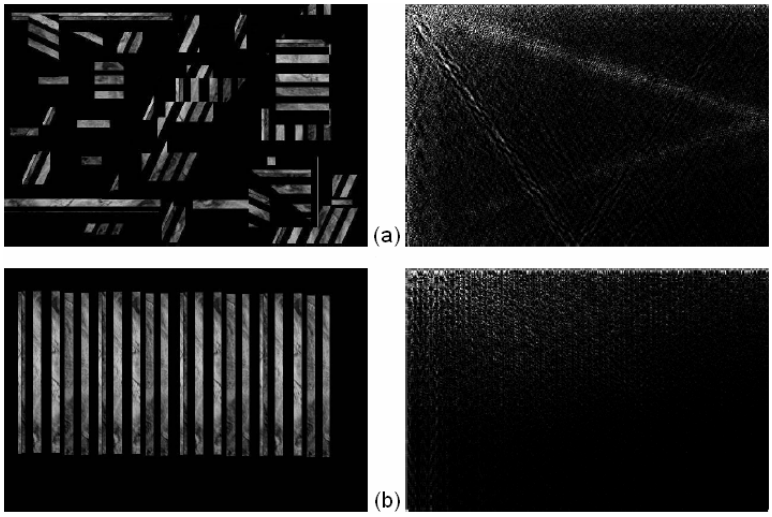


Figura 2.5: (a) Imagen descorrelacionada y su TDC; (b) Imagen correlacionada y su TDC.

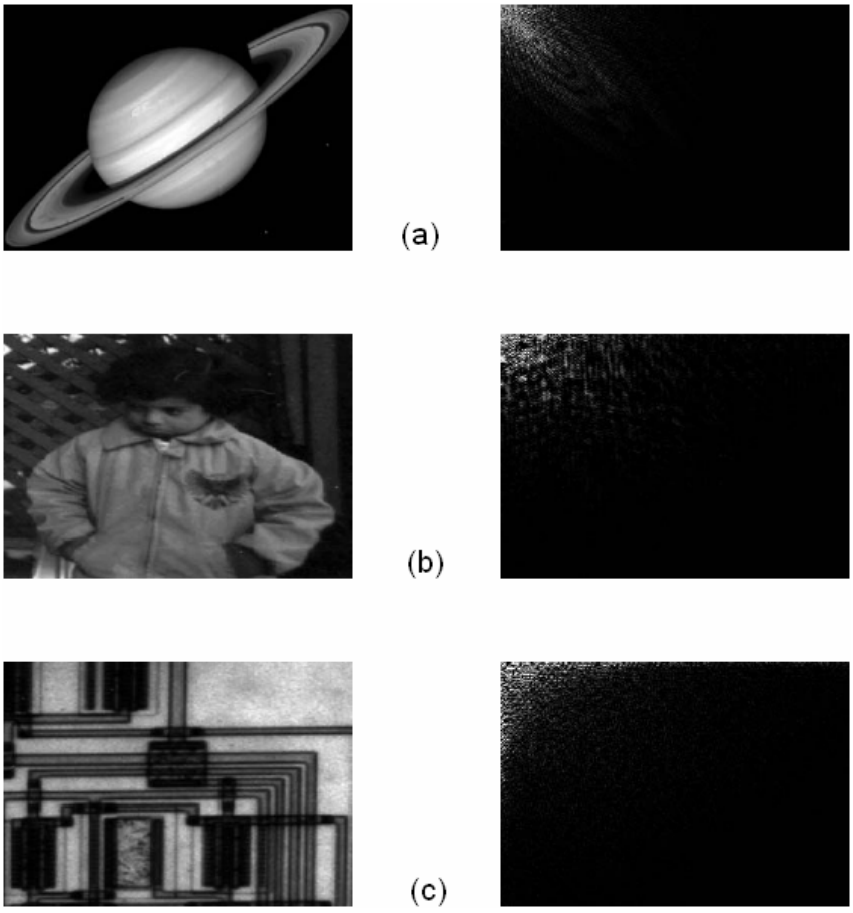


Figura 2.6: (a) Saturno y su TDC; (b) Niño y su TDC; (c) Circuito y su TDC.

bordes (es decir, variaciones de intensidad fuerte) y por lo tanto puede ser clasificada como una imagen de alta frecuencia con contenido espacial bajo. No obstante, los datos de la imagen exhiben alta correlación la cual es explotada por el algoritmo de TDC para proveer buena compactación de energía. La Fig.2.6(d) y (e) son imágenes con alto contenido en frecuencia y es-

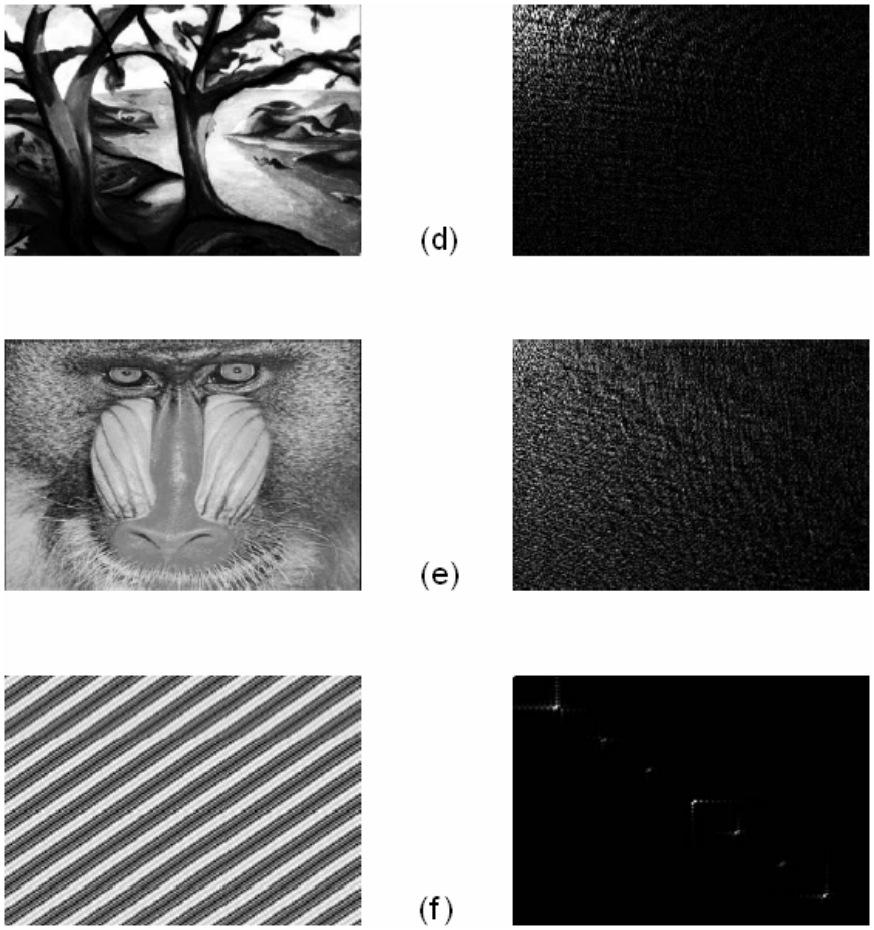


Figura 2.6: (d) Arboles y su TDC; (e) mandril y su TDC; (f) Onda senoide y su TDC.

pacial. Consecuentemente, los coeficientes de la transformada son repartidos en bajas y altas frecuencias. La Fig.2.6(e) muestra la periodicidad, entonces la TDC contiene impulsos con amplitudes proporcionales al peso de una frecuencia particular en la forma de onda particular. Los otros armónicos (relativamente insignificantes) de la onda senoidal pueden también ser observados por una observación más aproximada de su imagen TDC.

Por lo tanto, de la discusión precedente se puede inferir que la TDC rinde excelente compactación de energía para imágenes correlacionadas. Estudios han mostrado que la performance de compactación de energía de la TDC se aproxima a la optimalidad para imágenes que se aproximan a la correlación, es decir, la TDC provee (casi) una descorrelación óptima para tales imágenes [290].

2.2.1.3.3. Separabilidad

La ecuación de la TDC (2.4) puede ser expresada como,

$$\boldsymbol{C}(\boldsymbol{u},\boldsymbol{v}) = \alpha(\boldsymbol{u})\alpha(\boldsymbol{v})\sum_{\boldsymbol{x}=0}^{\boldsymbol{N}-1}\cos\left[\frac{\pi(2\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right]\sum_{\boldsymbol{y}=0}^{\boldsymbol{N}-1}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})\cos\left[\frac{\pi(2\boldsymbol{y}+1)\boldsymbol{v}}{2\boldsymbol{N}}\right] \tag{2.6}$$

para $\boldsymbol{u},\boldsymbol{v} = 0,1,2,\ldots,\boldsymbol{N}-1$.

Esta propiedad, conocida como *separabilidad*, tiene la ventaja principal que $\boldsymbol{C}(\boldsymbol{u},\boldsymbol{v})$ puede ser calculada en dos pasos, compuestos por sucesivas operaciones 1D sobre filas y columnas de la imagen. Esta idea es ilustrada gráficamente en la Fig.2.7. Los argumentos presentados pueden ser aplicados idénticamente para el cálculo de la TDC inversa (2.5).

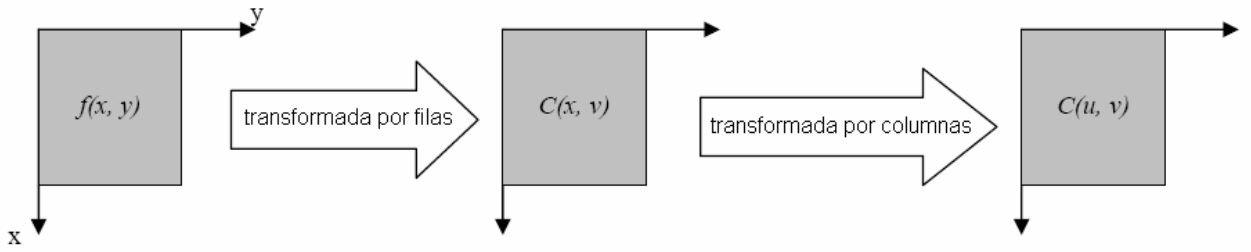


Figura 2.7: Cálculo de la TDC-2D usando la propiedad de separabilidad.

2.2.1.3.4. Simetría

Otra mirada a las operaciones sobre las filas y las columnas de la Ecuación (2.6) revela que dichas operaciones son funcionalmente idénticas. Tales transformaciones reciben el nombre de *transformaciones simétricas*. Una transformación simétrica y separable puede ser expresada en la forma [291]

$$\boldsymbol{T} = \boldsymbol{A}\,\boldsymbol{f}\,\boldsymbol{A} \tag{2.7}$$

donde $\boldsymbol{A}$ es una matriz de transformación simétrica de $N \times N$ con entradas $\boldsymbol{a}(\boldsymbol{i},\boldsymbol{j})$ dadas por

$$\boldsymbol{a}(\boldsymbol{i},\boldsymbol{j}) = \alpha(\boldsymbol{j})\sum_{\boldsymbol{j}=0}^{N-1}\cos\left[\frac{\pi(2\boldsymbol{j}+1)\boldsymbol{i}}{2\boldsymbol{N}}\right] \tag{2.8}$$

y f es la matriz imagen de $N \times N$.

Esta es una propiedad extremadamente útil la cual implica que la matriz de transformación () pueda ser precalculada *off-line* y entonces aplicada a la imagen proporcionando una mejora en los órdenes de magnitud en la eficiencia de cálculo.

2.2.1.3.5. Ortogonalidad

Con objeto de extender las ideas presentadas en la sección precedente, establecemos a la inversa de la transformación de (2.7) como

$$\boldsymbol{f} = \boldsymbol{A}^{-1}\,\boldsymbol{T}\,\boldsymbol{A}^{-1} \tag{2.9}$$

Como se discutió previamente, las funciones base TDC son ortonormales (ver Sección 2.2.1.1). Entonces, la matriz de transformación inversa de $\boldsymbol{A}$ es igual a su transpuesta, es decir, $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\boldsymbol{T}}$. Por lo tanto, y en adición a sus características de descorrelación, esta propiedad rinde alguna reducción en la complejidad de pre-cálculo.

2.2.1.4. Una TDC más rápida

Las propiedades discutidas en las últimas tres sub-secciones han establecido los fundamentos para un algoritmo de cálculo más rápido para la TDC. Los algoritmos de TDC rápida de arquitectura generalizada y específica han sido estudiadas ampliamente [292-306]. No obstante, y de acuerdo con las citas mencionadas, solo discutiremos un algoritmo que utiliza la Transformada Rápida de Fourier (TRF) para calcular la TDC y su inversa [36]. Este algoritmo es presentado para la TDC-1D, no obstante, puede ser extendido al caso 2D. En este punto, es importante notar que la TDC no es la parte real de la Transformada Discreta de Fourier (TDF). Esto puede ser fácilmente verificado al inspeccionar las matrices de transformación de la TDC y la TDF.

La secuencia 1D $\boldsymbol{f}(\boldsymbol{x})$ en (2.1) puede ser expresada como la suma de una secuencia par y otra impar

$$\bar{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{x}) + \boldsymbol{f}_{\boldsymbol{i}}(\boldsymbol{x}) \tag{2.10}$$

donde $\boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{x}) = \boldsymbol{f}(2\boldsymbol{x}) = \bar{\boldsymbol{f}}(\boldsymbol{x})$, $\boldsymbol{f}_{\boldsymbol{i}}(\boldsymbol{x}) = \boldsymbol{f}(2\boldsymbol{x}+1) = \bar{\boldsymbol{f}}(\boldsymbol{N}-\boldsymbol{x}-1)$, y $0 \le \boldsymbol{x} \le \left(\frac{\boldsymbol{N}}{2}\right) - 1$.

El término sumatoria en (2.1) puede ser partido para obtener

$$\begin{aligned}
\boldsymbol{C}(\boldsymbol{u}) &= \alpha(\boldsymbol{u})\left\{\sum_{\boldsymbol{x}=0}^{\left(\frac{\boldsymbol{N}}{2}\right)-1} \boldsymbol{f}(2\boldsymbol{x})\cos\left[\frac{\pi(4\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right] + \sum_{\boldsymbol{x}=0}^{\left(\frac{\boldsymbol{N}}{2}\right)-1} \boldsymbol{f}(2\boldsymbol{x}+1)\cos\left[\frac{\pi(4\boldsymbol{x}+3)\boldsymbol{u}}{2\boldsymbol{N}}\right]\right\} \\
&= \alpha(\boldsymbol{u})\left\{\sum_{\boldsymbol{x}=0}^{\left(\frac{\boldsymbol{N}}{2}\right)-1} \boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{x})\cos\left[\frac{\pi(4\boldsymbol{x}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right] + \sum_{\boldsymbol{x}=0}^{\left(\frac{\boldsymbol{N}}{2}\right)-1} \boldsymbol{f}_{\boldsymbol{i}}(\boldsymbol{x})\cos\left[\frac{\pi(4\boldsymbol{x}+3)\boldsymbol{u}}{2\boldsymbol{N}}\right]\right\} \\
&= \alpha(\boldsymbol{u})\sum_{\boldsymbol{x}=0}^{\boldsymbol{N}-1} \bar{\boldsymbol{f}}(\boldsymbol{x})\cos\left[\frac{\pi(4\boldsymbol{n}+1)\boldsymbol{u}}{2\boldsymbol{N}}\right] \\
&= \mathrm{Re}\left[\alpha(\boldsymbol{u})\boldsymbol{e}^{-\boldsymbol{j}\pi\boldsymbol{u}/2\boldsymbol{N}}\sum_{\boldsymbol{x}=0}^{\boldsymbol{N}-1} \bar{\boldsymbol{f}}(\boldsymbol{x})\boldsymbol{e}^{-\boldsymbol{j}2\pi\boldsymbol{u}\boldsymbol{x}/\boldsymbol{N}}\right] = \mathrm{Re}\left[\alpha(\boldsymbol{u})\boldsymbol{W}_{2\boldsymbol{N}}^{\boldsymbol{u}/2}\boldsymbol{TDF}\left\{\bar{\boldsymbol{f}}(\boldsymbol{x})\right\}_{\boldsymbol{N}}\right]
\end{aligned} \tag{2.11}$$

Para la transformación inversa, podemos calcular $\boldsymbol{f}(2\boldsymbol{x}) = \bar{\boldsymbol{f}}(2\boldsymbol{x})$ y los puntos impares pueden ser calculados al notar que $\boldsymbol{f}(2\boldsymbol{x}+1) = \bar{\boldsymbol{f}}[2(\boldsymbol{N}-1-\boldsymbol{x})]$ para $0 \le \boldsymbol{x} \le \left(\frac{\boldsymbol{N}}{2}\right) - 1$.

2.2.1.5. TDC versus TDF/TDKL

En este punto es importante mencionar la superioridad de la TDC sobre otras transformadas de la imagen. Más específicamente, al comparar la TDC con dos transformaciones lineales: 1) La TDKL [320] y 2) la TDF [312-319].

Como ya se vio en el capítulo anterior la TDKL es una transformada lineal donde las funciones base son tomadas de las propiedades estadísticas de los datos de la imagen, por lo que puede entonces ser adaptativa. Es óptima en el sentido de la compactación de la energía, es decir, concentra tanta energía como sea posible en el menor número de coeficientes posibles. No

obstante, el kernel de la transformación de la TDKL no es generalmente fijo, y entonces debe ser realizada la multiplicación de la matriz completa. En otras palabras, la TDKL es dependiente de los datos, y, por lo tanto, sin un pre-cálculo de transformación (como el caso de la TRF), la derivación de las bases respectivas para cada sub-bloque de la imagen requiere recursos de cálculo desmedidos. Aunque, algunos algoritmos de TDKL rápidos han sido sugeridos [308, 309], no obstante, la complejidad general de la TDKL es significativamente más alta que la respectiva de los algoritmos de la TDC y TDF. De acuerdo con las citas mencionadas, la familiaridad con la TDF ha sido asumida a través de dichos documentos. El kernel de la transformación de la TDF es lineal, separable y simétrico. Por lo tanto, como la TDC, posee imágenes de base fija y son posibles implementaciones rápidas. También evidencia características de buena decorrelación y compactación de la energía. No obstante, la TDF es una transformación compleja y por lo tanto estipula que deben ser codificadas ambas magnitudes de la imagen, es decir, módulo y fase. En relación a esto, varios estudios han mostrado que la TDC provee mejor compactación de energía que la TDF para imágenes más naturales. Además, la periodicidad implícita de la TDF da lugar a la frontera discontinuidades que se traducen en un importante contenido de altas frecuencias. Luego de la cuantización, el conocido *fenómeno de Gibbs* causa que los puntos sobre la frontera tomen valores erróneos [310].

2.2.1.6. Evaluación de la performance de la TDC: Una aproximación según la Teoría de la Información

En esta sección investigamos la performance de la TDC usando métricas de Teoría de la Información.

2.2.1.6.1. Reducción de la Entropía

Las características de decorrelación de la TDC deberían protagonizar un descenso en la entropía (o auto-información) de una imagen, la cual representa el límite teórico en la compresión de datos. Esto, a su vez, disminuirá el número de bits requeridos para representar la imagen. La Entropía (o medida de la información contenida en la imagen) se define como

$$\boldsymbol{H}(\boldsymbol{x}) = -\sum_{\boldsymbol{i}=1}^{\boldsymbol{N}} \boldsymbol{p}(\boldsymbol{x}_{\boldsymbol{i}}) \log(\boldsymbol{p}(\boldsymbol{x}_{\boldsymbol{i}})) \tag{2.12}$$

donde $\boldsymbol{p}(\boldsymbol{x}_{\boldsymbol{i}})$ es el valor de la función de masa de probabilidad (FMP) en $\boldsymbol{X} = \boldsymbol{x}_{\boldsymbol{i}}$, función de probabilidad o distribución de probabilidad de la variable aleatoria $\boldsymbol{x}$. A lo largo de la siguiente discusión, la FMP se refiere al histograma normalizado de la imagen, y en relación a los *bit-por-pixel* considerados. La primera columna de la Tabla.2.I da la entropía de las imágenes listadas en la Fig.2.6. El resto de las columnas dan las entropías de las respectivas imágenes TDC con un cierto número de coeficientes retenidos.

Tabla 2.I: Entropía de las imágenes y su TDC.

Imagen	Entropía de la imagen original	Entropía TDC (75 %)	Entropía TDC (50 %)	Entropía TDC (25 %)
Saturno	4.1140	1.9144	0.7504	0.2181
Niño	5.7599	1.7073	0.7348	0.2139
Circuito	6.9439	1.4961	0.6906	0.2120
Arboles	5.7006	1.1445	0.6181	0.2081
Mandril	7.3461	0.9944	0.5859	0.2058
Seno	3.1250	1.2096	0.6674	0.2125

Más específicamente, TDC (*M* %) significa que después de la operación de TDC solo el primer porcentaje *M* del total de coeficientes es retenido. Consideremos por ejemplo una imagen de 325 x 288, la TDC (25 %) retiene solamente el 25 % (352 x 288 x .25 = 25560) del total de (102240) coeficientes.

Debemos enfatizar nuevamente que esta disminución de la entropía representa una reducción en el número promedio de bits requeridos para una imagen particular. La segunda columna de la Tabla.2.I muestra claramente que la operación de la TDC reduce la entropía de todas las imágenes. La reducción en la entropía se hace más acentuada a medida que se disminuye el número de coeficientes a preservar (ver Tabla.2.I columnas 3, 4, y 5). Este resultado tan interesante implica que descartando algunos detalles de alta frecuencia se pueden producir importantes avances en la reducción de la entropía. Cabe señalar que la naturaleza inherentemente sin pérdida del codificador por transformada establece que la operación de descarte de coeficientes puede ser implementada en el cuantizador.

Una mirada más en detalle de la Tabla.2.I revela que la reducción en la entropía para la imagen del mandril es muy drástica. La entropía de la imagen original comprueba que tiene muchas componentes de alta frecuencia, así como detalles espaciales. Por lo tanto, codificarlo en el dominio espacial es muy ineficiente dado que los niveles de grises son algo que se distribuyen de manera uniforme a través de la imagen. No obstante, la TDC descorrelaciona los datos de la imagen por lo que se estira el histograma.

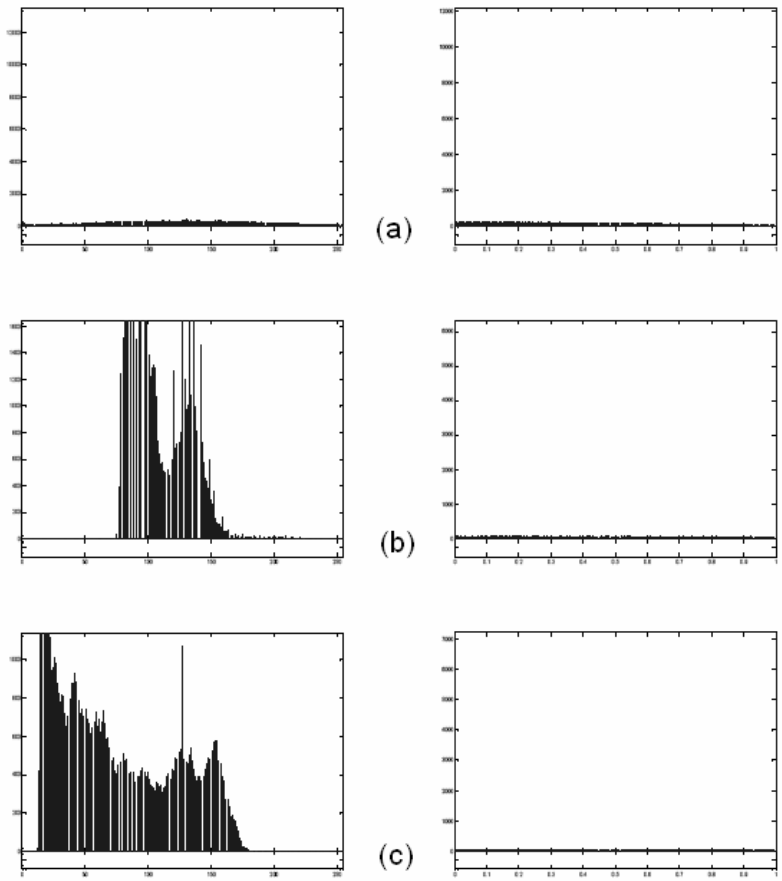


Figura 2.8: (a) Histograma de Saturno y su TDC; (b) Histograma del Niño y su TDC; y (c) Histograma del Circuito y su TDC.

Esta discusión se aplica también a las otras imágenes de alta frecuencia, concretamente Circuito y Arboles. La disminución en la entropía se explica fácilmente al observar los histogramas de la imagen original y de la imagen TDC codificada de la Fig.2.8.

Como una consecuencia de la propiedad de descorrelación, los datos de la imagen original es transformada en una forma que el histograma es estirado y la amplitud de la mayoría de los resultados transformados es muy pequeña.

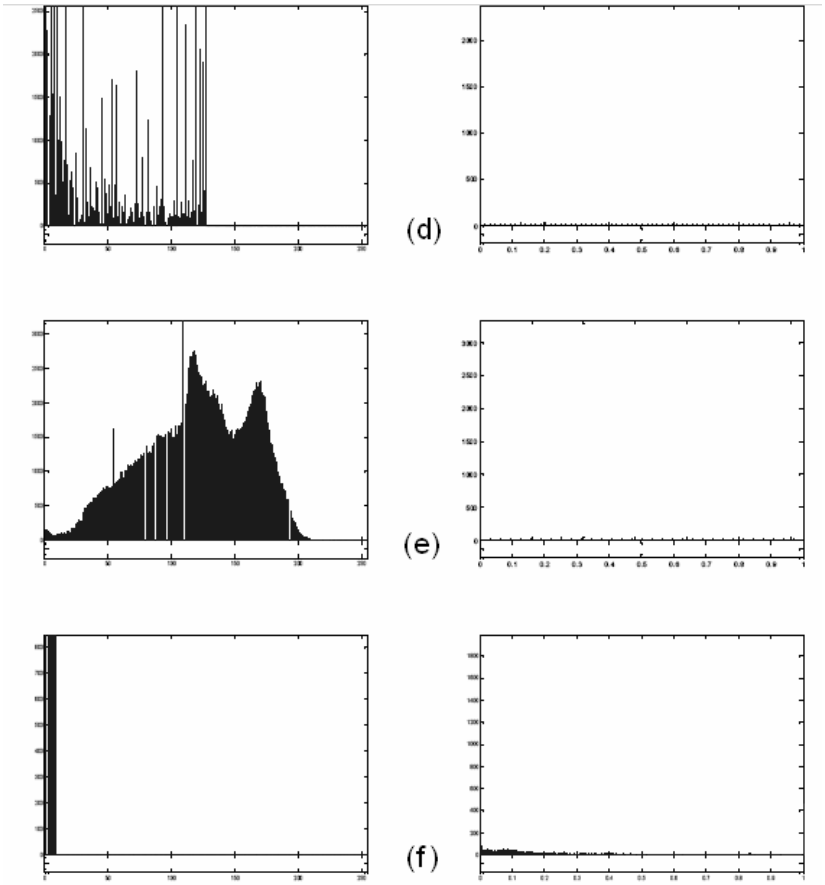


Figura 2.8: (d) Histograma de los Arboles y su TDC; (e) Histograma del mandril y su TDC; y (c) Histograma de la Senoide y su TDC.

La discusión precedente ignora una cuestión fundamental: ¿Cuánta distorsión visual en la imagen es introducida por el procedimiento de cuantización (en crudo) descrito anteriormente? Las Figuras 2.9 a 2.14 muestran las imágenes reconstruidas al realizar la operación de TDC inversa sobre los coeficientes cuantizados. Claramente, TDC (25 %) introduce el *efecto blur* [321] en todas las imágenes dado que solo ¼ del total del número de coeficientes son empleados en la reconstrucción. No obstante, TDC (50 %) provee casi idéntica reconstrucción en todas las imágenes excepto en la Figuras 2.12 (Arboles) y 2.14 (Senoide). Los resultados de la Fig.2.12 (Arboles) pueden ser explicados por el hecho de que las imágenes tienen un montón de detalles de alta-frecuencia descorrelacionados. Por lo tanto, descartando los coeficientes TDC de alta frecuencia resulta en una degradación de la calidad. La Fig.2.14 (Senoide) es fácilmente explicable por la por la examinación de sus TDC dados en la Fig.2.6(f). La remoción de los coeficientes de alta frecuencia resulta en la remoción de ciertas frecuencias que fueron

originalmente presentadas en la senoide. Luego de la pérdida de ciertas frecuencias no es posible alcanzar una reconstrucción perfecta.

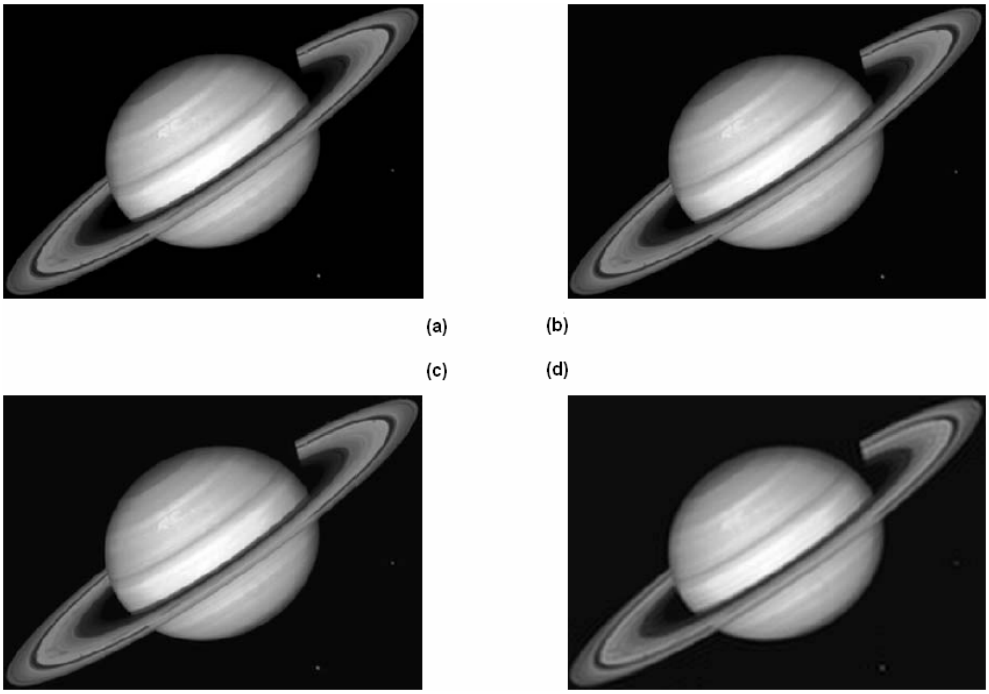


Figura 2.9: TDC inversa de Saturno; (a) TDC (100 %); (b) TDC (75 %); (c) TDC (50 %); (d) TDC (25 %).

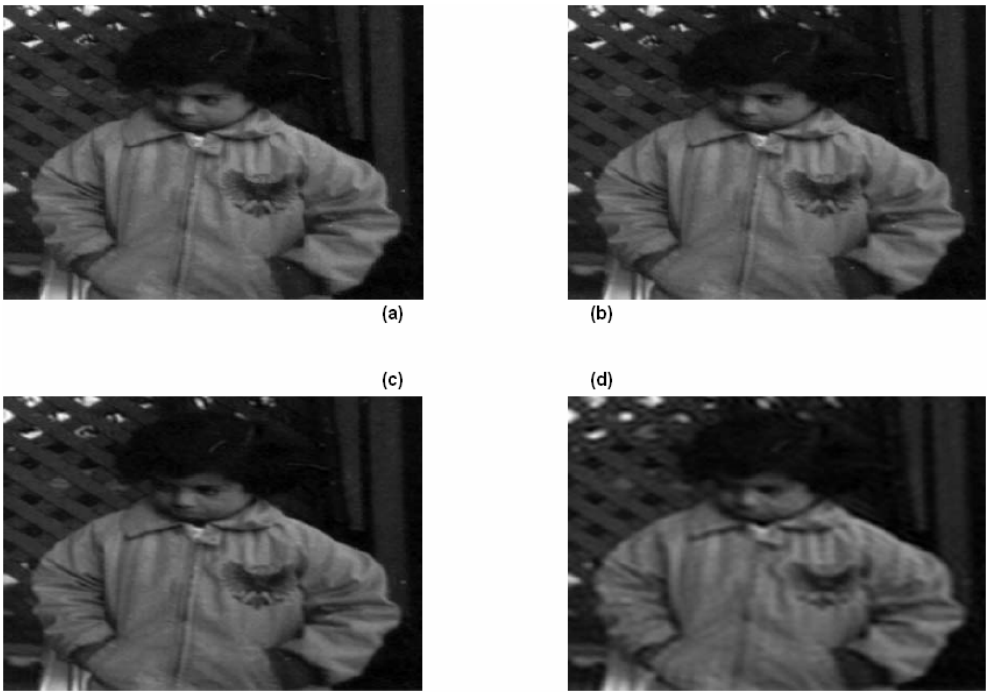


Figura 2.10: TDC inversa del Niño; (a) TDC (100 %); (b) TDC (75 %); (c) TDC (50 %); (d) TDC (25 %).

No obstante, TDC (75 %) provee excelente reconstrucción para todas las imágenes excepto la senoide. Esto constituye un resultado muy interesante dado que sugiere que basado sobre unos requerimientos de ancho de banda (heterogéneo) del receptor, los coeficientes TDC pueden ser

descartados por el cuantizador mientras que se disponga de una prestación de calidad aceptable.

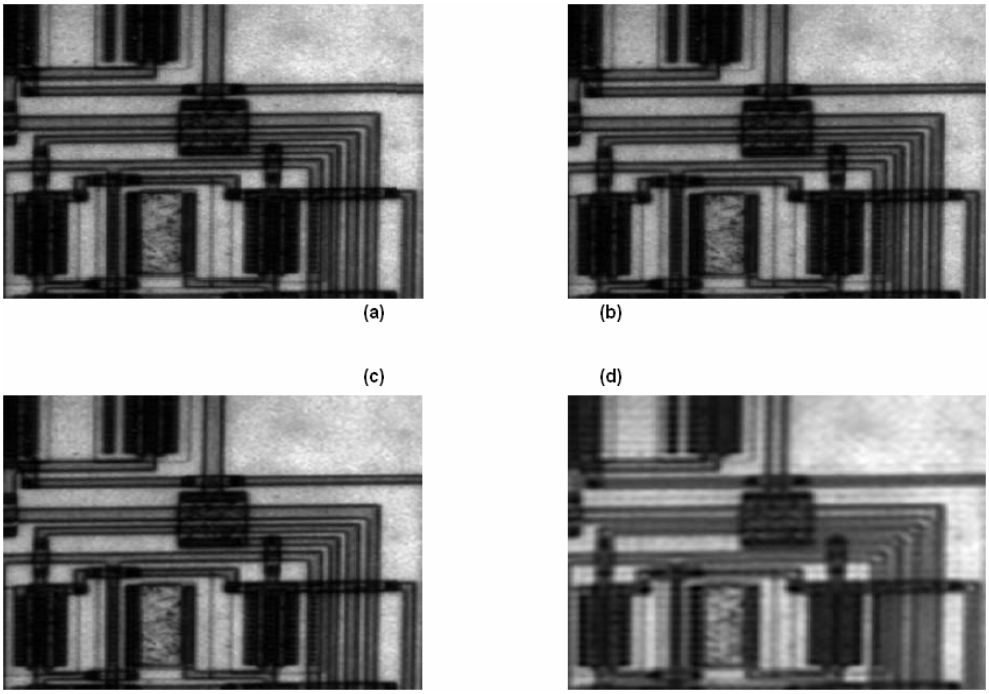


Figura 2.11: TDC inversa l Circuito; (a) TDC (100 %); (b) TDC (75 %); (c) TDC (50 %); (d) TDC (25 %).

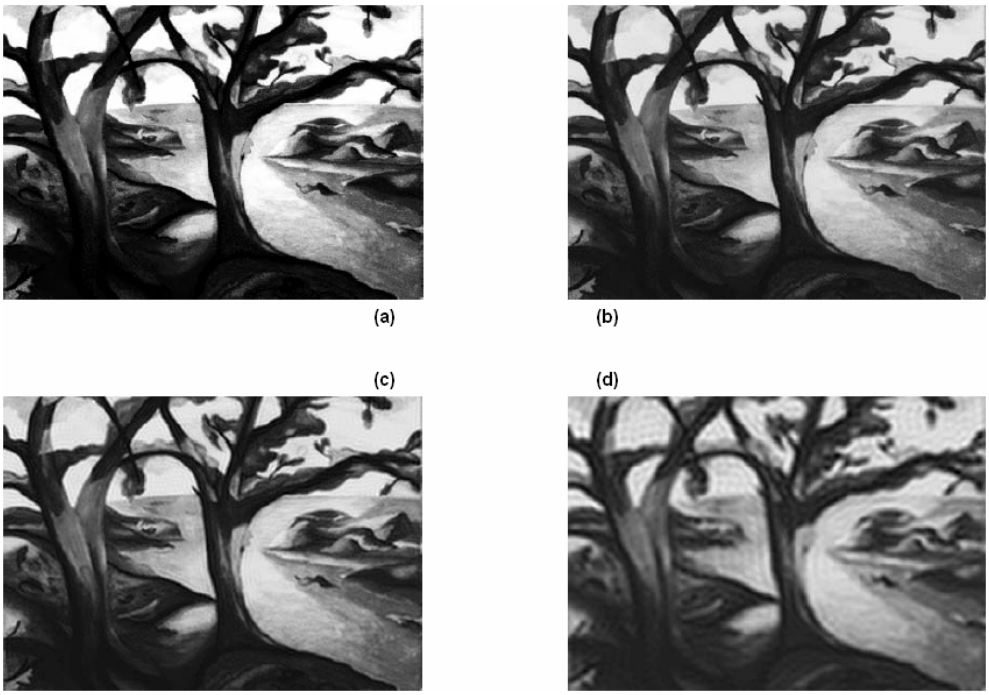


Figura 2.12: TDC inversa de los Arboles; (a) TDC (100 %); (b) TDC (75 %); (c) TDC (50 %); (d) TDC (25 %).

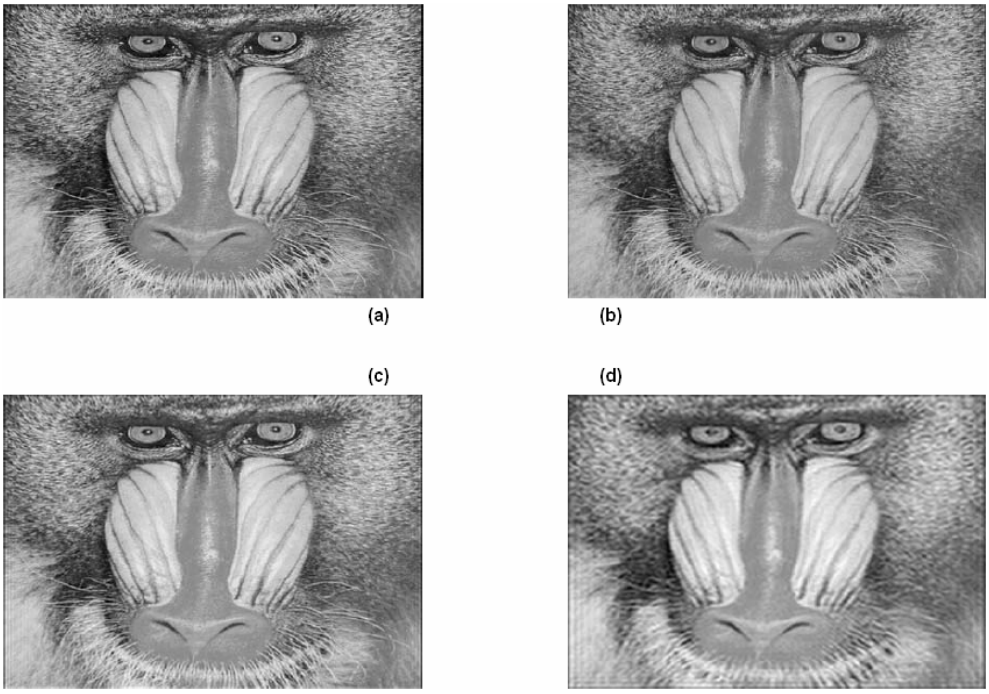


Figura 2.13: TDC inversa del mandril; (a) TDC (100 %); (b) TDC (75 %); (c) TDC (50 %); (d) TDC (25 %).

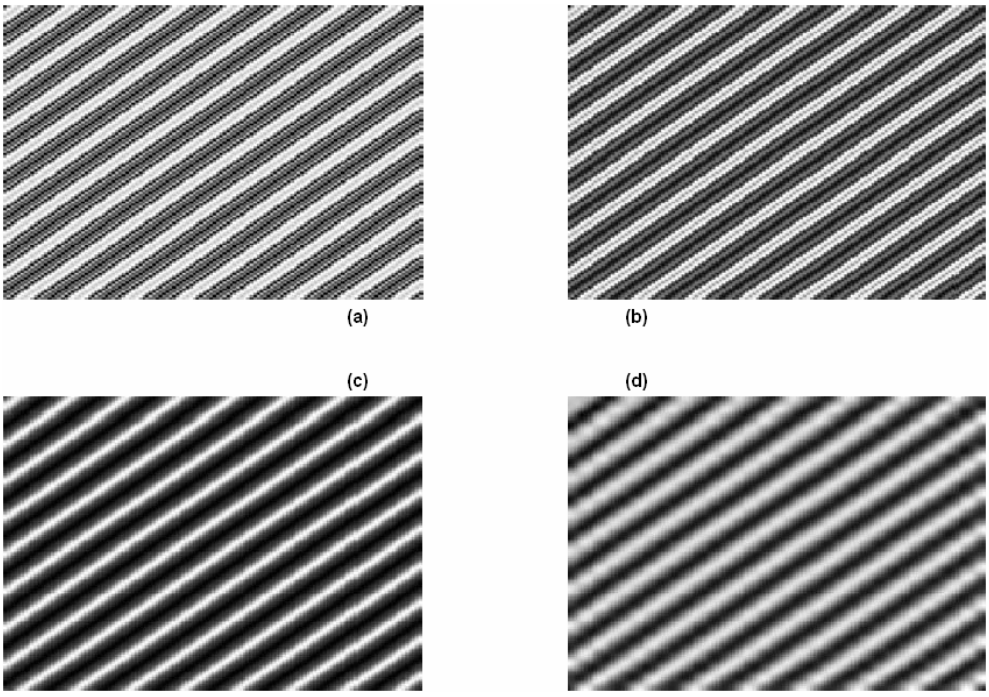


Figura 2.14: TDC inversa de la Senoide; (a) TDC (100 %); (b) TDC (75 %); (c) TDC (50 %); (d) TDC (25 %).

2.2.1.6.2. Información Mutua

Una transmisión de vídeo se compone de una secuencia de imágenes (referida a los cuadros) que son transmitidos en un determinado orden. Generalmente, el cambio de información entre cuadros consecutivos del video es bajo [322-327]. Por lo tanto, los valores de los pixeles en un cuadro pueden ser usados para predecir los pixeles adyacentes del próximo cuadro. Este

fenómeno es conocido como *correlación temporal*. Tal redundancia temporal puede ser removida para proveer una mejor compresión. No obstante, debería ser obvio que si los dos cuadros respectivos representan un cambio de la escena entonces la información mutua se reduce. Esto se muestra claramente en la Fig.2.15. Los primeros dos cuadros muestran una alta correlación temporal. No obstante, el cuadro-2 y el cuadro-3 representan un cambio en la escena, por lo tanto, la información que se solapa entre estos cuadros es relativamente menor.

Figura 2.15: Tres cuadros consecutivos de una secuencia de video.

Usamos la *Medida de la Información Mutua* para cuantificar la correlación temporal entre cuadros de una secuencia de video. La información mutua de dos cuadros de video puede, por lo tanto, ser definida como la cantidad de redundancia (correlación) entre dos cuadros. La Información Mutua esta dada por

$$\boldsymbol{I}(\boldsymbol{X};\boldsymbol{Y}) = \sum_{x}\sum_{y} \boldsymbol{p}(\boldsymbol{x},\boldsymbol{y}) \log\left(\frac{\boldsymbol{p}(\boldsymbol{x},\boldsymbol{y})}{\boldsymbol{p}(\boldsymbol{x})\boldsymbol{p}(\boldsymbol{y})}\right) \tag{2.13}$$

donde $\boldsymbol{p}(\boldsymbol{x},\boldsymbol{y})$ es la Función de Densidad de Probabilidad (FDP) conjunta de la variables aleatorias *X* e *Y* (es decir, cuadro-*X* y cuadro-*Y* en nuestro caso), $\boldsymbol{p}(\boldsymbol{x})$ y $\boldsymbol{p}(\boldsymbol{y})$ representan las FDP marginales de *X* e *Y* respectivamente.

Consideremos la información mutua de los cuadros de video mostrados en la Fig.2.15. Definamos a $\boldsymbol{p}(\boldsymbol{x},\boldsymbol{y})$ acoplando las variables de los pixeles adyacentes en cualquiera de los dos cuadros consecutivos. El rango de valores permitido para $\boldsymbol{p}(\boldsymbol{x},\boldsymbol{y})$ es (0, 0), (0, 1), (0, 2), …, (255, 255), Por lo tanto, el número total de resultados es $(255)^2$. De acuerdo a esto, $\boldsymbol{p}(\boldsymbol{x})$ y $\boldsymbol{p}(\boldsymbol{y})$ serán los histogramas normalizados de los cuadros *X* e *Y* respectivamente. La información mutua es como sigue:

Información Mutua entre cuadro-1 y cuadro-2 = 1.9336
Información Mutua entre cuadro-2 y cuadro-3 = 0.2785

Dado que, existe una alta correlación temporal entre los dos primeros cuadros, por lo tanto, su información mutua es bastante alta. No obstante, los cuadros 2 y 3 representan un cambio en la escena de video, por lo tanto, la información mutua de las dos imágenes es relativamente menor.

Esto resultados ponen en evidencia un fenómeno muy interesante, y que consiste en que los cuadros en la secuencia de video exhiben una alta correlación temporal. Tal redundancia temporal puede ser aprovechada por el codificador de la fuente para mejorar la eficiencia de codificación. En otras

palabras, los pixeles adyacentes de los cuadros de video pasado y futuro pueden ser usados para predecir los valores de los pixeles de un cuadro en particular. Esta correlación temporal es empleada por la mayoría de los sistemas de procesamiento de video contemporáneos.

2.2.2. La Transformada Discreta de Onditas (TDO): Haar

Las onditas son una herramienta matemática para la descomposición jerárquica de funciones. Ellas permiten que una función sea descripta en función de una componente general de la misma forma y de menor resolución, más detalles que poseen un amplio rango a reducir. Independientemente de si la función de interés es una imagen, una curva, o una superficie, las onditas ofrecen una técnica elegante para representar los niveles de detalle presente. Esta sección se destina a presentar los fundamentos matemáticos necesarios para el estudio y el uso de ellos. Concretamente, hablamos del caso simple de las onditas de Haar en una y dos dimensiones, y mostramos cómo éstas pueden ser utilizadas para la compresión de imágenes.

Aunque las onditas tienen sus raíces en la teoría de aproximación [175] y el procesamiento de señales [183], recientemente se han aplicado a muchos problemas en gráficos por computadora. Estas aplicaciones incluyen gráficos de edición de imágenes [171], compresión de imágenes [176], y consulta de imágenes [180]; control automático del nivel de detalle para la edición y rendering de curvas y superficies [177, 178, 182]; reconstrucción de superficies a partir de los contornos [184], y métodos rápidos de simulación para la solución de problemas en animación [181] e iluminación en general [173, 175, 179, 185]. Para una discusión acerca de onditas que va más allá del alcance de este trabajo, los lectores pueden recurrir a un trabajo esclarecedor [186].

Hemos establecido aquí en primer lugar la presentación de la forma más sencilla de onditas, que es la base de Haar. Cubrimos la transformada ondita uni-dimensional y las funciones base, y mostramos cómo estas herramientas pueden utilizarse para comprimir la representación de una función constante por partes. Entonces discutimos la generalización a dos dimensiones de la base de Haar, y demostramos cómo se aplican estas onditas a la compresión de una imagen. Dado que el Álgebra Lineal es fundamental para las matemáticas de onditas, en el Apéndice E se examinarán brevemente conceptos importantes.

2.2.2.1. Onditas en una dimensión

La base de Haar es la más simple de todas las bases de onditas. En primer lugar, discutiremos cómo una función uni-dimensional puede ser descompuesta utilizando onditas de Haar y, a continuación, describiremos las funciones de base actuales. Por último, mostraremos cómo utilizar la descomposición de las onditas de Haar como una técnica sencilla para la compresión de una función uni-dimensional [169].

2.2.2.1.1. Transformada Onditas de Haar unidimensional

Para tener una idea de cómo trabajan las onditas, empecemos con un ejemplo sencillo. Supongamos que tenemos una "imagen" uni-dimensional con una resolución de cuatro píxeles,

[9 7 3 5]

Podemos representar esta imagen en la base de *Haar* al calcular la TDO. Para hacer esto, primero promediamos los pixeles de a pares, para obtener una nueva imagen de más baja resolución

[8 4]

Evidentemente, cierta información se ha perdido en este proceso de promediación. Para recuperar el original de cuatro pixel del promedio de dos valores, tenemos que guardar algunos coeficientes de detalle, los cuales contienen la información que falta. En nuestro ejemplo, vamos a elegir 1 para el primer coeficiente de detalle, ya que la media calculada es de 1 menos 9 y 1 más 7. Este único número nos permite recuperar los dos primeros pixeles de nuestra imagen original de cuatro pixeles. Del mismo modo, el segundo detalle es el coeficiente -1, desde el 4+(-1) = 3 y 4-(-1) = 5. Por lo tanto, hemos descompuesto la imagen original en una de más baja resolución (dos píxeles) y un par de coeficientes de detalle. Repitiendo este proceso recursivamente sobre los promedios da una descomposición completa:

Resolución	Promedios	Coeficientes de detalle
4	[9 7 3 5]	
2	[8 4]	[1 -1]
1	[6]	[2]

Finalmente, definimos la *transformada ondita* (también llamada descomposición ondita) de la imagen original de 4-pixeles como los coeficientes de promedio, seguidos de los coeficientes de detalle en función de una resolución creciente. Entonces, la base de Haar uni-dimensional, la transformada ondita de la imagen original 4 de pixeles esta dada por

[6 2 1 -1]

La manera en que se calcula la TDO, promediando y diferenciando coeficientes en forma recursiva, se llama banco de filtros [170]. Téngase en cuenta que la información no se ha ganado o perdido en este proceso. La imagen original tenía cuatro coeficientes, y lo mismo ocurre con la transformada. También tomamos nota de que, dada la transformación, podemos reconstruir la imagen para cualquier resolución recursivamente añadiendo y restando los coeficientes de detalle de las versiones de más baja resolución.

El almacenamiento de la TDO de la imagen, en lugar de la imagen misma, tiene una serie de ventajas. Una de las ventajas de la TDO es que es a menudo un gran número de coeficientes de detalle resultan ser muy pequeños en magnitud, como en el ejemplo de la Fig2.16. Truncar, o eliminar, estos pequeños coeficientes de la representación sólo introduce pequeños errores en la imagen reconstruida, dando una forma de compresión de imágenes con "pérdidas". Vamos a discutir esta aplicación en particular de la TDO en el punto 2.2.2.1.3, después de que presentemos las funciones de base de Haar uni-dimensional.

2.2.2.1.2. Funciones base para onditas de Haar unidimensional

Hemos demostrado cómo una imagen uni-dimensional puede ser tratada como secuencias de coeficientes. Alternativamente, podemos pensar en imágenes como funciones constantes por partes en el intervalo semi-abierto [0, 1). Para ello, usaremos el concepto de *espacio vectorial*. Una imagen mono-pixel es solo una función que es constante sobre el intervalo completo [0, 1). Dejemos que V^0 sea el espacio vectorial de estas funciones. Una imagen bi-pixel tiene dos partes constantes sobre los intervalos [0, 1/2) y [1/2, 1). Llamaremos V^1 al espacio que contenga a todas estas funciones. Si seguimos de esta manera, el espacio V^j incluirá a todas las funciones de parte constante definidas sobre el intervalo [0, 1) con partes constantes sobre cada uno de los 2^j subintervalos iguales. Podemos ahora pensar a cada imagen uni-dimensional con 2^j pixeles como

un elemento, o vector, en V^j. Note que debido a que estos vectores son todas funciones definidas sobre el intervalo unidad, cada vector en V^j esta también contenido en V^{j+1}. Por ejemplo, siempre podemos describir a una función constante por partes con dos intervalos como una función constante por partes con cuatro intervalos, con cada intervalo en la primera función correspondiente a un par de intervalos en la segunda. Entonces, los espacios V^j están anidados; es decir,

$$\boldsymbol{V}^0 \subset \boldsymbol{V}^1 \subset \boldsymbol{V}^2 \subset \cdots$$

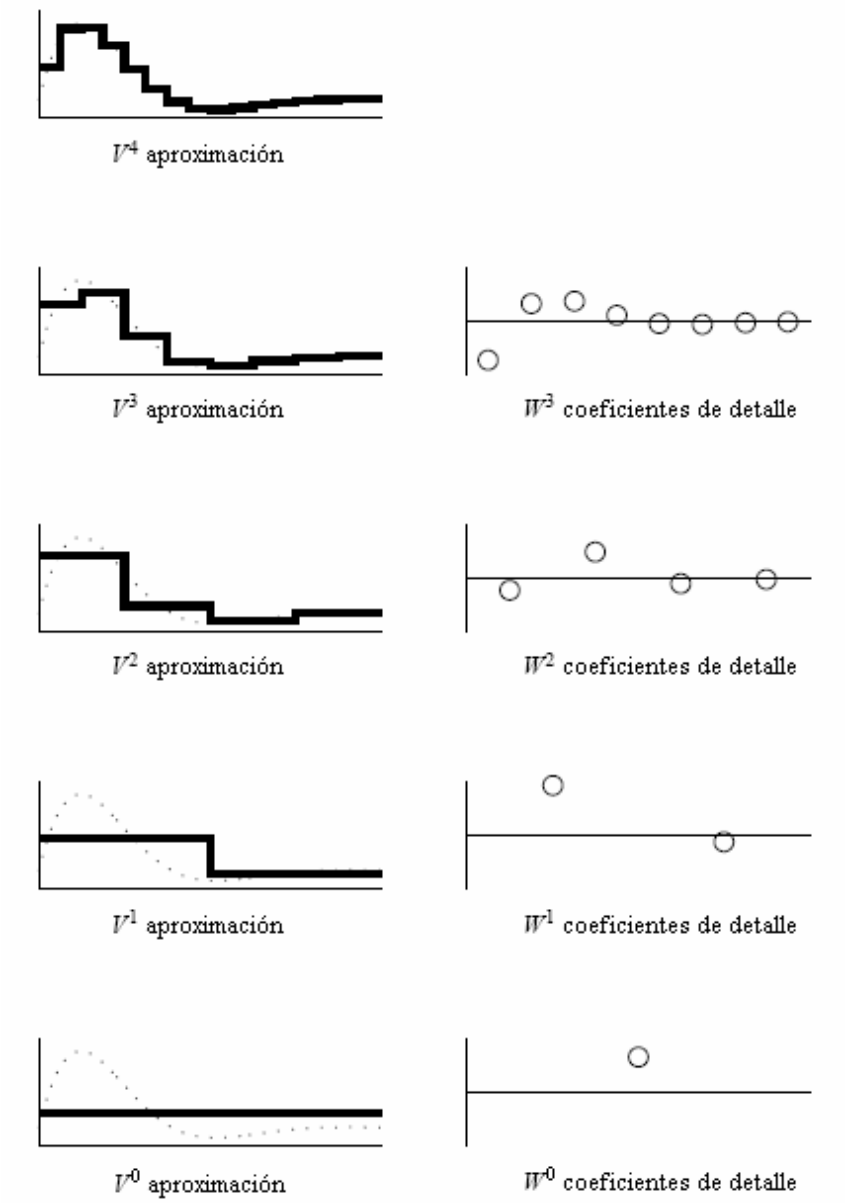


Figura 2.16: Una secuencia de aproximaciones de resolución-decreciente a una función (izquierda), junto con los coeficientes de detalle requeridos para recuperar la mejor aproximación (a la derecha). Note que en regiones donde la función verdadera esta cerca de ser plana, una aproximación constante por partes trabaja bien, por lo que los correspondientes coeficientes de detalle son relativamente pequeños.

La teoría del *análisis de multiresolución* requiere este conjunto de espacios anidados V^j [170].

Ahora necesitamos definir una base para cada espacio vectorial V^j. Las funciones base para los espacios V^j son llamadas *funciones de escala*, y son usualmente denotadas por el símbolo ϕ. Una

base simple para V^j esta dada por el conjunto de funciones "caja" escaladas y trasladadas:

$$\phi_i^j(x) = \phi(2^j x - i), \qquad i = 0, \ldots, 2^j - 1 \tag{2.14}$$

donde

$$\phi(x) = \begin{cases} 1 & \textbf{\textit{para}}\ 0 \le x < 1 \\ 0 & \textbf{\textit{para todo otro caso}} \end{cases} \tag{2.15}$$

Como un ejemplo, la Fig.2.17 muestra las cuatro funciones de caja formando una base para V^2.

El próximo paso es elegir un producto interno definido sobre los espacios vectoriales V^j. El producto interno "estándar",

$$\langle f \mid g \rangle = \int_0^1 f(x) g(x) dx \tag{2.16}$$

para dos elementos $f, g \in V^j$ lo hará muy bien para el ejemplo que ejecutaremos. Podemos definir ahora un nuevo espacio vectorial W^j como el *complemento ortogonal* de V^j en V^{j+1}. En otras palabras, vamos a dejar que W^j sea el espacio de todas las funciones en V^{j+1} que son ortogonales a todas las funciones en V^j en el marco de la elección del producto interno. Informalmente, podemos pensar a las onditas en W^j como un medio para representar las partes de una función en V^{j+1} que no pueden ser representados en V^j.

Una colección de funciones linealmente independientes $\psi_i^j(x)$ generadas en W^j recibe el nombre de onditas. Estas funciones base tienen dos importantes propiedades:

1. Las funciones base $\psi_i^j(x)$ de W^j, junto con las funciones base $\phi_i^j(x)$ de V^j, forman una base para V^{j+1}.

2. Cada función base $\psi_i^j(x)$ de W^j es ortogonal para cada función base $\phi_i^j(x)$ de V^j bajo el producto interno elegido.

Entonces, los "coeficientes de detalle" de la Sección 2.2.2.1.1 son realmente coeficientes de las funciones base de onditas. Las onditas correspondientes a la bases caja son conocidas como las onditas de *Haar*, dadas por

$$\psi_i^j(x) = \psi(2^j x - i), \qquad i = 0, \ldots, 2^j - 1 \tag{2.17}$$

donde

$$\psi(x) = \begin{cases} 1 & \textbf{\textit{para}}\ 0 \le x < 1/2 \\ -1 & \textbf{\textit{para}}\ 1/2 \le x < 1 \\ 0 & \textbf{\textit{para todo otro caso}} \end{cases} \tag{2.18}$$

La Fig.2.18 muestra dos onditas de Haar que abarca W^1.

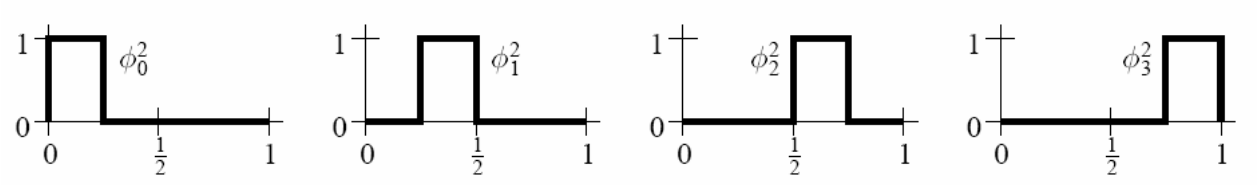


Figura 2.17: Bases caja para V^2.

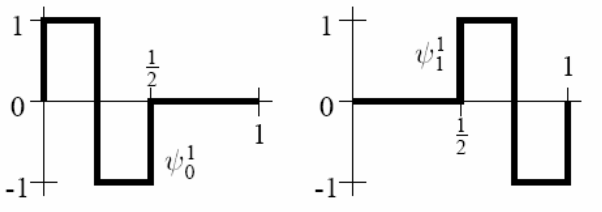
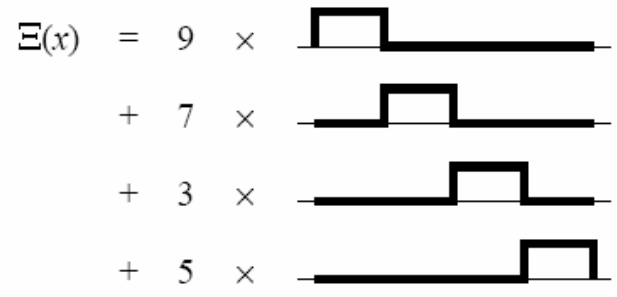


Figura 2.18: Onditas de Haar para W^1.

Antes de comenzar, vamos a ejecutar nuevamente nuestro ejemplo de la Sección 2.2.2.1.1, pero ahora aplicando estas ideas más sofisticadas. Comencemos expresando nuestra imagen original $\Xi(x)$ como una combinación lineal de funciones base caja en V^2:

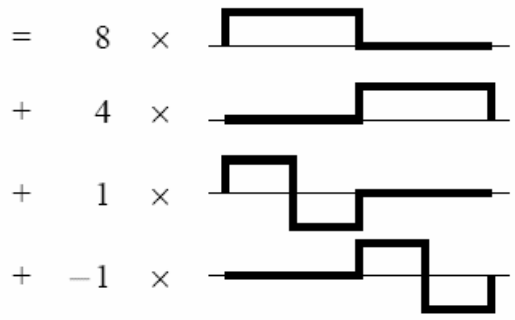
$$\Xi(x) = c_0^2\phi_0^2(x) + c_1^2\phi_1^2(x) + c_2^2\phi_2^2(x) + c_3^2\phi_3^2(x) \tag{2.19}$$

Una representación más gráfica es



Nótese que los coeficientes $c_0^2,\ldots,c_3^2$ son solo los cuatro valores de los pixeles originales [9 7 3 5]. Podemos reescribir la expresión para $\Xi(x)$ en función de las funciones base en V^1 y W^1, usando premediación y diferenciación por partes:

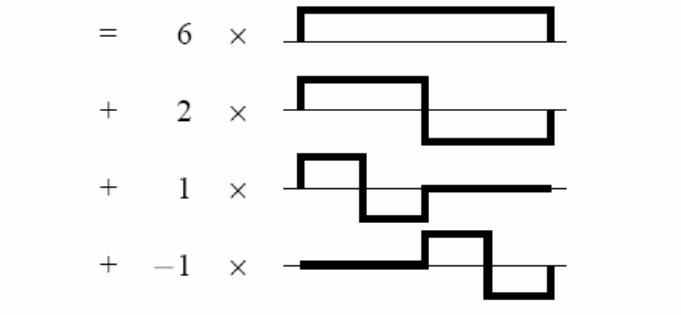
$$\Xi(x) = c_0^1\phi_0^1(x) + c_1^1\phi_1^1(x) + d_0^1\psi_0^1(x) + d_1^1\psi_1^1(x)$$



Estos cuatro coeficientes deberían lucir bien familiares.

Finalmente, reescribimos $\Xi(x)$ como una suma de funciones base en V^0, W^0, y W^1:

$$\Xi(x) = c_0^0\,\phi_0^0(x) + d_0^0\,\psi_0^0(x) + d_0^1\,\psi_0^1(x) + d_1^1\,\psi_1^1(x)$$



Una vez más, estos cuatro coeficientes son la transformada ondita de Haar de la imagen original. Las cuatro funciones que se muestran arriba constituyen la base de Haar para V^2. En lugar de usar las usuales cuatro funciones caja, podemos usar ϕ_0^0, ψ_0^0, ψ_0^1, y ψ_1^1 para representar al promedio total, el detalle amplio, los dos tipos de posibles detalles mas finos en una función en V^2. La base de Haar para V^j con $j > 2$ incluye estas funciones tan bien como las traslaciones más estrechas de la ondita $\psi(x)$.

2.2.2.1.2.1. Ortogonalidad

La base de Haar posee una importante propiedad conocida como *ortogonalidad*, la cual no siempre es compartida por otras bases de onditas. Una base ortogonal es una en la cual todas las funciones base, en este caso $\phi_0^0, \psi_0^0, \psi_0^1, \psi_1^1, \ldots$, son ortogonales a las otras. Nótese que la ortogonalidad es más fuerte que el requerimiento mínimo para onditas ψ_i^j que son ortogonales a todas las funciones escala en el mismo nivel de resolución j.

2.2.2.1.2.2. Normalización

Otra propiedad que es algunas veces deseable es la *normalización*. Una función base $u(x)$ está normalizada si $\langle u \mid u \rangle = 1$. Podemos normalizar la base de Haar al reemplazar nuestras definiciones más tempranas con

$$\begin{aligned} \phi_i^j(x) &= 2^{j/2}\,\phi(2^j x - i) \\ \psi_i^j(x) &= 2^{j/2}\,\psi(2^j x - i) \end{aligned} \qquad (2.20)$$

donde el factor constante $2^{j/2}$ se elige para satisfacer $\langle u \mid u \rangle = 1$ para el producto interno estándar. Con estas definiciones modificadas, los nuevos coeficientes normalizados son obtenidos al multiplicar cada antiguo coeficiente con potencia j para $2^{-j/2}$. Entonces, en el ejemplo de la sección previa, los coeficientes no-normalizados [6 2 1 -1] se convierten en los coeficientes normalizados

$$\left[\; 6 \;\; 2 \;\; \frac{1}{\sqrt{2}} \;\; \frac{-1}{\sqrt{2}} \;\right]$$

Como una alternativa para calcular primero los coeficientes no-normalizados y luego los normalizados, podemos incluir la normalización en el algoritmo de descomposición. Los siguientes dos algoritmos realizan la descomposición normalizada:

Algoritmo 2.1

```
Descomposición_en_pasos(C: arreglo [1. . h] de reales)
    para i = 1 a h/2 hacer
        C'[i] = (C[2i - 1] + C[2i])/√2
        C' [h/2 + i] = (C[2i - 1] - C[2i]) /√2
    fin para
    C = C'
Fin
```

Algoritmo 2.2

```
Descomposition(C: arreglo [1. . h] of reales)
    C = C/√h  (normaliza los coeficientes de entrada)
    mientras h > 1 hacer
        Descomposición_en_pasos(C[1. . h])
        h = h/2
    fin mientras
fin
```

Ahora podemos trabajar con una base ortonormal, es decir, ortogonal y normalizada. Usando una base ortonormal resulta ser útil cuando se comprime una función o una imagen, la cual se describe a continuación.

2.2.2.1.3. Aplicaciones I: Compresión

El objetivo de la compresión es expresar un conjunto inicial de datos usando algún conjunto de dato más pequeño, con o sin pérdida de información. Por ejemplo, supongamos que nos dan una función $f(x)$ expresada como una suma ponderada de funciones base $\boldsymbol{u}_1(\boldsymbol{x}),\ldots,\boldsymbol{u}_{\boldsymbol{m}}(\boldsymbol{x})$:

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} \boldsymbol{c}_{\boldsymbol{i}}\, \boldsymbol{u}_{\boldsymbol{i}}(\boldsymbol{x}) \tag{2.21}$$

El conjunto de datos en este caso consiste de los coeficientes $\boldsymbol{c}_1,\ldots,\boldsymbol{c}_{\boldsymbol{m}}$. Nos gustaría encontrar una función de aproximación $f(x)$ pero que requiera menos coeficientes, tal vez al usar una base diferente. Es decir, dada una tolerancia al error espcificada por el usuario ε (para compresión sin pérdidas, $\varepsilon = 0$), estamos buscando

$$\tilde{\boldsymbol{f}}(\boldsymbol{x}) = \sum_{\boldsymbol{i}=1}^{\tilde{\boldsymbol{m}}} \tilde{\boldsymbol{c}}_{\boldsymbol{i}}\, \tilde{\boldsymbol{u}}_{\boldsymbol{i}}(\boldsymbol{x}) \tag{2.22}$$

tal que $\tilde{\boldsymbol{m}} < \boldsymbol{m}$ y $\left\| \boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}(\boldsymbol{x}) \right\| \le \varepsilon$ para la misma norma. En general, usted puede tratar de construir un conjunto de funciones base $\tilde{\boldsymbol{u}}_1,\ldots,\tilde{\boldsymbol{u}}_{\tilde{\boldsymbol{m}}}$ que proveerían una buena aproximación con pocos coeficientes. Nos enfocaremos en cambio sobre el problema más simple de encontrar una

buena aproximación en una base fija. Una forma del problema de descompresión es ordenar los coeficientes $\boldsymbol{c}_1,\ldots,\boldsymbol{c}_m$ de modo que por cada $\tilde{\boldsymbol{m}} < \boldsymbol{m}$, los primeros $\tilde{\boldsymbol{m}}$ elementos de la secuencia darán la mejor aproximación de $\tilde{\boldsymbol{f}}(\boldsymbol{x})$ a $\boldsymbol{f}(\boldsymbol{x})$ como medida en la norma L^2. Como mostramos aquí, la solución a este problema es sencillo si la base es ortonormal, como es el caso con la base normalizada de Haar.
Sea σ una permutación de 1, …, *m*, sea $\tilde{\boldsymbol{f}}(\boldsymbol{x})$ una función que usa los coeficientes correspondientes para los primeros $\tilde{\boldsymbol{m}}$ números de la permutación σ:

$$\tilde{\boldsymbol{f}}(\boldsymbol{x}) = \sum_{\boldsymbol{i}=1}^{\tilde{\boldsymbol{m}}} \boldsymbol{c}_{\sigma(\boldsymbol{i})}\,\boldsymbol{u}_{\sigma(\boldsymbol{i})} \tag{2.23}$$

El cuadrado del error L^2 en esta aproximación es

$$\begin{aligned}
\left\| \boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}(\boldsymbol{x}) \right\|_2^2 &= \left\langle \boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}(\boldsymbol{x}) \,|\, \boldsymbol{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}(\boldsymbol{x}) \right\rangle \\
&= \left\langle \sum_{\boldsymbol{i}=\tilde{\boldsymbol{m}}+1}^{\boldsymbol{m}} \boldsymbol{c}_{\sigma(\boldsymbol{i})}\boldsymbol{u}_{\sigma(\boldsymbol{i})} \,|\, \sum_{\boldsymbol{j}=\tilde{\boldsymbol{m}}+1}^{\boldsymbol{m}} \boldsymbol{c}_{\sigma(\boldsymbol{j})}\boldsymbol{u}_{\sigma(\boldsymbol{j})} \right\rangle \\
&= \sum_{\boldsymbol{i}=\tilde{\boldsymbol{m}}+1}^{\boldsymbol{m}} \sum_{\boldsymbol{j}=\tilde{\boldsymbol{m}}+1}^{\boldsymbol{m}} \boldsymbol{c}_{\sigma(\boldsymbol{i})}\boldsymbol{c}_{\sigma(\boldsymbol{j})} \left\langle \boldsymbol{u}_{\sigma(\boldsymbol{i})} \,|\, \boldsymbol{u}_{\sigma(\boldsymbol{j})} \right\rangle \\
&= \sum_{\boldsymbol{i}=\tilde{\boldsymbol{m}}+1}^{\boldsymbol{m}} \left(\boldsymbol{c}_{\sigma(\boldsymbol{i})}\right)^2
\end{aligned} \tag{2.24}$$

El último paso sigue de la suposición que la base es ortonormal, así $\left\langle \boldsymbol{u}_{\boldsymbol{i}} \,|\, \boldsymbol{u}_{\boldsymbol{j}} \right\rangle = \delta_{\boldsymbol{ij}}$. Concluimos que para minimizar este error para cualquier $\tilde{\boldsymbol{m}}$ dado, la mejor elección para σ es la permutación que ordena los coeficientes en forma decreciente; es decir, σ satisface $\left|\boldsymbol{c}_{\sigma(1)}\right| \geq \cdots \geq \left|\boldsymbol{c}_{\sigma(\boldsymbol{m})}\right|$.

La Fig.2.16 demuestra cómo una función unidimensional podría ser transformada en coeficientes representando el promedio sobre todas las funciones promedio y varias resoluciones de detalle. Ahora repetimos el proceso, esta vez usando funciones de Haar normalizadas. Podemos aplicar la compresión en L^2 a los coeficientes resultantes simplemente removiendo o ignorando los coeficientes de más pequeños. Al variar la cantidad de compresión, obtenemos una secuencia de aproximaciones a la función original, como se muestra en la Fig.2.19.

2.2.2.2. Onditas en dos dimensiones

En preparación para la compresión de imágenes, necesitamos generalizar las onditas de Haar a dos dimensiones. Primero, consideraremos cómo realizar una descomposición de onditas del valor del pixel en una imagen bi-dimensional. Entonces describiremos las funciones de escala y onditas que forman una base de onditas bi-dimensional.

2.2.2.2.1. Transformada Onditas de Haar bidimensional

Hay dos formas en la que podemos usar onditas para transformar los valores del pixel en una imagen. Cada uno es una generalización para dos dimensiones de la transformada de onditas unidimensional descripta en la Sección 2.2.2.1.1.

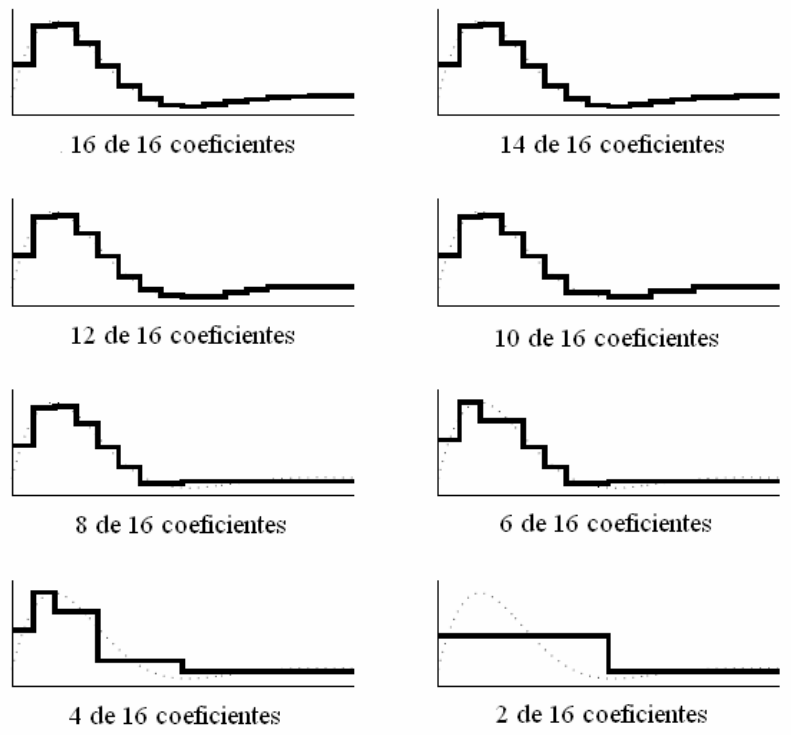


Figura 2.19: Aproximaciones a una función obtenida mediante la compresión de L^2: los coeficientes de detalle se eliminan con el fin de aumentar la magnitud.

Para obtener la *descomposición estándar* [172] de una imagen, aplicamos primero la transformada onditas uni-dimensional a cada fila de pixeles. Esta operación nos da un valor medio junto con los coeficientes de detalle para cada fila. A continuación, tratamos estas filas transformadas como si ellas mismas fueran una imagen y aplicamos la transformada uni-dimensional a cada columna. Los valores resultantes son todos coeficientes de detalle excepto para un único coeficiente promedio global. El Algoritmo 2.3 calcula la descomposición estándar. La Fig.2.20 ilustra cada paso de su operación.

Algoritmo 2.3

```
Descomposición_estándar(C: arreglo [1. . h, 1. . w] de reales)
    para filas = 1 a h hacer
        Descomposición(C[fila, 1. . w])
    fin para
    para col = 1 a w hacer
        Descomposición(C[1. . h, col])
    fin para
fin
```

El Segundo tipo de transformada ondita bi-dimensional, llamada descomposición no-estándar, alterna entre operaciones sobre filas y columnas. Primero, realizamos un paso de un promedio de par horizontal y diferenciando en los valores de los pixeles en cada fila de la imagen. Luego, aplicamos un promedio par vertical y diferenciando para cada columna del resultado. Para completar la transformación, repetimos este proceso recursivamente solo sobre el cuadrante que contiene los promedios en ambas direcciones. La Fig.2.21 muestra todos los pasos involucrados en el procedimiento de descomposición estándar de la figura.

Figura 2.20: Descomposición estándar de una imagen.

Figura 2.21: Descomposición no-estándar de una imagen.

Algoritmo 2.4

```
Descomposición_no-estándar(C: arreglo [1. . h, 1. . h] de reales)
    C = C/h (normaliza los coeficientes de entrada)
    mientras h > 1 hacer
        para filas = 1 a h hacer
            Descomposición_en_pasos (C[row, 1. . h])
        fin para
        para col = 1 a h hacer
            Descomposición_en_pasos(C[1. . h, col])
        fin para
        h = h/2
    fin mientras
fin
```

2.2.2.2.2. Funciones base de Haar bidimensional

Los dos métodos de descomposición de una imagen bidimensional presentan coeficientes que corresponden a dos diferentes conjuntos de funciones de base. La descomposición estándar de una imagen da coeficientes para una base formada por la *construcción estándar* [172] de una base bi-dimensional. Similarmente, la descomposición no-estándar da coeficientes para la *construcción no-estándar* de las funciones base.

La construcción estándar de la base de onditas bi-dimensional consiste de todos los posibles productos tensoriales de las funciones base uni-dimensional. Por ejemplo, cuando comenzamos con la base de Haar uni-dimensional para V^2, obtenemos la base bi-dimensional para V^2 mostrada en la Fig.2.22. Note que si aplicamos la construcción estándar a una base ortonormal en una dimensión, obtenemos una base ortonormal en dos dimensiones.

La construcción no-estándar de una base bi-dimensional procede en primer lugar la definición de una función de escalado bi-dimensional

$$\phi\phi(\boldsymbol{x},\boldsymbol{y}) = \phi(\boldsymbol{x})\phi(\boldsymbol{y}) \tag{2.25}$$

y tres funciones onditas,

$$\begin{aligned} \phi\psi(\boldsymbol{x},\boldsymbol{y}) &= \phi(\boldsymbol{x})\psi(\boldsymbol{y}) \\ \psi\phi(\boldsymbol{x},\boldsymbol{y}) &= \psi(\boldsymbol{x})\phi(\boldsymbol{y}) \\ \psi\psi(\boldsymbol{x},\boldsymbol{y}) &= \psi(\boldsymbol{x})\psi(\boldsymbol{y}) \end{aligned} \tag{2.26}$$

Ahora denotamos los niveles de escalado con un superíndice j (como hicimos en el caso uni-dimensional) y traslaciones horizontales y verticales con un par de subíndices k y l. Las bases no-estándar consisten de una única función rústica de escalado $\phi\phi^0_{0,0}(\boldsymbol{x},\boldsymbol{y}) = \phi\phi(\boldsymbol{x},\boldsymbol{y})$ a lo largo de las escalas y traslaciones de las tres funciones onditas $\phi\psi$, $\psi\phi$, y $\psi\psi$:

$$\begin{aligned} \phi\psi^j_{\boldsymbol{kl}}(\boldsymbol{x},\boldsymbol{y}) &= 2^j\phi\psi\left(2^j\boldsymbol{x}-\boldsymbol{k},2^j\boldsymbol{y}-\boldsymbol{l}\right) \\ \psi\phi^j_{\boldsymbol{kl}}(\boldsymbol{x},\boldsymbol{y}) &= 2^j\psi\phi\left(2^j\boldsymbol{x}-\boldsymbol{k},2^j\boldsymbol{y}-\boldsymbol{l}\right) \\ \psi\psi^j_{\boldsymbol{kl}}(\boldsymbol{x},\boldsymbol{y}) &= 2^j\psi\psi\left(2^j\boldsymbol{x}-\boldsymbol{k},2^j\boldsymbol{y}-\boldsymbol{l}\right) \end{aligned} \tag{2.27}$$

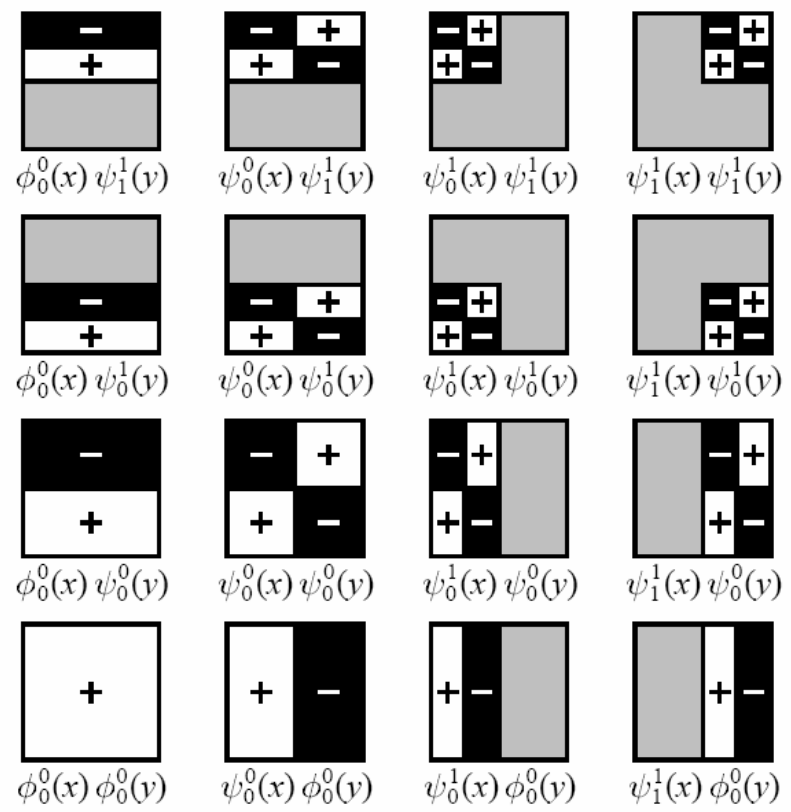


Figura 2.22: Construcción estándar de una base de onditas Haar bidimensional para V^2. En el caso no-normalizado, las funciones son +1 donde los signos mas aparecen, -1 donde los signos menos aparecen, y 0 regiones de grises.

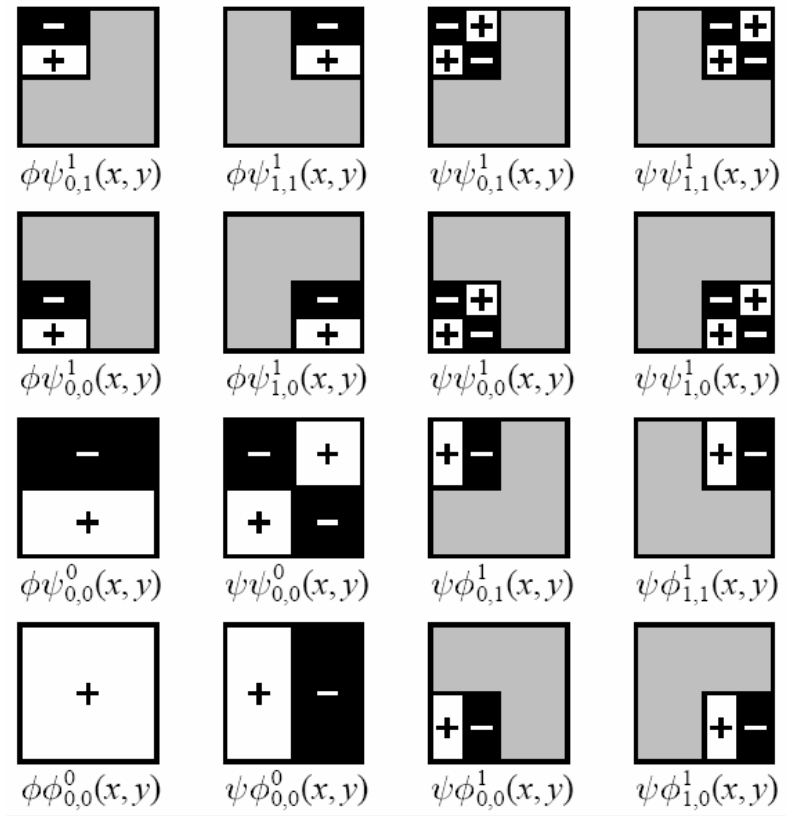


Figura 2.23: Construcción no-estándar de una base de onditas de Haar bidimensional para V^2.

La constante 2^j normaliza las onditas para dar una base ortonormal. La construcción no-estándar resulta en la base para V^2 mostrada en Fig.2.23. Hemos presentado aquí ambas aproximaciones, la estándar y la no-estándar para las transformadas onditas y las funciones base porque ambas tienen ventajas. La descomposición estándar de una imagen es atractiva porque simplemente requiere realizar transformadas uni-dimensionales sobre todas las filas y entonces sobre todas las columnas. Por otro lado, es ligeramente más eficiente para calcular la descomposición no-estándar. Para una imagen m x m, la descomposición estándar requiere un costo de $4(m^2 - m)$ operaciones, mientras la no-estándar requiere solamente un costo de $8/3\ (m^2 - 1)$ operaciones. Otra consideración es el *soporte* de cada función base, lo que significa que la porción de cada dominio de la función donde la función es distinta de cero. Todas las funciones de base de Haar no-estándar poseen soporte cuadrático, mientras que algunas funciones de base estándar tienen soporte no cuadrático. Dependiendo de la aplicación, una de estas opciones puede ser preferible a la otra.

2.2.2.2.3. Aplicaciones II: Compresión de imágenes

En la Sección 2.2.2.1.3 definimos compresión como la representación de una función usando menos coeficientes de función base que los que fueron originalmente dados. El método que discutimos para una función uni-dimensional se aplica igualmente bien a imágenes, las cuales tratamos como los coeficientes correspondientes a una base constante por partes bi-dimensional. La aproximación presentada aquí es solo introductoria; para un tratamiento más completo de la compresión de imágenes con onditas, ver el artículo de DeVore *et al.* [176].

Podemos resumir la compresión de imagines con onditas usando la norma L^2 en tres pasos:

1. Calcular los coeficientes $\boldsymbol{c}_1,\ldots,\boldsymbol{c}_{\boldsymbol{m}}$ que representan una imagen en una base de Haar bi-dimensional.

2. Ordenar los coeficientes en orden decreciente para producir la secuencia $\boldsymbol{c}_{\sigma(1)},\ldots,\boldsymbol{c}_{\sigma(\boldsymbol{m})}$.

3. Comenzando con $\tilde{\boldsymbol{m}} = \boldsymbol{m}$, encontrar el $\tilde{\boldsymbol{m}}$ más pequeño para el cual $\sum_{\boldsymbol{i}=\tilde{\boldsymbol{m}}+1}^{\boldsymbol{m}} \left(\boldsymbol{c}_{\sigma(\boldsymbol{i})}\right)^2 \le \varepsilon^2$, donde ε es el error L^2 aceptable.

El primer paso se logra mediante la aplicación de cualquiera de las dos transformadas de onditas de Haar bi-dimensional descriptas en la Sección 2.2.2.2.1, asegurándose de usar funciones base normalizadas. Cualquier técnica de ordenamiento estándar servirá para el segundo paso. No obstante, para imagines grandes el ordenamiento se hace excesivamente lento. El Algoritmo 2.5 resulta ser un método más eficiente que usar una estrategia de búsqueda binaria para encontrar un umbral bajo el cual la dimensión de los coeficientes que se considere insignificante. El procedimiento toma como entrada un arreglo uni-dimensional de coeficientes C (con cada coeficiente correspondiente a una función base bi-dimensional) y una tolerancia al error ε. Para cada estimación del umbral τ, el algoritmo calcula el cuadrado del error L^2 que debería resultar de los coeficientes descartados más pequeños que el umbral τ. Este error cuadrático s es comparado a ε^2 en cada iteración para decidir si la búsqueda binaria debería continuar en la mitad superior o inferior del intervalo actual. El algoritmo se detiene cuando el intervalo actual es tan estrecho que el número de coeficientes que vayan a desecharse ya no cambia. Este algoritmo de búsqueda binaria fue usado para producir las imágenes en la Fig.2.24. Estas imágenes demuestran las altas tasas de compresión que ofrecen las onditas, así como algunos de los artefactos que ellas producen.

Algoritmo 2.5

Compresión(C: **arreglo** [1. .m] **de reales**; ε : **real**)
  $\tau_{\min} = \min\{|\boldsymbol{C}[\boldsymbol{i}]|\}$
  $\tau_{\max} = \max\{|\boldsymbol{C}[\boldsymbol{i}]|\}$
  hacer
    $\tau = (\tau_{\min} + \tau_{\max})/2$
    $s = 0$
    para $i = 1$ **a** m **hacer**
      si $|\boldsymbol{C}[\boldsymbol{i}]| < \tau$ **entonces** $s = s + (C[i])^2$
    fin para
    si $\boldsymbol{s} < \varepsilon^2$ **entonces** $\tau_{\min} = \tau$ **sino** $\tau_{\max} = \tau$
  hasta $\tau_{\min} \approx \tau_{\max}$
  para $i = 1$ **a** m **hacer**
    si $|\boldsymbol{C}[\boldsymbol{i}]| < \tau$ **entonces** $C[i] = 0$
  fin para
fin

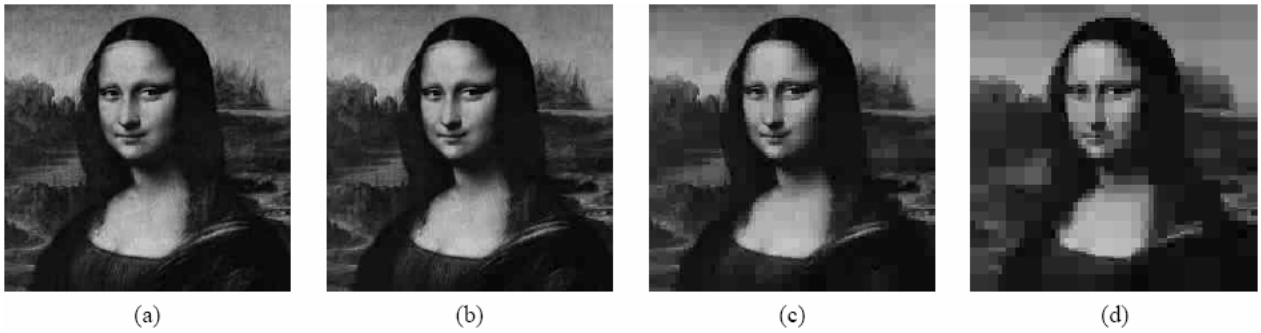


Figura 2.24: Compresión de imágenes con onditas en L^2: La imagen original (a) puede ser representada usando (b) 19% de sus coeficientes de onditas, con 5% de error relativo en L^2; (c) 3% de sus coeficientes de onditas, con 10% de error relativo en L^2; y (d) 1% de sus coeficientes de onditas, con 15% de error relativo en L^2.

DeVore *et al.* [176] sugirieron que la norma L^1 se adapta mejor a la tarea de compresión de imágenes. El Algoritmo 2.6 es un fragmento para un "codicioso" esquema de compresión L^1:

Algoritmo 2.6

para cada pixel(x, y) **hacer**
  $\delta[\boldsymbol{x}, \boldsymbol{y}] = 0$
fin para
para $i = 1$ **a** m **hacer**
  $\delta' = \delta$ + error de descarte $C[i]$
  si $\sum_{x,y} |\delta'[\boldsymbol{x}, \boldsymbol{y}]| < \varepsilon$ **entonces**
    descartar coeficientes $C[i]$
    $\delta = \delta'$
  fin si
fin para

Note que el resultado de este algoritmo depende del orden en el cual los coeficientes son visitados. Diferentes imágenes (y grados de compresión) pueden ser obtenidas al variar este orden—por ejemplo, al comenzar con la escala de coeficientes más fina, en lugar de los coeficientes más pequeños. También se puede ejecutar un procedimiento de optimización más sofisticado y restringido para seleccionar el número mínimo de los coeficientes sujetos al error establecido.

2.2.2.2.4. Noción de sub-bandas espectrales

La TDO-2D [176-186, 328-350] se corresponde con una serie de expresiones aproximadas de multi-resolución. En la práctica con imágenes, el análisis de multi-resolución es llevado a cabo usando cuatro canales de bancos de filtros (para cada nivel de descomposición) compuesta de un filtro pasa-bajo y otro pasa-alto en cada banco de filtro. Luego se muestrea a media tasa (se baja el muestreo a la mitad) de la frecuencia previa. Se repite este procedimiento, dado que – en principio – es posible obtener transformada ondita de cualquier orden. El procedimiento de disminución del muestreo mantiene totalmente constante el parámetro de escala (igual a ½) de las sucesivas transformadas de onditas generando un beneficio en la implementación computacional. Dado que se aplica a una imagen, el filtrado es implementado en una forma separable, es decir, por filas y columnas.

Cabe señalarse en este punto, que el mismo proceso que se describirá aquí para filtrado de la imagen (mediante un umbralizado de las sub-bandas de alta frecuencia) sirve para el proceso de compresión.

De [172] y [177], la TDO de una imagen consiste de cuatro canales de frecuencia por cada nivel de descomposición. Por ejemplo, para el nivel-*i* de descomposición, tenemos:

$LL_{r,i}$: Coeficientes Ruidosos de Aproximación.
$LH_{r,i}$: Coeficientes Ruidosos de Detalle Vertical,
$HL_{r,i}$: Coeficientes Ruidosos de Detalle Horizontal, y
$HH_{r,i}$: Coeficientes Ruidosos de Detalle Diagonal.

La sub-banda LL se descompone recursivamente en cada escala, como se ilustra en la Fig. 2.25 [176], [177].

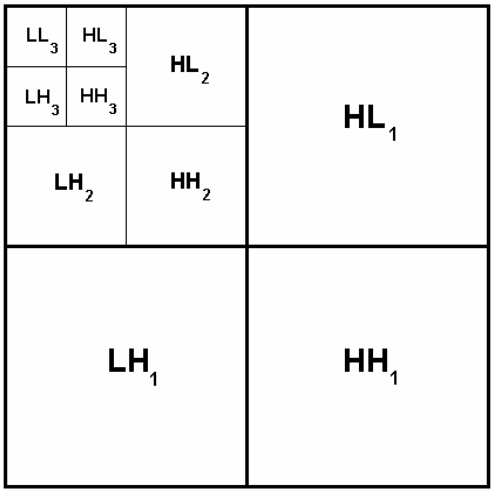


Fig. 2.25 Preparación de los datos de la imagen. Descomposición recursiva de la sub-banda LL.

Para lograr una reducción de ruido adaptativa de escala-espacial, necesitamos preparar la cadena de datos de los coeficientes 1-D los cuales contienen la información en escala-espacial de las imágenes 2D. Esto es algo similar a la exploración "zigzag" de los coeficientes de la TDC (ver Apéndice A) en aplicaciones de codificación de imágenes [331]. En este paso de preparación de los datos, los coeficientes de la TDO-2D son reordenados como una serie de coeficientes 1-D en orden espacial a fin de que las muestras adyacentes de la misma representen áreas locales de la imagen original [333]. La Fig.2.26 muestra el interior de la TDO-2D con las cuatro sub-bandas de la imagen transformada [338], la cual será usada en la Fig.2.27. Cada salida de la Fig.2.26 representa una sub-banda del proceso de división de la matriz de coeficientes 2-D correspondientes a la Fig. 2.25.

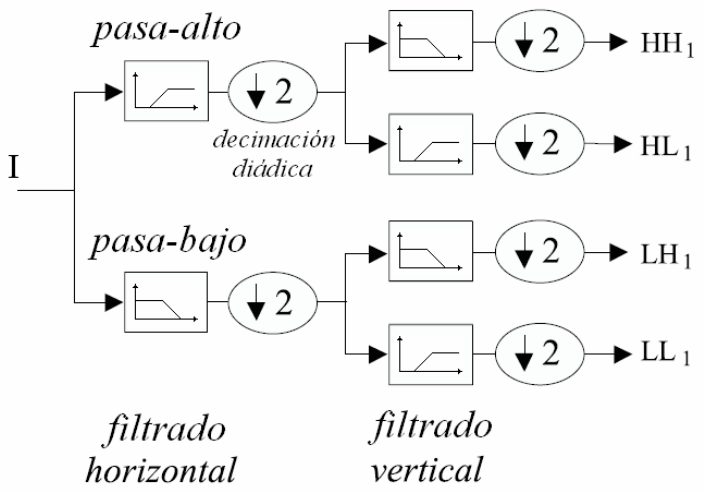


Fig.2.26: TDO bi-dimensional. Un paso de descomposición. División habitual de las sub-bandas.

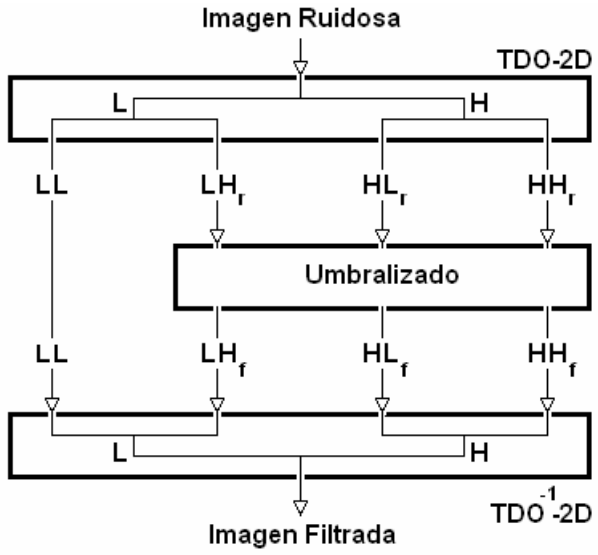


Fig.2.27: Técnica de Umbralizado

2.2.2.2.4.1. Umbralizado de ruido para onditas

Los coeficientes de onditas calculados por la TDO representan cambios en la imagen en una resolución en particular. Al mirar la imagen en varias resoluciones, debería ser posible filtrar los ruidos, al menos en teoría. Sin embargo, la definición de ruido es complicada.

De hecho, "lo que para una persona es ruido, para otra es una señal". En particular esto depende de que resolución se este buscando. Un algoritmo para remover el ruido blanco Gaussiano se resume en D.L. Donoho y I.M. Johnstone [328], [329], y está sintetizado en la Fig.2.27.

El procedimiento consiste en:

1) Calcular una TDO y ordenar los coeficientes en orden creciente de frecuencia. Esto resultará en un arreglo conteniendo la imagen promediada mas un conjunto de coeficientes de longitud 1, 2, 4, 8, etc. El umbral de ruido será calculado sobre el espectro de coeficientes de frecuencias más altas (este es el espectro más grande).

2) Calcular la *desviación media absoluta* (DMA) sobre el espectro de coeficientes más grandes. La media es calculada del valor absoluto de los coeficientes. La ecuación para la DMA se muestra abajo:

$$\delta_{DMA} = \frac{media(|\, C_{r,i} \,|)}{0.6745} \tag{2.28}$$

donde $C_{r,i}$ puede ser $LH_{r,i}$, $HL_{r,i}$, o $HH_{r,i}$ para el nivel-*i* de descomposición. El factor 0.6745 en el denominador re-escala el numerador a fin de que δ_{MDA} sea también un estimador de la desviación estándar del ruido blanco Gaussiano [175], [331], [332].

3) Para calcular el umbral de ruido λ hemos usado una versión modificada de la ecuación que ha sido discutida en los trabajos de D.L. Donoho y I.M. Johnstone [213-217]. La ecuación es:

$$\lambda = \delta_{MDA} \sqrt{2log[N]} \tag{2.29}$$

donde N es el número de pixeles en la sub-imagen, i.e., HL, LH o HH.

4) Aplicar el algoritmo de umbralizado a los coeficientes. Hay dos versiones populares:

4.1. *Umbralizado abrupto*. El umbralizado abrupto pone cualquier coeficiente menor o igual que el umbral a cero, ver Fig.2.28(a).

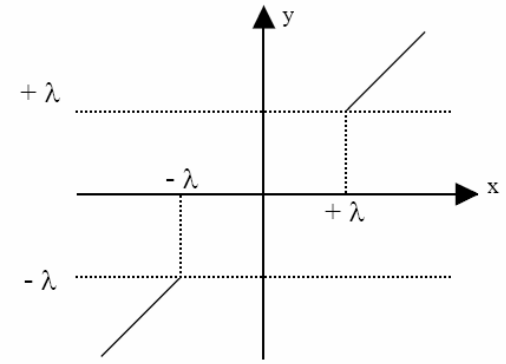


Fig.2.28(a): Umbralizado abrupto.

donde
x debe ser $LH_{r,i}$, $HL_{r,i}$, o $HH_{r,i}$,

y debe ser
$HH_{f,i}$: Coeficientes de Detalles Filtrados Diagonal,
$HL_{f,i}$: Coeficientes de Detalles Filtrados Horizontal,
$LH_{f,i}$: Coeficientes de Detalles Filtrados Vertical,
para el nivel-*i* de descomposición.

Algoritmo 2.7

```
para fila = 1:N
  para columna = 1:N
    si |Cn,i[fila][columna]| <= λ,
      Cn,i[fila][columna] = 0.0;
    fin si
  fin para
fin para
```

4.2. *Umbralizado suave*. El umbralizado suave cualquier coeficiente menor o igual que el umbral a cero, ver Fig.2.28(b). El umbral es sustraído de cualquier coeficiente que es mayor que el umbral. Esto mueve los coeficientes de la imagen hacia cero.

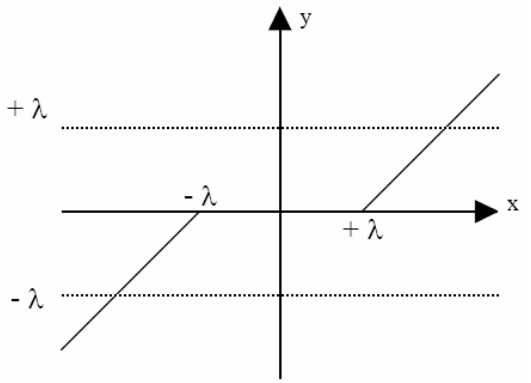


Fig.2.28(b) Umbralizado suave.

Algoritmo 2.8

```
para fila = 1:N
  para columna = 1:N
    si |Cn,i[fila][columna]| <= λ,
      Cn,i[fila][columna] = 0.0;
    sino
      Cn,i[fila][columna] = Cn,i[fila][columna] - λ;
    fin si
  fin para
fin para
```

2.2.3. La Transformada de Walsh-Hadamard (TWH)

La TWH [285] es un ejemplo de una clase generalizada de la Transformada de Fourier. La misma realiza una operación ortogonal, simétrica, involutiva (la matriz de la transformación es igual a la de su inversa) y lineal sobre 2^m números reales (o números complejos, aunque las matrices de Hadamard propiamente dichas son puramente reales.

La TWH puede ser considerada como siendo conformada de matrices de TDF de dimensión 2, y de hecho es equivalente a una TDF multidimensional de dimensión 2 x 2 x … x 2 x 2. Descompone un vector de entrada arbitraria en una superposición de funciones de Walsh (Fig.2.29).

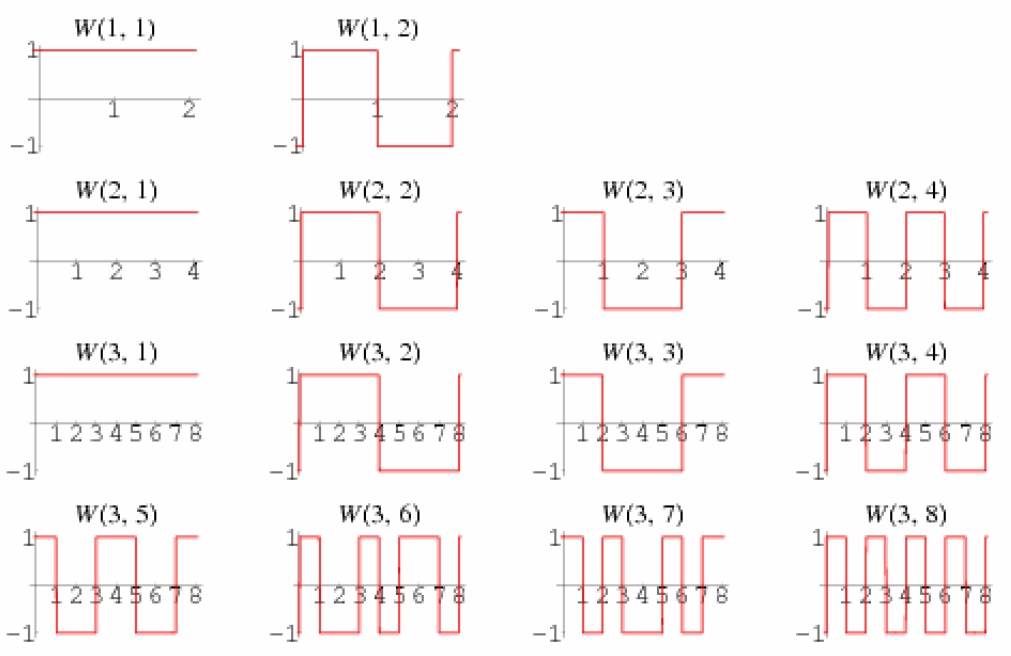


Fig.2.29: Funciones de Walsh.

2.2.3.1. Definiciones

La TWH H_m es una matriz de 2^m x 2^m, la matriz de Hadamard (escalada por un factor de normalización), que transforma 2^m números reales x_n en 2^m números reales X_n. Podemos definir la TWH de dos maneras:

1. Recursivamente
2. Usando la representación binaria (base 2) de los índices *n* y *k*.

1. Recursivamente, definimos la transformada de Hadamard H_0 de 1 x 1 por la identidad $H_0 = 1$, y entonces definir H_m para m > 0 como:

$$H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix} \tag{2.30}$$

donde $\frac{1}{\sqrt{2}}$ es una normalización, la cual es algunas veces omitida. Entonces, a excepción de dicho factor de normalización, las matrices de Hadamard están hechas enteramente de 1 y -1.

2. Equivalentemente, podemos definir la matriz de Hadamard por su (*k*, *n*)-ésima entrada al escribir

$$k = k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \cdots + k_1 2 + k_0 \tag{2.31}$$

y

$$n = n_{m-1}2^{m-1} + n_{m-2}2^{m-2} + \cdots + n_1 2 + n_0 \tag{2.32}$$

donde k_j y n_j son dígitos binarios (0 o 1) de k y n, respectivamente. En este caso, tenemos:

$$\left(H_m\right)_{k,n} = \frac{1}{2^{m/2}}(-1)^{\sum_j k_j n_j} \tag{2.33}$$

Esto es exactamente la TDF multidimensional 2 x 2 x … x 2 x 2, normalizada a la unidad, si consideramos a las entradas y salidas como arreglos multidimensionales indexados por k_j y n_j , respectivamente.

Algunos ejemplos de matrices de Hadamard son los siguientes.

$$H_0 = +1 \tag{2.34}$$

$$H_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.35}$$

Este último caso es precisamente la TDF de dimensión 2. Puede ser considerado también como la TDF sobre el grupo aditivo de dos-elementos de $Z/(2)$.

$$H_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{2.36}$$

$$H_3 = \frac{1}{2^{3/2}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \tag{2.37}$$

Las filas de las matrices de Hadamard son las funciones de Walsh.

2.2.3.2. Complejidad computacional

La TH puede ser calculada en m x log(m) operaciones, usando el algoritmo de la Transformada Rápida de Hadamard (TRH) [36].

2.2.4. La Transformada de Discreta de Hartley (TDH)

La TDH [286] es un caso particular de la TDF [312-319]. No obstante, a diferencia de esta última que esta definida en el campo complejo, la TDH esta definida en el campo de los reales. La misma toma la forma

$$F(p,q)=\frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)\,cas\left(\frac{2\pi}{N}(pm+qn)\right) \tag{2.38}$$

$$f(m,n)=\frac{1}{N}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} F(p,q)\,cas\left(\frac{2\pi}{N}(pm+qn)\right) \tag{2.39}$$

siendo cas(•) = cos(•)+sen(•).

La misma posee versión rápida al igual que la TRF y será sumamente útil como parte de las soluciones propuestas en el Capítulo 3.

2.3. Conclusiones del capítulo

En este capítulo se presentaron las TDC y TDO, en este último caso y en particular la Transformada de Haar, así como las Transformadas de Walsh-Hadamard y la Transformada Discreta de Hartley, las cuales y como se verá en detalle en el Capítulo 3 permitirán vehiculizar SMEP más eficientemente al emplear la TDKL.

Capítulo 3

3. Efecto de utilizar SMEP en la TDKL

3.1. Introducción

En este capítulo se verá la formalización del concepto de SMEP, su origen proveniente del Álgebra de matrices [347-349], así como las soluciones propuestas para alcanzarlo.

3.2. Origen, justificación y utilidad del aporte

A continuación se describe el origen y la cronología del aporte.

En el año 2006 y con objeto de la primera revisión crítica del diseño de la misión y observa-torio SAC-D/Aquarius (ver Fig.3.1), el cual se trata de una plataforma satelital que porta 8 instrumentos de teleobservación de la tierra, se reunieron en Buenos Aires 120 científicos, ingenieros y técnicos, además de miembros de la Comisión Nacional de Actividades Espaciales (CONAE) y la National Aeronautics and Space Administration (NASA), están presentes miembros de la Agencia Spaziale Italiana (ASI), Centre National d"Etudes Spatiales (CNES), Canadian Space Agency (CSA), Instituto Nacional de Pesquisas Espaciais (INPE), los responsables de componentes e instrumentos tanto de la CONAE como de Comisión Nacional de Energía Atómica (CNEA), Universidad Nacional de la Plata (UNLP) y Centro de Investigaciones Opticas (CIOP) dependiente del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), además de miembros de Investigación Aplicada S.E. (INVAP) como contratista principal de la fabricación del satélite.

Los 8 instrumentos generan un gran volumen de datos a ser almacenados hasta que la plataforma pasa sobre la Estación Terrena de Córdoba (ETC), por lo cual el almacenamiento y procesamiento a bordo se hace inmanejable sin un eficiente sistema de compresión y almacenamiento.

Surge la propuesta de emplear un algoritmo similar de compresión inter-cuadro (entre las bandas multi e hiper-espectrales) [69-71, 156, 157, 159], e intra-cuadro (entre los bloques a ser dividida cada banda) con objeto de simplificar y optimizar el proceso de embebido de software en una única FPGA (del inglés ***Field Programmable Gate Array***) de a bordo.

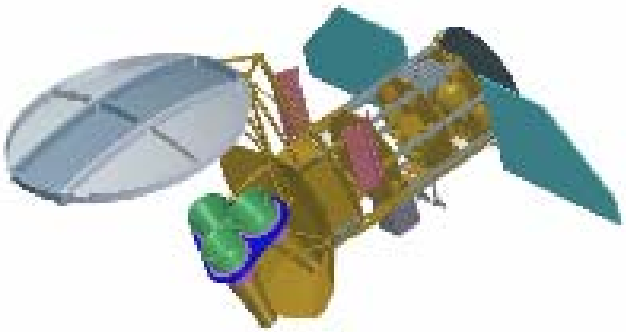


Figura 3.1: Esquema del satélite Aquarius/SAC-D (gentileza CONAE).

Dichas bandas multi e hiperespectrales se presentan ordenadas según la disposición típica que muestra la Fig.3.2. Un ejemplo muy ilustrativo se encuentra en la Fig.3.3 donde las bandas son numeradas con un aumento de la longitud de onda de visible a infrarrojo.

Dichas bandas deben ser descorrelacionadas mediante la TDKL en un proceso similar al mostrado en la Fig.3.3, donde las columnas de la matriz **V** son los autovectores de la matriz de autocovarianza asociada con el conjunto de de bandas mutiespectrales de la Fig.3.3.

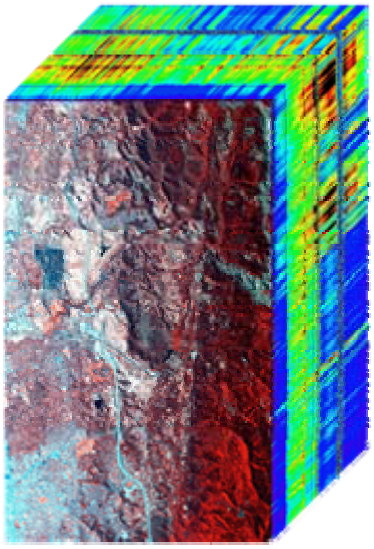


Figura 3.2: Imágenes satélites multi e hiperespectrales.

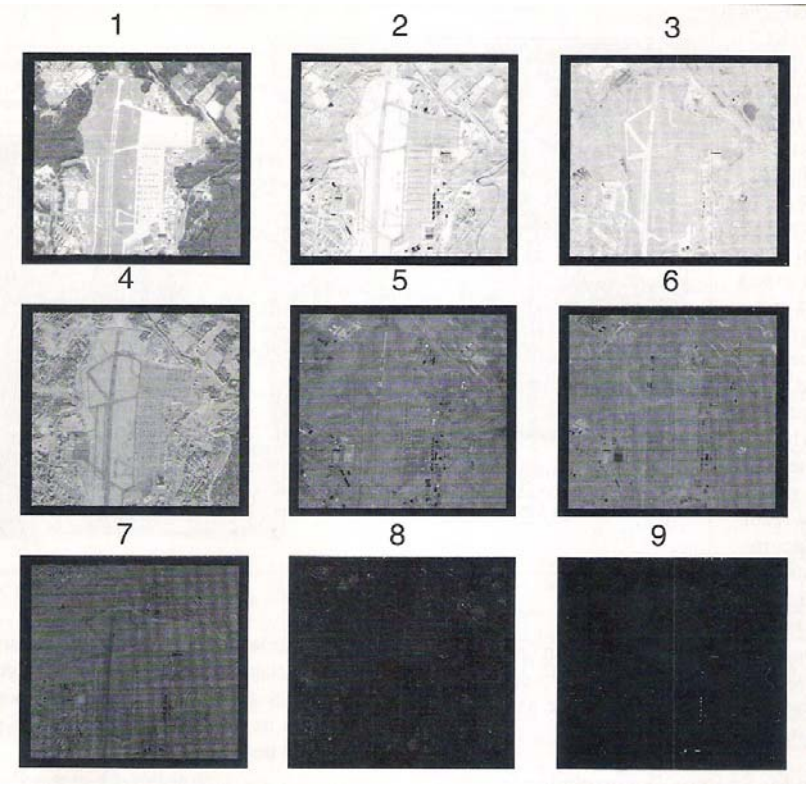


Figura 3.3: Bandas multiespectrales numeradas de menor a mayor longitud de onda (visible al infrarrojo).

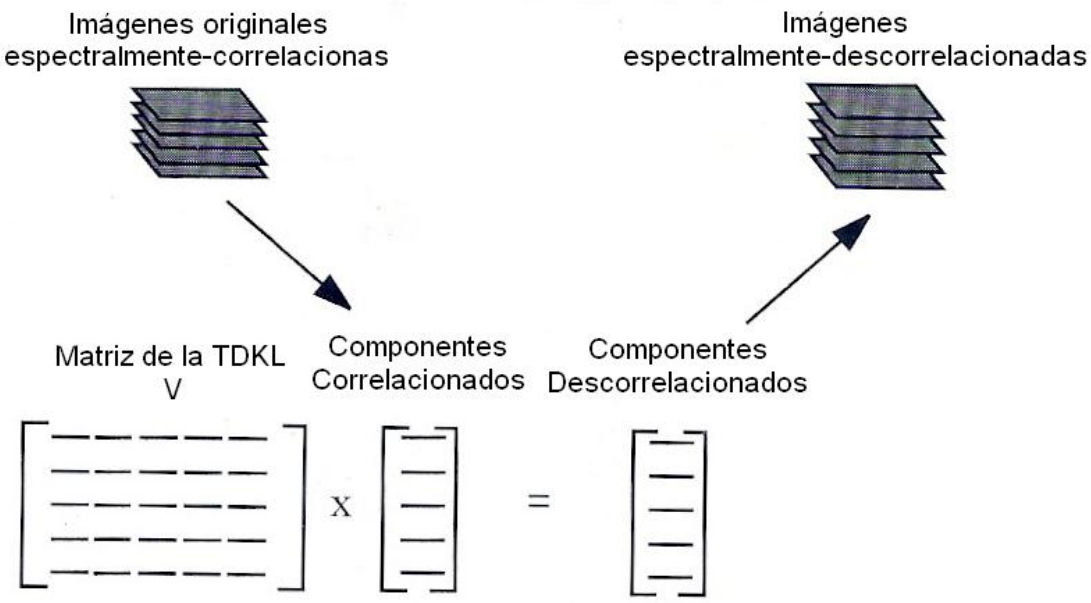


Figura 3.4: Proceso de descorrelación de las bandas multiespectrales mediante la TDKL.

La fuerte correlación interbanda se refleja en la matriz de coeficientes de correlación de la Fig.3.5, la cual expone el solapamiento de información entre bandas (todas contra todas) [69-71, 156, 157, 159].

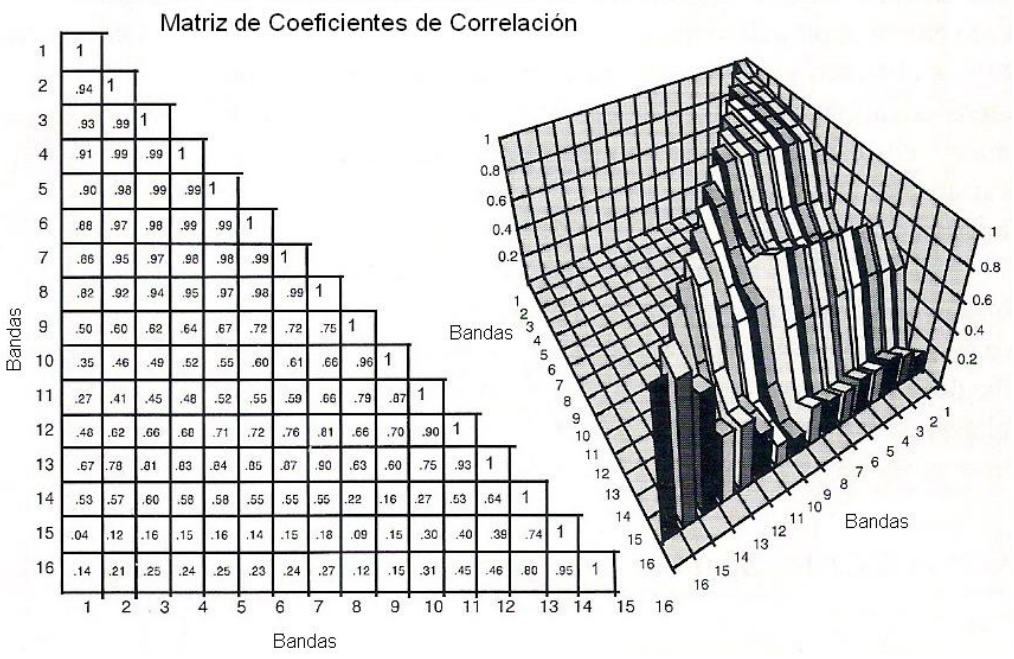


Figura 3.5: La Matriz de Coeficientes de Correlación.

El proceso descripto de descorrelación interbanda se encuentra inscripto en el codificador de compresión del paquete multiespectral del algoritmo de Tescher [156, 157, 159] y Pentland [45,

51, 78, 97, 98, 322], donde la Fig.3.6 nos muestra el codificador, mientras que la Fig.3.7 nos muestra el decodificador [69-71, 156, 157, 159].

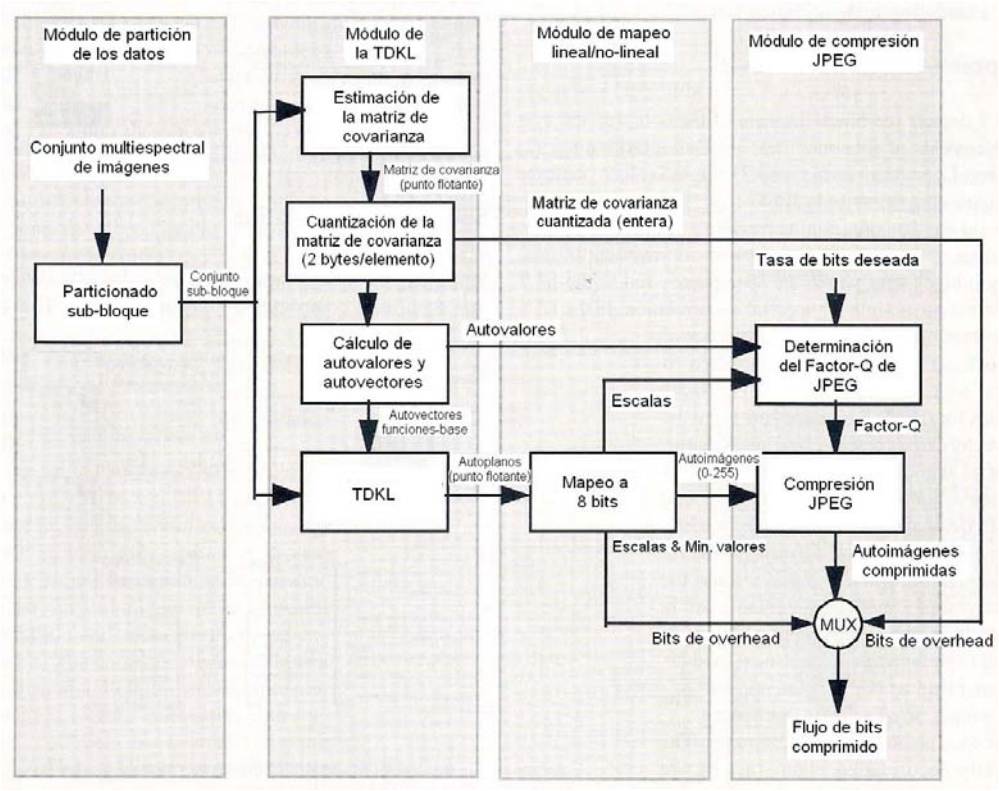


Figura 3.6: Codificador multiespectral.

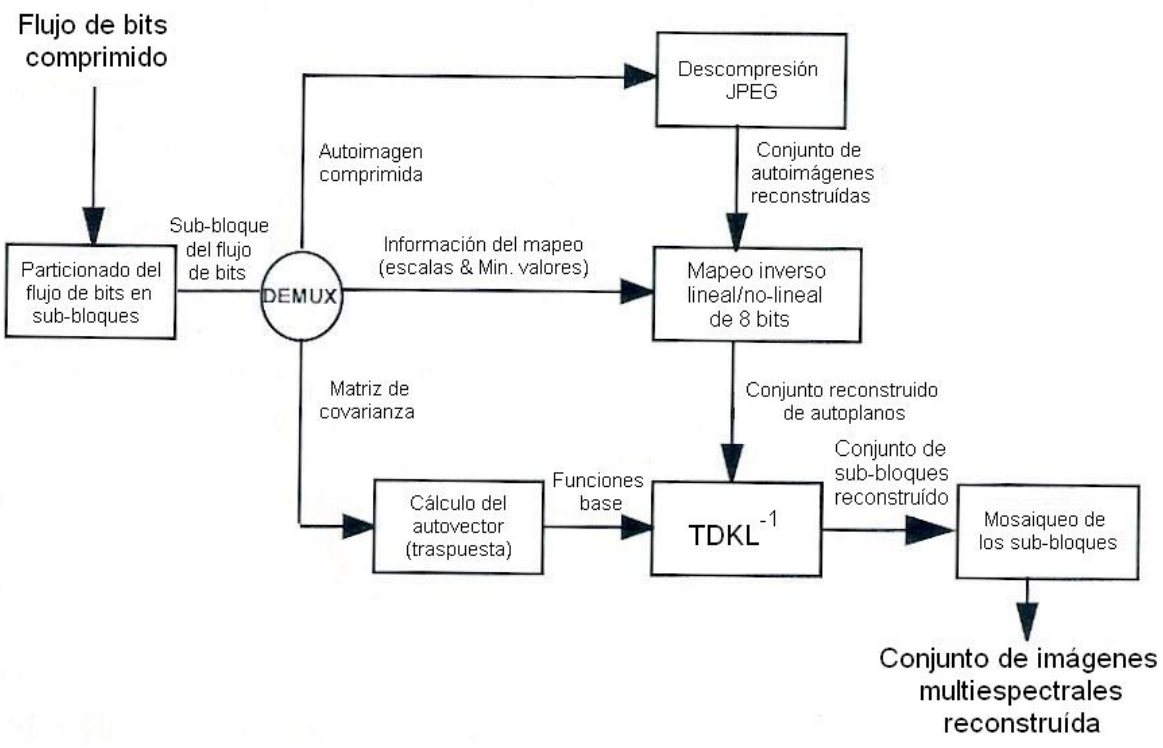


Figura 3.7: Decodificador multiespectral.

En la Fig.3.6 podemos observar que el segundo módulo se corresponde con la aplicación de la TDKL vista en el Capítulo 1, mientras que el cuarto módulo con la aplicación del algoritmo JPEG (ver Apéndice D) a cada banda. Obviamente, la diferencia tan evidente entre la técnica de descorrelación basada en TDKL frente al algoritmo JPEG complica considerablemente la implementación tanto del codificador como del decodificador en un sistema de software embebido. Finalmente, y ante el mencionado problema se opta por tratar cada banda por separado según las siguientes opciones:

a. No se las comprime
b. Se las comprime con JPEG [469-484] o JPEG2000 [485-506]
c. Se las comprime mediante algún algoritmo propietario basado en onditas [169-332].

3.3. Ubicuidad del aporte en relación a la TDKL

Existen tres líneas de investigación actualmente pendientes acerca de TDKL según Tescher y Pentland (responsables del procesamiento de imágenes multi e hiperespectrales del Jet Propulsion Laboratory (JPL) de la NASA:

1. Mejorar el algoritmo de Jacobi, es decir, menor costo computacional, para obtener una versión rápida generalizada de la TDKL. Decimos generalizada, por cuanto Anil K. Jain [309, 151, 307, 36, 46] trabajó en un caso particular de la versión rápida de la TDKL para imágenes muy ergódicas. ***Es decir, el problema sigue abierto.***

2. TDKL distribuída, la cual permitiría separar el kernel de la transformación en sub-kerneles y enviar cada uno a un hilo diferente de procesamiento distribuido. Actualmente Martin Vetterli [222, 223, 297, 301, 333] está trabajando en una aproximación a un espacio lineal equivalente, es decir, logra descorrelacionar, pero con autovalores (y por ende autovectores) distintos de la transformación original. ***Es decir, el problema sigue abierto.***

3. KLT eficiente aplicada a una imagen aislada con mosaicos grandes (menor costo computacional).

El punto 3 representa el aporte de este trabajo.

3.4. Clasificación de las métricas

La clasificación de las distintas métricas, así como la posición donde recoger los datos para su cálculo, nos permitirán:

a. Identificar el mejor tipo de grilla de sub-bloques con mayor nivel de SMEP frente a una camada de distintos posibles, obtenidas mediante distintas técnicas.

b. Tratar de entender como identificar con más precisión a dicha población a los efectos de que tal cosa nos permita individualizar la métrica más apropiada a ser aplicada a los bloques de una imagen con objeto de establecer si los mismos poseen buen nivel de SMEP antes de aplicar las configuraciones de codificación y decodificación en base a la TDKL. Es decir, saber si un tipo de grilla posee buen nivel de SMEP casi sin costo computacional alguno.

3.4.1. Ambito de revelación del SMEP en base al análisis de métricas de posición

La Fig.3.8 representa el diagrama de identificación de la mejor grilla basado en métricas. En este contexto estableceremos tres familias bien diferenciadas de métricas según su posición, a saber:

a. Métricas de fondo (MSE y PSNR)

b. Métricas intermedias (basadas en los autovalores)

c. Métrica de entrada (aquella que deseamos encontrar)

Como podemos observar en la Fig.3.8, se van presentando diferentes tipos de grillas (basadas en distintas técnicas) en las cuales parcelar una misma imagen, e ir evaluándolas primero con las métricas de fondo, es decir, buscamos aquella grilla que produzca el menor MSE y el mayor PSNR entre la imagen original y la recuperada de todo el proceso de codificación/decodificación. La cual (para un mismo porcentaje de poda de los autovalores) coincide con aquella grilla para la cual el espectro normalizado de autovalores cae a cero más rápidamente. La grilla ganadora se identifica en la Fig.3.8 mediante un asterisco.

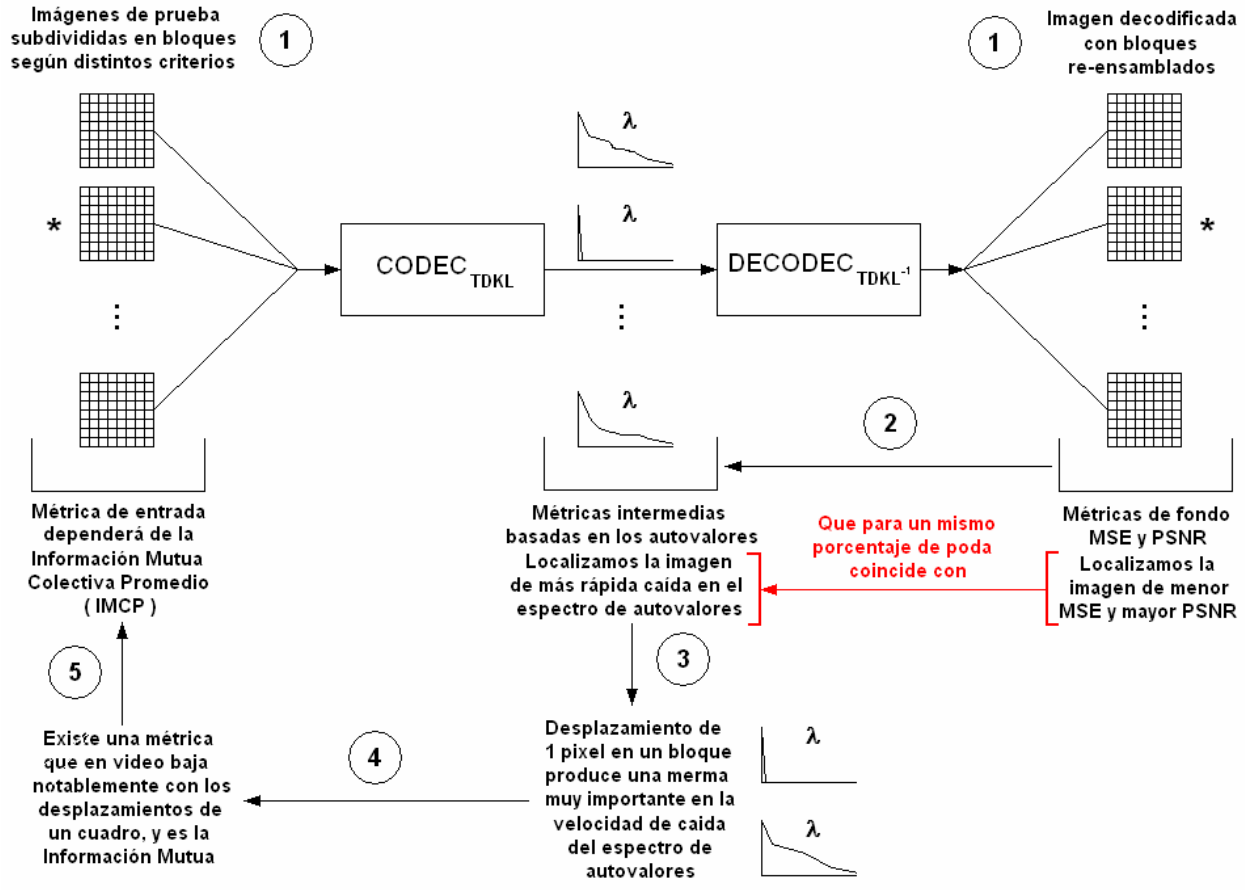


Figura 3.8: Diagrama para la identificación de la mejor grilla basada en métricas.

Ahora bien, lo que queremos obtener con este análisis es una métrica a la entrada del diagrama de la Fig.3.8, es decir, una métrica tal que aplicada a la grilla nos diga si posee un buen nivel de

SMEP antes de aplicar el proceso de codificación/decodificación. Cuanto más cerca de la entrada éste dicha métrica (cuanto antes se la aplique), menor costo computacional habrá que pagar para saber si el tipo de grilla obtenido posee buen SMEP con todo lo que ello conlleva.

3.4.2. La métrica para SMEP

Una pista que nos permitirá identificar a la métrica de entrada insinuada en la sección anterior se basa en el espectro normalizado de autovalores aplicado entre codificación y decodificación. Si en el caso de una muy buena grilla (alto SMEP) desplazamos un mosaico un solo pixel a la derecha o hacia abajo veremos que el espectro normalizado de autovalores se degrada significativamente. Otro ejemplo del ámbito de imágenes donde se dá un fenómeno parecido, es en el caso de video, donde cuadros con alta redundancia al sufrir el desplazamiento de un solo pixel en uno de ellos degradan notablemente a la métrica que se usa en dichos casos, siendo dicha métrica la Información Mutua [322-327], la cual puede verse en detalle en la Sección 2.2.1.6.2 del Capítulo 2.

Lo expuesto anteriormente nos dá un indicio acabado de que tipo de métrica estamos buscando, dado que la Información Mutua atañe a solo dos bloques entre si, nosotros necesitamos una métrica con simulares atributos pero para todo el conjunto de mosaicos tomados de a dos. La métrica que se necesita es la Información Mutua Colectiva Promedio (IMCP) y que mostramos a continuación:

$$\boldsymbol{IMCP} = \frac{1}{\left(\dfrac{\boldsymbol{nm}!}{(\boldsymbol{nm}-2)!\,2!} \right)} \sum_{\boldsymbol{i}=1}^{\boldsymbol{nm}-1} \sum_{\boldsymbol{j}=\boldsymbol{i}+1}^{\boldsymbol{nm}} \boldsymbol{I}(\boldsymbol{b}_{\boldsymbol{i}};\boldsymbol{b}_{\boldsymbol{j}}) \tag{3.1}$$

3.5. Formalización del atributo

Los indicios del atributo nos llevan a formularnos una serie de preguntas en pos de formalizarlo

En qué consiste?

1. En una Similitud Morfológica Equi-Posicional (SMEP) de los bloques.
2. En una elevada Información Mutua entre bloques.

En qué casos se dá?

Casos extremos (rango = 1)
1. Para bloques de 1x1 pixeles, pero con un inadmisible costo computacional.
2. En bandas multi e hiperespectral (no es el caso de monocuadro).

Casos intermedios
3. A través de una transformada que haga algo parecido a Haar.

Qué consecuencias acarrea su presencia?

1. Mayor eficiencia de decorrelación
   1.1. Menor MSE y mayor PSNR
   1.2. El espectro de autovalores tiende más rápidamente a cero con mosaicos más grandes, lo que origina que un pequeño porcentaje de los primeros autovalores (los más grandes) se lleven el mayor porcentaje de la traza ofreciendo más eficiencia en la poda, mayor tasa de compresión y mayor energía visual conservada.
   1.3. Mayor Información Mutua intermosaicos.

2. Menor costo computacional al ser los mosaicos más grandes y por ende de menor dimensión la matriz de aurocorrelación

3.6. Similitud Morfológica Equi-Posicional (SMEP)

De la Sección 1.2.5 sabemos que la forma de aumentarle el rendimiento de decorrelación a la TDKL es mediante:

1. Disminución del tamaño de los mosaicos con el consiguiente e inadmisible aumento de la CC.

2. Disminución del tamaño de los mosaicos y la aplicación de una aproximación supuestamente rápida de la TDKL, lo cual, como se puede apreciar en el Apéndice C no es recomendable por varias razones.

3. Dejar los mosaicos con un tamaño razonablemente grande y aplicando alguna técnica que permita explotar los atributos interbloques vistos en la Sección 1.2.5.3, los cuales nos indican que cuando los mosaicos tienen una similitud morfológica y equi-posicional el espectro de autovalores acelera su tendencia a cero.

Para este último, en el caso límite, si la tendencia a cero fuera tan pronunciada que solo el primer autovalor fuera distinto de cero, lo razonable sería pensar que todos los mosaicos son idénticos al primero multiplicados por un escalar. Es decir, si aplicando una apropiada exploración sobre los pixeles de los mosaicos estos se convirtieran en filas de una matriz de representación $\boldsymbol{X}$, las mismas deberían ser (en principio) linealmente dependientes para asegurar lo dicho, y en ese caso en particular de rango 1. Su contraparte, justifica lo visto en la Sección 1.2.5.1, dado que si deseáramos dejar grande el tamaño de los bloques sin aplicar la técnica insinuada nos encontraríamos que (en general) no existe ningún escalar que multiplicado por un mosaico nos dé otro mosaico, ver Fig.1.16. Esto nos indica que en este caso dichas filas son linealmente independientes, que es el caso común de aplicación de la TDKL [350-354].

3.6.1. Definición de SMEP

SMEP es el atributo mediante el cual, el rendimiento de decorrelación de la TDKL que es aplicada sobre un grupo de mosaicos de una imagen, depende de la similitud morfológica equiposicional del contenido de los mismos. En el límite, el SMEP será máximo cuando habiéndose explorado los pixeles de dichos mosaicos para convertirlos en filas de una matriz de representación $\boldsymbol{X}$, las filas de esta (vectores fila) sean linealmente dependientes, siendo el $rango(\boldsymbol{X}) = 1$. Como se verá en la Sección 3.2.3 esto provocará que el único autovalor distinto de cero sea el primero, obteniéndose de esta manera una matriz $\boldsymbol{C}_y$ altamente rala.

3.6.2. Distintos niveles de SMEP

El siguiente ejemplo se basa en aquel presentado en la Fig.1.17 de la Sección 1.2.5.3. La diferencia reside en la nueva grilla (3), la cual, aunque también como la del caso de la Fig.1.17(b) posee SMEP, la de la Fig.3.9 en cambio lo hace para el caso límite en el cual todos los bloques son una manifestación del primero multiplicado por un escalar, donde dicho escalar es

$$0 < escalar < 1 \qquad (3.2)$$

Por lo tanto y como puede observarse en la Fig.3.10 (la cual representa el espectro normalizado de autovalores para el caso con y sin SMEP de la Fig.1.17 y el nuevo con SMEP extremo de rango = 1) para la Grilla 3 sólo el primer autovalor es distinto de cero, lo que como se verá en la siguiente sección el rango de la matriz de representación asociada es 1.

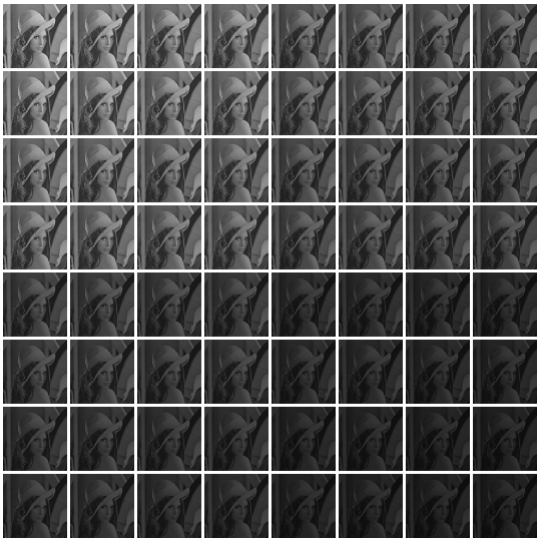


Figura 3.9: Grilla para bloques con SMEP extremo (rango 1).

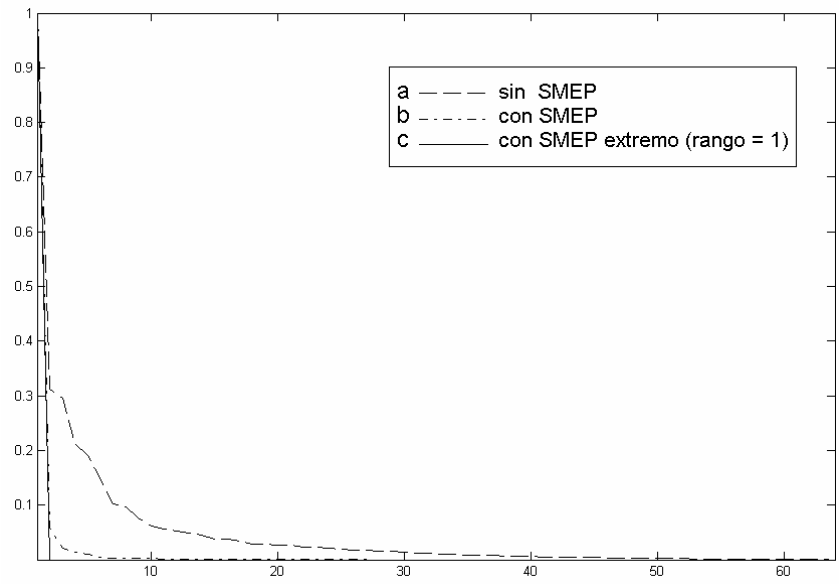


Figura 3.10: Autovalores normalizados contra el primero: a) Grilla 1: sin SMEP, b) Grilla 2: con SMEP, y c) Grilla 3: con SMEP extremo (rango 1).

Otras grillas que producen buen SMEP a partir de un sistema generador por filas, el cual será desarrollado a partir de la Sección 3.2.4 se presentan en la Fig.3.11. Obsérvese la similitud de los bloques en morfología y posición. Su espectro normalizado de autovalores es similar al de Haar.

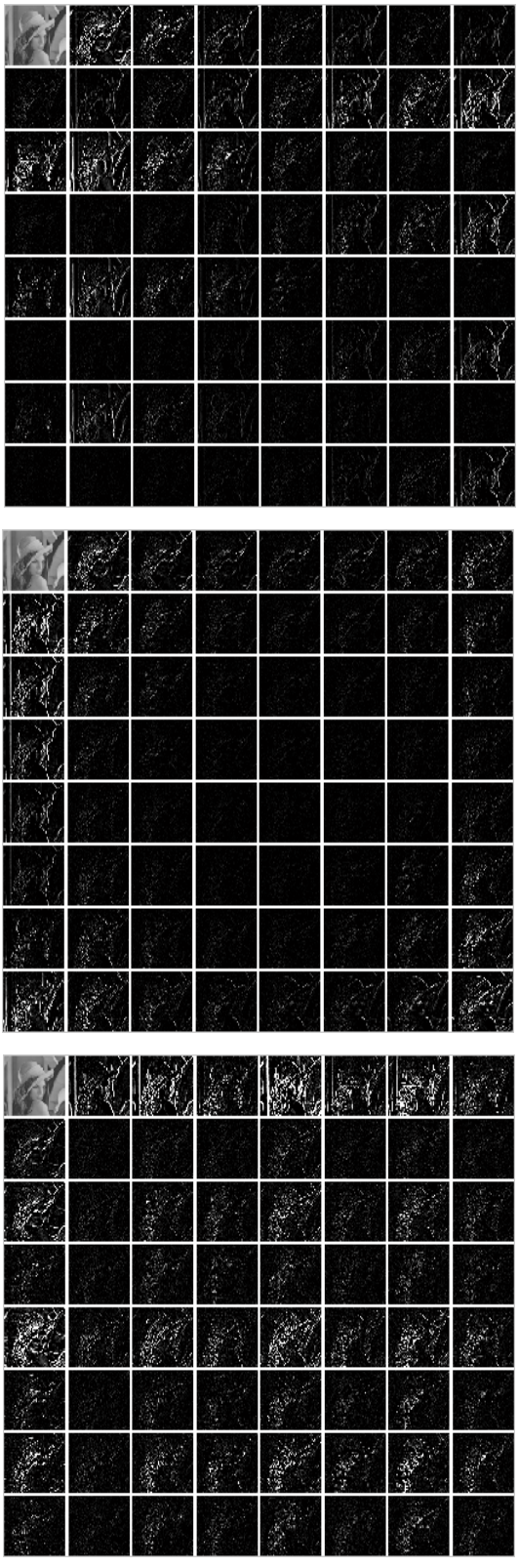


Figura 3.11: Grillas para bloques con buen nivel de SMEP para: a) TDSC, b) TDH, y c) TWH.

3.6.3. Propiedad L

Esta propiedad recibe su nombre por la forma que toma el espectro de autovalores (absoluto o normalizado) de la Fig.3.10(c), el cual es el de una letra l mayúscula o "L".

Dada la matriz de representación $\boldsymbol{X} \in \Re^{nm\times(nfm*ncm)}$, asociada a los mosaicos de la Fig.3.9, y siendo la matriz simétrica $\boldsymbol{C}_{\mathbf{x}} \in \Re^{nm\times nm}$ su matriz de autocorrelación, así como la matriz simétrica y diagonal $\boldsymbol{C}_{\mathbf{y}} \in \Re^{nm\times nm}$ la matriz resultante de la aplicación de la TDKL sobre las filas de la matriz $\boldsymbol{X}$, podemos enunciar la siguiente propiedad:

> *rango*($\boldsymbol{X}$) = *rango*($\boldsymbol{C}_{\mathbf{x}}$) = *rango*($\boldsymbol{C}_{\mathbf{y}}$) = *número de autovalores distintos de cero* [364-379].

Esto quiere decir, que en el caso de la Fig.3.9 al ser todos sus mosaicos una manifestación del primero multiplicado por un escalar real, las filas correspondientes en la matriz de representación asociada $\boldsymbol{X}$ serán todas linealmente dependientes entre sí, por lo que el rango de $\boldsymbol{X}$ será 1, al igual que el de $\boldsymbol{C}_{\mathbf{x}}$ y el de $\boldsymbol{C}_{\mathbf{y}}$, dando lugar a que sólo el primer autovalor de $\boldsymbol{C}_{\mathbf{y}}$ sea distinto de cero, como puede observarse en la Fig.3.10(c). Esta propiedad es el corazón del SMEP, aunque como mencionamos oportunamente resulta ser su caso límite. Toda aproximación que permita disponiendo de una inversa, un caso intermedio entre el límite (rango = 1) y la tradicional e ineficiente aplicación de la TDKL será muy bienvenida.

Otras propiedades importantes para el caso límite son:

> *traza*($\boldsymbol{C}_{\mathbf{x}}$) = *traza*($\boldsymbol{C}_{\mathbf{y}}$) = λ_1 (toda la energía visual se concentra en el primer sub-bloque)
> *determinante*($\boldsymbol{C}_{\mathbf{x}}$) = *determinante*($\boldsymbol{C}_{\mathbf{y}}$) = 0
> *rango*($\boldsymbol{X}$) = *rango*($\boldsymbol{C}_{\mathbf{x}}$) = *rango*($\boldsymbol{C}_{\mathbf{y}}$) = 1

3.6.4. Sistema generador por filas

En pos de obtener una aproximación al SMEP límite de la sección anterior (rango = 1), formulamos una herramienta que nos permitirá acceder convenientemente a las soluciones propuestas de la Sección 3.2.6.

A continuación definimos tres sistemas generadores por filas, todos en sintaxis de MATLAB® y basados en:

a) TDC-1D
b) TDH-1D
c) TWH-1D

a) El primero consiste en la aplicación directa de la TDC-1D vista en la Sección 2.2.1.1.
En MATLAB®: G = *dctmtx*(*nfG*), con $G \in \Re^{nfG\times ncG}$, siendo *nfG* = *ncG.*

b) El segundo consiste también en la aplicación directa pero de la TDH vista en la Sección 2.2.4.

Esta no será una función built-in de MATLAB® como la anterior por lo que deberemos desarrollarla nosotros mismos:

```
function G = tdhmtx(nfG)

for nf = 0:nfG -1
    for nc = 0:nfG -1
        G(nf+1, nc+1) = (cos(2*pi*nf*nc/nfG) + sin(2*pi*nf*nc/nfG))/nfG;
    end
end
```

c) El tercero consiste en la aplicación directa de la TWH-1D vista en la Sección 2.2.3.
En MATLAB®: G = *hadamard*(*nfG*)

3.6.5. Máscara convolutiva

En base a los sistemas generadores por filas desarrollados en la sección anterior, formulamos una herramienta que nos permitirá completar las necesidades de las combinaciones que siguen. La misma se basa en la convolución bidimensional o convolución por máscara [371-398] entre una máscara ***M*** que se va desplazando de izquierda a derecha y de arriba hacia abajo sobre una imagen ***I***, y esta última. Dicha convolución proviene del modelo de filtro basado en la teoría de Bayes, para lo cual el filtrado de una imagen *I* corrupta por ruido blanco Gaussiano es considerada como,

$$I = b + n \tag{3.3}$$

donde *n* es el ruido Gaussiano independiente, y necesitamos estimar la imagen deseada *b* (con el menor ruido posible) lo mejor que se pueda, según algún criterio. Este es un problema clásico en Teoría de la Estimación. Nuestro problema se reduce a estimar *b* a partir de la observación ruidosa *I*. Para dicho propósito emplearemos el estimador *maximum a posteriori* (*MAP*). Tal estimador ha sido ampliamente utilizado en problemas de restauración y reconstrucción de imágenes, derivando apropiadamente sus funciones de densidad de probabilidad (FDP). Modelos invariantes e invariantes en el tiempo son discutidos en [311] y que empleamos en el presente trabajo de manera tal de deducir el estimador MAP más apropiado para nuestros propósitos.

El estimador *MAP* clásico para (3.3) es

$$\hat{b}(I) = \arg\max_{b} p_{b|I}(b|I) \tag{3.4}$$

Usando la regla de Bayes obtenemos

$$\begin{aligned} \hat{b}(I) &= \arg\max_{b} [\, p_{I|b}(I|b) \,.\, p_{b}(b) \,] \\ &= \arg\max_{b} [\, p_{n}(I - b) \,.\, p_{b}(b) \,] \end{aligned} \tag{3.5}$$

Por lo tanto, estas ecuaciones nos permitirán escribir esta estimación en función de la FDP del ruido (p_n) y la FDP de la imagen estimada (p_b). Por otra parte, partimos de la suposición de que el

ruido de FDP p_n es Gaussiano de valor medio nulo con varianza σ_n, i.e.,

$$p_{\mathrm{n}}(n) = \frac{1}{\sigma_{\mathrm{n}}\sqrt{2\pi}}.exp\left(-\frac{n^2}{2\,\sigma_{\mathrm{n}}^2}\right). \tag{3.6}$$

En la práctica, generalmente, se plantean dos problemas con el enfoque Bayessiano, los cuales surgen cuando es empleada una FDP $p_b(b)$ exacta pero complicada:

1) puede ser difícil estimar los parámetros de p_b para una imagen específica, especialmente a partir de datos ruidosos, y

2) los estimadores para estos modelos pueden no tener una solución de forma cerrada y simple, con la consiguiente dificultad en su obtención. La solución para estos problemas requiere usualmente técnicas numéricas [311].

Continuemos desarrollando el estimador *MAP* y exponerlo para los casos Gaussiano y Laplaciano. La Ecuación (3.5) es también equivalente a

$$\hat{b}(I) = \arg\max_{\mathrm{b}}\left[log(p_{\mathrm{n}}(I-b)+log(p_{\mathrm{b}}(b))\right] \tag{3.7}$$

Como en [311], definamos $f(b) = \log(p_b(b))$. Los cuales empleando (3.6), (3.7) se convierte en

$$\hat{b}(I) = \arg\max_{\mathrm{b}}\left[-\frac{(I-b)^2}{2\sigma_{\mathrm{n}}^2} + f(b)\right] \tag{3.8}$$

Esto es equivalente a resolver la siguiente ecuación para $\hat{b}$ si $p_b(b)$ si asumimos que es extrictamente convexa y diferenciable.

$$\frac{I-\hat{b}}{\sigma_{\mathrm{n}}^2} + f'(\hat{b}) = 0 \tag{3.9}$$

Si asumimos que $p_b(b)$ es una FDP de valor medio nulo con varianza σ^2, entonces la misma puede ser escrita como

$$p_{\mathrm{b}}(b) = exp\left(-\frac{\hat{b}^{\mathrm{T}}\left(M^{-1}-I_m\right)\hat{b}}{2\sigma_{\mathrm{n}}^2}\right) \tag{3.10}$$

donde M es una matriz que representa a la máscara de convolución y I_m es la matriz identidad, por lo cual

$$f(\hat{b}) = -\frac{\hat{b}^{\mathrm{T}}\left(M^{-1}-I_m\right)\hat{b}}{2\sigma_{\mathrm{n}}^2} \tag{3.11}$$

con

$$f'(\hat{b}) = -\frac{\left(M^{-1} - I_m\right)\hat{b}}{\sigma_n^2} \qquad (3.12)$$

Reemplazando (3.12) en (3.9), el estimador puede ser escrito como

$$\hat{b}(I) = M.I \qquad (3.13)$$

donde la Ecuación (3.13) representa una convolución 2D entre la máscara *M*, y la imagen ruidosa *I*. Matemáticamente podemos escribir la convolución como:

$$b_{u,v}(p,q) = \sum_{f=1}^{nfM} \sum_{c=1}^{ncM} M_{u,v}(f,c) I\big((p-1)nfM + f, (q-1)ncM + c\big) \qquad (3.14)$$

Mientras que su inversa es

$$I\big((p-1)nfM + u, (q-1)ncM + v\big) = \sum_{f=1}^{nfM} \sum_{c=1}^{ncM} M_{u,v}^{-1}(f,c)\, b_{f,c}(p,q) \qquad (3.15)$$

donde:

M = máscara convolutiva, $\boldsymbol{M} \in \Re^{nfM \times ncM}$
b = es el sub-bloque o mosaico que surge como resultado de la aplicación de la máscara ***M*** sobre la imagen $\boldsymbol{I} \in \Re^{nfI \times ncI}$
$p \in [1, nfI / nfM]$
$q \in [1, ncI / ncM]$
$f \in [1, nfM]$
$c \in [1, ncM]$
$u \in [1, nfM]$
$v \in [1, ncM]$

Tanto la convolución directa como la inversa son sin solapamiento o disjuntas.

A continuación, desarrollemos un ejemplo genérico a los efectos de explicar la génesis de la máscara en función del sistema generador por filas y la vinculación entre las distintas variables involucradas y sus dimensiones.

Considerando a las matrices generadas en la Sección 3.2.4, podemos observar que para una matriz $G \in \Re^{nfG \times ncG}$, tenemos

$$G = \left[\begin{array}{cccc} g(1,1) & g(1,2) & g(1,3) & g(1,4) \\ \hline g(2,1) & g(2,2) & g(2,3) & g(2,4) \\ \hline g(3,1) & g(3,2) & g(3,3) & g(3,4) \\ \hline g(4,1) & g(4,2) & g(4,3) & g(4,4) \end{array}\right] \rightarrow M = \left[\begin{array}{cc|cc} g(1,1) & g(1,2) & g(2,1) & g(2,2) \\ g(1,3) & g(1,4) & g(2,3) & g(2,4) \\ \hline g(3,1) & g(3,2) & g(4,1) & g(4,2) \\ g(3,3) & g(3,4) & g(4,3) & g(4,4) \end{array}\right] = \left[\begin{array}{cc} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{array}\right] \qquad (3.16)$$

$$M_{1,1} = \begin{bmatrix} g(1,1) & g(1,2) \\ g(1,3) & g(1,4) \end{bmatrix} \tag{3.17a}$$

$$M_{1,2} = \begin{bmatrix} g(2,1) & g(2,2) \\ g(2,3) & g(2,4) \end{bmatrix} \tag{3.17b}$$

$$M_{2,1} = \begin{bmatrix} g(3,1) & g(3,2) \\ g(3,3) & g(3,4) \end{bmatrix} \tag{3.17c}$$

$$M_{2,2} = \begin{bmatrix} g(4,1) & g(4,2) \\ g(4,3) & g(4,4) \end{bmatrix} \tag{3.17d}$$

Donde cada máscara tendrá dimensiones $ncM = \sqrt{ncG}$ y $nfM = \sqrt{nfG}$.

Podemos realizar un procedimiento similar para el caso de la obtención de las máscaras inversas, comenzando invirtiendo la matriz ***G*** y aplicando un procedimiento similar al descrito.

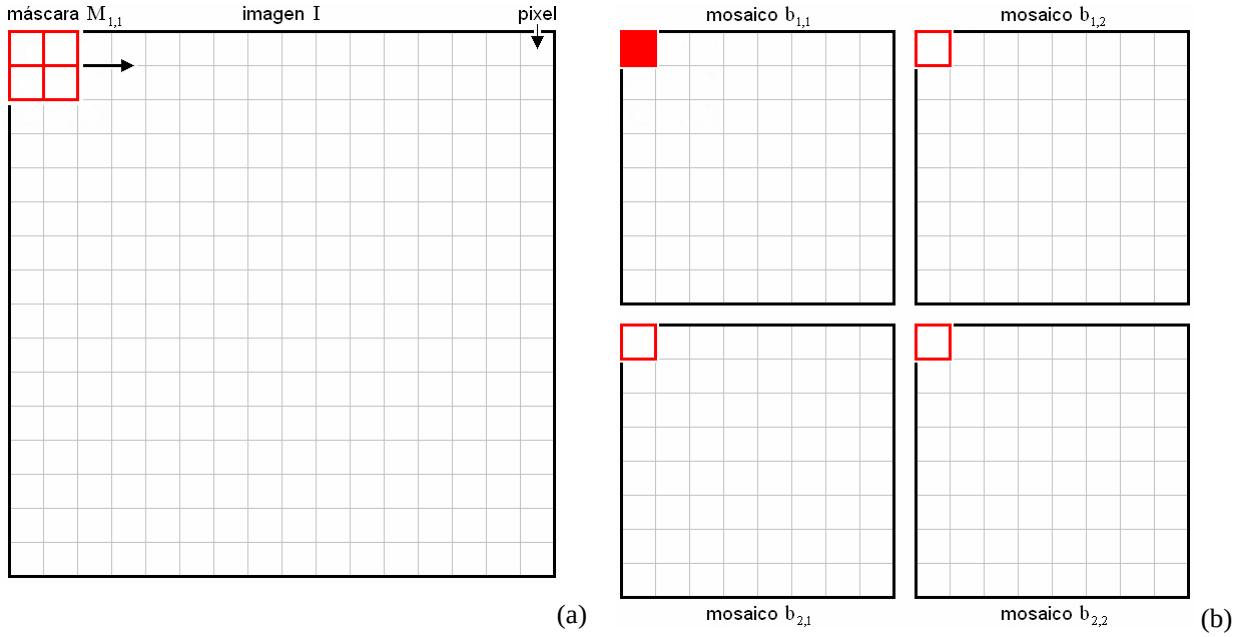


Figura 3.12: (a) Imagen y sobre ella la máscara de 2-*por*-2. (b) Mosaicos resultantes (2x2).

Como puede observarse en la Ecuación (3.16) las filas de la matriz *G* se exploran por filas para constituir los bloques de la matriz *M*, los cuales reciben el nombre de máscaras (Ecuación 3.16). Para estas mismas dimensiones, y aplicando la Ecuación (3.14) obtenemos los mosaicos de la Fig.3.12(b). La Fig.3.12(a) muestra la aplicación en barrido primero de izquierda a derecha y luego de arriba hacia debajo de cada máscara.

Si en cambio, la matriz *G* fuera de 16-*por*-16 obtendríamos como resultado los mosaicos de la Fig.3.13(b). La Fig.3.13(a) representa un esquema similar al de la Fig.3.12(a) pero para la máscara aludida.

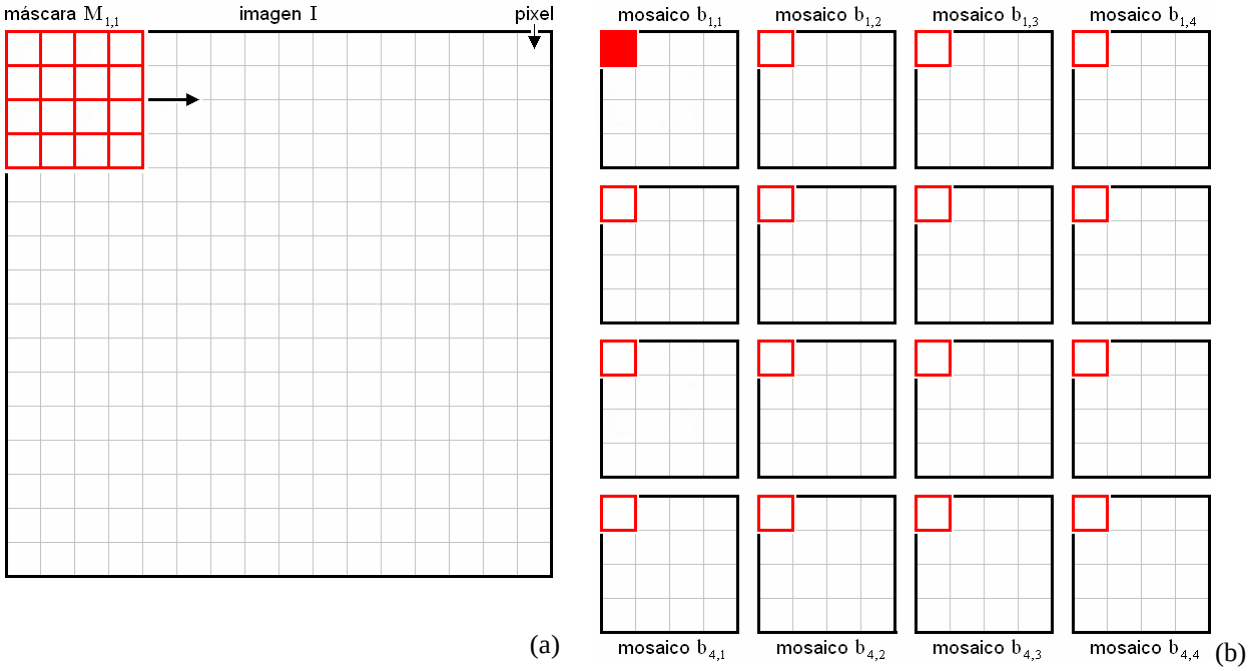


Figura 3.13: (a) Imagen y sobre ella la máscara de 4-*por*-4. (b) Mosaicos resultantes (4x4).

Obsérvese en ambas figuras que la acción de la máscara $M_{1,1}$ da lugar al pixel relleno del mosaico $b_{1,1}$. La acción de las máscaras consecutivas dará lugar a los pixeles remarcados pero no rellenos de los restantes mosaicos.

3.6.6. Soluciones propuestas

Dado que ninguna transformación con inversa puede ir de la matriz de representación ***X*** asociada a la grilla 1 de la Fig.1.17(a), a la matriz de representación ***X*** asociada a la grilla 3 de la Fig.3.9, deberemos tomar alguna solución de compromiso. En virtud de lo dicho, se genera un problema de recuperación de la imagen en el decodificador del receptor. Por lo tanto, lo único que podemos hacer es aplicar un método aproximado con SMEP que disponga de dicha inversa, para lo cual proponemos las siguientes combinaciones en base que:

a) TDC-2D genera cosas parecidas a partir de bloques diferentes,
b) TDO-2D genera sub-bandas morfológicamente parecidas y equi-posicionales (a pesar de la diferencia de frecuencia al ser aplicada a una imagen),
c) TDH-1D como sistema generador da lugar a algo similar a la TDC-2D
d) TWH-1D como sistema generador da lugar a algo similar a la TDO con base de Haar

Como podemos observar de la Fig.3.10, la Grilla 1 cuyos mosaicos no contienen SMEP es la empleada en una típica aplicación de la TDKL a una imagen monocuadro. En cambio las Grillas 2 y 3 poseen distintos grados de SMEP, siendo la última el caso extremo para rango = 1. No obstante, y como puede observarse, no existe tanta diferencia entre ambas, con la ventaja de que la Grilla 2 esta asociada a un rango = ***nm***, por lo cual podemos disponer de varios métodos que nos permitan pasar de la Grilla 1 a la Grilla 2 en forma práctica y con inversa. Dichos métodos, constituyen el corazón del presente trabajo.

Vamos a organizar las combinaciones propuestas en grupos, a saber:

Grupo A:

TDC-2D a toda la imagen + exploración zig-zag de los sub-bloques + TDKL sobre los sub-bloques.

TDC-2D a cada sub-bloque + exploración por filas de los sub-bloques + TDKL sobre los sub-bloques.

TDC-2D recursivamente generando cuadrantes + exploración de Morton de los cuadrantes (sub-bloques, ver Sección A.2) + TDKL sobre los sub-bloques.

Grupo B:

TDO-2D recursivamente generando cuadrantes + exploración de Morton de los cuadrantes (sub-bloques, ver Sección A.2) + TDKL sobre los sub-bloques.

TDO-2D recursivamente generando grilla + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.

Grupo C:

TDC-1D como sistema generador de sub-bloques + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.

TDH-1D como sistema generador de sub-bloques + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.

TWH-1D como sistema generador de sub-bloques + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.

3.6.7. Codificador y decodificar basado en SMEP

El diagrama en bloques (Fig.1.7) se complementa con la intervención del sistema de generación del SMEP en el codec y restaurador de SMEP en el decodec para la compresión de imágenes con pérdidas mediante la TDKL. La Fig.3.14 representa dicho diagrama en bloques.

Como una consecuencia natural de lo mencionado, surgen los algoritmos del codec y decodec, Algoritmo 3.1 y 3.2 respectivamente. En síntesis, el Algoritmo 3.1 representa la compresión con pérdidas en base a TDKL, el cual está compuesto por los primeros seis bloques de la Fig.3.14, es decir,

ALGORITMO 3.1

(a) División en bloques de la imagen
(b) Generador de SMEP
(c) TDKL
(d) Evaluación y poda
(e) Cuantización
(f) Compresión entrópica

Mientras que el Algoritmo 3.2 representa la descompresión con pérdidas en base a la $TDKL^{-1}$, el cual está compuesto por los últimos seis bloques de la Fig.3.14, es decir,

ALGORITMO 3.2

(a) Descompresión entrópica
(b) Decuantización
(c) Completar con ceros
(d) $TDKL^{-1}$
(e) Restaurador de SMEP
(f) Reconstrucción de la imagen

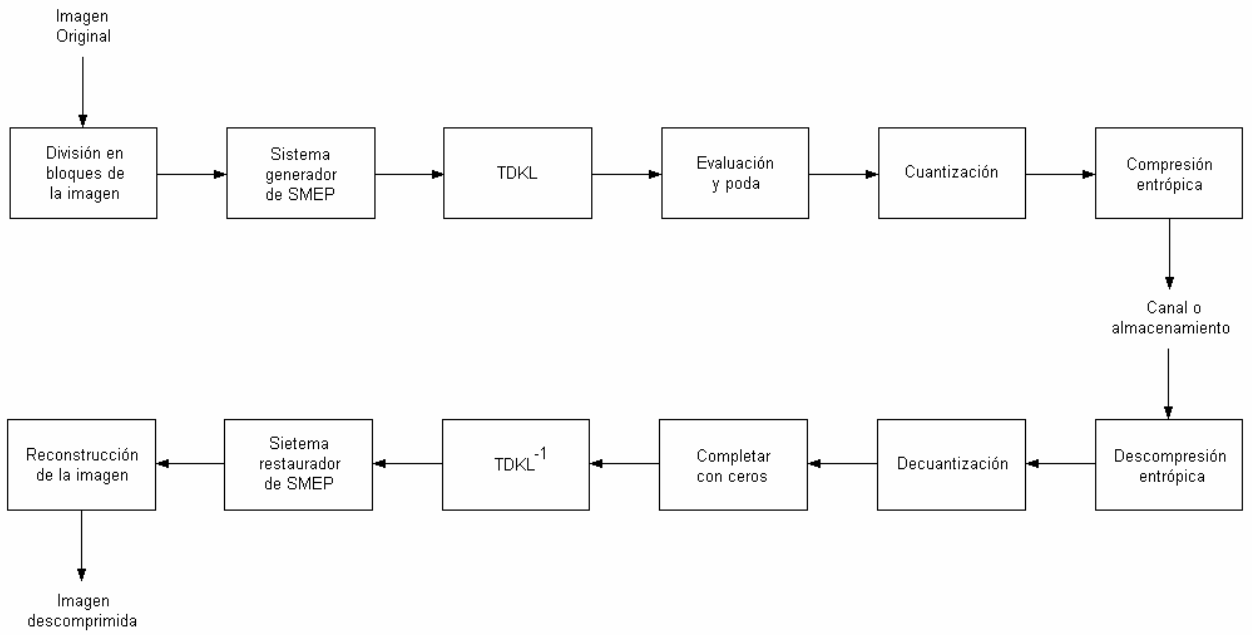


Figura 3.14: Diagrama en bloques de la compresión de imágenes con pérdidas mediante la TDKL empleando SMEP.

3.7 Conclusiones del capítulo

En este capítulo se presentaron las combinaciones propuestas, que permiten alcanzar un alto grado de SMEP con inversa, haciendo más eficiente la aplicación de la TDKL sobre una imagen monocuadro. Dichas combinaciones serán comparadas en el capítulo siguiente con las demás técnicas existentes sin SMEP a los efectos de evaluar las potencialidades de las primeras.

Capítulo 4

4. Simulaciones computacionales

4.1. Introducción

En este capítulo se presentan las simulaciones comparativas entre las técnicas propuestas con SMEP y las tradicionales sin SMEP. Además se realizará un grupo de simulaciones adicionales para probar el mejor exponente con SMEP vs JPEG y JPEG2000; estos últimos a modo de los mas sobresalientes representantes de aquellos ya posicionados en el acervo científico-tecnológico. En todos los casos, se evaluarán las distintas técnicas simuladas en base a las métricas oportunamente descriptas en el Capítulo 1.

4.2. Simulaciones computacionales

Las simulaciones en base a las técnicas provenientes del Capítulo 3 son:

Grupo A:

1. *TDC-2D a toda la imagen + exploración zig-zag de los sub-bloques + TDKL sobre los sub-bloques.*

2. *TDC-2D a cada sub-bloque + exploración por filas de los sub-bloques + TDKL sobre los sub-bloques.*

3. *TDC-2D recursivamente generando cuadrantes + exploración de Morton de los cuadrantes (sub-bloques) + TDKL sobre los sub-bloques.*

Grupo B:

4. *TDO-2D recursivamente generando cuadrantes + exploración de Morton de los cuadrantes (sub-bloques) + TDKL sobre los sub-bloques.*

5. *TDO-2D recursivamente generando grilla + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.*

Grupo C:

6. *TDC-1D como sistema generador de sub-bloques + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.*

7. *TDH-1D como sistema generador de sub-bloques + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.*

8. *TWH-1D como sistema generador de sub-bloques + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.*

Compararemos los tres grupos de combinaciones precedentes, con las siguientes:

Grupo D:

9. *Dividir recursivamente la imagen generando cuadrantes + exploración de Morton de los cuadrantes (sub-bloques) + TDKL sobre los sub-bloques.*

10. *Dividir recursivamente la imagen generando grilla + exploración por filas de los sub-bloques de la grilla + TDKL sobre los sub-bloques.*

Grupo E:

11. *TDO-2D recursivamente generando cuadrantes + exploración de Morton de los cuadrantes (sub-bloques) + poda típica de las sub-bandas de alta frecuencia de toda TDO-2D.*

12. *TDO-2D recursivamente generando grilla + exploración por filas de los sub-bloques de la grilla + poda típica de las sub-bandas de alta frecuencia de toda TDO-2D.*

Grupo F:

13. *TDC-2D a toda la imagen + umbralizado.*

14. *TDO-2D recursivamente a la sub-banda de baja frecuencia en cada nivel.*

15. *Dividir la imagen en sub-bloques + constituir la matriz 3D con dichos sub-bloques + TDKL sobre los sub-bloques.*

Grupo G:

16. *JPEG*

17. *JPEG2000*

Fundamentalmente, realizaremos las siguientes simulaciones:

a) Todas las técnicas que posean TDKL (con o sin SMEP) entre si: 1 al 10.

b) El mejor emergente de la simulación (a) vs. todas las técnicas que no posean TDKL: 11 a 14.

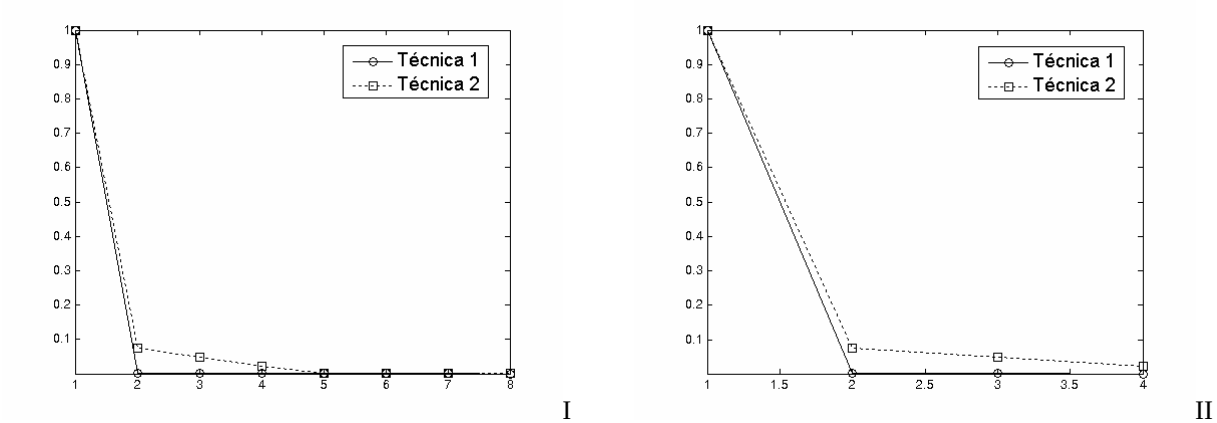
c) El mejor emergente de la simulación (a) vs. JPEG y JPEG2000: 16 y 17.

4.2.1. Primer grupo de simulaciones

Dado que este primer grupo de simulaciones se basa exclusivamente en técnicas que emplean TDKL (con y sin SMEP) se dispone de un mayor número de métricas compartidas para poder evaluar las ventajas comparativas entre las distintas técnicas. Como se muestra en la Tabla 4.I la mejor es ampliamente la Técnica 1 para todas las métricas empleadas. El empleo de la TDKL en forma tradicional (Técnica 15, es decir, solo) se puede apreciar en la Técnica 10, dado que son idénticos. Todas estas simulaciones se realizan sobre Lena en grises de 512x512 pixeles.

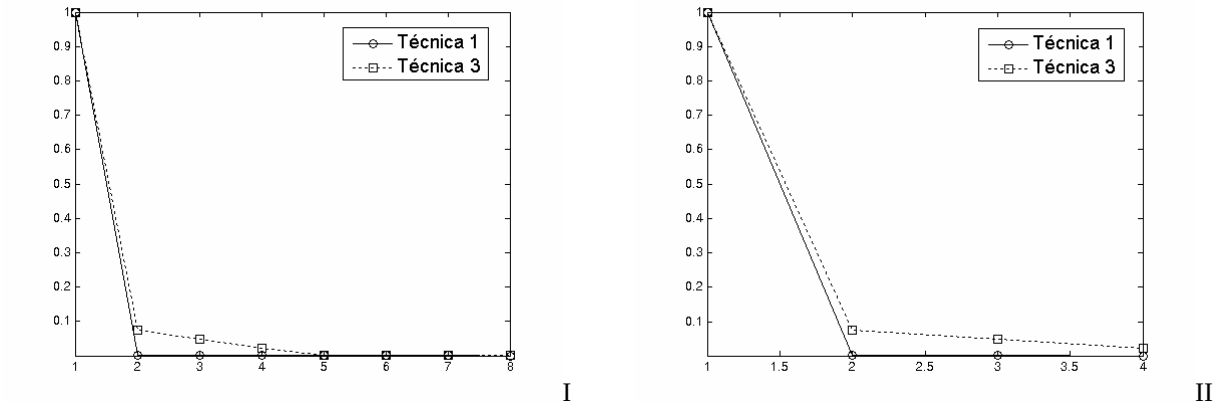
Tabla 4.I: Métricas vs Técnicas Empleadas

Técnica Empleada	Métricas									
	TC	bpp	MSE	PSNR	IMCP	E_{at}	E_{pa}	H_n	E_d	TEUCP
1	15.6935	0.0640	48.5315	31.2706	2.3071	1059100	99.7067	0.0149	0.0002	57.1250
2	15.6935	0.0640	52.4557	30.9329	2.1914	1059200	99.6830	0.1287	0.0024	56.1563
3	15.6935	0.0640	52.4557	30.9329	2.1914	1059200	99.6830	0.1287	0.0024	58.6406
4	15.6935	0.0640	65.2347	29.9860	2.0758	6867500	96.1092	0.1534	0.0022	59.5313
5	15.6935	0.0640	65.2347	29.9860	2.0758	6867500	96.1092	0.1534	0.0022	57.6563
6	15.6935	0.0640	65.2347	29.9860	1.9543	107300	96.1092	0.1534	0.0022	62.4219
7	15.6935	0.0640	65.2347	29.9860	1.9017	1676.6	96.1092	0.1534	0.0022	63.6719
8	15.6935	0.0640	65.2347	29.9860	2.0758	6867500	96.1092	0.1534	0.0022	63.0781
9	15.6935	0.0640	424.5221	21.8518	0.7723	62129	56.2694	0.6733	0.0355	58.4219
10	15.6935	0.0640	424.5221	21.8518	0.7723	62129	56.2694	0.6733	0.0355	57.3281



I II

Figura 4.1(a): AN para las técnicas 1 vs. 2, graficando para los I. 8 y II. 4 primeros sub-bloques.

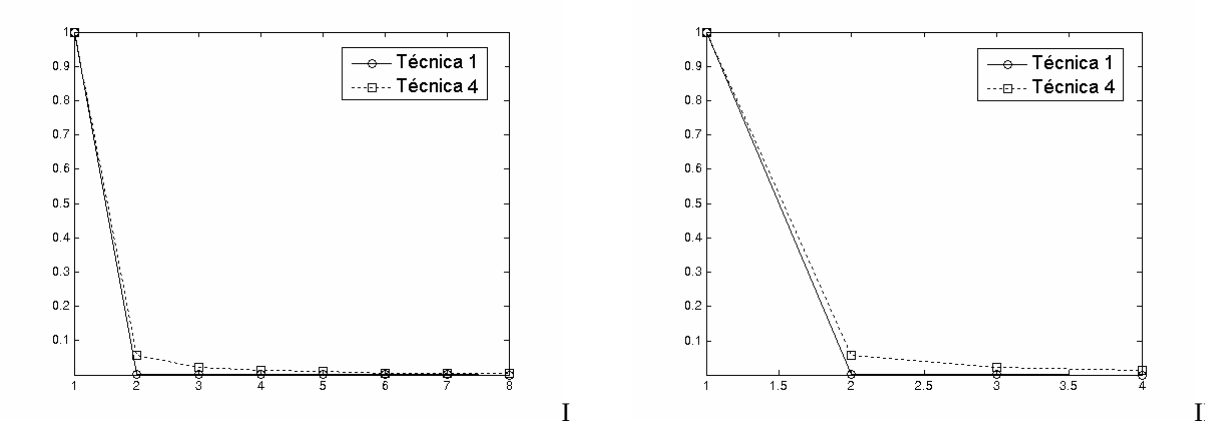


I II

Figura 4.1(b): AN para las técnicas 1 vs. 3, graficando para los I. 8 y II. 4 primeros sub-bloques

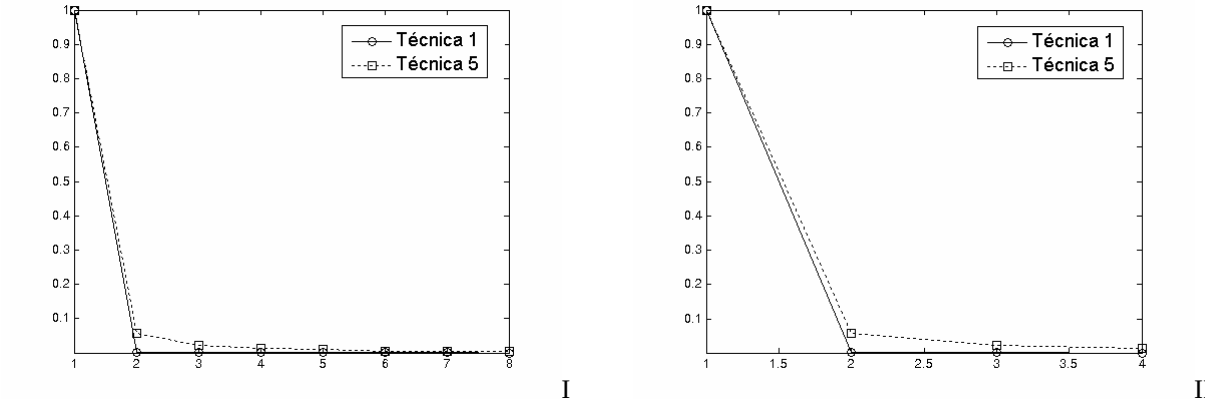
Aunque en este primer grupo de simulaciones la TPTB = 16 y el tamaño de los mosaicos se tomó en todos los casos igual a 64, la serie representada por la Fig.4.1 muestra el detalle de los primeros 8 y 4 autovalores normalizados de las distintas técnicas simuladas.

Como puede apreciarse en detalle, es la Técnica 1 la que posee mayor SMEP de todas, haciéndolo casi en el caso límite, pues casi solo el primer autovalor es distinto de cero.



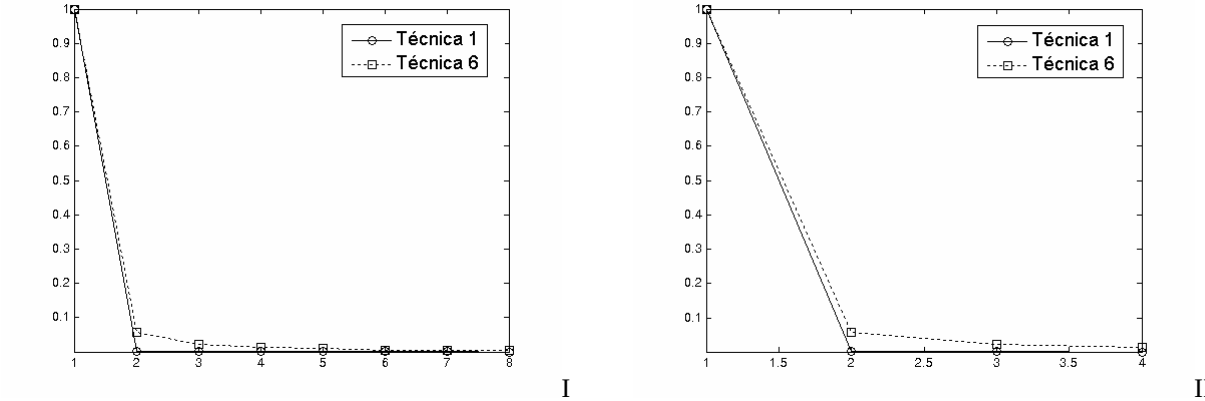
I II

Figura 4.1(c): AN para las técnicas 1 vs. 4, graficando para los I. 8 y II. 4 primeros sub-bloques



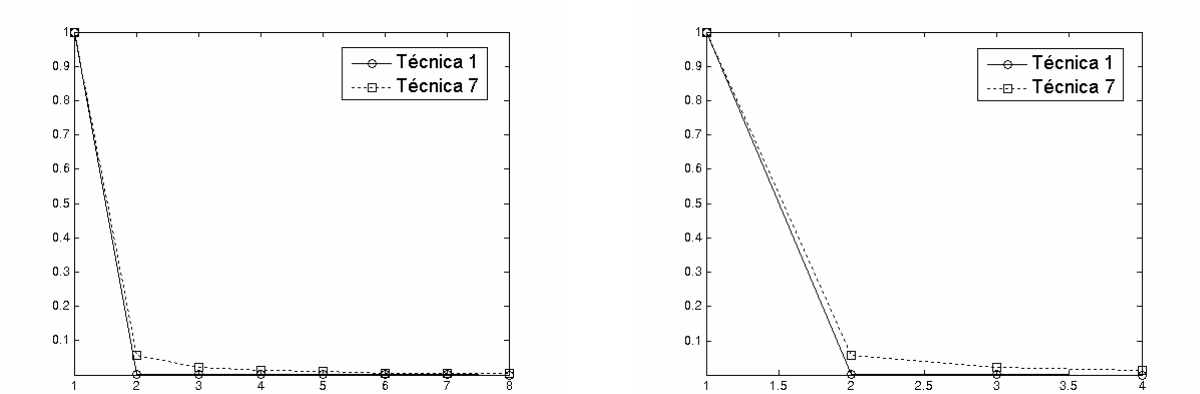
I II

Figura 4.1(d): AN para las técnicas 1 vs. 5, graficando para los I. 8 y II. 4 primeros sub-bloques



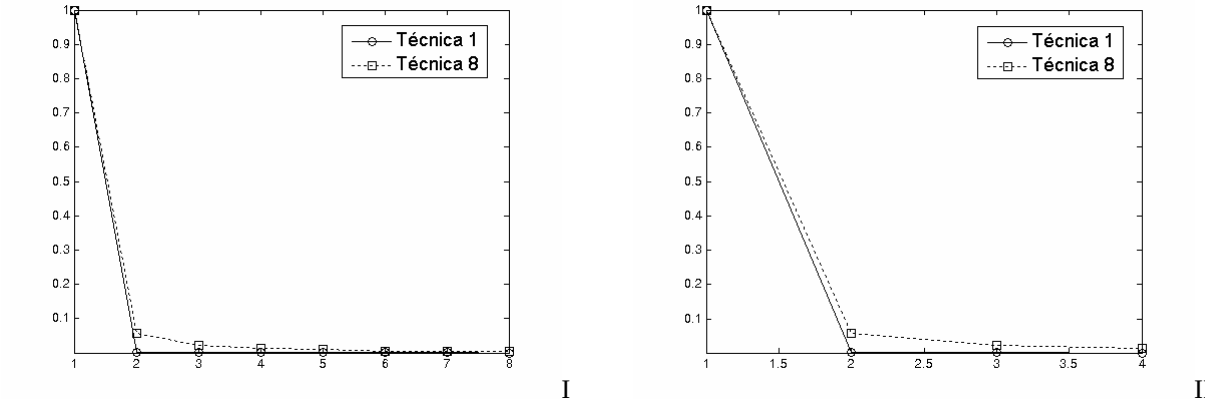
I II

Figura 4.1(e): AN para las técnicas 1 vs. 6, graficando para los I. 8 y II. 4 primeros sub-bloques



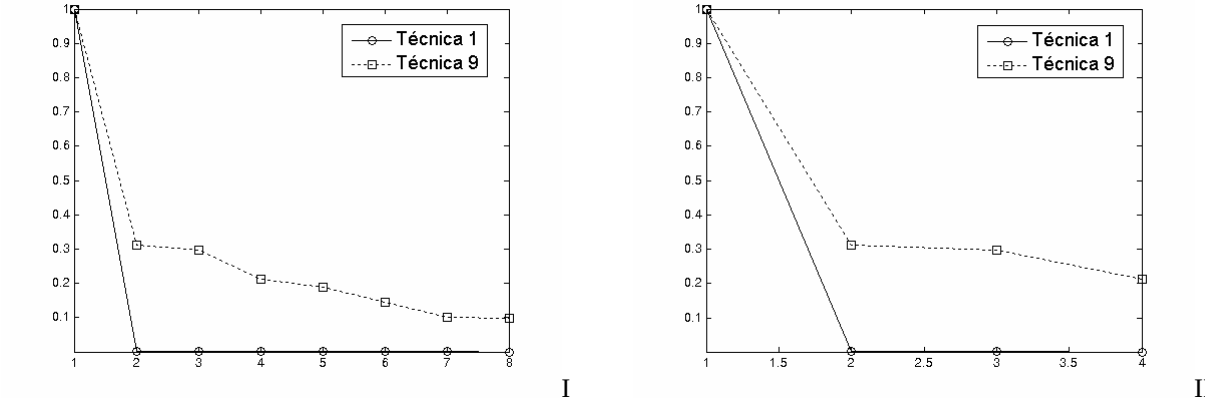
I II

Figura 4.1(f): AN para las técnicas 1 vs. 7, graficando para los I. 8 y II. 4 primeros sub-bloques



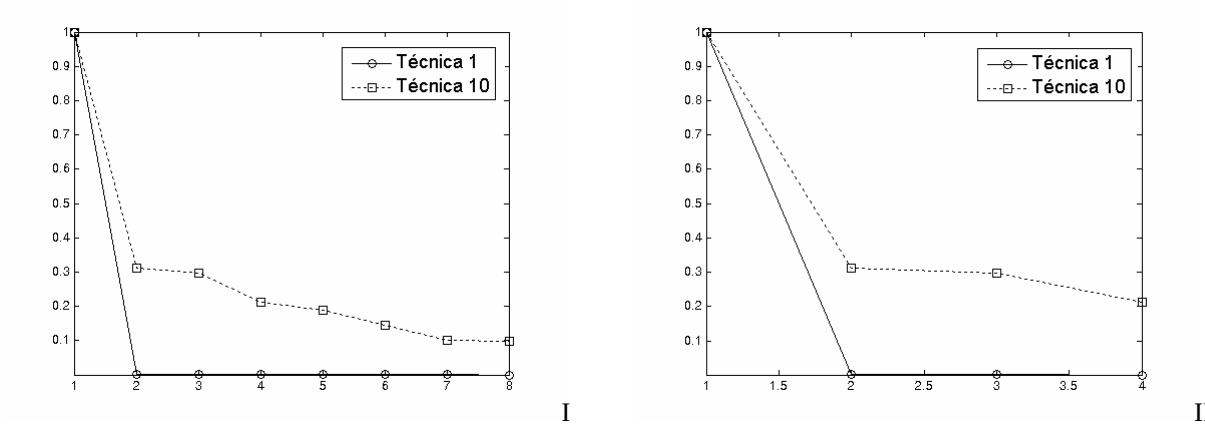
I II

Figura 4.1(g): AN para las técnicas 1 vs. 8, graficando para los I. 8 y II. 4 primeros sub-bloques



I II

Figura 4.1(h): AN para las técnicas 1 vs. 9, graficando para los I. 8 y II. 4 primeros sub-bloques

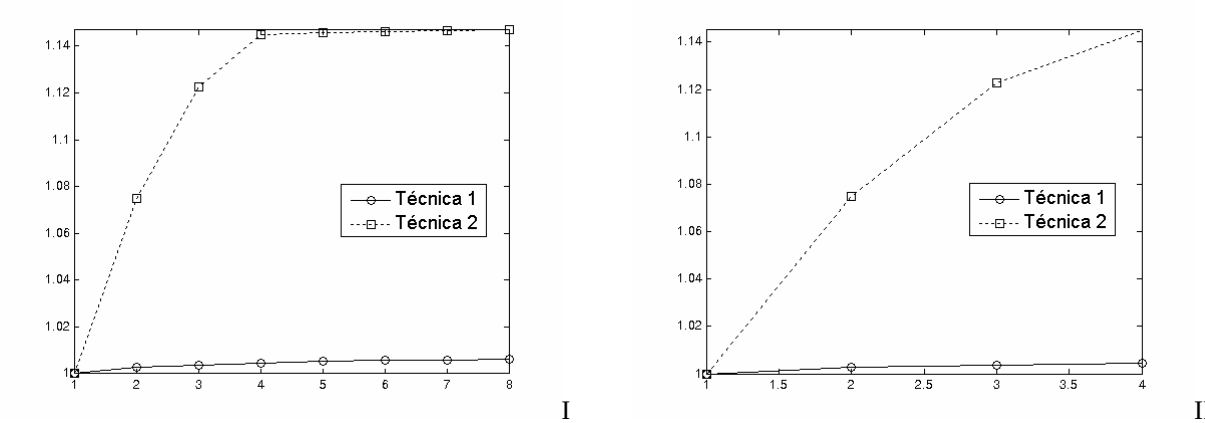


I II

Figura 4.1(i): AN para las técnicas 1 vs. 10, graficando para los I. 8 y II. 4 primeros sub-bloques

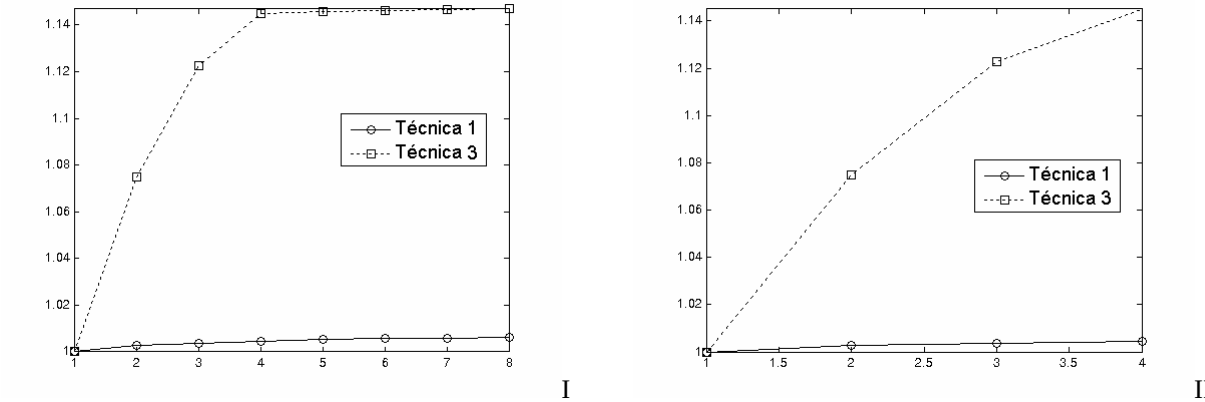
La serie representada por la Fig.4.2 muestra los autovalores normalizados acumulados, como una noción indirecta de la energía involucrada.

En todos los casos es sumamente claro el poco aporte acumulado de los autovalores distintos del primero frente al mismo en la Técnica 1 con respecto a las demás, lo que indica, que en la primera la energía visual se concentró práctica y exclusivamente en el primer autovalor de la serie.



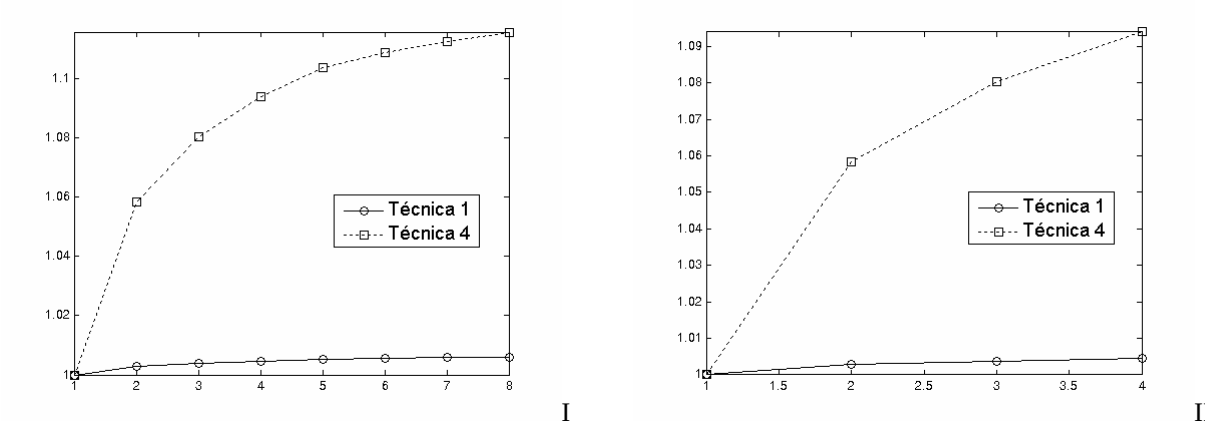
I II

Figura 4.2(a): ANA para las técnicas 1 vs. 2, graficando para los I. 8 y II. 4 primeros sub-bloques



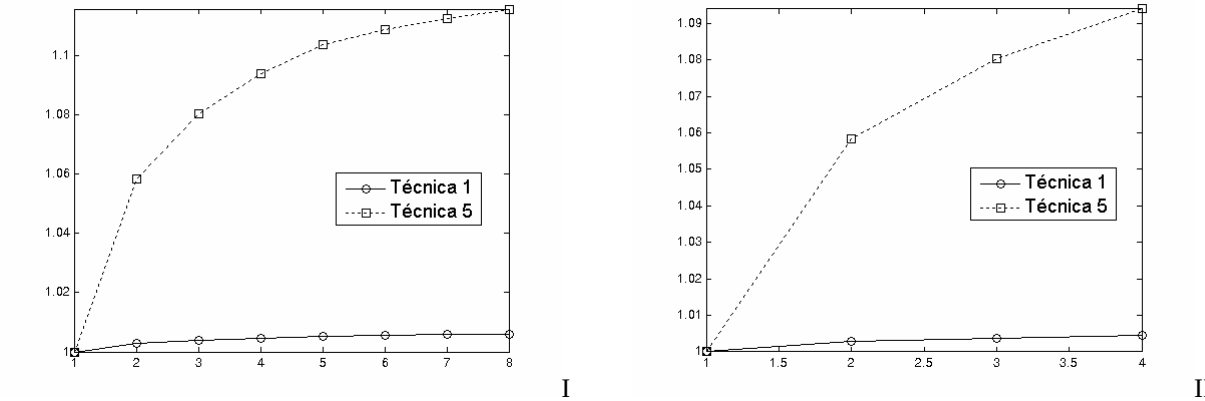
I II

Figura 4.2(b): ANA para las técnicas 1 vs. 3, graficando para los I. 8 y II. 4 primeros sub-bloques



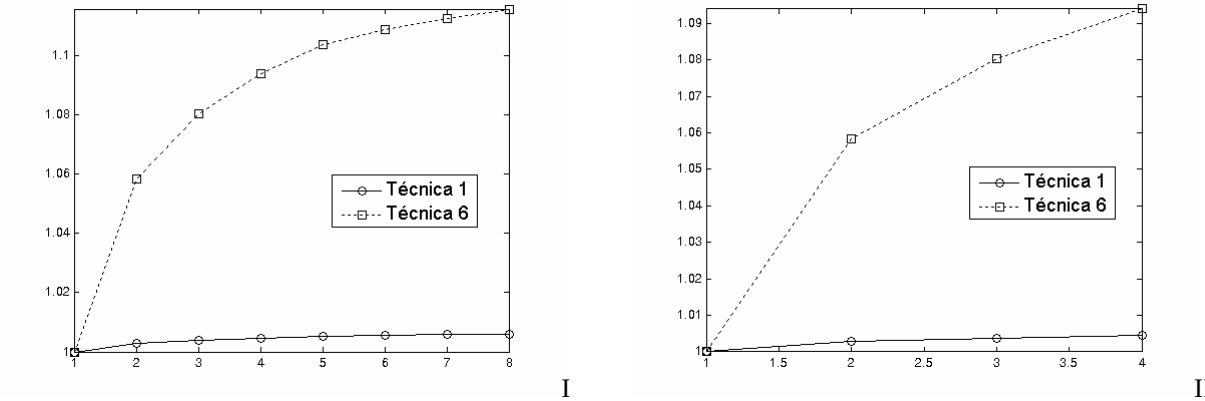
I II

Figura 4.2(c): ANA para las técnicas 1 vs. 4, graficando para los I. 8 y II. 4 primeros sub-bloques



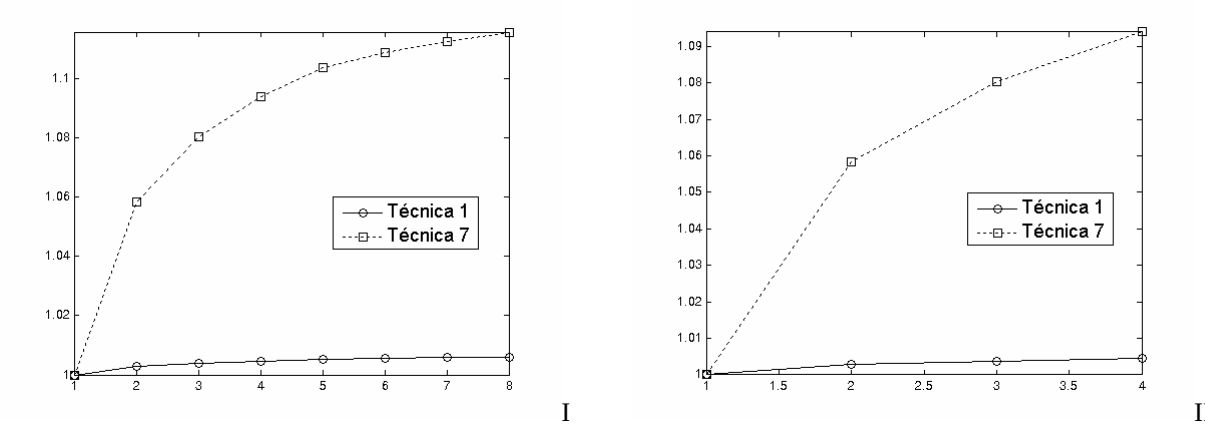
I II

Figura 4.2(d): ANA para las técnicas 1 vs. 5, graficando para los I. 8 y II. 4 primeros sub-bloques



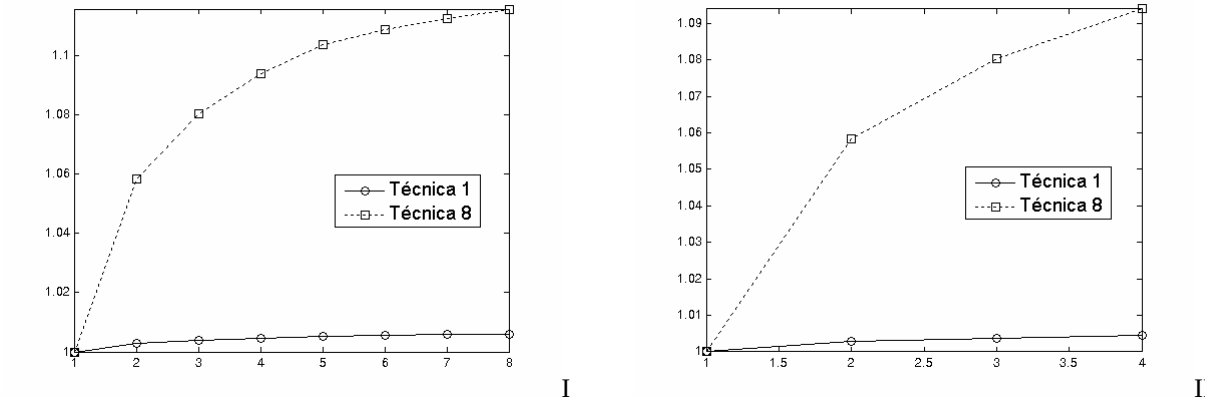
I II

Figura 4.2(e): ANA para las técnicas 1 vs. 6, graficando para los I. 8 y II. 4 primeros sub-bloques



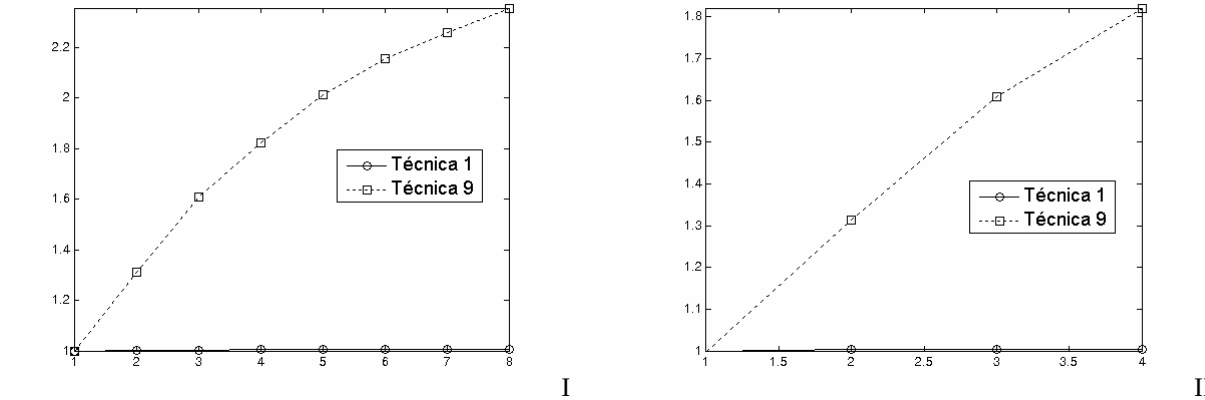
I II

Figura 4.2(f): ANA para las técnicas 1 vs. 7, graficando para los I. 8 y II. 4 primeros sub-bloques



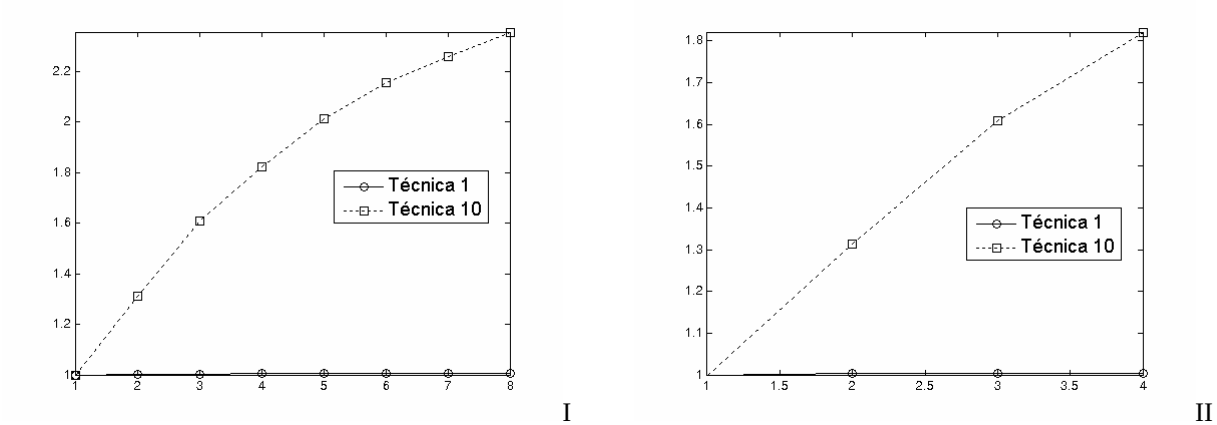
I II

Figura 4.2(g): ANA para las técnicas 1 vs. 8, graficando para los I. 8 y II. 4 primeros sub-bloques



I II

Figura 4.2(h): ANA para las técnicas 1 vs. 9, graficando para los I. 8 y II. 4 primeros sub-bloques



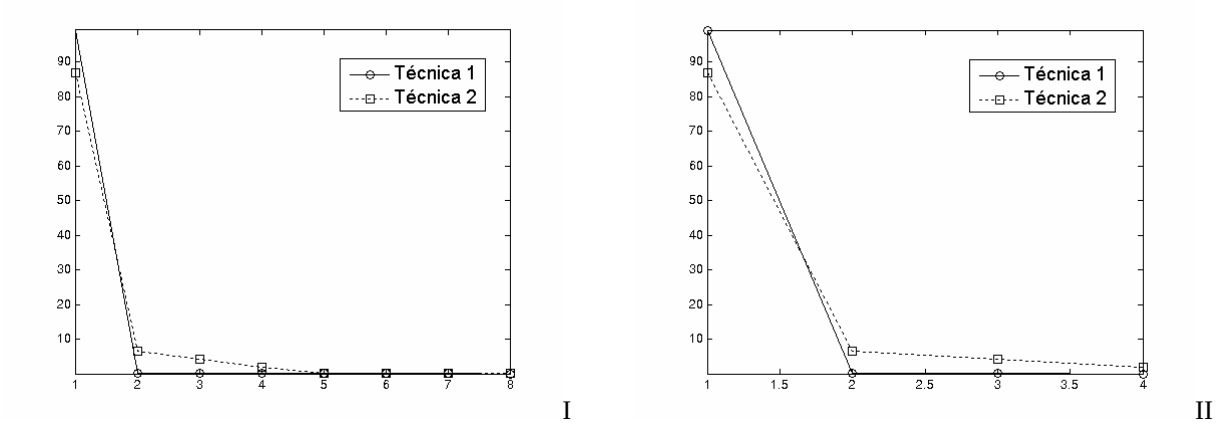
I II

Figura 4.2(i): ANA para las técnicas 1 vs. 10, graficando para los I. 8 y II. 4 primeros sub-bloques

La Fig.4.3 muestra a modo de monitor en forma porcentual y por ende más inequívoca los aportes individuales de cada autovalor.

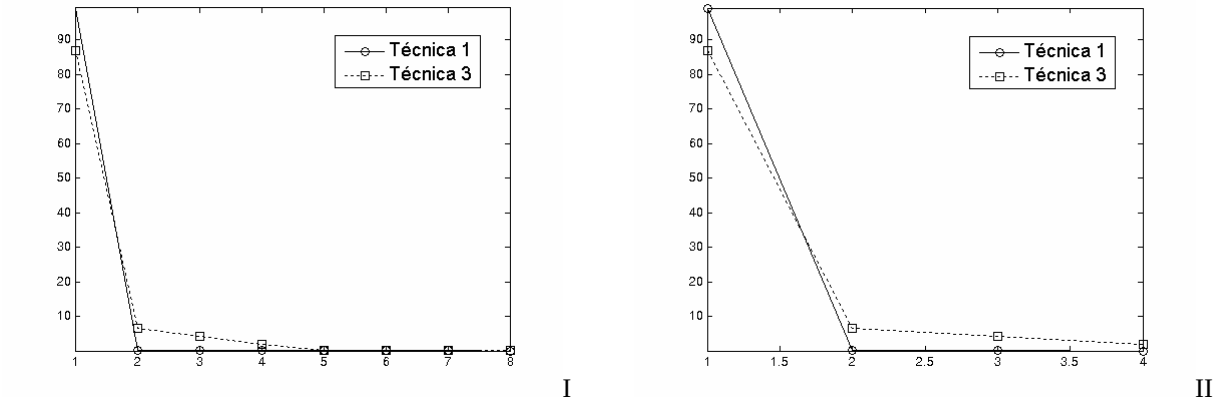
Nuevamente aquí vuelve a triunfar la Técnica 1 por sobre todas las demás.

La Técnica 1 es la única que arranca casi en el 100 % mientras el resto no supera el 90 % nunca.



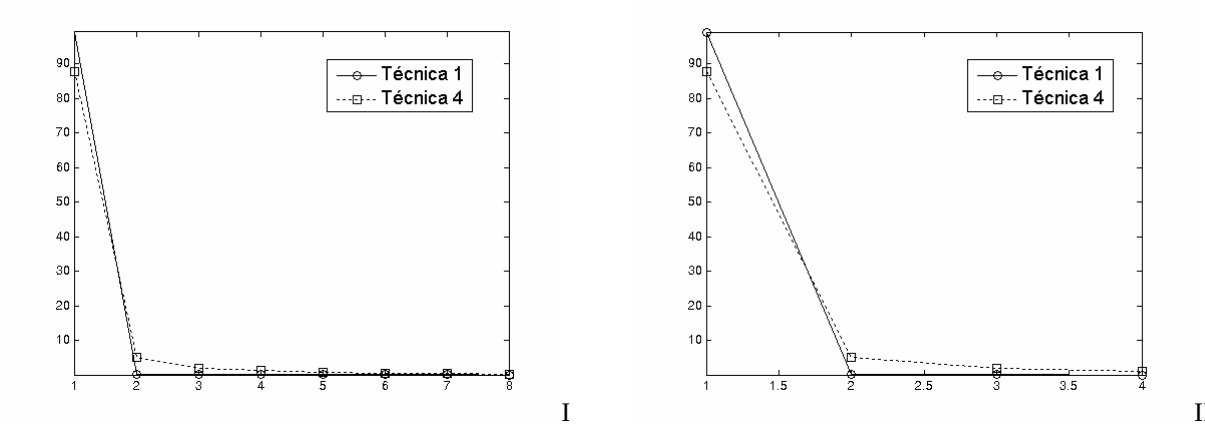
I II

Figura 4.3(a): Ercp para las técnicas 1 vs. 2, graficando para los I. 8 y II. 4 primeros sub-bloques



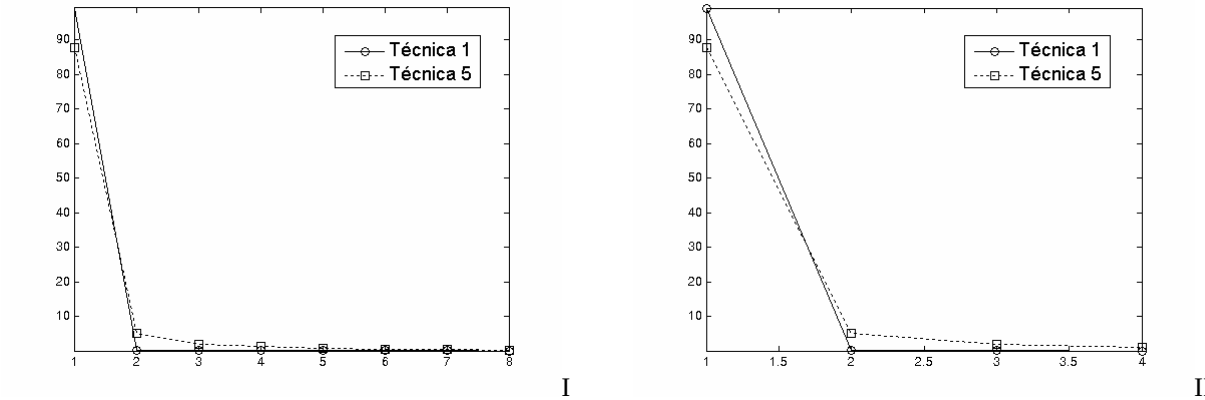
I II

Figura 4.3(b): Ercp para las técnicas 1 vs. 3, graficando para los I. 8 y II. 4 primeros sub-bloques



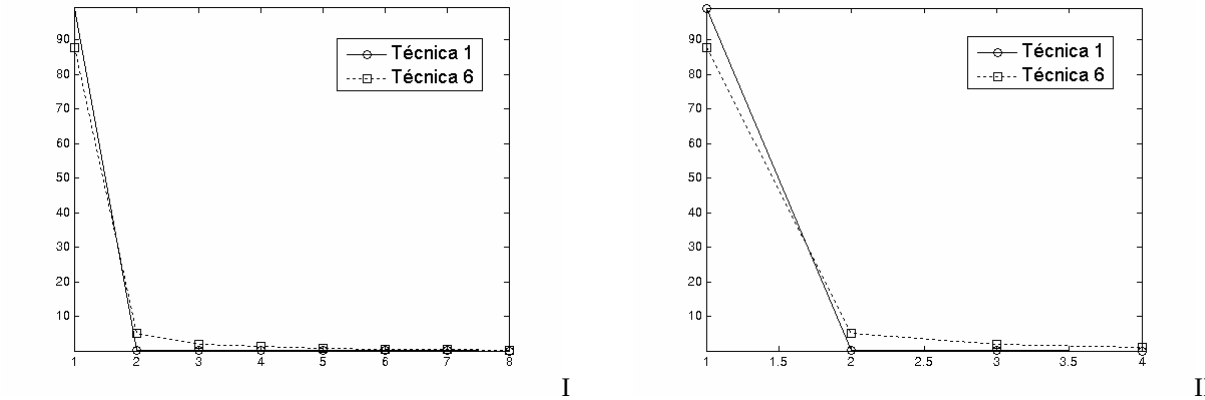
I II

Figura 4.3(c): Ercp para las técnicas 1 vs. 4, graficando para los I. 8 y II. 4 primeros sub-bloques



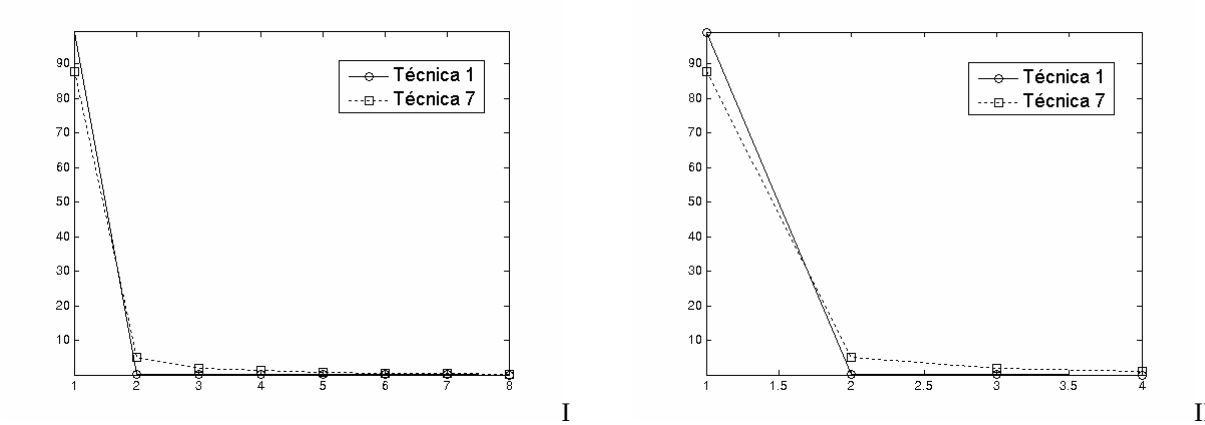
I II

Figura 4.3(d): Ercp para las técnicas 1 vs. 5, graficando para los I. 8 y II. 4 primeros sub-bloques



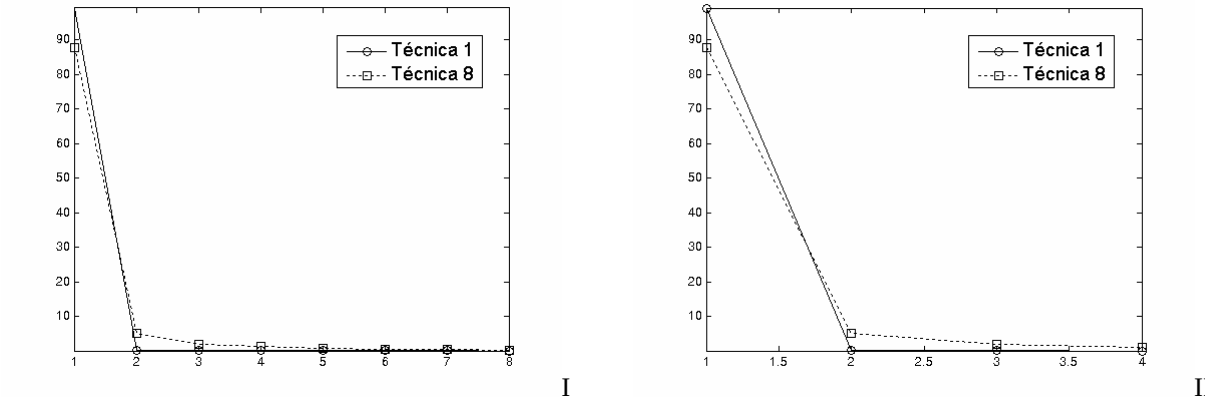
I II

Figura 4.3(e): Ercp para las técnicas 1 vs. 6, graficando para los I. 8 y II. 4 primeros sub-bloques



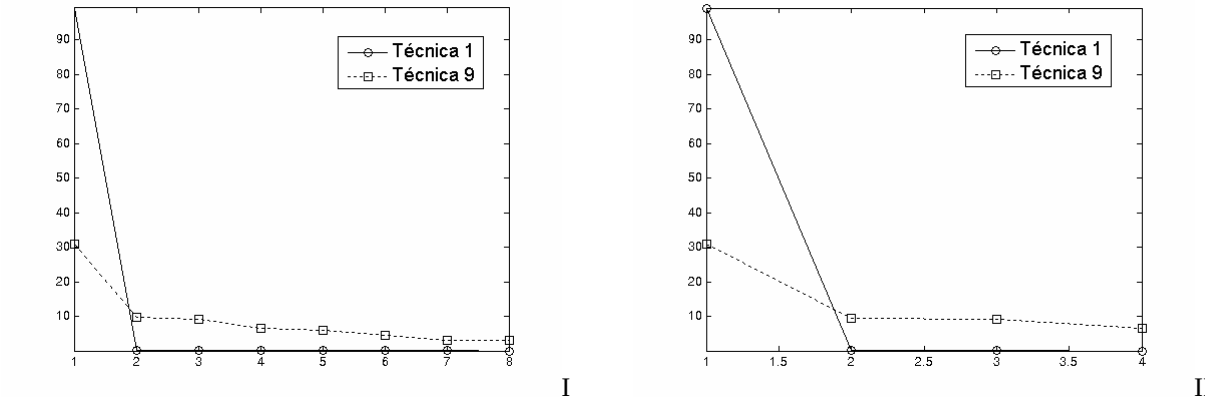
I II

Figura 4.3(f): Ercp para las técnicas 1 vs. 7, graficando para los I. 8 y II. 4 primeros sub-bloques



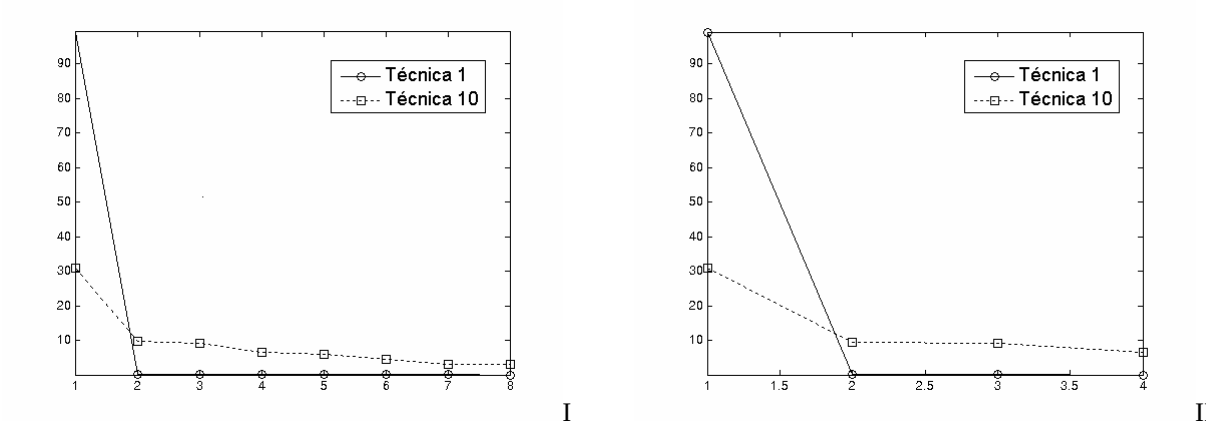
I II

Figura 4.3(g): Ercp para las técnicas 1 vs. 8, graficando para los I. 8 y II. 4 primeros sub-bloques



I II

Figura 4.3(h): Ercp para las técnicas 1 vs. 9, graficando para los I. 8 y II. 4 primeros sub-bloques

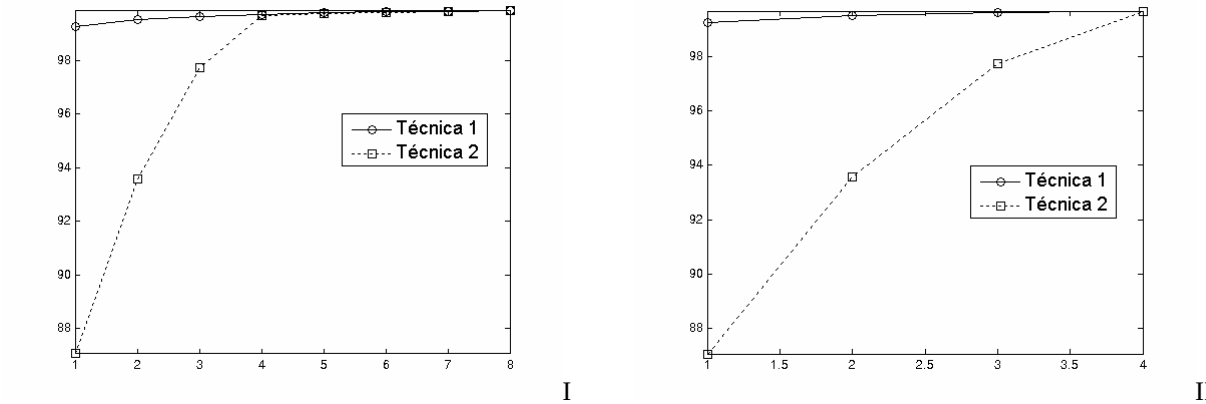


I II

Figura 4.3(i): Ercp para las técnicas 1 vs. 10, graficando para los I. 8 y II. 4 primeros sub-bloques

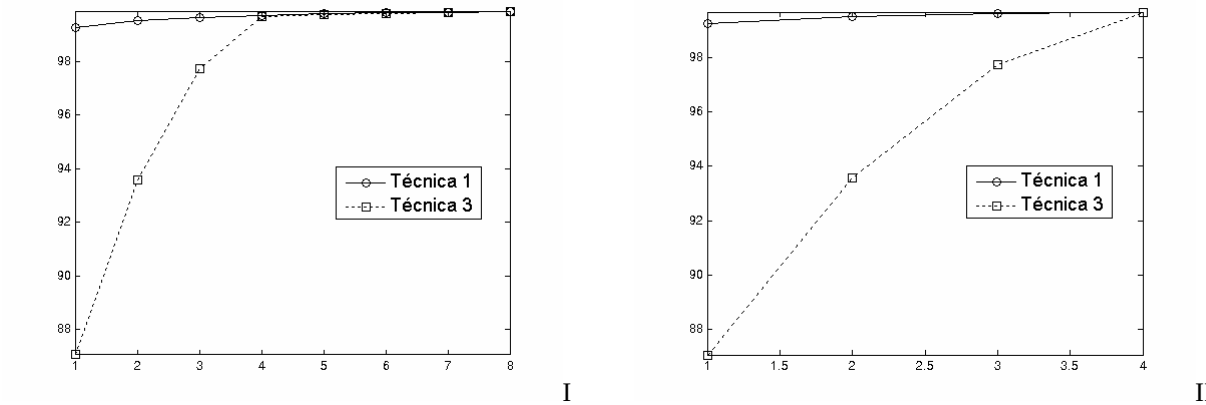
La Fig.4.4 muestra el acumulado de las simulaciones anteriores. Obsérvese de donde arranca la Técnica 1 (algo más del 99 %) frente al resto (por debajo del 88 %).

Esto es un claro e inequívoco monitor de la naturaleza SMEP que posee la Técnica 1, frente al resto que la poseen en proporciones considerablemente menores, aunque muy superiores a la Técnica 10 que sería la que es la empleada hasta el momento de TDKL sin las herramientas que vehiculizan SMEP.



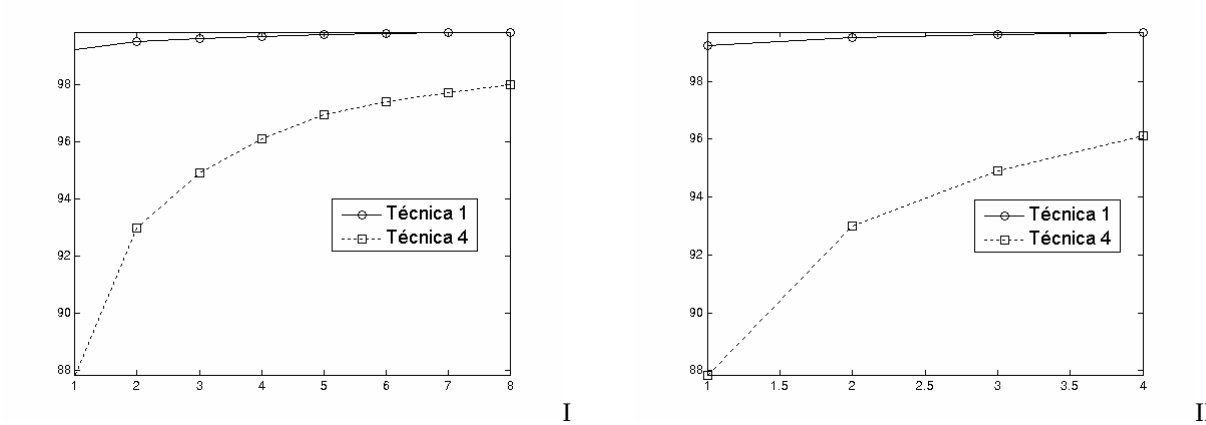
I II

Figura 4.4(a): Ercpa para las técnicas 1 vs. 2, graficando para los I. 8 y II. 4 primeros sub-bloques



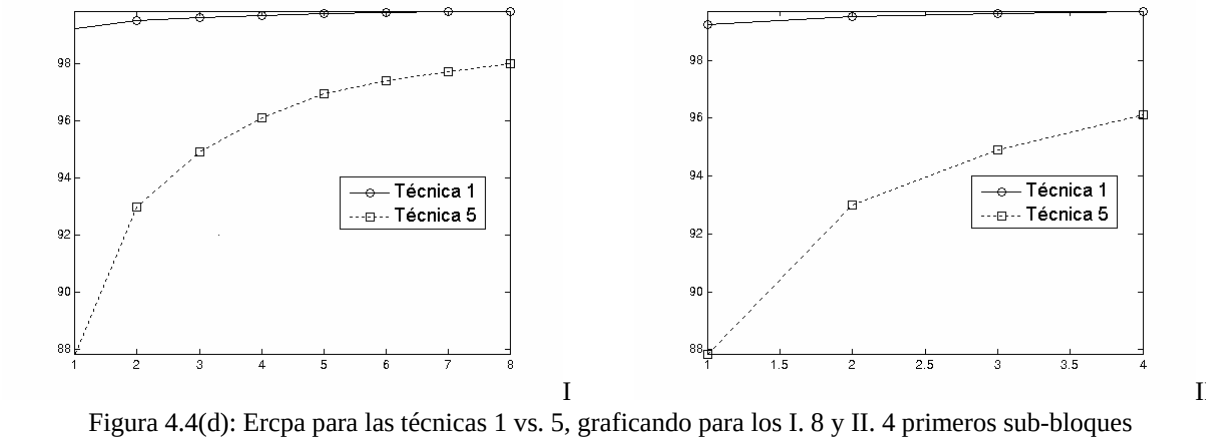
I II

Figura 4.4(b): Ercpa para las técnicas 1 vs. 3, graficando para los I. 8 y II. 4 primeros sub-bloques



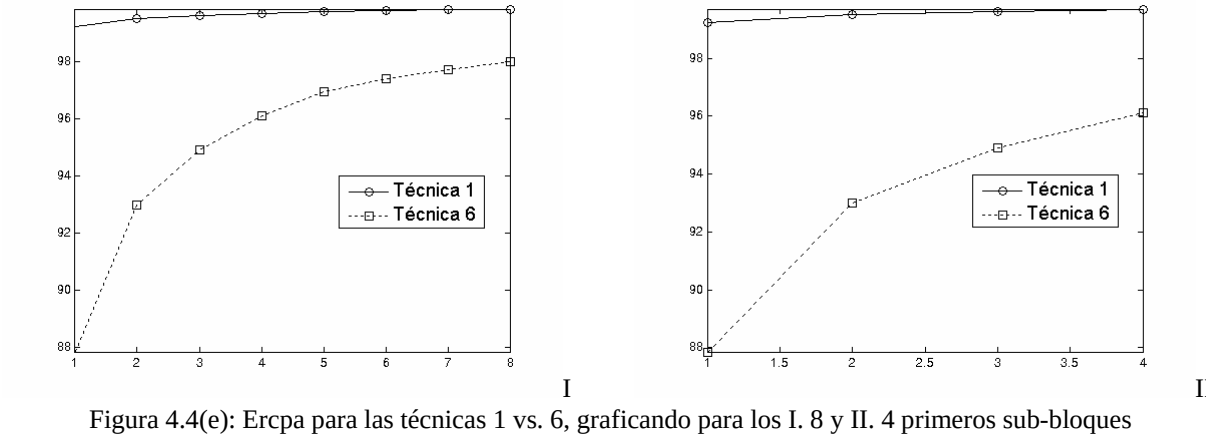
I II

Figura 4.4(c): Ercpa para las técnicas 1 vs. 4, graficando para los I. 8 y II. 4 primeros sub-bloques



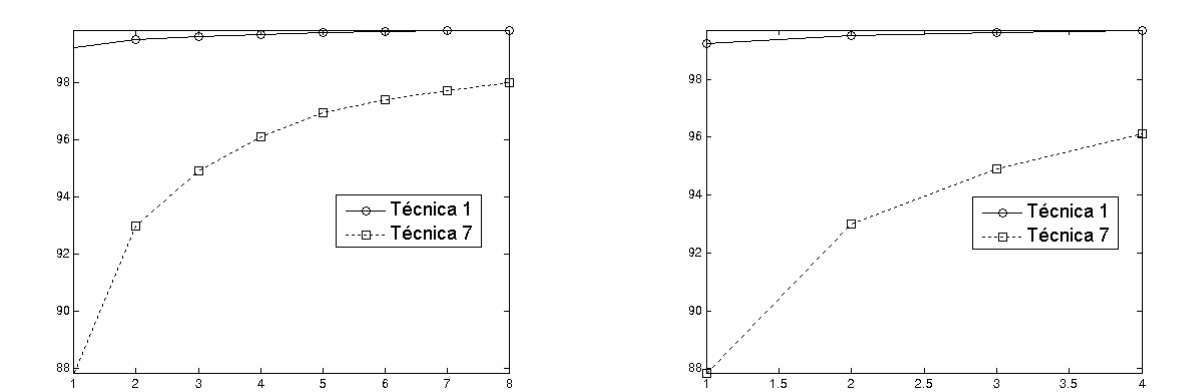
I II

Figura 4.4(d): Ercpa para las técnicas 1 vs. 5, graficando para los I. 8 y II. 4 primeros sub-bloques



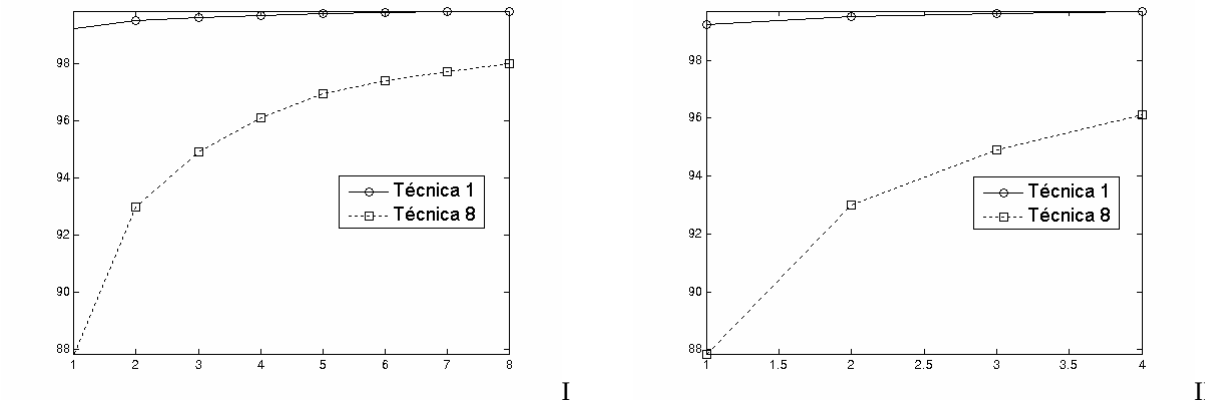
I II

Figura 4.4(e): Ercpa para las técnicas 1 vs. 6, graficando para los I. 8 y II. 4 primeros sub-bloques



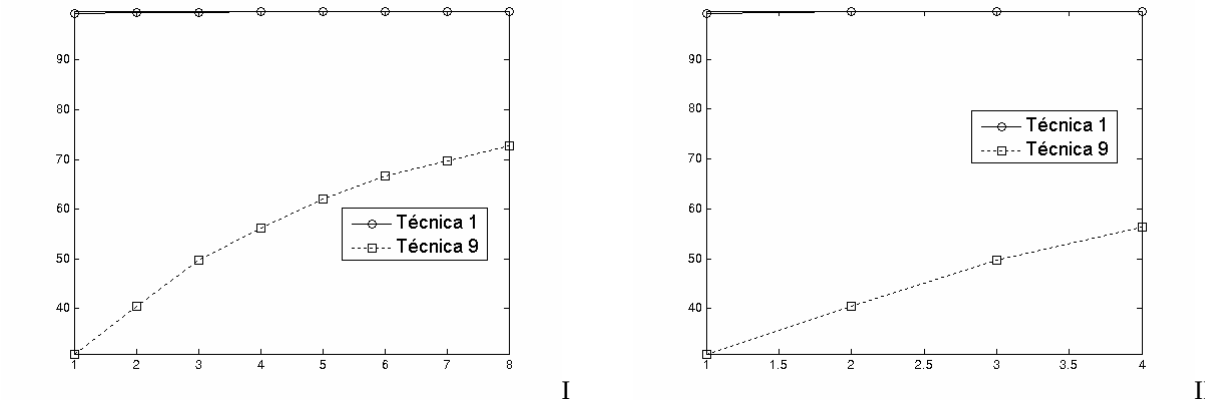
I II

Figura 4.4(f): Ercpa para las técnicas 1 vs. 7, graficando para los I. 8 y II. 4 primeros sub-bloques



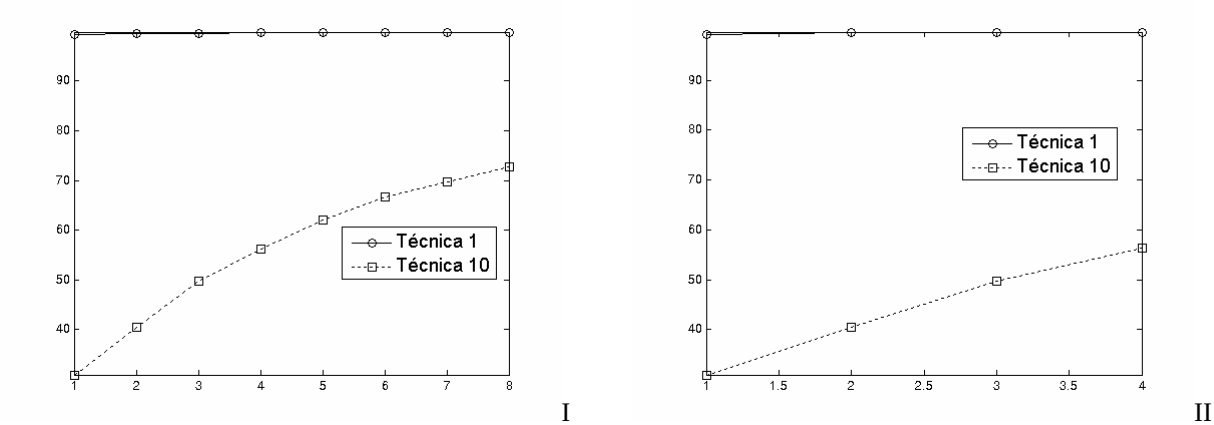
I II

Figura 4.4(g): Ercpa para las técnicas 1 vs. 8, graficando para los I. 8 y II. 4 primeros sub-bloques



I II

Figura 4.4(h): Ercpa para las técnicas 1 vs. 9, graficando para los I. 8 y II. 4 primeros sub-bloques



I II

Figura 4.4(i): Ercpa para las técnicas 1 vs. 10, graficando para los I. 8 y II. 4 primeros sub-bloques

La Fig.4.5 muestra de a pares la imagen decodificada de cada técnica empleada contra la original sin comprimir. Obsérvese la calidad visual de la Técnica 1 frente al resto.

I II

Figura 4.5(a): Imagen decodificada, I. Técnica 1 vs. II. Original.

I II

Figura 4.5(b): Imagen decodificada, I. Técnica 2 vs. II. Original.


I


II

Figura 4.5(c): Imagen decodificada, I. Técnica 3 vs. II. Original.


I


II

Figura 4.5(d): Imagen decodificada, I. Técnica 4 vs. II. Original.


I


II

Figura 4.5(e): Imagen decodificada, I. Técnica 5 vs. II. Original.


I


II

Figura 4.5(f): Imagen decodificada, I. Técnica 6 vs. II. Original.


I

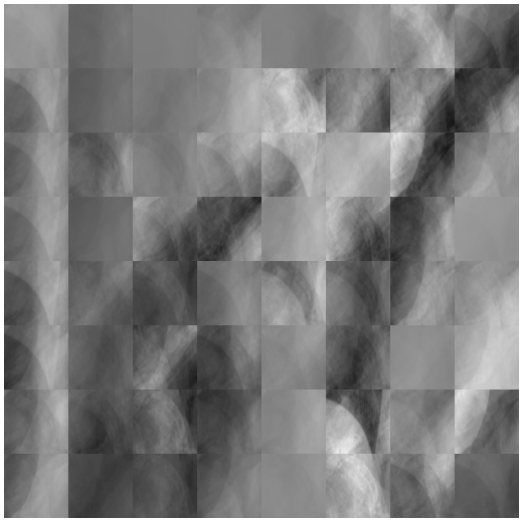

II

Figura 4.5(g): Imagen decodificada, I. Técnica 7 vs. II. Original.


I


II

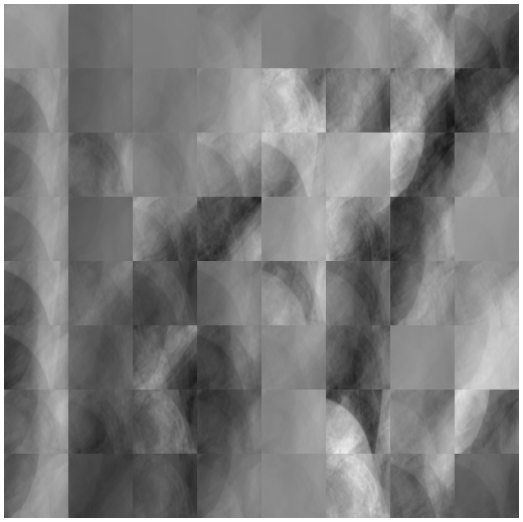
Figura 4.5(h): Imagen decodificada, I. Técnica 8 vs. II. Original.



I

II

Figura 4.5(i): Imagen decodificada, I. Técnica 9 vs. II. Original.



I

II

Figura 4.5(j): Imagen decodificada, I. Técnica 10 vs. II. Original.

La Fig.4.6 nos ejemplifica el error píxel-a-pixel de las distintas técnicas entre la imagen original sin comprimir vs las imágenes decodificadas en cada caso. Obsérvese lo insignificante de dicho error para la Técnica 1. De hecho, la Fig.4.6(a) tuvo que ser realzada para poder distinguirse visualmente dicho error.

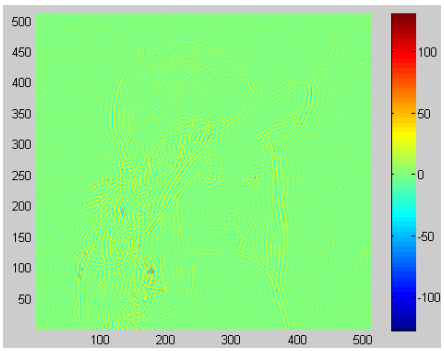


Figura 4.6(a): Error píxel-a-pixel para Técnica 1.

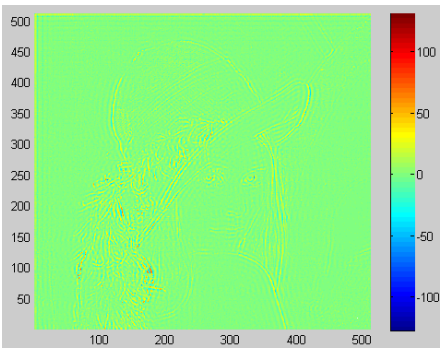


Figura 4.6(b): Error píxel-a-pixel para Técnica 2.

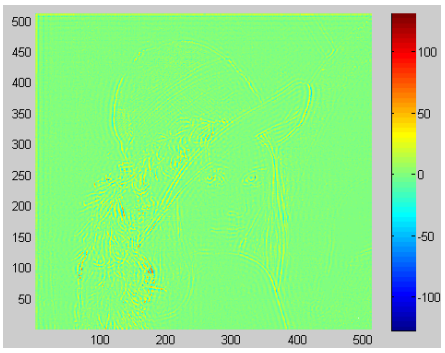


Figura 4.6(c): Error píxel-a-pixel para Técnica 3.

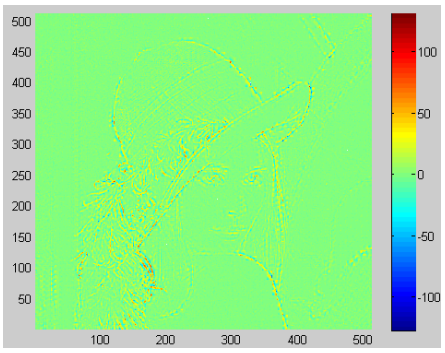


Figura 4.6(d): Error píxel-a-pixel para Técnica 4.

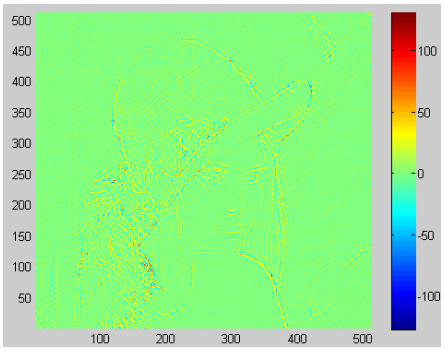


Figura 4.6(e): Error píxel-a-pixel para Técnica 5.

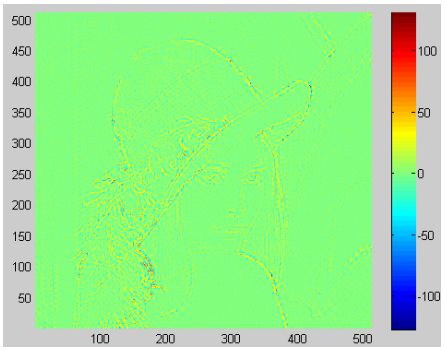


Figura 4.6(f): Error píxel-a-pixel para Técnica 6.

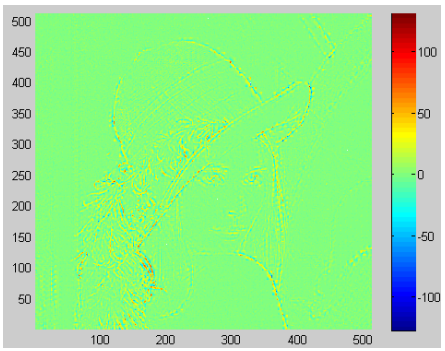


Figura 4.6(g): Error píxel-a-pixel para Técnica 7.

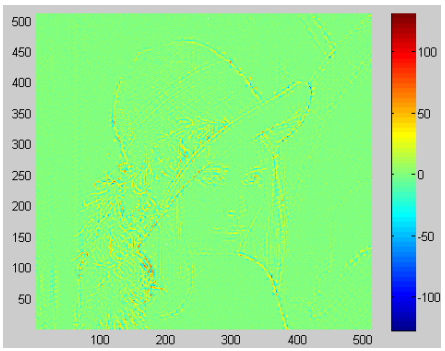


Figura 4.6(h): Error píxel-a-pixel para Técnica 8.

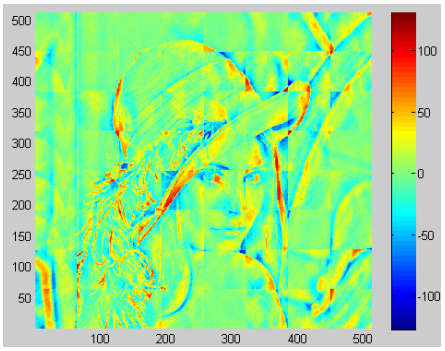


Figura 4.6(i): Error píxel-a-pixel para Técnica 9.

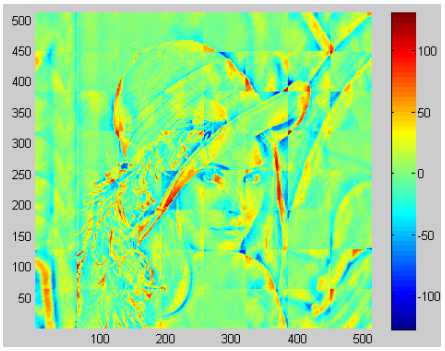


Figura 4.6(j): Error píxel-a-pixel para Técnica 10.

4.2.2. Segundo grupo de simulaciones

En este segundo grupo de simulaciones compararemos los resultados de la Técnica 1 vs. las Técnicas 11 a 14 sin TDKL. Para la Técnica 1 empleamos mosaicos de 64-por-64 pixeles, mien-

tras que para las Técnicas 11 y 12 los mosaicos son de 128-por-128 pixeles. Para la Técnica 13 utilizamos un umbral de 200, y finalmente pata la Técnica 14 se realizaron 2 niveles de splitting.

La Tabla 4.II muestra las ventajas cuantitativas de la Técnica 1 por sobre el resto. La Fig.4.7 muestra de a pares la imagen decodificada de cada técnica vs. la original sin comprimir. Finalmente la Fig.4.8 muestra los errores píxel-a-pixel de todas.

Tabla 4.II: Métricas vs Técnicas Empleadas

Técnica Empleada	Métricas			
	TC	bpp	MSE	PSNR
1	15.6935	0.0640	48.5315	31.2706
11	11.9110	0.0843	75.4200	29.3559
12	11.9110	0.0843	75.4200	29.3559
13	15.6577	0.0641	209.9089	24.9105
14	11.8836	0.0845	75.2143	29.3678


I


II

Figura 4.7(a): Imagen decodificada, I. Técnica 1 vs. II. Original.


I


II

Figura 4.7(b): Imagen decodificada, I. Técnica 11 vs. II. Original.


I


II

Figura 4.7(c): Imagen decodificada, I. Técnica 12 vs. II. Original.


I


II

Figura 4.7(d): Imagen decodificada, I. Técnica 13 vs. II. Original.


I


II

Figura 4.7(e): Imagen decodificada, I. Técnica 14 vs. II. Original.

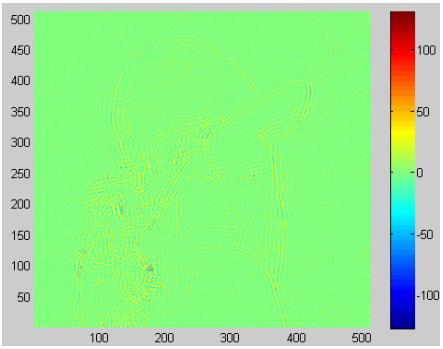


Figura 4.8(a): Error píxel-a-pixel para Técnica 1.

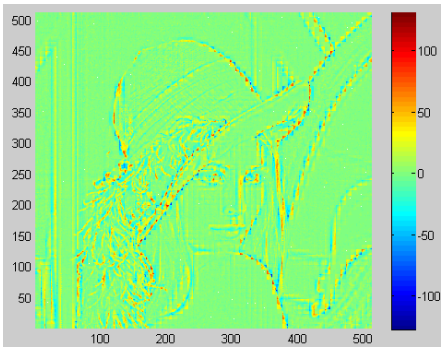


Figura 4.8(b): Error píxel-a-pixel para Técnica 11.

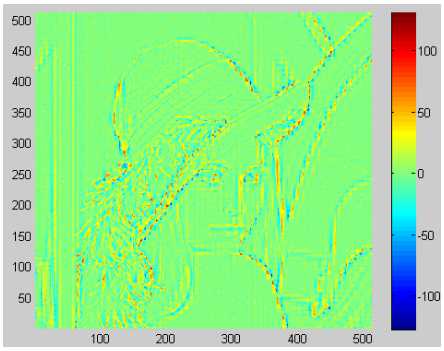


Figura 4.8(c): Error píxel-a-pixel para Técnica 12.

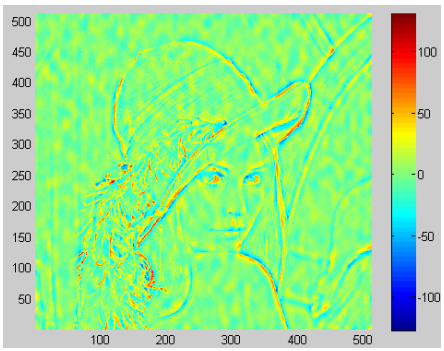


Figura 4.8(d): Error píxel-a-pixel para Técnica 13.

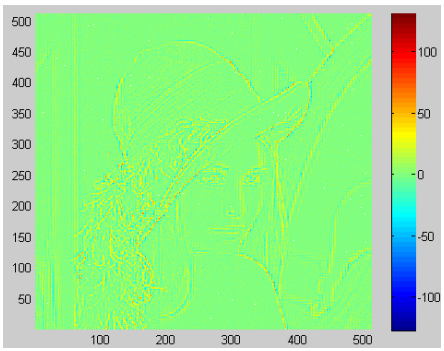


Figura 4.8(e): Error píxel-a-pixel para Técnica 14.

4.2.3. Tercer grupo de simulaciones

Este tercer grupo de simulaciones consiste en comparar las prestaciones de la Técnica 1 contra JPEG y JPEG2000.

Tabla 4.III: Métricas vs Técnicas Empleadas

Técnica Empleada	Métricas			
	TC	bpp	MSE	PSNR
1	15.6935	0.0640	9.5315	38.2706
16	8.9358	0.1124	8.7869	38.6924
17	9.9921	0.1005	10.4207	37.9518

La Técnica 1 empleó bloques de 64-por-64 pixeles, de hecho, si lo hubiera hecho con bloques de 16 u 8 como el caso de JPEG y JPEG2000 se habría obtenido una mejora notable y muy por encima de ambas técnicas, pero a costa de un mayor tiempo de cálculo.

 I  II

Figura 4.9(a): Imagen decodificada, I. Técnica 1 vs. II. Original.

 I  II

Figura 4.9(b): Imagen decodificada, I. Técnica 16 vs. II. Original.

 I  II

Figura 4.9(c): Imagen decodificada, I. Técnica 17 vs. II. Original.

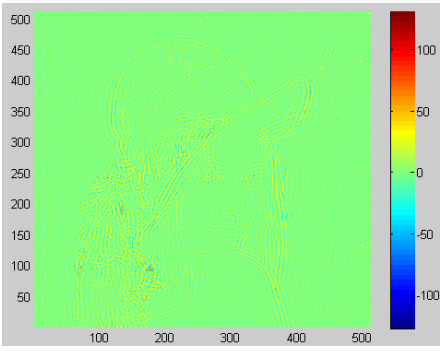


Figura 4.10(a): Error píxel-a-pixel para Técnica 1.

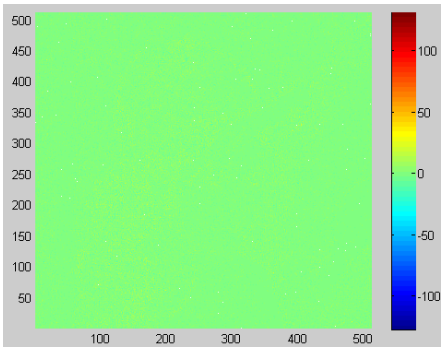


Figura 4.10(b): Error píxel-a-pixel para Técnica 16.

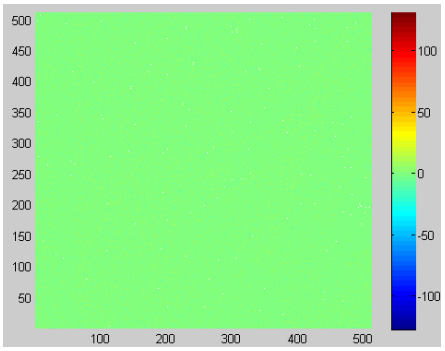


Figura 4.10(c): Error píxel-a-pixel para Técnica 17.

Ya con bloques simplemente de 32-por-32 la Técnica 1 supera a JPEG y JPEG2000 ampliamente y con una baja CC.

Tanto la codificación de JPEG, como la de JPEG2000 se realizó en MATLAB® en base a los principios del Apéndice D) al igual que con las 14 técnicas restantes.

4.3. Conclusiones del capítulo

En este capítulo se puso de relieve la supremacía de los métodos con SMEP frente aquellos sin SMEP. Además, se hizo lo propio entre el mejor método con SMEP y JPEG y JPEG2000. En ambos casos tanto a partir de métricas irrefutables como así también en base a la calidad visual de las correspondientes imágenes descomprimidas.

Capítulo 5

5. Conclusiones finales y futuras investigaciones

5.1. Conclusiones finales

Se ha demostrado fehacientemente a lo largo del presente trabajo, que el atributo conocido como SMEP permite aplicar más eficientemente la TDKL donde antes no se lo había podido hacer, es decir, al caso de imágenes monocuadro. Todas las simulaciones han señalado mediante las más apropiadas métricas que las técnicas con SMEP elevan el rendimiento de decorrelación de la TDKL manteniendo baja la CC. Por otra parte, las simulaciones del Capítulo 4 demostraron además, que la técnica con SMEP obtuvo mejores resultados que JPEG y JPEG2000 con todo lo que ello representa. La Técnica 1 empleó bloques de 64-por-64 pixeles, de hecho, si lo hubiera hecho con bloques de 16 u 8 como el caso de JPEG y JPEG2000 se habría obtenido una mejora notable y muy por encima de ambas técnicas, pero a costa de un mayor tiempo de cálculo. Ya con bloques simplemente de 32-por-32 supera a ambas ampliamente y con baja CC.

Finalmente, las conclusiones finales son:

• Es más eficiente al descorrelacionar cuando es elevada la Información Mutua entre bloques

• Permite casi el atributo L con cuadros grandes bajando dramáticamente la CC de la TDKL

5.2. Futuras investigaciones

Las futuras líneas de investigación deben orientarse a aplicar SMEP en señales (1D) y video (3D), de manera tal de extender el espacio de aplicación de esta nueva herramienta a ámbitos no involucrados en el presente trabajo. Esperando lograr similares resultados como los aquí alcanzados y a la vez posicionando a la TDKL donde jamás se pensó que podía llegar considerando su CC.

Por último, las futuras líneas de investigación más relevantes son:

• Aplicar SMEP a bloques solapados

• Investigar si otras transformadas producen SMEP

Apéndice A

A Tipos de exploración de los mosaicos

A.1 Introducción

Este apéndice contiene los distintos métodos de exploración de los bloques. Mas allá de cómo se generen los mosaicos, el procedimiento consiste en explorar la imagen por bloques y recolectar los mismos de manera consecuente con una mayor o menor afinidad entre ellos según un determinado criterio en el contexto de su uso.

A.2 Tipos de exploración

La Fig.A.1 muestra seis diferentes tipos de exploración espacial de los bloques [499, 400]. En el presente trabajo usaremos los métodos (a), (d) y (f).

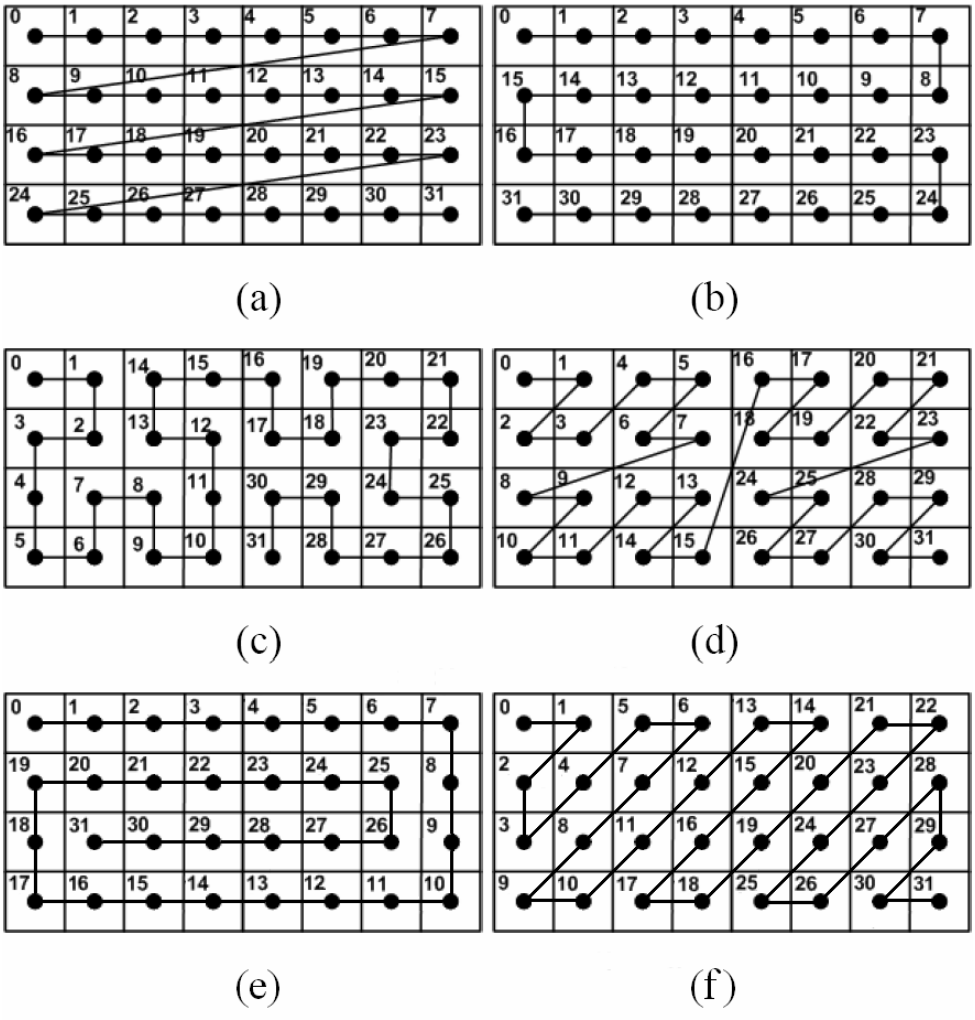


Fig.A.1: Diferentes métodos de exploración de los bloques. a) Orden de barrido por filas, b) orden de filas primas, c) orden de Peano-Hilbert, d) orden Z-Morton, e) orden espiral, y f) orden en zig-zag.

En la Fig.A.1 cada celda numerada representa un sub-bloque (por ejemplo en el ámbito del dominio de la TDO, ver Capítulo 2, Sección 2.2.2.2.4) la cual debe estar espacialmente ordenada (en orden creciente) en una matriz tridimensional antes de aplicar a esta última la TDKL, ver Fig.A.2.

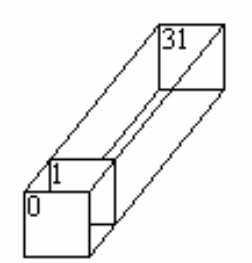


Fig.A.2: Construcción de una matriz-3D con los sub-bloques en orden creciente.

Como se pudo observar en la Fig.A.1, los pixeles, los cuales han de ser tratados o no con TDO, se encuentran concentrados formando lo que llamamos bloques o mosaicos. En los bloques se agrupan cantidades de 2×2, 4×4, 8×8 … pixeles, los cuales pueden ser fácilmente extraídos, dado que los pixeles en estos bloques son transmitidos uno a continuación del otro (el ordenamiento por filas no posee esta importante característica porque los pixeles son transmitidos serialmente fila tras fila). Esta característica puede ser aprovechada para procesamiento espacial de imágenes, tal como es el caso de una reducción en la resolución. En orden a reducir la resolución de la imagen por un factor de dos, hay que calcular la media de cuatro píxeles (un bloque de 2×2 pixeles). Con estos ordenamientos (Morton y barrido por filas), se puede realizar de una manera simple y sencilla, sin requerir múltiples elementos de almacenamiento. Estos cálculos pueden ser extensivos a bloques de dimensiones 4×4, 8×8, etc.

Finalmente, el tipo de barrido de bloques por zig-zag se empleará en el Capítulo 3 (Soluciones propuestas), mientras que el barrido de pixeles por zig-zag se emplea en los formatos de compresión JPEG y JPEG2000 (ver Apéndice D).

A.3 Conclusiones del apéndice

En este apéndice se mostraron los tipos de exploración de bloques más frecuentemente usados en la práctica, así como los más recomendables para el presente trabajo.

Apéndice B

B Algunos elementos de compresión de imágenes con pérdidas mediante transformada

B.1 Introducción

En este apéndice se desarrollan las herramientas relativas a la compresión de imágenes con pérdidas en base a la codificación genérica por transformada en forma complementaria a las desarrolladas en el Capítulo 1. Esto quiere decir, que se analizarán las que en el mismo quedaran pendientes, a saber:

• Una noción de codificación genérica por transformada, sus aspectos más relevantes y propiedades
• Cuantización, sus variantes y la versión más apropiada para esta tesis
• Compresión entrópica, en particular de Huffman y Aritmética.

B.2 Elementos constitutivos

B.2.1 Codificación genérica por transformada

El sistema de compresión de imágenes con pérdidas no recupera la imagen original pixel a pixel. En su lugar toma ventaja de las limitaciones del ojo humano mediante las cuales el sistema visual tiende a aproximar la imagen recuperada a la original. Estos métodos pueden alcanzar tasas de compresión bastamente superiores que los métodos sin pérdidas, no obstante, los mismos deben ser utilizados con cuidado [401, 402]. Las técnicas de compresión con pérdidas generalmente solo trabajan bien con fotografías de la vida real; las mismas dan frecuentemente pésimos resultados con otros tipos de imágenes tales como las binarizadas, o texto. Sometiendo una imagen a través de varios ciclos de compresión-descompresión con pérdidas sucederá que la imagen se degradará más allá de los estándares aceptables. Por lo tanto, una compresión con pérdidas debería ser usada exclusivamente luego de que todos los otros eventuales procesos aplicados a la imagen ya han tenido lugar, es decir, no debería ser empleado como un formato de almacenamiento intermedio. Solo el ojo humano les asignará una apariencia similar a la imagen recuperada y a la original. En cambio, si una computadora posee un sistema de reconocimiento de imágenes, entonces, la misma reconocerá las diferencias entre ambas [403]. En otras palabras, considere el codificador de transformada genérica de la Fig.B.1 el cual consiste de una transformada bi-dimensional, un cuantizador, y un codificador entrópico (siendo oportuno aclarar en

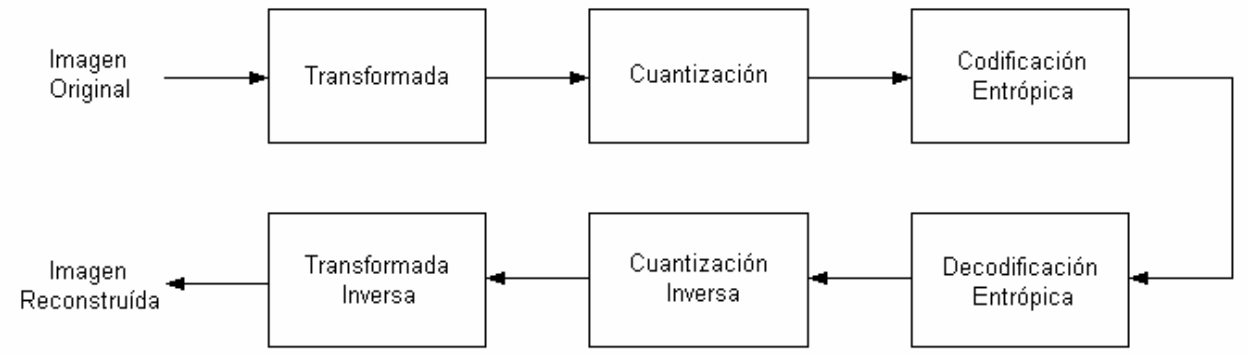


Figura B.1: Codificación genérica por transformada, para imágenes digitales.

este punto que entre la transformada y la cuantización se descartan coeficientes poco significativos con algún criterio de agresividad). Nosotros podemos observar en este punto que las pérdidas ocurren durante la cuantización y luego de la transformada. Por lo tanto, en orden a conducir a buen puerto nuestro análisis, debemos repetir la transformada para retornar al estado donde las pérdidas ocurren y examinar los efectos de la cuantización sobre los coeficientes de la transformación [404, 405].

B.2.1.1 Esquema de compresión

La codificación por transformada es un esquema de compresión donde se aplica una transformación $\boldsymbol{T}$ a un conjunto de datos dados $\{\boldsymbol{x_i}\}_{\boldsymbol{i}=1\ldots\boldsymbol{n}} \subset \Re^{\boldsymbol{m}}$ en orden a obtener una nueva representación $\{\boldsymbol{y_i}\}_{\boldsymbol{i}=1\ldots\boldsymbol{n}} \subset \Re^{\boldsymbol{m}}$ que resulta tener mejores propiedades con respecto a la cuantización y codificación:

$$\boldsymbol{y} = \boldsymbol{T}\,\boldsymbol{x}, \quad \boldsymbol{y} = \{\boldsymbol{y_i}\}_{\boldsymbol{i}=1\ldots\boldsymbol{n}}, \quad \boldsymbol{x} = \{\boldsymbol{x_i}\}_{\boldsymbol{i}=1\ldots\boldsymbol{n}}, \tag{B.1}$$

La compresión de imágenes que emplea codificación por transformada se constituye usualmente como un esquema de compresión con pérdidas. Esto significa que la imagen original puede ser reconstruida a partir de su código solo con cierta exactitud. El error de reconstrucción recibe el nombre de *distorsión*. Como se muestra en la Fig.B.2, la codificación por transformada consiste básicamente de los siguientes pasos:

1. la imagen es segmentada en mosaicos, cada mosaico es representado por un vector resultante en la representación de la imagen $\boldsymbol{x}$.
2. Una transformación $\boldsymbol{T}$ es aplicada a los datos vectoriales.
3. Los coeficientes resultantes de la transformación son cuantizados usando cuantización escalar (la cual se desarrollará en detalle en este apéndice), es decir, el codificador con pérdidas $\boldsymbol{Q_e}$ es aplicado sobre la transformada, el cual resulta en una representación discreta $\boldsymbol{Q_e}(\boldsymbol{Tx})$.
4. La transformada discretizada es codificada mediante una función $\boldsymbol{C}$ en una cadena de bits, es decir, el código. La representación de la imagen $\boldsymbol{C}(\boldsymbol{Q_e}(\boldsymbol{Tx}))$ puede entonces ser almacenada o transmitida a un receptor.
5. Para reconstruir una imagen desde su código, la transformada discretizada $\boldsymbol{Q_e}(\boldsymbol{Tx})$ es reobtenida desde el decodificador.
6. La función de decodificación $\boldsymbol{Q_d}$ del cuantizador es aplicada, reconstruyendo la transformada distorsionada $\boldsymbol{Q_d}(\boldsymbol{Q_e}(\boldsymbol{Tx})) = \boldsymbol{Q}(\boldsymbol{Tx})$.
7. La transformación inversa $\boldsymbol{T}^{-1}$ reconstruye el vector de representación distorsionado de la imagen $\tilde{\boldsymbol{x}} = \boldsymbol{T}^{-1}(\boldsymbol{Q}(\boldsymbol{Tx}))$.
8. Finalmente, la imagen es reconstruida desde su vector de representación.

En lugar de la aproximación de codificación de transformada presentada, la que generalmente es la mejor forma de codificar los datos vectoriales (sin pérdidas), es decir, los mosaicos de la imagen, debería ser en general una codificación vectorial o esquema de cuantización vectorial [136]. No obstante, dado que la dimensión de su codebook (tabla de codificación) sería inmensa y dado que este resulta necesario para representar una imagen y teniendo en cuenta que la complejidad computacional de la búsqueda en el codebook es inaceptable, el método descrito resulta ser inaplicable en la práctica [403]. Entonces, el método de compresión de imágenes con

pérdidas más ampliamente usado (usado por ejemplo en el formato JPEG estándar, el cual es desarrollado en el Apéndice D) resulta ser el de aplicar una transformación y – más o menos – independientemente cuantización escalar y compresión entrópica sobre los componentes resultantes.

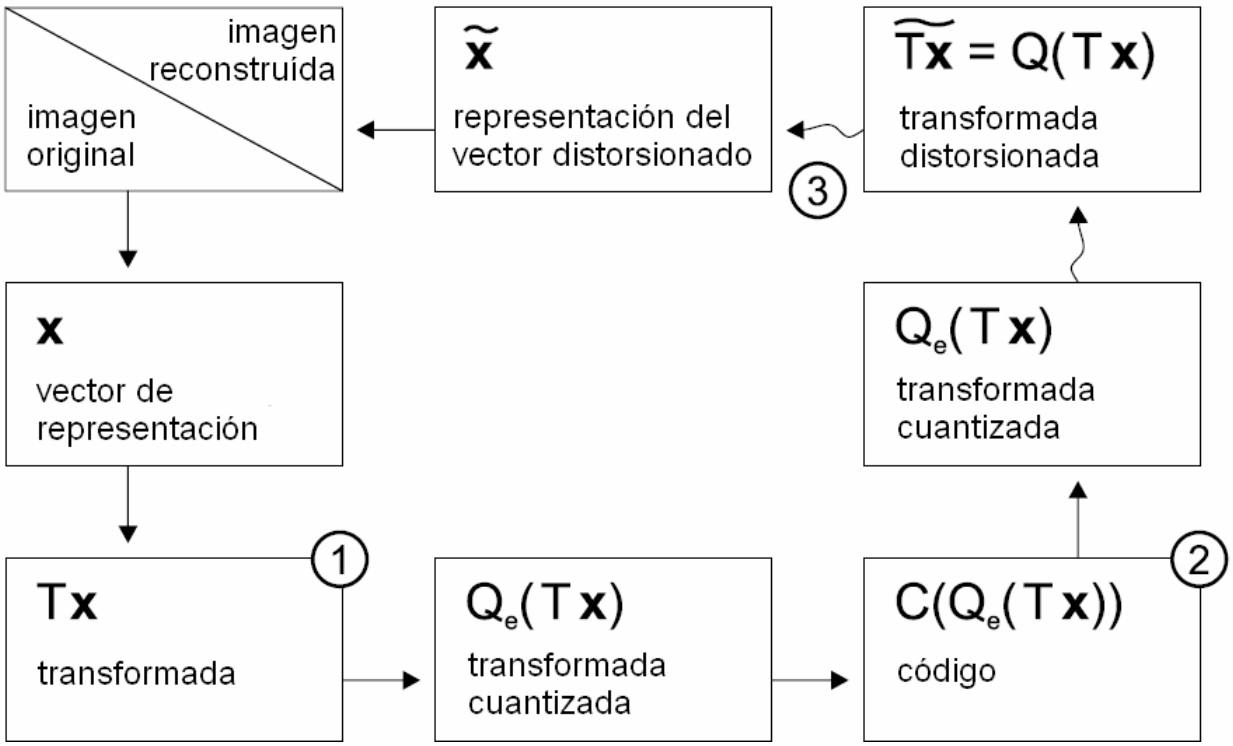


Figura B.2: Pasos de la codificación por transformada.

En la mayoría de los casos, las transformaciones lineales son usadas para reducir la cantidad de redundancia en la representación de la señal, por ejemplo, la TDKL, la TDC o la TDO, siendo estas dos últimas independiente de los datos. Esto resulta en un código más corto que la representación canónica que si nosotros aplicamos cuantizadores escalares independientes a los coeficientes. Hace varios años, el objetivo de la codificación por transformada se orientó a la recorrelación de los datos en pos a reducir las redundancias. Mas tarde se demostró que la decorrelación, de hecho, conduce a la transformación óptima – al menos en el caso de señales de entrada Gaussianas en un límite de resolución elevado [404]. No obstante, la recorrelación no es ni necesaria ni suficiente para la codificación de transformada óptima en general.

Como se describió en el *paso 1*, en función de poder realizar la transformación sobre los datos de la imagen, vamos a segmentar a la misma en mosaicos de *nfm* x *ncm* píxeles y reordenar la intensidad de los valores de los píxeles para obtener una representación vectorial $\boldsymbol{x} \in \Re^{\boldsymbol{m}}$, donde *m* = *nfm* x *ncm* (ver Fig.1.3). Inversamente, podemos reconstruir la imagen a partir de su vector de representación, dada la dimensión de la imagen, es decir, el número de filas y columnas de los mosaicos de la imagen. La performance del esquema de compresión depende principalmente de la transformada usada. Optimizando la transformación *T* para compresión de imágenes, se minimiza la longitud del código, es decir, de la transformada cuantizada codificada

$$\boldsymbol{y}_{CQ} = \boldsymbol{C}\left(\boldsymbol{Q}\left(\boldsymbol{T}\boldsymbol{x}\right)\right) \tag{B.2}$$

Mientras que al mismo tiempo se minimiza la distorsión

$$\boldsymbol{D = d(x, \tilde{x})\ para\ \tilde{x} = T^{-1}\left(Q^{-1}\left(C^{-1}Y\right)\right)} \qquad \text{(B.3)}$$

donde *C* representa al codificador, *Q* al cuantizador y *d* es la medida de la distancia. Un paso de cuantización es en la mayoría de los casos necesario dado que en general la transformación retorna valores no-enteros, incluso tratándose de fuentes discretas. El codificador *C* convierte los valores entregados por el cuantizador a una cadena binaria de bits decodificable en forma única. Se debe notar que la distorsión resulta del paso de cuantización y algunas veces de una transformación no estrictamente invertible. Discutiremos tanto la cuantización como la codificación más adelante en el presente apéndice.

Dependiendo de diferentes suposiciones acerca de la cuantización y la subsecuente codificación de los coeficientes de la transformada, se pueden presentar diferentes criterios de performance para las transformaciones [405], mediante los cuales se puede realizar la transformación sin especificar cuantizadores y codificadores concretos. Esto significa que este tipo de análisis se realiza directamente sobre los entradas transformadas (*paso 1* en la Fig.B.2)

Existen muchas alternativas a los efectos de implementar diferentes cuantizadores, así como las medidas de la longitud del código resultante. Por lo tanto, podemos analizar la representación de la imagen en el *paso 2* de la Figura B.2. Para una longitud de código dado, es decir, una limitación directa de la exactitud de cuantización, compararemos entonces la distorsión de las imágenes reconstruidas de las transformadas probadas.

B.2.1.2 Propiedades de las transformadas

Como se puede apreciar en la Fig.B.3 la codificación por transformada implica la aplicación de un proceso multi-etapas el cual se encuentra sujeto a una serie de propiedades. No obstante, dicho

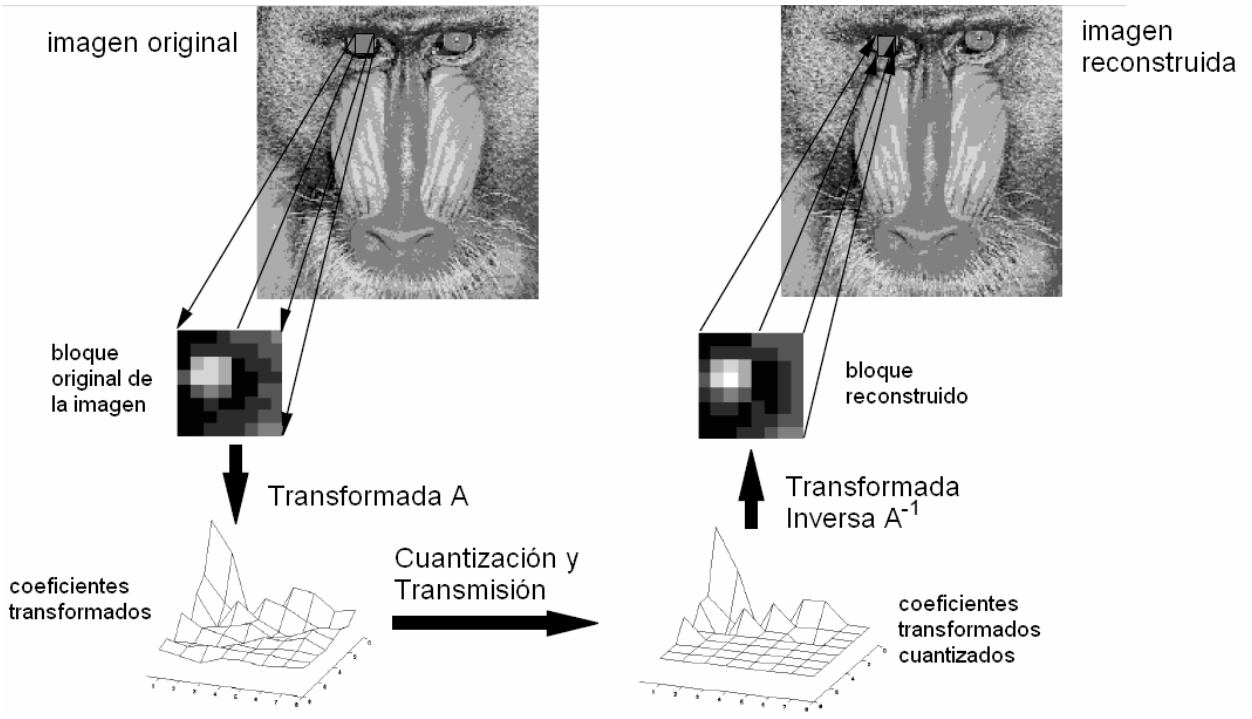


Figura B.3: Codificación por transformada.

proceso se inaugura con una transformación directa del tipo

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x} \tag{B.4}$$

donde las dimensiones involucradas (según el Capítulo 1) serán: $\boldsymbol{y} \in \Re^{nm\times(nfm\times ncm)}$, $\mathbf{A} \in \Re^{nm\times nm}$ y $\boldsymbol{x} \in \Re^{nm\times(nfm\times ncm)}$. Siendo los elementos constitutivos de $\boldsymbol{x}$, los bloques en los cuales se sub-dividió a la imagen original $\boldsymbol{I}$ organizados como columnas.

EL proceso se completa con una transformación inversa del tipo [406]

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\ \boldsymbol{y} = \boldsymbol{A}^{T}\,\boldsymbol{y} \tag{B.5}$$

De más está decir, que se cumple estrictamente con el principio de linealidad, mediante el cual $\boldsymbol{x}$ es representado como una combinación lineal de las *funciones base*.

Por otra parte, el Teorema de Parseval sostiene que una transformada es una rotación del vector señal alrededor del origen de un espacio vectorial – en este caso y en función de las dimensiones involucradas, de dimensiones – (*nfI/nfm*) x (*ncI/ncm*) [149, 36].

Una transformación lineal es separable, si la transformada de un bloque de la señal de dimensiones (*nfI/nfm*) x (*ncI/ncm*) puede ser expresada por

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}\,\boldsymbol{A}^{T} \tag{B.6}$$

siendo $\boldsymbol{y}$ la matriz de coeficientes transformados y $\boldsymbol{A}$ la matriz de transformación ortonormal.

La transformación inversa será entonces

$$\boldsymbol{x} = \boldsymbol{A}^{T}\ \boldsymbol{y}\,\boldsymbol{A} \tag{B.7}$$

Es de vital importancia en la práctica el hecho de que la transformada requiere dos multiplicaciones de matrices de dimensiones (*nfI/nfm*) x (*ncI/ncm*) en lugar de una única multiplicación de un vector de dimensión *1* x ((*nfI/nfm*)x(*ncI/ncm*)) con una matriz de dimensión ((*nfI/nfm*)x(*ncI/ncm*)) x ((*nfI/nfm*)x(*ncI/ncm*)) obteniendo se así una reducción en la complejidad de O((*nfI/nfm*)x(*ncI/ncm*)x(*nfI/nfm*)x(*ncI/ncm*)) a O((*nfI/nfm*)x(*ncI/ncm*)x(*nfI/nfm*)), asumiendo para este último caso que *nfI = ncI* y *nfm = ncm.*

Otro aspecto importante es que la transformada bidimensional puede ser implementada como dos transformadas unidimensionales a lo largo de las filas y columnas del bloque de señal (Fig.B.4).

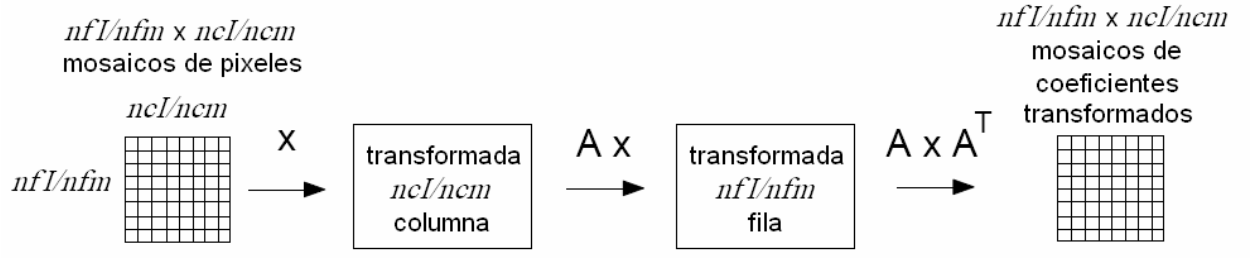


Figura B.4: Transformada ortonormal separable.

Las ecuaciones B.4 a B.7 constituyen las propiedades de las transformadas involucradas en el presente trabajo.

Los criterios de selección de una transformada particular son:

- Decorrelación, concentración de la energía (por ejemplo, TDKL, TDC, otras)

- Funciones base visualmente agradables (por ejemplo, ruido pseudo-aleatorio, *m*-secuencias, transformadas solapadas). Ver. Fig.B.5 [407-414, 401], donde TH es la Transformada de Haar, TWH es la Transformada de Walsh-Hadamard y TS es la transformada de Slant [149, 36]. Para todos los casos las funciones base son consideradas en relación a bloques de 8 x 8.

- Baja complejidad computacional.

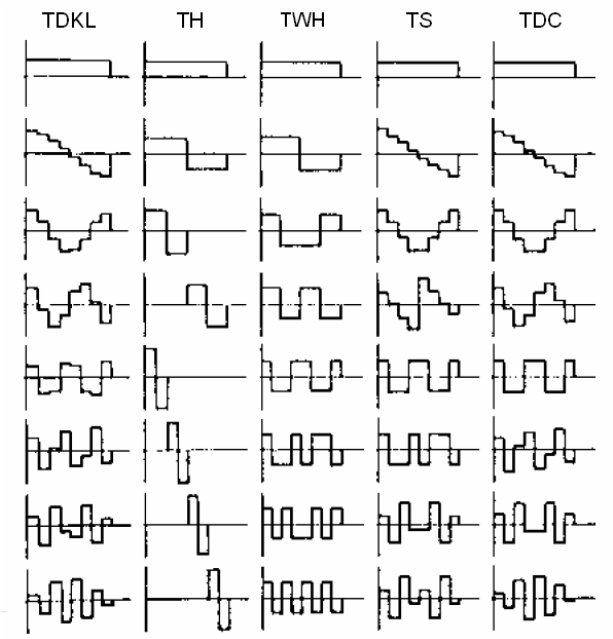


Figura B.5: Bases de varias transformadas.

La concentración de la energía medida para imágenes naturales típicas, para bloques de medida 1x32 [149, 36], puede observarse en la Fig.B.6 donde la TDKL es óptima y la TDC solo es superada por la TDKL.

Finalmente en la codificación por transformada adaptativa (Fig.B.7) la cuantización y la codificación entrópica son optimizadas separadamente para cada clase.

Las clases típicas serán:
• Bloques sin detalles
• Estructuras horizontales
• Estructuras verticales
• Diagonales
• Texturas sin una orientación predilecta

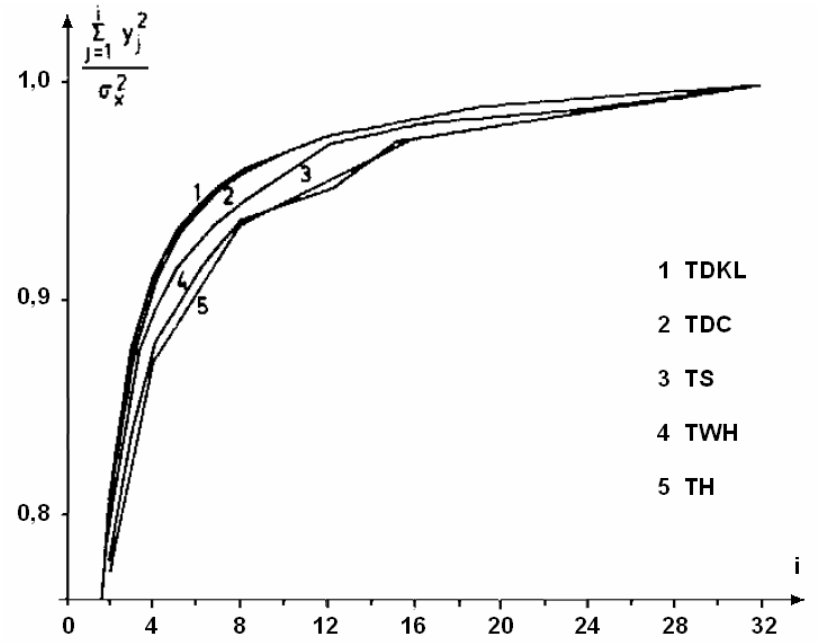


Figura B.6: Concentración de energía medida para varias transformadas.

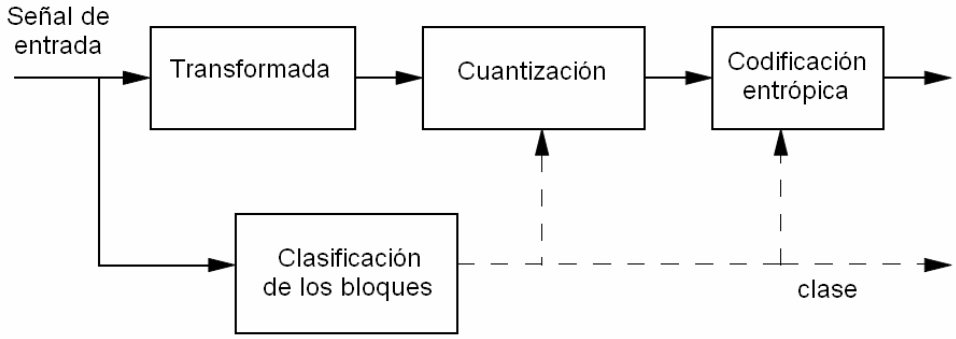


Figura B.7: Codificación por transformada adaptativa.

B.2.2 Paso de cuantización en compresión de imágenes con pérdidas.

La cuantización reduce la profundidad en bits de los coeficientes de los bloques a expensas de la precisión. Dos diferentes procedimientos de cuantización están permitidos en estos casos:

a) Cuantización escalar, y
b) Cuantización basada en el código de Trellis

La cuantización escalar consiste de un simple truncamiento de los bits menos significativos, frecuentemente obtenidos por un desplazamiento a la derecha de la magnitud de los coeficientes de los bloques [415]. Los coeficientes difieren solo en los dígitos que se eliminaron siendo esto indistinguible luego de la decuantización [415]. Esto resulta en una función truncada de a pasos como la mostrada en la Fig.B.8. La cuantización escalar sufre del así llamado problema de la *zona-muerta* [415]. A saber, si el paso de cuantización Δ_b es usado por un bloque *b*, los coeficientes cuya magnitud cae por debajo del umbral son convertidos a cero. Entonces la información acerca de los coeficientes en una esfera de radio Δ_b alrededor de cero se perderá

completamente, ver Fig.B.8. Siendo $p[n]$ el número de desplazamientos a la derecha para el coeficiente *n-ésimo*, $s[n]$, en el *b-ésimo* bloque, entonces, la cuantización escalar puede ser expresada por la fórmula:

$$\boldsymbol{q}^{p[n]}[\boldsymbol{n}] = \boldsymbol{signo}(\boldsymbol{s}[\boldsymbol{n}]) \times \left\lfloor \frac{|\boldsymbol{s}[\boldsymbol{n}]|}{\Delta_{\boldsymbol{b}}^{p[n]}} \right\rfloor \tag{B.8}$$

donde:

$\lfloor \bullet \rfloor$ significa "*redondear hacia abajo*", y

$$\Delta_{\boldsymbol{b}}^{p[n]} = 2^{p[n]} \times \Delta_{\boldsymbol{b}} \tag{B.9}$$

La ecuación de decuantización será:

$$\hat{\boldsymbol{s}}[\boldsymbol{n}] = \begin{cases} \left\lceil \left(\boldsymbol{q}^{p[n]}[\boldsymbol{n}] - \boldsymbol{r} \times 2^{M_b - p[n]} \right) \times \Delta_{\boldsymbol{b}}^{p[n]} \right\rceil & \boldsymbol{q}^{p[n]}[\boldsymbol{n}] < 0 \\ 0 & \boldsymbol{q}^{p[n]}[\boldsymbol{n}] = 0 \\ \left\lfloor \left(\boldsymbol{q}^{p[n]}[\boldsymbol{n}] + \boldsymbol{r} \times 2^{M_b - p[n]} \right) \times \Delta_{\boldsymbol{b}}^{p[n]} \right\rfloor & \boldsymbol{q}^{p[n]}[\boldsymbol{n}] > 0 \end{cases} \tag{B.10}$$

donde:

$\lceil \bullet \rceil$ significa "*redondear hacia arriba*", y

M_b es el número máximo de bits en la magnitud de los coeficientes y $\boldsymbol{r} \in [0,1)$ es usualmente establecido como $\boldsymbol{r}$ = 0.5. La cuantización basada en el código de Trellis es mucho más complicada pero usualmente da mejores resultados y no sufre del problema de la *zona-muerta*. Para más detalles acerca de esta cuantización se puede recurrir a [415].

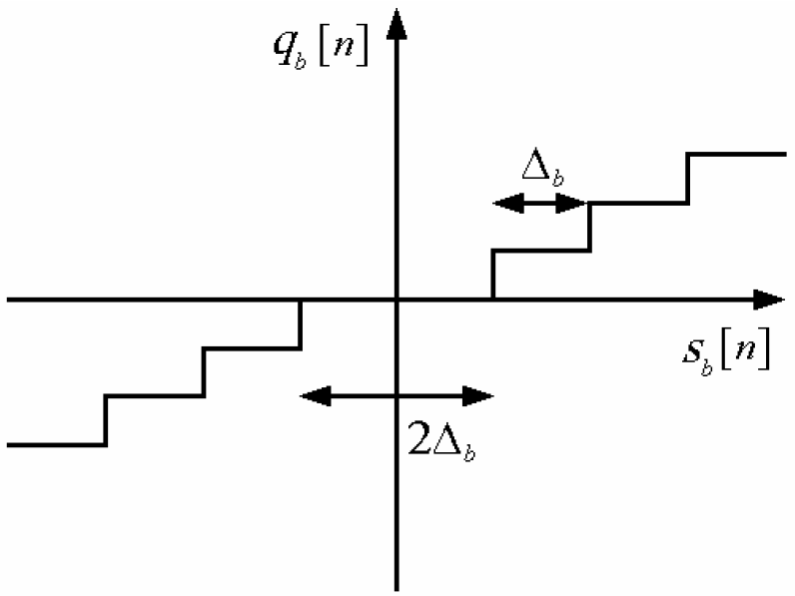


Figura B.8: Cuantización escalar: Una *zona-muerta* de $2 \times \Delta_{\boldsymbol{b}}$ es generada entorno de cero. $s_b[n]$ indica el valor de entrada, mientras que $q_b[n]$ es la señal cuantizada de salida.

B.2.3 Compresión entrópica o remoción de la redundancia

La compresión entrópica no produce pérdidas en el esquema de compresión planteado hasta el momento. Por otra parte, la compresión entrópica asigna a cada símbolo un código de longitud promedio igual a la entropía de la fuente [416]. Los compresores entrópicos constan de dos partes fundamentales [416] cuya interacción se describe gráficamente en la Figura B.9:

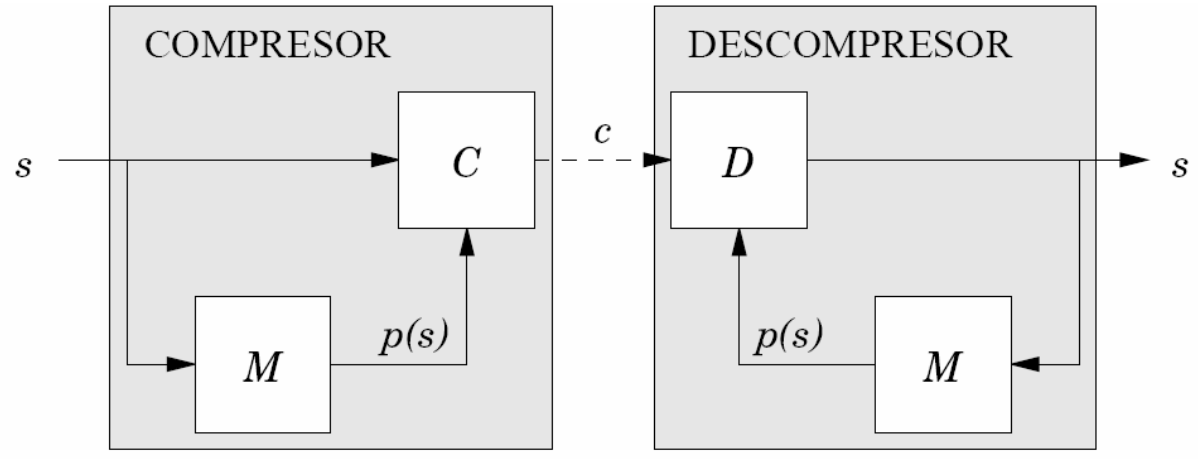


Figura B.9: Modelo de codificación/decodificación entrópica.

- Un modelo probabilístico *M* que asigna a cada símbolo de entrada *s* una probabilidad de ocurrencia *p*(*s*), bajo ciertas restricciones [416], es decir, dado que el codificador tratará de asignar al símbolo-fuente *s* codificado $-\log_2 p(s)$ bits de código. Para que la codificación sea posible, debe ocurrir que $0 < p(s) < 1$:

  1. Si $p(s) = 0$, el símbolo-fuente sería codificado con una palabra-código de longitud infinita. Además, *s* no pertenecería al alfabeto-fuente.

  2. Si $p(s) = 1$, la palabra-código generada obtendría una longitud de 0 bits y el decodificador sería incapaz de detectarla. Por otra parte, el alfabeto fuente sólo contendría un símbolo-fuente.

- Un codificador *C* que realiza las funciones de traductor, capaz de asignar a cada símbolo de entrada *s* un código *c* de longitud ideal igual a la expresada por

  $$\boldsymbol{l}_s(\boldsymbol{c}_i) = -\log_s \boldsymbol{p}(\boldsymbol{a}_i) \qquad \text{(B.11)}$$

  Donde las longitudes de las palabras-código tienen que ser igual a la información asociada al símbolo-fuente representado. En otras palabras, la longitud debe ser igual a la entropía de la fuente expresada en bits de código/símbolo. El traductor inverso o decodificador *D* recupera la representación original de los símbolos, conociendo la misma información de contexto que el traductor directo, que es proporcionada por un modelo probabilística idéntico al del descompresor.

La clave para obtener el máximo rendimiento en un compresor entrópico es la estimación exacta de las probabilidades de los símbolos. Si es demasiado baja, el código asignado aumenta de longitud y como el símbolo asociado ocurre en más ocasiones de las esperadas, la longitud de la

secuencia-código aumenta. Por otra parte, si la probabilidad es demasiado alta, estaremos codificando un símbolo que no es tan frecuente como creemos mediante un código muy corto, a costa de alargar la longitud de los códigos asignados al resto de símbolos que tendrán asignada una probabilidad menor que la real.

El rendimiento de los compresores entrópicos depende tanto de los codificadores de longitud variable como de los modelos probabilísticos usados, pero su independencia es tal que pueden ser estudiados por separado. Por otra parte, los compresores entrópicos en general son más lentos que los basados en diccionarios. Esto se debe a que ahora se busca la máxima compresión posible, incluso a costa de desarrollar algoritmos pesados que permitan aprovechar ese "grado de optimalidad" [416], y que los compresores basados en diccionarios no explotan sobre secuencias de longitud finita.

Los conceptos e ideas introducidos en esta sección son fundamentales para el diseño de los compresores de imágenes reversibles que serán analizados en el siguiente capítulo. Sin embargo, en éste se analizarán aquellos métodos de compresión que pueden ser considerados como el "estado del arte" en compresión de texto [417-446].

B.2.3.1 Capacidad de un canal discreto sin ruido

En la teoría de la información, el canal de comunicación constituye el enlace entre el codificador del emisor y el decodificador del receptor. Un canal discreto transmite símbolos-código (bits de código cuando sea binario, como casi siempre ocurre). Si además no existe ruido de transmisión, a la salida tenemos los mismos símbolos-código y en la misma secuencia (orden) que a la entrada, transcurrido un cierto tiempo. Puesto que no existen alteraciones durante la transmisión, podemos afirmar que la cantidad de información de salida es igual a la de entrada.

En el contexto de la compresión de datos, el canal simboliza el dispositivo de transmisión o de almacenamiento. Por tanto, la compresión de datos va a ser óptima si aprovechamos al máximo la capacidad de transmisión del canal.

Se define la capacidad de un canal como la máxima cantidad de información que puede transmitirse en cierto intervalo de tiempo. Todos los canales, incluso los ruidosos, pueden transmitir una cantidad de información infinita durante un periodo de tiempo ilimitado, así que lo normal es hablar de cantidad de información transmitida por unidad de tiempo. Este es el concepto de capacidad de un canal manejado en muchos ámbitos de transmisión de datos, independientemente del tipo de canal (analógico o digital).

Sin embargo, para definir la capacidad de un canal discreto, se puede hablar de capacidad en términos de la cantidad de información transmitida por símbolo-código o palabra código enviada. Cuando la duración o longitud de los símbolos-código es constante y tenemos un total de s símbolos-código, la capacidad de un canal puede calcularse como

$$\boldsymbol{W} = \log_2 \boldsymbol{s} \qquad \text{(B.12)}$$

cantidad medida en bits de información/bits de código. En el caso de trabajar con códigos y fuentes binarias, la capacidad de un canal (sin ruido) es de 1 bit de información/bit de código transmitido. En este caso, se está utilizando un código óptimo desde el punto de vista de la compresión de datos.

Es interesante resaltar que las dimensiones de la capacidad de un canal y de la eficiencia de un código coinciden.

B.2.3.2 Un codificador entrópico universal

Antes de comenzar a estudiar los diferentes codificadores entrópicos usados en compresión de datos, vamos a adelantar la idea que es la esencia de todos ellos. Esto nos va a permitir comprender con más facilidad cómo funcionan, qué esperar de ellos y en qué se parecen todos los compresores entre si.

Como fue expresado en la sección anterior, un código no es redundante si cada bit de información es representado exactamente por un bit de código. Los compresores basados en diccionarios no estiman las probabilidades de los símbolos y por lo tanto es difícil entender que la afirmación anterior también es cierta en este caso, pero así es. Cuando las cadenas son sustituidas por índices, lo que en realidad se hace es codificar un gran símbolo-fuente compuesto por varios caracteres usando un código de longitud constante de al menos 1 bit.

El mismo efecto puede obtenerse en el caso de la codificación entrópica porque cada carácter se codifica usando un código de longitud variable. Ahora estimamos las probabilidades de los símbolos (caracteres) y construimos un código de longitud igual al número de bits de información correspondientes.

Por lo tanto, el objetivo de todo codificador es encontrar una representación lo más compacta posible para un determinado símbolo *s*. En teoría, debería ser posible asignar fracciones de bit de código dependiendo de su probabilidad de ocurrencia. La ventaja principal de los compresores basados en diccionarios sobre los entrópicos es que al extender la fuente no es necesario trabajar con fracciones de bits lo que produce el correspondiente aumento en velocidad. Bajo estas condiciones, sean los siguientes algoritmos: Algoritmo B.1 de codificación y Algoritmo B.2 de decodificación:

Algoritmo B.1

Mientras *s* no esté determinado sin incertidumbre (por el decodificador):

(a) Realizar una afirmación sobre *s* que permita al decodificador averiguar algo acerca de la identidad de *s*. Intentar que dicha afirmación tenga tantas posibilidades de ser cierta como de ser falsa.

(b) Emitir un bit de código indicando el resultado de la afirmación.

El decodificador asociado debería constar de los siguientes pasos:

Algoritmo B.2

Mientras *s* no esté determinado sin incertidumbre:

(a) Realizar la misma afirmación que el decodificador.

(b) Recibir un bit de código indicando el resultado de la información.

No es difícil apreciar que si se cumple la premisa de que la afirmación sea equiprobable, estaremos encontrando un código 100% eficiente. Sin embargo, lo que ocurre en la práctica es que formular afirmaciones ciertas exactamente en un 50% es muy complicado y por tanto el código contiene cierto nivel de redundancia.

Planteemos una implementación sencilla de los anteriores algoritmos. Supongamos que trabajamos con un alfabeto de 256 símbolos y que conocemos sus probabilidades ya que existe un modelo que las calcula. El paso no trivial que hay que describir es el de la formulación de la afirmación equiprobable. Supongamos que ordenamos los símbolos por su probabilidad de forma decreciente. Una afirmación lo más equiprobable posible sería: “el símbolo a codificar pertenece al conjunto formado por uno o más símbolos que se forma cuando recorremos la lista de símbolos y la suma de las probabilidades de todos ellos alcanza el valor más próximo a 0.5”.

El decodificador puede hacer exactamente la misma predicción porque para formarla no se usa el símbolo codificado. Así, con la llegada del bit, el decodificador conoce si el símbolo está en el primer conjunto o en el segundo. El símbolo es encontrado cuando en el conjunto seleccionado sólo existe un símbolo.

Una forma eficiente de realizar las particiones (binarias) en nuestra lista ordenada es representarla en como un árbol binario en el que los símbolos más probables estén más cerca de la raíz y los más improbables estén lo más alejados de ésta como sea posible. En realidad, justamente esto es lo que hace la codificación de Huffman.

B.2.3.3 La codificación de Huffman

El código de Huffman [417] fue inventado por David A. Huffman en 1952 y desde entonces ha sido intensivamente utilizado en compresión de datos, porque genera un código de longitud entera instantáneo y óptimo.

B.2.3.4 El codificador

Huffman ideó un método para construir un árbol binario donde cada hoja representa a un símbolo, con la propiedad de que la distancia de una hoja s a la raíz del árbol es exactamente

$$\lceil -\log_2 \boldsymbol{p}(\boldsymbol{s}) \rceil \tag{B.13}$$

A continuación asignó un dígito binario a cada rama del árbol y diseño así un código de longitud variable que representa a los símbolos más probables con un código más corto y viceversa. El algoritmo de construcción del árbol de Huffman es el siguiente:

1. Crear una lista con tantos nodos como símbolos diferentes vayamos a codificar. Cada nodo representa a un símbolo diferente y almacena su probabilidad.

2. Mientras existan al menos dos nodos en la lista:

   (a) Extraer de la lista los dos nodos con menor probabilidad. Si existen más de dos, la elección es irrelevante.

   (b) Insertar en la lista un nodo que sea padre de los dos nodos extraídos, formando un árbol binario equilibrado de tres nodos. Este nuevo nodo tiene una probabilidad que es la suma de las probabilidades de los hijos y no representa a un símbolo en concreto.

La Figura B.10 muestra un ejemplo de construcción de un árbol de Huffman. Por comodidad trabajaremos con pesos enteros y no con probabilidades reales. Supongamos que codificamos 5 símbolos representados por A, B, C, D y E con pesos 15, 7, 6, 6 y 5 respectivamente. El primer

paso (Figura B.10-a) consiste en crear una lista de 5 nodos. Gráficamente esta lista estaría formada por los nodos que queden en el nivel superior.

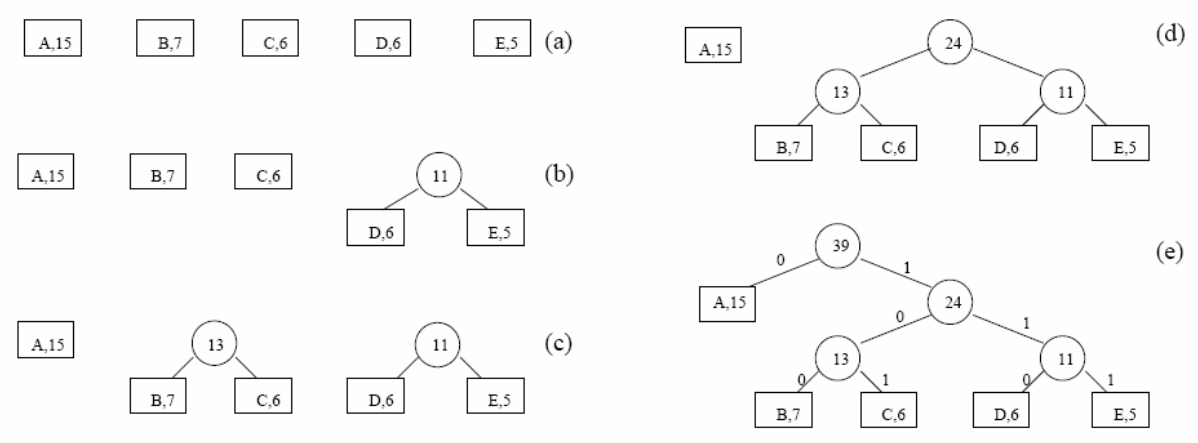


Figura B.10: Ejemplo de construcción de un árbol de Huffman.

A continuación extraemos los nodos con símbolos D y E, que son los de menor peso. También podríamos seleccionar el nodo C y el árbol de Huffman resultante sería distinto pero equivalente desde el punto de vista de la compresión de datos. Los pesos de D y E son 6 y 5 y por lo tanto, insertamos un nuevo nodo en la lista con peso 11 (Figura B.10-b). Los dos siguientes nodos utilizados para construir un nuevo subárbol son el B y el C (Figura B.10-c). A continuación, los dos subárboles forman un árbol mayor (Figura B.10-d) y por último, el nodo A se procesa quedando el árbol de Huffman completo (Figura B.10-e). El número de iteraciones necesarias para crear un árbol de Huffman que codifique *N* símbolos es *N*, ya que en cada iteración extraemos dos nodos de la lista e insertamos uno.

Una vez empleada la información que proporciona el modelo para construir el árbol es posible codificar y descodificar. La codificación de un símbolo consiste en emitir el código formado por la concatenación de las etiquetas de cada una de las ramas que es necesario recorrer desde la raíz hasta la hoja que representa al símbolo. Por ejemplo, si las ramas izquierdas se etiquetan con el bit de código 0 y las derechas con el 1 (ver Figura B.10-e), el código de Huffman calculado es c_A = 0, c_B = 100, c_C = 101, c_D = 110 y c_E = 111.

B.2.3.5 El decodificador

La descodificación es justamente el proceso inverso. Un código describe con los bits que lo forman un camino desde la raíz hasta una hoja del árbol de Huffman. El símbolo descodificado es el asociado a la hoja.

Por ejemplo, supongamos que codificamos la secuencia de símbolos ABACDE. La secuencia de códigos resultante es (teniendo en cuenta el árbol de Huffman anteriormente construido): 01000101110111. Para descodificarla, inicialmente nos situamos en la raíz del árbol. Como el primer bit es un 0, el primer símbolo descodificado es una A (ver Figura B.10). Para decodificar el siguiente símbolo nos situamos de nuevo en la raíz y utilizamos los bits 100 para llegar hasta el símbolo B. Este proceso se repite hasta que no existen más bits de código que descodificar.

B.2.3.6 Fuentes de información

Una fuente de información discreta es aquella que produce mensajes como sucesiones de

símbolos-fuente (a las que llamaremos en general secuencias-fuente) extraídos de entre una colección finita de símbolos-fuente que conforman un alfabeto-fuente. La notación

$$\boldsymbol{A} = \{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r\} \tag{B.14}$$

indica un alfabeto-fuente *A* compuesto de *r* símbolos-fuente a_i. A *r* se le llama base de la fuente de información discreta. Nótese que hablar de bits de información no significa tomar *r* = 2.

En el ámbito de la teoría de la información, la emisión de símbolos para la construcción de mensajes se considera gobernada por un proceso estocástico, lo que quiere decir que la fuente de información produce los símbolos-fuente de acuerdo con unas ciertas probabilidades

$$\boldsymbol{P} = \{\boldsymbol{p}(\boldsymbol{a}_1), \cdots, \boldsymbol{p}(\boldsymbol{a}_r)\} \tag{B.15}$$

La probabilidad asociada a un símbolo-fuente debe ser mayor que cero, para que el símbolo-fuente tenga razón de existir como tal en el alfabeto-fuente (o de lo contrario nunca formaría parte de ninguna secuencia-fuente) e inferior a uno, para que existan al menos dos símbolos-fuente diferentes en el alfabeto-fuente.

Además se cumple que, teniendo una muestra (mensaje) razonablemente grande de la fuente de información, dicha muestra es representativa (estadísticamente hablando) del conjunto de todas las cadenas o secuencias de Markov que pueden ser producidas por la fuente, y por esta razón, diremos que se trata de un "proceso ergódico". Una fuente de información se dice ergódica si una muestra suficientemente corta de ella es representativa (estadísticamente hablando) de toda la información que es capaz de emitir. Gracias a esta propiedad, el famoso experimento realizado por Shannon que consistió en sintetizar mensajes escritos en inglés utilizando una fuente de este tipo, tras haber analizado suficientes secuencias de texto escrito, fue un éxito.

Los procesos de Markov nos hablan acerca de fuentes de información que generan los símbolos-fuente con una probabilidad que puede depender o no de los símbolos anteriormente producidos. Cuando, la probabilidad de un símbolo-fuente no depende del previamente emitido, hablamos de fuentes sin memoria, de fuentes de memoria nula o de fuentes de Markov de orden 0. Cuando ocurre lo contrario, hablamos de fuentes con memoria o de fuentes de Markov de orden *K*, donde *K* es el número de símbolos-fuente que forman el contexto que determina la probabilidad de generación del siguiente símbolo-fuente.

B.2.3.7 Compresión adaptativa

La longitud de los códigos de Huffman depende de la correcta estimación de las probabilidades de los símbolos. Existen fuentes que emiten los símbolos cuyas probabilidades permanecen constantes (fuentes ergódicas, ver Sección anterior) y por lo tanto puede usarse un único árbol de Huffman para codificarla eficientemente. El modelo probabilístico se calcula una única vez o es conocido de antemano tal como ocurre en la compresión de imágenes [447-449]. Los modelos que permanecen inalterables durante el proceso de compresión se llaman modelos estáticos.

Cuando las distribuciones de probabilidad no son conocidas de antemano, la utilización de modelos estáticos requiere que la secuencia a comprimir sea recorrida dos veces. En la primera pasada se calculan las probabilidades de los símbolos y en la segunda se codifica. El modelo debe ser comunicado de forma explícita al descompresor, típicamente en forma de cabecera previa a la secuencia de códigos.

El otro tipo de modelos probabilísticos que existen se llaman modelos dinámicos o adaptativos. En general proporcionan mejores tasas de compresión que los estáticos debido fundamentalmente a que las fuentes de información no son ergódicas. Sin embargo, también provocan procesos de compresión y descompresión más lentos, pues hay que emplear recursos para gestionar el modelo.

La ventaja más interesante de usar modelos adaptativos es que pueden estimar las probabilidades de los símbolos sin necesidad de realizar dos pasadas (diremos en estos casos que la secuencia puede ser tratada como una corriente o stream), sin embargo, cuando la fuente es ergódica, los modelos estáticos son realmente eficientes, su rendimiento es comparable al de los no adaptativos [418-420].

La idea que hay detrás de la compresión entrópica adaptativa es la misma que la que usan los compresores basados en diccionarios: codificar usando sólo la información que hasta ese instante conoce el descompresor. El Algoritmo B.3 de compresión adaptativa realiza los siguientes pasos:

Algoritmo B.3

1. Inicialmente todos los símbolos son equiprobables y su probabilidad es mayor que 0.
2. Mientras existan símbolos que codificar:
   (a) Codificar el siguiente símbolo.
   (b) Actualizar la probabilidad del símbolo codificado.

Mientras que el Algoritmo B.4 de descompresión adaptativa consiste en:

Algoritmo B.4

1. Idéntico al paso 1 del compresor.
2. Mientras existan símbolos que decodificar:
   (c) Decodificar el siguiente símbolo.
   (d) Actualizar la probabilidad del símbolo decodificado.

B.2.3.8 Actualización en el árbol de Huffman

La actualización de un símbolo en el árbol provoca que la probabilidad del símbolo actualizado y de todos los nodos internos que son antecesores (hasta llegar a la raíz) del símbolo sean incrementadas. La modificación de los pesos de los nodos puede provocar que los nodos afectados tengan que ascender por el árbol en dirección a la raíz, lo que acarrea una modificación de su estructura.

Usar el algoritmo de construcción presentado en la Sección **B.2.3.3** es demasiado costoso pues su complejidad es de $O(n^2)$ si n es el número de símbolos contemplados. Para acelerar este proceso, el árbol se representa de forma ordenada atendiendo a los pesos de los nodos usando un *array* $a[]$ con $2 \times n \cdot 1$ elementos [421-424]. El nodo raíz se almacena en $a[0]$, su hijo de mayor peso en $a[1]$ y el de menor en $a[2]$. A continuación se hace lo mismo con los hijos, comenzando por el de más peso, siempre que los nodos no sean hojas. Como resultado el array se ordena de forma decreciente y siempre debería cumplir esta propiedad.

Usando *a*[], es sencillo comprobar si la actualización de un símbolo provoca una modificación de la estructura del árbol porque esto sólo ocurre cuando el array deja de estar ordenado. En dicho caso, el nodo cuyo peso supera al que está por encima de él en el array debe ser intercambiado con el actualizado. Para comprender mejor este proceso, en la Figura B.11-(A) se muestra un ejemplo.

Cuando incrementamos en una unidad el peso del símbolo *a* (lo que llamamos una actualización en un algoritmo de compresión adaptativo), tenemos que incrementar el peso también de su nodo padre, de su nodo abuelo y así sucesivamente hasta llegar a la raíz, tal y como se describe en la Figura B.11-(B). Si incrementamos de nuevo el símbolo *a* se viola la propiedad de que se puedan recorrer los nodos en orden decreciente (se degenera el árbol). Por esta razón, antes de continuar con la actualización de los pesos hasta la raíz, los nodos 8 y 5 deben ser intercambiados (Figura B.11-(C)). Después del intercambio el proceso de actualización continua, comprobándose en cada actualización el cumplimiento de la propiedad del recorrido ordenado. De esta forma se consigue el árbol de Huffman de la Figura B.11-(D).

Hasta ahora, la forma del árbol no ha variado ya que el incremento del peso de una de las hojas es insuficiente para mover un nodo interno. Para que esto ocurra vemos que por ejemplo, el símbolo *a* debe ser incrementado al menos dos veces más. Tras el primer incremento las cosas quedan como se indica en la Figura B.11-(E). Como puede verse no se incumple la propiedad del recorrido ordenado. Pero el segundo incremento de *a* provoca varios intercambios. En primer lugar, al acumular un peso igual a 5, debe ser intercambiado por el nodo almacenado en 4. Dicha circunstancia es la que se muestra en la Figura B.11-(F). Tras el intercambio continua la propagación del incremento del nodo padre del símbolo *a* situado en la posición 2 (Figura B.11-(G)). El incremento de este nodo provoca de nuevo la violación del recorrido ordenado y los nodos 1 y 2 deben ser intercambiados (Figura B.11-(H)). Cuando éste ya se ha realizado, se produce el incremento del nodo raíz (Figura B.11-(I)) y el árbol de Huffman queda construido (Figura B.11-(J)). Debido a que normalmente se utilizan datos de tipo entero (con pocos bits de precisión) para almacenar la probabilidad de un símbolo, es necesario ejecutar algún proceso de escalado de todas las probabilidades de los nodos del árbol de Huffman y así evitar desbordamientos. Dicho proceso consiste típicamente en dividir (truncando) todos los recuentos entre 2. Esta operación no modifica la forma del árbol y además tiene un efecto beneficioso. Es frecuente encontrar secuencias de símbolos en las que muchos de ellos aparecen en zonas determinadas de la secuencia, como consecuencia de que las fuentes no son realmente ergódicas. La operación de escalado provoca que debido a los truncamientos, las probabilidades de estos símbolos sean menores que si no se produjera escalado, y esto ocurre precisamente cuando no son utilizados. Visualmente lo que ocurre es que dichos símbolos se “hunden” en el árbol más rápidamente y dejan que los símbolos más usados puedan codificarse con menos bits.

B.2.3.9 Extensión de una fuente de información

La extensión de una fuente es el nombre que recibe el proceso de construir una fuente llamada fuente extendida a partir de una fuente de información, generando cada símbolo-fuente extendido como la concatenación de dos o más símbolos-fuente. Como consecuencia, la base de la fuente extendida es r^N, donde N es el orden de la extensión y r es el tamaño del alfabeto. El objetivo de extender una fuente es aumentar suficientemente el número de símbolos-fuente. Como veremos, haciendo esto es posible encontrar representaciones más eficientes para las secuencias-fuente originales. Por ejemplo, si el alfabeto-fuente $a = \{a_1, a_2, a_3\}$, el alfabeto-fuente extendido a^2 de orden 2 será

$$a^2 = \{a_1a_1, a_1a_2, a_1a_3, a_2a_1, a_2a_2, a_2a_3, a_3a_1, a_3a_2, a_3a_3\} \quad \text{(B.16)}$$

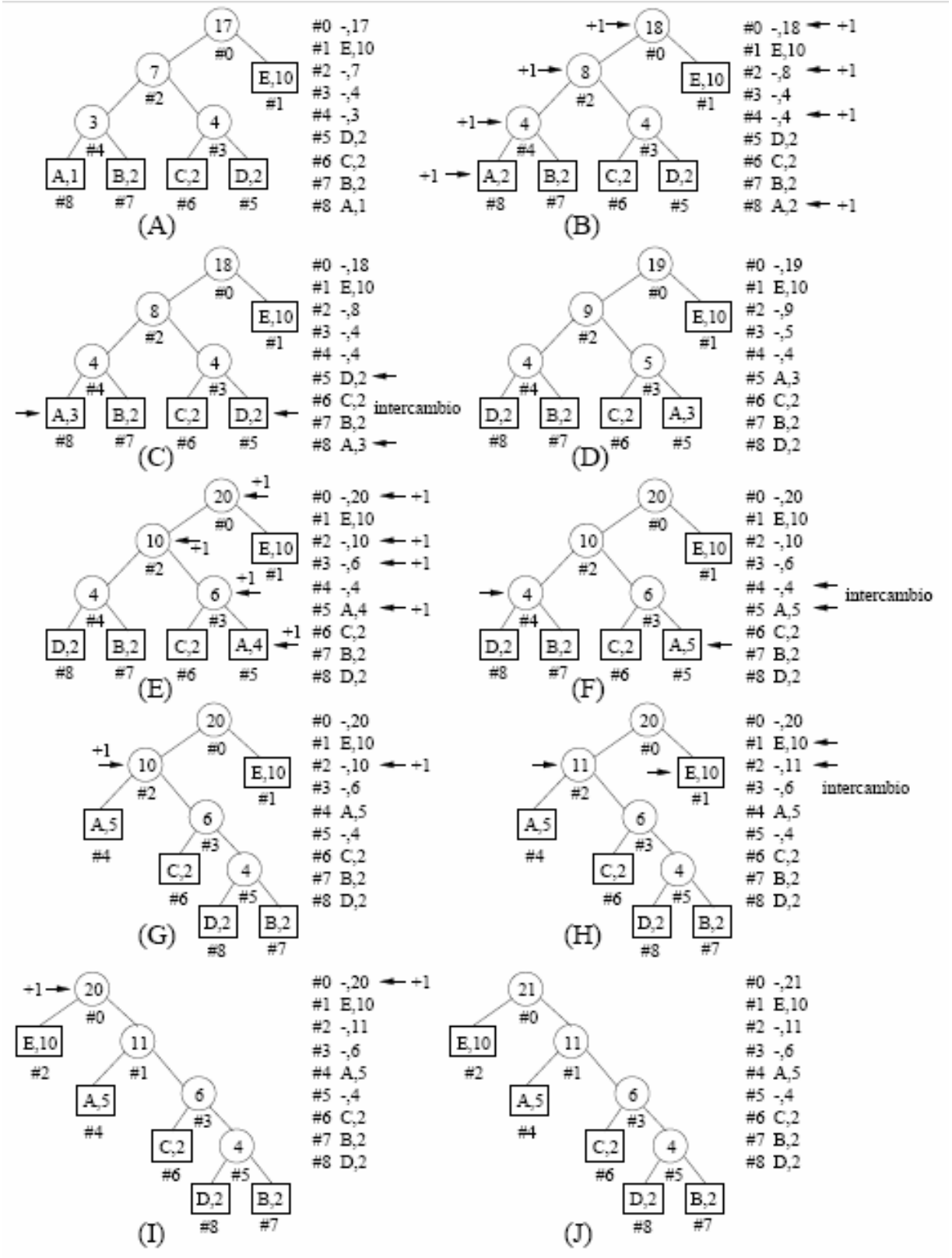


Figura B.11: Ejemplo de actualización de un árbol de Huffman.

B.2.3.10 La codificación aritmética

La codificación aritmética fue introducida por Abramson [425] en 1963 y desarrollada por Pasco [426] en 1976. Desde entonces, ha recibido una atención especial porque es una forma de

representación de la información muy eficiente y permite una independencia máxima entre el modelo probabilístico y el codificador.

Supongamos que necesitamos codificar una fuente de información binaria (por ejemplo, una imagen de fax) que emite símbolos con probabilidades desiguales. La codificación de Huffman es incapaz de expresar eficientemente esta fuente de información porque representaría cada símbolo con 1 bit de código al menos que es el tamaño de la representación actual. La solución dada a este problema durante mucho tiempo ha consistido en extender primero la fuente (ver la Sección anterior) y luego usar la codificación de Huffman sobre un alfabeto mayor. La eficiencia de la solución sólo es ideal cuando el orden de la extensión es infinito y por tanto, nunca podemos representar la fuente sin redundancia.

Abramson, Pasco y más tarde Rissanen y Landon [427, 428] desarrollaron un método de codificación que permitía codificar fuentes binarias sin necesidad de extender la fuente. El método se llamó codificación aritmética binaria. La idea es muy sencilla, aunque sobre su implementación software y hardware se han escrito decenas de artículos [429-440].

Consiste en asignar a cada posible secuencia de símbolos un subintervalo dentro del intervalo [0, 1) cuyo tamaño es igual al producto de todas las probabilidades de los símbolos que forman la secuencia. La posición del subintervalo dentro del intervalo real [0, 1) está determinado por la secuencia concreta de símbolos. El algoritmo básico de codificación consiste en emitir un número cualquiera perteneciente al subintervalo final. El número de bits de código necesarios para representar el código aritmético, es igual a la entropía de la secuencia de símbolos y por esta razón, la codificación aritmética se considera 100% eficiente.

B.2.3.11 La codificación aritmética binaria

Históricamente la codificación aritmética binaria ha recibido una atención especial debido a las siguientes razones:

- El cálculo del intervalo siguiente sólo afecta en una iteración a *L* (símbolo más bajo) o a *H* (símbolo más alto), pero nunca a ambos: si el símbolo a codificar es un 0, sólo hay que calcular *H* y viceversa. El tiempo de cálculo del intervalo siguiente se divide por tanto por la mitad aproximadamente.

- Se puede utilizar para codificar fuentes de información multisímbolo si construimos un árbol de Huffman en el que cada nodo tiene asociada una probabilidad de transición hacia su rama izquierda o derecha [441]. Se usa un árbol de Huffman porque esta estructura minimiza el número de decisiones para codificar un símbolo en función de su probabilidad. Cada vez que atravesamos un nodo interno, se codifica aritméticamente la probabilidad de la rama escogida que depende del nodo asociado.

- Los modelos probabilísticos para dos símbolos son muy sencillos ya que con una única probabilidad se describe a la fuente. Además, en el proceso de descodificación la búsqueda del subintervalo que contiene el código aritmético se trivializa porque sólo existen dos alternativas.

Estas circunstancias (y en especial la última) han provocado que la mayoría de las implementaciones físicas sean para codificadores y decodificadores binarios y el ejemplo más claro lo tenemos en el Q-Coder [442-445]. Sin embargo, cuando los alfabetos son multisímbolo (256

típicamente), el codificador aritmético multisímbolo supera en velocidad a la versión binaria. Esto es especialmente cierto en el caso de una implementación software en la que es mucho más determinante el número de instrucciones ejecutadas que la complejidad aritmética de éstas. Incluso en el caso de una codificación de fuentes binarias, la menor velocidad de funcionamiento del algoritmo multisímbolo puede ser suplida procesando la fuente de forma extendida, incluso tratándose de implementaciones hardware [446].

B.3 Conclusiones del apéndice

En este apéndice se analizaron todos los elementos complementariamente y no desarrollados en el Capítulo 1 en función del esquema de compresión/descompresión basado en la TDKL asociados a la Fig.1.7 de la Sección 1.2.3.

Apéndice C

C Implementaciones rápidas de ACP

C.1 Introducción

Como se mencionó en la Sección 1.2.5.2, en este apéndice describiremos en detalle las dos familias de aproximaciones rápidas al ACP.

C.2 Aproximaciones rápidas al ACP

C.2.1 Versiones conexionistas

C.2.1.1 Algoritmo de Oja y Karhunen

Si la secuencia de entrada esta compuesta por componentes de $\boldsymbol{x}$ normales de valor medio nulo, entonces este algoritmo [134-136] consiste en el siguiente modelo Hebbiano

$$\boldsymbol{V}_{t+1} = \boldsymbol{V}_t + \eta_t\left[-\partial_{\boldsymbol{V}}\boldsymbol{J}(\boldsymbol{V}_t)\right] \tag{C.1}$$

donde todas las variables involucradas fueron definidas en el Capítulo 1, mientras que el parámetro η_t se define como la tasa de aprendizaje. Este modelo no converge a las componentes principales, excepto para un espacio vectorial de un solo componente. [134-136].

La función de mérito asociada será de la forma

$$\boldsymbol{J}(\boldsymbol{V}) = \boldsymbol{E}\left[\boldsymbol{x}^T\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{x}\right] + \frac{1}{2}\boldsymbol{tr}\left[\Lambda\boldsymbol{V}^T\boldsymbol{V}\right] \tag{C.2}$$

donde Λ está conformado por los operadores de Lagrange. Derivando $\boldsymbol{J}(\boldsymbol{V})$ con respecto a $\boldsymbol{V}$ tenemos

$$\partial_{\boldsymbol{V}}\boldsymbol{J}(\boldsymbol{V}) = \boldsymbol{E}\left[\boldsymbol{V}\boldsymbol{x}\right]\boldsymbol{x}^T + \Lambda\boldsymbol{V}^T = 0 \tag{C.3}$$

donde su versión aproximada instantánea será

$$\partial_{\boldsymbol{V}}\boldsymbol{J}(\boldsymbol{V}) = \boldsymbol{y}_t\ \boldsymbol{x}_t^T + \Lambda\boldsymbol{V}_t^T \tag{C.4}$$

despejando Λ tenemos

$$\Lambda = -\boldsymbol{y}\ \boldsymbol{y}^T \tag{C.5}$$

Reemplazando Λ en (C.4) y a su vez esta en (C.1) finalmente arribamos al modelo de Oja

$$\boldsymbol{V}_{t+1} = \boldsymbol{V}_t + \eta\left[\boldsymbol{y}_t\boldsymbol{x}_t^T - \boldsymbol{y}_t\boldsymbol{y}_t^T\boldsymbol{V}_t\right] \tag{C.6}$$

C.2.1.2 Algoritmo Hebbiano Generalizado (AHG)

Una pequeña modificación al modelo de Oja debida a Sanger [137] soluciona el problema del caso anterior convergiendo a las componentes principales. Ahora bien, dado que

$$\boldsymbol{y}_t = \boldsymbol{V}_t \ \boldsymbol{x}_t \tag{C.7}$$

Bajo ciertas limitaciones ortogonales, la matriz de pesos $\boldsymbol{V}$ para la cual asumimos que la *i-ésima* fila de $\boldsymbol{V}$ debe tener norma unitaria y debe ser ortogonal a la *j-ésima* fila, donde $j = 1, \ldots, i\text{-}1$ [450]. Usando los multiplicadores de Lagrange para incorporar las limitaciones en la función objetivo, podemos reescribir la función de mérito $\boldsymbol{J}(\boldsymbol{V})$ como [451]:

$$\boldsymbol{J}(\boldsymbol{V}) = \boldsymbol{E}\left[\boldsymbol{x}^T\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{x}\right] + \frac{1}{2}\boldsymbol{tr}\left[\Lambda\left(\boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{I}\right)\right] \tag{C.8}$$

donde la matriz de multiplicadores de Lagrange Λ está limitada en este caso a una matriz triangular inferior. Tomando gradientes de $\boldsymbol{J}(\boldsymbol{V})$ e igualando a cero

$$\partial_V \boldsymbol{J}(\boldsymbol{V}) = \boldsymbol{E}\left[\boldsymbol{V}\boldsymbol{x}\right]\boldsymbol{x}^T + \Lambda\boldsymbol{V} = 0 \tag{C.9}$$

A diferencia del caso de Oja, aplicaremos aquí las condiciones de optimalidad de [452]

$$\Lambda\left(\boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{I}\right) = 0 \tag{C.10}$$

Multiplicando por derecha a (C.9) por $\boldsymbol{V}^T$, usando (C.10), y teniendo en cuenta que Λ debe ser triangular inferior

$$\Lambda = -\boldsymbol{ti}\left(\boldsymbol{y}\,\boldsymbol{y}^T\right) \tag{C.11}$$

donde $\boldsymbol{ti}(\bullet)$ significa hacer cero a todos los elementos por encima de la diagonal principal de la matriz $(\bullet)$. Introduciendo (C.11) en (C.9) y aproximando estocásticamente la esperanza $\boldsymbol{E}[\boldsymbol{y}]$ con su estimado instantáneo (C.7), donde $\boldsymbol{x}_i \in \Re^n$ es la observación en el tiempo t

$$\partial_{V_t} \boldsymbol{J}(\boldsymbol{V}) = \boldsymbol{y}_t\,\boldsymbol{x}_t^T - \boldsymbol{ti}\left(\boldsymbol{y}_t\,\boldsymbol{y}_t^T\right)\boldsymbol{V}_t \tag{C.12}$$

El gradiente ascendente en (C.12) finalmente arriba al *algoritmo Hebbiano generalizado* (AHG) de Sanger [137]:

$$\boldsymbol{V}_{t+1} = \boldsymbol{V}_t + \eta_t\left[\boldsymbol{y}_t\boldsymbol{x}_t^T - \boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^T\right)\boldsymbol{V}_t\right] \tag{C.13}$$

Para una apropiada ganancia escalar η_t, (C.13) tenderá a converger a la solución de componentes principales conforme $\boldsymbol{t} \to \infty$; aunque su convergencia global no está probada [138]. A diferencia de Oja, AHG requiere que la *i-ésima* fila de $\boldsymbol{V}$ sea ortogonal a todas las otras filas de $\boldsymbol{V}$, es decir, que $\boldsymbol{V}$ sea ortogonal. El algoritmo de Oja converge a una base ortogonal arbitraria, no necesariamente la base de la ACP para el subespacio abarcado por los primeros $r < n$ autovalores.

Finalmente

$$\eta_t = \frac{\tau}{\boldsymbol{t} + \tau}\eta_0 \tag{C.14}$$

con η_0 y τ parámetros de ajuste positivos [138] .

C.2.1.3 Algoritmo Hebbiano Kernelizado (AHK)

La ventaja de AHG sobre el ACP convencional es que no requiere almacenar a la matriz de covarianza. Esto es particularmente útil cuando la dimensión de la matriz de covarianza es grande, como es el caso mencionado en el Capítulo 1 cuando los mosaicos en los que se divide la imagen para su tratamiento son pequeños. Similarmente, y como se verá en la Sección C.2.2.1 la formulación directa del ACP Kernelizado se torna impráctica para bloques pequeños dado que la matriz de kernel asociada es muy grande para ser almacenada en memoria. Para este caso, se reformula el AHG en un EHKR (ver Sección C.2.2.1) para obtener una aproximación eficiente de memoria del ACP Kernelizado.

Entonces, la regla actualizada de AHG de la Ecuación (C.13) se representa en el EHKR *F* como

$$\boldsymbol{V}_{t+1} = \boldsymbol{V}_t + \eta_t \left[\boldsymbol{y}_t \, \Phi(\boldsymbol{x}_t)^T - \boldsymbol{ti}\left(\boldsymbol{y}_t \boldsymbol{y}_t^T\right)\boldsymbol{V}_t \right] \tag{C.15}$$

donde las filas de $\boldsymbol{V}_t$ son ahora vectores en *F* e $\boldsymbol{y}_t = \boldsymbol{V}_t \, \Phi(\boldsymbol{x}_t)$. $\Phi(\boldsymbol{x}_t)$ es un patrón presentado en el tiempo *t* el cual es aleatoriamente seleccionado de los puntos de datos mapeados $\{\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_l)\}$. Para una más sencilla notación, asumimos que hay una función *J*(*t*) la cual mapea *t* a $\boldsymbol{i} \in \{1, \ldots, \boldsymbol{l}\}$ asegurando $\Phi(\boldsymbol{x}_t) = \Phi(\boldsymbol{x}_i)$. De la solución directa de ACP Kernelizado, es sabido que $\boldsymbol{V}_t$ puede ser expandido en los puntos de datos mapeados $\Phi(\boldsymbol{x}_i)$. Esto restringe el espacio de búsqueda a combinaciones lineales de la $\Phi(\boldsymbol{x}_i)$ tal que $\boldsymbol{V}_t$ puede ser expresado como

$$\boldsymbol{V}_t = \boldsymbol{A}_t \Phi \tag{C.16}$$

con un matriz de *r* x *l* $\boldsymbol{A}_t = \left(\boldsymbol{a}_{1,t}^T, \ldots, \boldsymbol{a}_{r,t}^T\right)^T$ de los coeficientes de expansión. La *i-ésima* fila $\boldsymbol{a}_i = \left(\boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,l}\right)$ de $\boldsymbol{A}_t$ corresponde a los coeficientes de expansión del *i-ésimo* autovector de *K* en el $\Phi(\boldsymbol{x}_i)$ (ver Sección C.2.2.1), es decir, $\boldsymbol{V}_{i,t} = \Phi^T \boldsymbol{a}_{i,t}$. Usando esta representación, la regla de actualización se convierte en

$$\boldsymbol{A}_{t+1}\Phi = \boldsymbol{A}_t\Phi + \eta_t \left[\boldsymbol{y}_t \, \Phi(\boldsymbol{x}_t)^T - \boldsymbol{ti}\left(\boldsymbol{y}_t \boldsymbol{y}_t^T\right)\boldsymbol{A}_t\Phi \right] \tag{C.17}$$

Estos puntos de datos mapeados $\Phi(\boldsymbol{x}_t)$ pueden ser representados como $\Phi(\boldsymbol{x}_t) = \Phi^T \boldsymbol{b}_t$ con un vector unidad canónica $\boldsymbol{b}_t = (0, \ldots, 1, \ldots, 0)^T$ en $\Re^l$ (solo el *J*(*t*)-ésimo elemento es 1). Usando esta notación, la regla de actualización puede ser escrita solo en función de los coeficientes de expansión como

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \eta_t \left[\boldsymbol{y}_t \, \boldsymbol{b}_t^T - \boldsymbol{ti}\left(\boldsymbol{y}_t \boldsymbol{y}_t^T\right)\boldsymbol{A}_t \right] \tag{C.18}$$

O equivalentemente en su forma de componentes

$$\boldsymbol{a}_{ij,t+1} = \begin{cases} \boldsymbol{a}_{ij,t} + \eta\, \boldsymbol{y}_{i,t} - \eta\, \boldsymbol{y}_{i,t} \sum_{k=1}^{i} \boldsymbol{a}_{kj,t}\, \boldsymbol{y}_{k,t} & \textit{si } \boldsymbol{J}(\boldsymbol{t}) = \boldsymbol{j} \\ \boldsymbol{a}_{ij,t} - \eta\, \boldsymbol{y}_{i,t} \sum_{k=1}^{i} \boldsymbol{a}_{kj,t}\, \boldsymbol{y}_{k,t} & \textit{si } \boldsymbol{J}(\boldsymbol{t}) \neq \boldsymbol{j} \end{cases} \tag{C.19}$$

donde

$$\boldsymbol{y}_{i,t} = \sum_{k=1}^{l} \boldsymbol{a}_{ik,t}\, \Phi(\boldsymbol{x}_k)\Phi(\boldsymbol{x}_t) = \sum_{k=1}^{l} \boldsymbol{a}_{ik,t}\, \boldsymbol{k}(\boldsymbol{x}_k, \boldsymbol{x}_t) \tag{C.20}$$

el cual no requiere a $\Phi(\boldsymbol{x})$ en su forma explícita conforme a proveer una implementación práctica de la AHG en *F*. Este algoritmo es debido a Kim [138] y permite una implementación mucho más simple comparada con AHG.

C.2.1.4 AHK con Meta-Descenso Estocástico (MDE)

C.2.1.4.1 La ganancia decae con la inversa de los autovalores

Consideremos el término $\boldsymbol{y}_t \boldsymbol{x}_t^T = \boldsymbol{V}_t \boldsymbol{x}_t \boldsymbol{x}_t^T$ apareciendo del lado derecho de la Ecuación (C.13). En la solución deseada, las filas de $\boldsymbol{V}_t$ contienen las componentes principales, es decir, conteniendo a los autovectores de $\boldsymbol{C}_x = \boldsymbol{X}\,\boldsymbol{X}^T$. Los elementos de $\boldsymbol{y}_t$ entonces escalan con los autoralores asociados de $\boldsymbol{C}_x$. Grandes diferencias en los autovalores pueden por lo tanto conducir a un mal condicionamiento (por lo tanto, enlentecer la convergencia) del AHG; lo mismo para AHK.

Podemos contrarrestar este problema al establecer al AHK con un vector ganancia $\eta_t \in \Re_+^r$ el cual provee cada autovector estimado con su parámetro de ganancia individual

$$[\eta_t]_i = \frac{1}{[\lambda_t]_i} \frac{\tau}{\boldsymbol{t} + \tau} \eta_0 \tag{C.21}$$

con η_0 y τ parámetros de ajuste positivos como en (C.14).

Finalmente, la regla de actualización se convertirá entonces en

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \boldsymbol{diag}(\eta_t)\Gamma_t \tag{C.22}$$

donde *diag*(•) mapea un vector en una matriz diagonal, con

$$\Gamma_t = \boldsymbol{y}_t\, \boldsymbol{e}_{p_t}^T - \boldsymbol{ti}(\boldsymbol{y}_t \boldsymbol{y}_t^T)\boldsymbol{A}_t \tag{C.23}$$

donde $\boldsymbol{e_i}$ es el vector unidad en la dirección *i*.

C.2.1.4.2 Meta-Descenso Estocástico Escalar

Sea $\boldsymbol{V}$ un espacio vectorial, $\theta \in \boldsymbol{V}$ un vector parámetro, y $\boldsymbol{J} : \boldsymbol{V} \to \Re$ la función objetivo que deseamos optimizar. Asumimos que $\boldsymbol{J}$ es doblemente diferenciable en casi todos lados. Denominaremos $\boldsymbol{J}_t : \boldsymbol{V} \to \Re$ la aproximación estocástica de la función objetivo en el tiempo *t*. Nuestro objetivo es encontrar θ tal que $\boldsymbol{E}_t\left[\boldsymbol{J}_t(\theta)\right]$ sea minimizada. Adaptamos θ mediante un gradiente estocástico descendente

$$\theta_{t+1} = \theta_t - \boldsymbol{e}^{\rho_t} \boldsymbol{g}_t, \text{ donde } \boldsymbol{g}_t = \partial_{\theta_t} \boldsymbol{J}_t(\theta_t) \tag{C.24}$$

El gradiente estocástico descendente es sensible a los valores de la *ganancia-logarítmica* $\rho_t \in \Re$: si es muy pequeño, (C.24) tomará muchas iteraciones para converger; mientras que si es muy grande, (C.24) puede diverger.

Una solución es adaptar ρ_t por un gradiente descendente meta-nivel simultáneo. Entonces podríamos buscar minimizar el valor del objetivo en la próxima iteración al ajustar ρ_t en proporción al gradiente $\partial_{\rho_t} \boldsymbol{J}_{t+1}(\theta_{t+1})$. Usando la regla de la cadena y (C.24) encontramos

$$\begin{aligned} \rho_{t+1} &= \rho_t - \mu \partial_{\rho_t} \boldsymbol{J}_{t+1}(\theta_{t+1}) \\ &= \rho_t - \mu \left[\partial_{\theta_{t+1}} \boldsymbol{J}_{t+1}(\theta_{t+1})\right]^T \partial_{\rho_t} \theta_{t+1} \\ &= \rho_t + \mu \boldsymbol{e}^{\rho_t} \boldsymbol{g}_{t+1}^T \boldsymbol{g}_t \end{aligned} \tag{C.25}$$

donde la *meta-ganancia* $\mu \geq 0$ es un parámetro de ajuste escalar. Intuitivamente, la adaptación de ganancia de (C.25) esta impulsada por el ángulo entre sucesivas mediciones del gradiente: Si este es menor que 90º, entonces $\boldsymbol{g}_{t+1}^T \boldsymbol{g}_t > 0$, y ρ_t aumentará. Inversamente, si el ángulo es mayor que 90º (gradiente oscilante), entonces ρ_t disminuirá porque $\boldsymbol{g}_{t+1}^T \boldsymbol{g}_t < 0$. Entonces (C.25) sirve a gradientes sucesivos decorrelacionados, los cuales llevan a mejorar la convergencia de (C.24).

Un inconveniente de (C.25) reside en que la recorrelación ocurre solo a través de un único paso de tiempo, convirtiendo a la adaptación de la ganancia extremadamente sensitiva a correlaciones de *corto-plazo* espúreas en los datos. El Meta-Descenso estocástico (MDS; Schraudolph, 1999 [454] 2002 [453]) aborda esta cuestión al emplear una traza exponencialmente decreciente de gradientes a través del tiempo:

$$\begin{aligned} \rho_{t+1} &= \rho_t - \mu \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} \boldsymbol{J}_{t+1}(\theta_{t+1}) \\ &= \rho_t - \mu \left[\partial_{\theta_{t+1}} \boldsymbol{J}_{t+1}(\theta_{t+1})\right]^T \sum_{i=0}^{t} \xi^i \partial_{\rho_{t-i}} \theta_{t+1} \\ &= \rho_t - \mu \boldsymbol{g}_{t+1}^T \boldsymbol{v}_{t+1} \end{aligned} \tag{C.26}$$

donde el vector $\boldsymbol{v}_{t+1} \in \boldsymbol{V}$ caracteriza la dependencia de θ_{t+1} sobre su ganancia histórica para una escala de tiempo gobernada por el factor de decaimiento $0 \leq \xi \leq 1$, un parámetro de ajuste escalar.

Para computar $\boldsymbol{v}_{t+1}$ eficientemente, expandimos θ_{t+1} en función de su definición recursiva (C.24):

$$\begin{aligned}\boldsymbol{v}_{t+1} &= \sum_{i=0}^{t} \xi^i \, \partial_{\rho_{t-i}} \theta_{t+1} \\ &= \sum_{i=0}^{t} \xi^i \, \partial_{\rho_{t-i}} \theta_t - \sum_{i=0}^{t} \xi^i \, \partial_{\rho_{t-i}} \left[e^{\rho_t} \boldsymbol{g}_t \right] \\ &\approx \xi \, \boldsymbol{v}_t - e^{\rho_t} \left(\boldsymbol{g}_t + \partial_{\theta_t} \boldsymbol{g}_t \sum_{i=0}^{t} \xi^i \, \partial_{\rho_{t-i}} \theta_t \right)\end{aligned} \tag{C.27}$$

Aquí hemos usado $\partial_{\rho_t} \theta_t = 0$, y aproximado

$$\sum_{i=0}^{t} \xi^i \, \partial_{\rho_{t-i}} \rho_t \approx 0 \tag{C.28}$$

lo que equivale a afirmar que la adaptación de *ganancia-logarítmica* debe estar en equilibrio sobre la escala de tiempo determinada por ξ. Notando que $\partial_{\theta_t} \boldsymbol{g}_t$ es el Hessiano de $\boldsymbol{J}_t(\theta_t)$, arribamos a una simple adaptación iterativa

$$\boldsymbol{v}_{t+1} = \xi \, \boldsymbol{v}_t - e^{\rho_t} \left(\boldsymbol{g}_t + \xi \, \boldsymbol{H}_t \boldsymbol{v}_t \right) \tag{C.29}$$

Dado que los parámetros iniciales θ_0 no dependen de la ganancia, $\boldsymbol{v}_0 = 0$. Nótese que para $\xi = 0$ (C.29) y (C.26) se reducen a la adaptación de ganancia de un solo paso (C.25).

El costo del producto *Hessiano-vector* $\boldsymbol{H}_t \boldsymbol{v}_t$ debería ser elevado si se hace ingenuamente. Afortunadamente, existen métodos eficientes para calcular esta cantidad directamente sin calcular el Hessiano [453-456]. En esencia, estos métodos trabajan al propagar $\boldsymbol{v}$ como un diferencial (es decir, derivada direccional) a través del cálculo del gradiente:

$$\boldsymbol{d}\theta_t = \boldsymbol{v}_t \;\Rightarrow\; \boldsymbol{H}_t \boldsymbol{v}_t = \boldsymbol{d}\boldsymbol{g}_t \tag{C.30}$$

En otras palabras, si ponemos el diferencial $\boldsymbol{d}\theta_t$ del vector parámetro a $\boldsymbol{v}_t$, entonces el diferencial resultante del gradiente $\boldsymbol{g}_t$ (una función de θ_t) es el producto *Hessiano-vector* $\boldsymbol{H}_t \boldsymbol{v}_t$.

C.2.1.4.3 MDE para AHK

Este algoritmo debido a Günter *et al* [141-142] se desarrolla a continuación.

La actualización de AHK de (C.22) puede ser interpretada como *r* actualizaciones acopladas en EHKR, uno por cada fila de $\boldsymbol{A}_t$, cada una asociada con una ganancia escalar. Para aplicar MDE aquí introducimos un vector de *ganancia-logarítmica* adicional $\rho_t \in \Re^r$

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + e^{\boldsymbol{diag}(\rho_t)} \, \boldsymbol{diag}(\eta_t) \Gamma_t \tag{C.31}$$

(El exponencial de una matriz diagonal se obtiene simplemente por exponenciación de las elementos de la diagonal individualmente). Por lo tanto, estamos aplicando MDE a AHK para un gradiente descendente *precondicionado* por la inversa de los autovalores estimados [139-140]. Felizmente, MDE trabajará con tal precondicionador, y se beneficiará al hacerlo.

En un EHKR, el MDE se adapta a una *ganancia-logarítmica* escalar cuya actualización es manejada por el producto interno entre el gradiente y un diferencial de los parámetros del sistema, todos en el EHKR [457]. En el caso de AHK, $\Gamma_t\Phi'$ puede ser interpretado como el gradiente en el EHKR de la función de mérito (C.8) maximizada por AHK. Por lo tanto, la adaptación de MDE de ρ_t en (C.31) esta manejada por los elementos de la diagonal de $\langle \Gamma_t\Phi', \boldsymbol{B}_t\Phi' \rangle_{\mathrm{H}}$, donde $\boldsymbol{B}_t = \boldsymbol{d}\,\boldsymbol{A}_t$ denota a la matriz de los coeficientes de expansión de r x l para los parámetros diferenciales del MDE, análogo al vector $\boldsymbol{v}$ en la Sección C.2.1.4.2:

$$\begin{aligned}\rho_t &= \rho_{t-1} - \mu\,\boldsymbol{diag}\left(\langle \Gamma_t\Phi', \boldsymbol{B}_t\Phi' \rangle_{\mathrm{H}}\right)\\ &= \rho_{t-1} - \mu\,\boldsymbol{diag}\left(\Gamma_t\Phi'\Phi'^{\boldsymbol{T}}\,\boldsymbol{B}_t^{\boldsymbol{T}}\right)\\ &= \rho_{t-1} - \mu\,\boldsymbol{diag}\left(\Gamma_t\boldsymbol{K}'\,\boldsymbol{B}_t^{\boldsymbol{T}}\right)\end{aligned} \tag{C.32}$$

El cálculo de $\Gamma_t\boldsymbol{K}'$ realizado en forma ingenua en (C.32) debería tener una CC de $O(rl^2)$, la cual para l grandes es prohibitivamente elevada. No obstante, podemos reducir esta CC a $O(rl)$ al notar que (C.23) implica que

$$\begin{aligned}\Gamma_t\boldsymbol{K}' &= \boldsymbol{y}_t\,\boldsymbol{e}_{p_t}^{\boldsymbol{T}}\,\boldsymbol{K}' - \boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\boldsymbol{T}}\right)\boldsymbol{A}_t\boldsymbol{K}'\\ &= \boldsymbol{y}_t\,\boldsymbol{k}'^{\boldsymbol{T}}_{p_t} - \boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\boldsymbol{T}}\right)\boldsymbol{A}_t\boldsymbol{K}'\end{aligned} \tag{C.33}$$

donde la matriz de r x l $\boldsymbol{A}_t\boldsymbol{K}'$ puede ser almacenada y actualizada incrementalmente via (C.31):

$$\boldsymbol{A}_{t+1}\boldsymbol{K}' = \boldsymbol{A}_t\boldsymbol{K}' + \boldsymbol{e}^{\boldsymbol{diag}(\rho_t)}\,\boldsymbol{diag}\left(\eta_t\right)\Gamma_t\boldsymbol{K}' \tag{C.34}$$

El cálculo inicial de $\boldsymbol{A}_1\boldsymbol{K}'$ tendrá todavía y en general una CC de $O(rl^2)$ pero es asequible dado que solo debe ser efectuado una vez. Alternativamente, la CC de este paso puede ser reducida fácilmente a $O(rl)$ haciendo a $\boldsymbol{A}_1$ adecuadamente rala.

Finalmente, aplicamos la actualización estándar de MDE (C.29) de los parámetros diferenciales:

$$\boldsymbol{B}_{t+1} = \xi\,\boldsymbol{B}_t + \boldsymbol{e}^{\boldsymbol{diag}(\rho_t)}\,\boldsymbol{diag}\left(\eta_t\right)\left(\Gamma_t + \xi\,\boldsymbol{d}\,\Gamma_t\right) \tag{C.35}$$

El diferencial $\boldsymbol{d}\,\Gamma_t$ del gradiente, análogo a $\boldsymbol{d}\,\boldsymbol{g}_t$ en (C.30), puede ser calculado aplicando las reglas del cálculo:

$$\begin{aligned}\boldsymbol{d}\Gamma_t &= \boldsymbol{d}\left[\boldsymbol{y}_t\,\boldsymbol{e}_{p_t}^{\boldsymbol{T}} - \boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\boldsymbol{T}}\right)\boldsymbol{A}_t\right]\\ &= \left(\boldsymbol{dA}_t\right)\boldsymbol{k}'^{\boldsymbol{T}}_{p_t}\,\boldsymbol{e}_{p_t}^{\boldsymbol{T}} - \boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\boldsymbol{T}}\right)\left(\boldsymbol{dA}_t\right) - \left[\boldsymbol{d}\,\boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\boldsymbol{T}}\right)\right]\boldsymbol{A}_t\\ &= \boldsymbol{B}_t\boldsymbol{k}'^{\boldsymbol{T}}_{p_t}\,\boldsymbol{e}_{p_t}^{\boldsymbol{T}} - \boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{\boldsymbol{T}}\right)\boldsymbol{B}_t - \boldsymbol{ti}\left(\boldsymbol{B}_t\boldsymbol{k}'^{\boldsymbol{T}}_{p_t}\,\boldsymbol{y}_t^{\boldsymbol{T}} + \boldsymbol{y}_t\boldsymbol{k}'^{\boldsymbol{T}}_{p_t}\,\boldsymbol{B}_t^{\boldsymbol{T}}\right)\boldsymbol{A}_t\end{aligned} \tag{C.36}$$

usando el hecho de que dado que $\boldsymbol{k}'$ y $\boldsymbol{e}$ son ambos independientes de $\boldsymbol{A}$, tenemos $\boldsymbol{d}\left(\boldsymbol{k}'^{T}_{P_t}\ \boldsymbol{e}^{T}_{P_t}\right)=0$. Insertando (C.23) y (C.36) en (C.35) finalmente la regla de actualización es

$$\boldsymbol{B}_{t+1}=\xi\boldsymbol{B}_t+\boldsymbol{e}^{\boldsymbol{diag}(\rho_t)}\,\boldsymbol{diag}(\eta_t)\left[\left(\boldsymbol{A}_t+\xi\boldsymbol{B}_t\right)\boldsymbol{k}'^{T}_{P_t}\ \boldsymbol{e}^{T}_{P_t}-\boldsymbol{ti}\left(\boldsymbol{y}_t\boldsymbol{y}_t^{T}\right)\left(\boldsymbol{A}_t+\xi\boldsymbol{B}_t\right)-\xi\boldsymbol{ti}\left(\boldsymbol{B}_t\boldsymbol{k}'^{T}_{P_t}\ \boldsymbol{y}_t^{T}+\boldsymbol{y}_t\boldsymbol{k}'^{T}_{P_t}\ \boldsymbol{B}_t^{T}\right)\boldsymbol{A}_t\right] \tag{C.37}$$

C.2.2 Versiones no-conexionistas

C.2.2.1 ACP Kernelizado

Este algoritmo debido a Schölkopf *et al* [141, 142] se desarrolla a continuación.

Cuando los datos de interés son altamente no-lineales, el ACP falla al intentar capturar la estructura subyacente. Una extensión no-lineal de ACP, la ACP Kernelizada calcula las componentes principales en un posible Espacio de Hilbert Kernelizado de Reproducción (EHKR) de alta dimensión *F* el cual es relativo al espacio de entradas para un mapeo no-lineal $\Phi:\Re^N\rightarrow\boldsymbol{F}$ [138]. Una importante propiedad de un EHKR es que el producto interno de dos puntos mapeados por Φ puede ser evaluado usando funciones kernel

$$\boldsymbol{k}(\boldsymbol{x},\boldsymbol{y})=\Phi(\boldsymbol{x})\cdot\Phi(\boldsymbol{y}) \tag{C.38}$$

el cual nos permite calcular el valor del producto interno sin tener que llevar a cabo el mapeo explícito Φ. Dado que ACP puede ser formulado en función de los productos internos, podemos calcularlo también implícitamente en un EHKR. Asumiendo que los datos están centrados en *F* (es decir, $\sum_{\boldsymbol{k}=1}^{\boldsymbol{l}}\Phi\left(\boldsymbol{x}_{\boldsymbol{k}}\right)=0$) la matriz de covarianza toma la forma

$$\boldsymbol{C}=\frac{1}{\boldsymbol{l}}\Phi^{\boldsymbol{T}}\Phi \tag{C.39}$$

donde $\Phi=\left(\Phi\left(\boldsymbol{x}_1\right)^{\boldsymbol{T}},\ldots,\Phi\left(\boldsymbol{x}_{\boldsymbol{l}}\right)^{\boldsymbol{T}}\right)^{\boldsymbol{T}}$. Ahora se deben encontrar los autovalores $\lambda\geq0$ y autovectores $\boldsymbol{v}\in\boldsymbol{F}\setminus0$ satisfaciendo

$$\lambda\boldsymbol{v}=\boldsymbol{C}\,\boldsymbol{v} \tag{C.40}$$

Para todas las soluciones *v* con $\lambda\neq0$ que se encuentran en el ámbito de $\left\{\Phi\left(\boldsymbol{x}_1\right),\ldots,\Phi\left(\boldsymbol{x}_{\boldsymbol{l}}\right)\right\}$. [141, 142], consideraremos el siguiente problema equivalente

$$\lambda\,\Phi\boldsymbol{v}=\boldsymbol{C}\,\Phi\boldsymbol{v} \tag{C.41}$$

y podemos representar *v* en función de un vector *l*-dimensional *q* como $\boldsymbol{v}=\Phi^{\boldsymbol{T}}\boldsymbol{q}$. Combinando esta con (C.39) y (C.41) y definiendo una matriz de kernel *K* de *l* x *l* por $\boldsymbol{K}=\Phi\,\Phi^{\boldsymbol{T}}$ conduce a $\boldsymbol{l}\,\lambda\,\boldsymbol{K}\,\boldsymbol{q}=\boldsymbol{K}^2\,\boldsymbol{q}$. La solución puede ser obtenida al resolver el *problema de autovalores kernelizados* [141, 142]

$$\boldsymbol{l}\,\lambda\ \boldsymbol{q}=\boldsymbol{K}\,\boldsymbol{q} \tag{C.42}$$

Se debe notar que la dimensión de la matriz de kernel esta en escala con el cuadrado del número de ejemplos. Por lo tanto, esto se convierte en un problema computacionalmente inviable de resolver directamente en relación a los autovalores kernelizados para un gran número de ejemplos. Esto es el motivo de la aparición del AHK presentado en la Sección C.2.1.3. Para una discusión más detallada sobre ACP y ACP Kernelizado incluyendo aspecto atenientes a la complejidad computacional se puede recurrir a [138].

C.2.2.2 ACP Rápido

Este algoritmo debido a Sharma *et al* [143] será desarrollado en esta sección.

C.2.2.2.1 Prolegómenos

Comenzaremos con una reseña para obtener la transformada ACP de punto fijo. La transformada ACP puede ser hallada al minimizar el error cuadrático medio, Para ver esto, téngase en cuenta al vector de características $\boldsymbol{x} \in \Re^{\boldsymbol{d}}$ (espacio *d-dimensional*), al vector de características reducido dimensionalmente $\boldsymbol{y} \in \Re^{\boldsymbol{h}}$ y al vector de características reconstruido $\hat{\boldsymbol{x}} \in \Re^{\boldsymbol{d}}$. Entonces el MSE puede ser representado aquí como

$$\boldsymbol{MSE} = \boldsymbol{E}\left[\left\|\boldsymbol{x} - \hat{\boldsymbol{x}}\right\|^2\right] \tag{C.43}$$

donde $\boldsymbol{E}[\bullet]$ es el operador de esperanza con respecto a $\boldsymbol{x}$ y $\|\bullet\|^2$ es el valor de la norma al cuadrado. Sabemos que la transformación ACP $\boldsymbol{V}$ es en este caso de dimensión *d* x *h* y son empleados para generar una reducción dimensional desde un espacio *d-dimensional* a otro espacio de características *h-dimensional*, es decir, $\boldsymbol{V} : \boldsymbol{x} \rightarrow \boldsymbol{y}$ o bien $\boldsymbol{y} = \boldsymbol{V}^{\boldsymbol{T}} \boldsymbol{x}$. Se debe notar que en esta transformación, $\boldsymbol{x}$ se asume de valor medio nulo, si la media no es cero entonces $\boldsymbol{y}$ puede ser representada como

$$\boldsymbol{y} = \boldsymbol{V}^{\boldsymbol{T}}\left(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}}\right) \tag{C.44}$$

donde $\boldsymbol{m}_{\boldsymbol{x}} = \boldsymbol{E}[\boldsymbol{x}]$ (como se vió en el Capítulo 1). Dada $\boldsymbol{y}$ (de la Ecuación (C.44)) uno puede simplemente aplicar la transformación inversa hacia el espacio de características original con el mismo error de reconstrucción finito. El error de reconstrucción será

$$\hat{\boldsymbol{x}} = \boldsymbol{V}\,\boldsymbol{y} + \boldsymbol{m}_{\boldsymbol{x}} \tag{C.45}$$

Sustituyendo (C.44) en (C.45) tenemos

$$\hat{\boldsymbol{x}} = \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}}\left(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}}\right) + \boldsymbol{m}_{\boldsymbol{x}} \tag{C.46}$$

La Ecuación (C.43) puede ser re-escrita utilizando (C.46) como

$$\boldsymbol{MSE} = \boldsymbol{E}\left[\left\|\left(\boldsymbol{I}_{\boldsymbol{d}\times\boldsymbol{d}} - \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}}\right)\left(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}}\right)\right\|^2\right] \tag{C.47}$$

Los *h* vectores de la base deseada de $\boldsymbol{V}$ se proyectan al subespacio *d-dimensional* y son

mutuamente ortonormales, es decir, $\boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}} = \boldsymbol{I}_{\boldsymbol{h}\times\boldsymbol{h}}$. Usando la condición de ortonormalidad de la Ecuación (C.47) puede ser a su vez más simplificada como

$$\boldsymbol{MSE} = \boldsymbol{E}\left[\boldsymbol{g}(\boldsymbol{V}, \boldsymbol{x})\right] \tag{C.48}$$

donde $\boldsymbol{g}(\boldsymbol{V}, \boldsymbol{x})$ es una función escalar y está dada por

$$\boldsymbol{g}(\boldsymbol{V}, \boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})^{\boldsymbol{T}} \left(\boldsymbol{I}_{\boldsymbol{d}\times\boldsymbol{d}} - \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}}\right)(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}}) \tag{C.49}$$

La derivada de $\boldsymbol{E}\left[\boldsymbol{g}(\boldsymbol{V}, \boldsymbol{x})\right]$ con respecto a $\boldsymbol{V}$ [143] será

$$\frac{\partial}{\partial \boldsymbol{V}} \boldsymbol{E}\left[\boldsymbol{g}(\boldsymbol{V}, \boldsymbol{x})\right] = -2\boldsymbol{E}\left[(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})^{\boldsymbol{T}} \boldsymbol{V}\right] \tag{C.50}$$

Dada la Ecuación (C.50) y la condición de ortonormalidad de $\boldsymbol{V}$, podemos aplicar el algoritmo de punto-fijo [458] para resolver para los valores de $\boldsymbol{V}$, es decir,

$$\begin{aligned} \boldsymbol{V} &= \boldsymbol{E}\left[(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})^{\boldsymbol{T}}\right]\boldsymbol{V} \\ \boldsymbol{V} &= \boldsymbol{ortonormalizar}(\boldsymbol{V}) \end{aligned} \tag{C.51}$$

Notemos que el signo negativo desaparece de la Ecuación (C.50) dado que $\boldsymbol{V}$ y $-\boldsymbol{V}$ definen la misma dirección. Además, dado que la esperanza es con respecto a $\boldsymbol{x}$, la transformación $\boldsymbol{V}$ queda afuera de la misma. En consecuencia podemos expresar (C.51) como

$$\begin{aligned} \boldsymbol{V} &= \boldsymbol{C}_{\boldsymbol{x}}\, \boldsymbol{V} \\ \boldsymbol{V} &= \boldsymbol{ortonormalizar}(\boldsymbol{V}) \end{aligned} \tag{C.52}$$

donde $\boldsymbol{C}_{\boldsymbol{x}} = \boldsymbol{E}\left[(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})(\boldsymbol{x} - \boldsymbol{m}_{\boldsymbol{x}})^{\boldsymbol{T}}\right]$ es la covarianza de $\boldsymbol{x}$. La ortonormalización de $\boldsymbol{V}$ se puede realizar usando el procedimiento de ortonormalización de Gram-Schmidt. Entonces, el cálculo de los *h* autovectores no requiere del procedimiento de descomposición en autovalores. Esto significa que el método cíclico de Jacobi puede ser evitado por completo para dicha autodescomposición [143]. Esto debería reducir ciertamente la CC del proceso de encontrar los autovectores de ACP. Además, al usar un algoritmo de punto-fijo hace que éste converja muy rápido en solo unas pocas iteraciones sin la necesidad de ningún conocimiento previo de una tasa de aprendizaje en forma inicial como es el caso del descenso por el gradiente basado en los métodos de maximización de la esperanza. La próxima sección atañe al algoritmo para el cálculo de los autovectores para ACP.

C.2.2.2.2 Algoritmo ACP Rápido

Una forma para calcular todos los *h* vectores de la base ortonormal es usando el método de Gram-Schmidt. El autovector o eje principal más dominante/líder puede ser considerado en primer lugar. Similarmente, todos los restantes *h*-1 vectores base (ortonormal a los previamente medidos vectores base) serán medidos uno por uno en un orden reducido de dominancia. La medida previa de los (*p*-1)-*ésimos* vectores base será utilizada para encontrar el *p-ésimo* vector base. El algoritmo para el *p-ésimo* vector base convergerá cuando los valores nuevos y viejos de $\boldsymbol{V}_{\boldsymbol{p}}$

apunten en la misma dirección, es decir, $\boldsymbol{V}_p^{+T}\boldsymbol{V}_p \approx 1$ (donde $\boldsymbol{V}_p^{+T}$ es el nuevo valor de $\boldsymbol{V}_p$). Es usualmente económico usar un error de tolerancia finito para satisfacer el criterio de convergencia

$$\boldsymbol{abs}\left(\boldsymbol{V}_p^{+T}\boldsymbol{V}_p\right) < \varepsilon \tag{C.53}$$

donde ε es una tolerancia predefinida o umbral y ***abs***(•) significa *valor absoluto*. El Algoritmo C.1 representa al algoritmo ACP Rápido para el cálculo de autovectores líderes

Algoritmo C.1

1. Elegir *h*, el número de ejes principales o autovectores requeridos a estimar.
   Calcular la covarianza $\boldsymbol{C}_x$ y poner $p = 1$
2. Inicializar el autovector $\boldsymbol{V}_p$ de dimensión $d \times 1$, por ejemplo, aleatoriamente
3. $\boldsymbol{V}_p = \boldsymbol{C}_x \boldsymbol{V}_p$
4. $\boldsymbol{V}_p = \boldsymbol{V}_p - \sum_{j=1}^{p-1}\left(\boldsymbol{V}_p^T\boldsymbol{V}_j\right)\boldsymbol{V}_j$
5. $\boldsymbol{V}_p = \dfrac{\boldsymbol{V}_p}{\left\|\boldsymbol{V}_p\right\|}$
6. Si $\boldsymbol{V}_p$ no converge, ir al paso 3.
7. $p = p+1$ e ir al paso 2. hasta que $p = h$

Mientras que la CC del mismo puede ser observada en la Tabla C.I.

Tabla C.I: CC del algoritmo

Pasos de mayor procesamiento involucrados	CC
Covarianza C_x (paso 1)	$O(d^2n)$
Gram-Schmidt para V_p (*p-ésimo* vector base)	$O(dpL)$
Gram-Schmidt para todos los vectores base $p = 1 \ldots h$	$O(dh^2L)$
Actualización para todos los vectores base $p = 1 \ldots h$ (paso 3)	$O(d^2hL)$
Estimado Total	$O(d^2hL+d^2n) \approx O(d^2h+d^2n)$

C.2.2.3 ACP Recursivo

Este algoritmo debido a Principe *et al* [144] comienza con la suposición de que una secuencia de vectores de entrada $\boldsymbol{x}_k$ son estacionarios en un sentido amplio de valor medio nulo y están arribando, donde *k* es el índice temporal de la muestra. La covarianza estimada en el tiempo *k* para los vectores de entrada es

$$\boldsymbol{C}_{x,k} = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{x}_i\boldsymbol{x}_i^T = \frac{(k-1)}{k}\boldsymbol{C}_{x,k-1} + \frac{1}{k}\boldsymbol{x}_k\boldsymbol{x}_k^T \tag{C.54}$$

Haciendo $\boldsymbol{C}_{x,k} = \boldsymbol{V}_k\Lambda_k\boldsymbol{V}_k^T$ y $\boldsymbol{C}_{x,k-1} = \boldsymbol{V}_{k-1}\Lambda_{k-1}\boldsymbol{V}_{k-1}^T$, donde $\boldsymbol{V}$ y Λ representan a la matriz de los autovectores ortonormales y la matriz diagonal de autovalores, respectivamente. También

definimos $\boldsymbol{y_k} = \boldsymbol{V_k^T x_k}$. Sustituyendo estas definiciones en (C.54), obtenemos la siguiente fórmula recursiva para los autovectores y autovalores:

$$\boldsymbol{V_k}(\boldsymbol{k}\Lambda_k)\boldsymbol{V_k^T} = \boldsymbol{V_{k-1}}\left[(\boldsymbol{k}-1)\Lambda_{k-1} + \boldsymbol{y_k y_k^T}\right]\boldsymbol{V_{k-1}^T} \tag{C.55}$$

Claramente, si podemos determinar la autodescomposición de la matriz $\left[(\boldsymbol{k}-1)\Lambda_{k-1} + \boldsymbol{y_k y_k^T}\right]$, la cual se expresa mediante $\boldsymbol{Q_k D_k Q_k^T}$, donde $\boldsymbol{Q}$ es ortonormal y $\boldsymbol{D}$ es diagonal, entonces (C.55) se convierte

$$\boldsymbol{V_k}(\boldsymbol{k}\Lambda_k)\boldsymbol{V_k^T} = \boldsymbol{V_{k-1}Q_k D_k Q_k^T V_{k-1}^T} \tag{C.56}$$

Por comparación directa, la regla de actualización recursiva para los autovectores y autovalores esta determinada por

$$\begin{aligned} \boldsymbol{V_k} &= \boldsymbol{V_{k-1}Q_k} \\ \Lambda_k &= \frac{\boldsymbol{D_k}}{\boldsymbol{k}} \end{aligned} \tag{C.57}$$

A pesar del hecho de que la matriz $\left[(\boldsymbol{k}-1)\Lambda_{k-1} + \boldsymbol{y_k y_k^T}\right]$ tiene una estructura especial mucho mas simple que aquella de la matriz de covarianza general, determinar la autodescomposición de $\boldsymbol{Q_k D_k Q_k^T}$ analíticamente es difícil. No obstante, especialmente si *k* es grande, el problema puede

Algoritmo C.2

1. Inicializar $\boldsymbol{V_0}$ y $\Lambda_{\boldsymbol{0}}$
2. En cada instante de tiempo *k* hacer lo siguiente
   (a) Obtener una muestra de entrada $\boldsymbol{x_k}$
   (b) Establecer el parámetro de profundidad de memoria $\boldsymbol{\lambda_k}$
   (c) Calcular $\boldsymbol{y_k} = \boldsymbol{V_{k-1}^T x_k}$
   (d) Encontrar las perturbaciones P_V y P_Λ correspondientes a
   $(1-\lambda_k)\Lambda_{k-1} + \lambda_k \boldsymbol{y_k y_k^T}$
   (e) Actualizar las matrices de autovectores y autovalores
   $\tilde{\boldsymbol{V}}_k = \boldsymbol{V_{k-1}}(\boldsymbol{I} + \boldsymbol{P_V})$
   $\tilde{\Lambda}_k = (1-\lambda_k)\Lambda_{k-1} + \boldsymbol{P}_\Lambda$
   (f) Normalizar las normas de autovectores estimados por $\boldsymbol{V_k} = \tilde{\boldsymbol{V}}_k \boldsymbol{T_k}$, donde $\boldsymbol{T_k}$ es una matriz diagonal conteniendo las inversas de las normas de cada columna de $\tilde{\boldsymbol{V}}_k$
   (g) Corregir los autovalores estimados mediante $\Lambda_k = \tilde{\Lambda}_k \boldsymbol{T_k^{-2}}$, donde $\boldsymbol{T_k^{-2}}$ es una matriz diagonal conteniendo las normas cuadradas de las columnas de $\tilde{\boldsymbol{V}}_k$.

ser resuelto en una forma más simple usando una aproximación por análisis de matrices de perturbación [144].

Finalmente, el Algoritmo C.2 representa al ACP Recursivo. Donde P_V y P_Λ son matrices de pequeñas perturbaciones.

C.2.2.4 TDKL basada en una red sistólica

Este algoritmo debido a Güleryüz *et al* [145, 146] consiste en la aplicación transversal de una versión matricial (de expansión o de productos externos) del algoritmo de Gram-Schmidt a las filas de la matriz $\boldsymbol{X}$ de la Sección 1.2, lo cual desemboca en un espacio lineal equivalente en el cual los autovalores no están ordenados en forma decreciente en la matriz diagonal resultante del proceso de decorrelación, por lo que la poda es imposible. No obstante, aplicando un algoritmo de ordenamiento (y conservando tal secuencia) los autovalores ordenados poseen una caída muy lenta en pos de un eventual poda, pero con un muy bajo rendimiento de descompresión.

Concretamente las filas de la matriz $\boldsymbol{X}$ guardarán entre sí la siguiente relación

$$\begin{aligned}
\widehat{\boldsymbol{H}}_1^{\boldsymbol{T}} &= \boldsymbol{H}_1^{\boldsymbol{T}} \\
\widehat{\boldsymbol{H}}_2^{\boldsymbol{T}} &= \boldsymbol{H}_2^{\boldsymbol{T}}\left(1-\Gamma_1\widehat{\boldsymbol{H}}_1^{\boldsymbol{T}}\right) \\
&\vdots \\
\widehat{\boldsymbol{H}}_N^{\boldsymbol{T}} &= \boldsymbol{H}_N^{\boldsymbol{T}}\left(1-\Gamma_1\widehat{\boldsymbol{H}}_1^{\boldsymbol{T}}-\cdots-\Gamma_{N-1}\widehat{\boldsymbol{H}}_{N-1}^{\boldsymbol{T}}\right)
\end{aligned} \tag{C.58}$$

donde **1** es una matriz identidad de (*nfm***ncm*) x (*nfm***ncm*). Un atributo complementario permite,

$$\begin{bmatrix} \widehat{\boldsymbol{H}}_1^{\boldsymbol{T}} \\ \widehat{\boldsymbol{H}}_2^{\boldsymbol{T}} \\ \vdots \\ \widehat{\boldsymbol{H}}_N^{\boldsymbol{T}} \end{bmatrix} \left[\Gamma_1\,\Gamma_2\cdots\Gamma_N\right] = 1 \quad \Rightarrow \quad \widehat{\boldsymbol{H}}_{\boldsymbol{m}}^{\boldsymbol{T}}\Gamma_{\boldsymbol{n}} = \begin{cases} 0_{\boldsymbol{nm}\times\boldsymbol{nm}} & \boldsymbol{si\ m \neq n} \\ 1_{\boldsymbol{nm}\times\boldsymbol{nm}} & \boldsymbol{si\ m = n} \end{cases} \tag{C.59}$$

para *m*, *n* = 1, … *N*.

La inversa surge automáticamente de (C.58).

C.2.2.5 ACP basado en el filtro de Kalman

Este algoritmo debido a Zhong *et al* [147] consiste en plantear un problema de estimación a partir del siguiente modelo autoregresivo [462-468]

$$\begin{aligned}
\boldsymbol{X}_{t+1} &= \boldsymbol{A}\,\boldsymbol{X}_t + \boldsymbol{v}_t, \quad \boldsymbol{X}_0 = \boldsymbol{x}_0, \quad \boldsymbol{v}_t \sim \boldsymbol{N}(0,\boldsymbol{Q}) \\
\boldsymbol{Y}_t &= \boldsymbol{C}\,\boldsymbol{X}_t + \boldsymbol{w}_t, \quad \boldsymbol{w}_t \sim \boldsymbol{N}(0,\boldsymbol{R})
\end{aligned} \tag{C.60}$$

donde la matriz de evolución de estados $\boldsymbol{A}$ es *nm* x *nm*, la cual asumimos estable $|\boldsymbol{A}| < 1$, siendo $\boldsymbol{v}$ y $\boldsymbol{w}$ los ruidos de estado y medición respectivamente. Por otra parte, $\boldsymbol{Q}$ es la matriz de

covarianza de $\boldsymbol{v}$, mientras $\boldsymbol{R}$ lo es de $\boldsymbol{w}$.

La función de mérito para un filtro de Kalman robusto será

$$\boldsymbol{J}_k = \left(\boldsymbol{Y} - \boldsymbol{C}\widehat{\boldsymbol{X}}_k\right)^T \boldsymbol{M}^{k-1}\boldsymbol{R}^{-1}\left(\boldsymbol{Y} - \boldsymbol{C}\widehat{\boldsymbol{X}}_k\right) + \left(\widehat{\boldsymbol{X}}_k - \widehat{\boldsymbol{X}}^-\right)^T \boldsymbol{Q}\left(\widehat{\boldsymbol{X}}_k - \widehat{\boldsymbol{X}}^-\right) \tag{C.61}$$

donde $\boldsymbol{M}^{k-1}$ es la matriz de ponderación en el paso *k*:

$$\boldsymbol{M}^{k-1} = \boldsymbol{diag}\left(\boldsymbol{w}\left(\boldsymbol{z}_1^{k-1}\right), \boldsymbol{w}\left(\boldsymbol{z}_2^{k-1}\right), \ldots, \boldsymbol{w}\left(\boldsymbol{z}_m^{k-1}\right)\right) \tag{C.62}$$

La matriz $\boldsymbol{M}^{k-1}$ es actualizada en cada paso *k* usando el nuevo estado $\widehat{\boldsymbol{X}}_k$.

En resumen, el Algoritmo C.3 representa al filtro de Kalman robusto.

Algoritmo C.3

1. Calcular la matriz ACP modificada

$$\boldsymbol{C}' = \left(\boldsymbol{C}\boldsymbol{P}_t^- \boldsymbol{C}^T + \boldsymbol{R}\right)\boldsymbol{M}^T\boldsymbol{C}\left(\boldsymbol{P}_t^- \boldsymbol{C}^T \boldsymbol{M}^T \boldsymbol{C}\right)^{-1}$$

2. Calcular la ganancia del Kalman robusto

$$\boldsymbol{K} = \left(\boldsymbol{C}'^T\, \boldsymbol{M}\, \boldsymbol{C}'\right)^{-1}\boldsymbol{C}'^T$$

3. Actualizar la estimación con la medición $\boldsymbol{Y}_k$

$$\widehat{\boldsymbol{X}}_t = \widehat{\boldsymbol{X}}_t^- + \boldsymbol{K}\,\boldsymbol{M}\left(\boldsymbol{Y}_t - \boldsymbol{C}\,\widehat{\boldsymbol{X}}_t^-\right)$$

4. Actualizar el error de covarianza $\boldsymbol{P}_t$

$$\boldsymbol{P}_t = \left(\boldsymbol{I} - \boldsymbol{K}\,\boldsymbol{M}\,\boldsymbol{C}\right)\boldsymbol{P}_t^-$$

La CC del procesa descripto es $O(m^2n+n^3)$, donde *m* es la dimensión del vector de observación y *n* es la dimensión del modelo de parámetros auto-regresivo de promedios móviles (por sus siglas en inglés, ARMA). El término dominante es m^2, dado que en el caso de esta tesis $\boldsymbol{m} >> \boldsymbol{n}$.

C.2.2.6 Algoritmo de Gradiente Estocástico (AGE)

Este algoritmo debido a Dehaene [148] consiste originalmente de las siguientes suposiciones. Sea $\boldsymbol{x}_k \in \Re^N$ un vector de una señal de entrada dada en el tiempo *k* (*k* = 1, 2, …), generado por

$$\boldsymbol{x}_k = \boldsymbol{M}_k\,\boldsymbol{s}_k + \boldsymbol{n}_k \tag{C.63}$$

donde $\boldsymbol{M}_k$ es una matriz columna de rango completo $\boldsymbol{N} \times \kappa$, $\boldsymbol{s}_k \in \Re^\kappa \left(\kappa < \boldsymbol{N}\right)$ es la señal fuente, con la matriz de correlación no-singular $\boldsymbol{E}\left\{\boldsymbol{s}_k \boldsymbol{s}_k^T\right\}$ y finalmente $\boldsymbol{n}_k$ es la señal de ruido en la que mayormente se cumple $\boldsymbol{E}\left\{\boldsymbol{n}_k \boldsymbol{n}_k^T\right\} = \sigma_N^2 \boldsymbol{I}$. El espacio de columnas de $\boldsymbol{M}_k$ recibe el nombre de subespacio de señales, y su complemento ortogonal se llama subespacio de ruido. El problema del subespacio de seguimiento [148] consiste en estimar el subespacio de señal en cada instante *k*, dado $\boldsymbol{x}_k$.

Un algoritmo simple de gradiente estocástico esta dado como se indica en el Algoritmo C.4 donde las columnas $\boldsymbol{A}_k$ dado un estimado del subespacio de la señal en cada instante *k*,

Algoritmo C.4

1. Inicializar $\boldsymbol{A}_0$
2. ***for*** $\boldsymbol{k} = 1, \ldots, \infty$

   Paso 1: Actualización temporal

   $\tilde{\boldsymbol{A}}_k = \boldsymbol{A}_{k-1} + \alpha\, \boldsymbol{x}_k \boldsymbol{x}_k^T \boldsymbol{A}_{k-1}$

   Paso 2: Re-ortogonalización

   $\tilde{\boldsymbol{A}}_k = \boldsymbol{Q}\boldsymbol{R}$ (factorización QR)

   $\boldsymbol{A}_k = \boldsymbol{Q}$

   end

C.2.3. Simulaciones computacionales y comparaciones contra ACP

C.2.3.1. Métricas

Existe un gran número de métricas que podemos usar para la evaluación comparativa de la eficiencia de obtención de los autovalores y los autovectores. El principal problema que surge en la creación de dichas métricas reside en su aplicabilidad a métodos tan intrínsicamente disímiles, es decir,

1. matemático puro (ACP)
2. conexionistas
3. no conexionistas

por lo que deben diseñarse métricas *per-se* debiéndose realizar un exhaustivo análisis de aquellas pocas cosas que dichas familias de métodos tienen en común. En principio, la aspiración de obtener los autovalores y autovectores de un conjunto de vectores de representación. No obstante, y como se ha visto, estos métodos son tan distintos que en algunos casos la comparación solo será cualitativa mediante predicados en el contexto de una taxonomía, en lugar de una tabla.

En esta sección se diseñarán algunas métricas que hemos (*prima facie*) establecido como las más conspicuas para la ocasión, dado que no se han encontrado precedentes en la literatura del tema.

C.2.3.1.1. Complejidad computacional (CC)

Esta métrica está relacionada con la cantidad de operaciones de adición y multiplicación realizadas sobre los elementos de las matrices y vectores empleadas en el cálculo del algoritmo en cuestión. Se representa mediante el operador “O”, y en el caso particular de la ACP el mismo resulta $O((nfV \times nV)^3)$, léase, la cantidad de operaciones de adición y multiplicación necesarias para el cálculo de la ACP dependen de la cantidad de vectores (*nfV*) multiplicada por la cantidad de vectores (*nV*) al cubo.

C.2.3.1.2. Tiempo empleado por la Unidad Central de Procesamiento (TEUCP)

Este tiempo lo mediremos en segundos, para lo cual emplearemos para todas las simulaciones del presente trabajo a la función *built-in* de MATLAB® cputime [39].

C.2.3.1.3. Error Cuadrático Medio de Autovectores (ECMA)

El mismo se define como

$$\boldsymbol{ECMA} = \frac{1}{\boldsymbol{N}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{N}} \left(\boldsymbol{v_i} - \hat{\boldsymbol{v}}_{\boldsymbol{i}}\right)^T \left(\boldsymbol{v_i} - \hat{\boldsymbol{v}}_{\boldsymbol{i}}\right) \tag{C.64}$$

donde
$\boldsymbol{N}$ es el número de autovectores y autovalores
$\boldsymbol{v_i}$ son los autovectores como resultado de ACP
$\hat{\boldsymbol{v}}_{\boldsymbol{i}}$ son los autovectores como resultado del método aproximado

C.2.3.1.4. Error Cuadrático Medio de autovalores (ECMa)

El mismo se define como

$$\boldsymbol{ECMa} = \frac{1}{\boldsymbol{N}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{N}} \left(\lambda_{\boldsymbol{i}} - \hat{\lambda}_{\boldsymbol{i}}\right)^2 \tag{C.65}$$

donde
$\lambda_{\boldsymbol{i}}$ son los autovalores como resultado de ACP
$\hat{\lambda}_{\boldsymbol{i}}$ son los autovalores como resultado del método aproximado

C.2.3.2. Métricas Taxonómicas

A diferencia de las métricas anteriores, las cuales evalúan cuantitativamente el rendimiento de obtención de las componentes principales por parte de los métodos aproximados, las que siguen lo hacen cualitativamente, mediante preposiciones y predicados. Dichas métricas engrosarán una taxonomía, la cual se presenta en la próxima sección. Las mismas son:

a) Dificultad de Codificación (DC)
b) Completitud (Comp)
c) Estabilidad y Convergencia (E&C)
d) Coincidencia del Primero (CP)
e) Espacio Lineal Equivalente (ELE)
f) Sensibilidad a las Condiciones Iniciales y la Correcta Elección de los Parámetros (SCI&CEP)

Cabe realizar una serie de aclaraciones pertinentes:

1. **Dificultad de Codificación (DC):** Todas estas métricas se confrontan contra el método puro de obtención de las componentes principales, es decir, el ACP. De hecho, si partimos de la base de que el ACP se realiza mediante el algoritmo de Jacobi [25-27], la única que resulta de codificación más complicada que el mismo será AHK+MDE (calificación *alta*). El resto están a la par (calificación *mediana*) o son más simples de codificar (calificación *baja*).

2. **Completitud (Comp):** Por completitud entendemos que en el caso de los métodos conexionistas y a los efectos de obtener el mismo número de componentes principales que ACP, hay que definir elementos ficticios que acompañen a la matriz $\boldsymbol{X}$ de representación a modo de $\boldsymbol{X}+\boldsymbol{\Delta X}$. Cosa que no sucede en los métodos no-conexionistas. Por lo cual dicha métrica se presenta como una pregunta: *Hay que completar con ficticios*?

3. **Estabilidad y Convergencia (E&C): E**sta métrica hace referencia a la tendencia del algoritmo empleado en cuestión como sistema dinámico *convergiendo o no a las componentes principales o no*. Es decir, si converge aunque más no sea a un sistema lineal equivalente, pero converge o bien, ni siquiera eso.

4. **Espacio Lineal Equivalente (ELE):** Esta métrica tiene que ver con el hecho de que si bien el método aproximado no arriba a todos los mismos componentes principales que ACP, por lo menos, si el espacio muestral es de un solo vector, el autovector resultante (y su correspon diente autovalor asociado) son coincidentes con el método ACP. En definitiva, esta métrica se vincula fuertemente con las dos anteriores, y tiene lugar cuando el método aproximado descorrelaciona pero no para los mismos componentes principales que ACP, arribando a un espacio lineal diferente del mismo.

5. **Sensibilidad a las Condiciones Iniciales y la Correcta Elección de los Parámetros (SCI&CEP):** Esto se da con gran énfasis en todo tipo de algoritmo recursivo, ya sea, los conexionistas como algunos no-conexionistas que responden a esta forma. Dicha sensibilidad decidirá en qué rangos el método arriba algún tipo de solución o directamente diverge.

C.2.3.3. Simulaciones y análisis comparativo de resultados

Dada una matriz de representación $\boldsymbol{X}$ de dimensiones *nf*$\boldsymbol{X}$-por-*nc*$\boldsymbol{X}$ el problema a simular consiste en obtener los autovectores y autovalores del problema que se describirá. Como puede observarse de la Tabla C.2 y la Taxonomía C.1, las dos mejores aproximaciones las constituyen el AHG y el ACP Kalmaniano, siendo el mejor este último, en virtud de sus ventajas comparativas numéricas y taxonómicas. Aunque en rigor, ningún método reemplaza al método puro matemático [459, 460], incluyendo el método de Jacobi [39, 459, 460], es muy interesante la compilación de resultados obtenidos aquí para todos los métodos aproximados. Curiosamente, el mejor de los conexionistas resulta ser el AHG, incluso por encima de sus aparentes perfeccionamientos y mejoramientos, es decir, AHK y AHK+MDE, aunque ésto ya es insinuado entre líneas por ciertos autores [461].

Para estas simulaciones se empleó un vector de representación $\boldsymbol{X}$ de 15-por-3 y las simulaciones se corrieron todas sobre MATLAB® (Mathworks, Natick, MA) [39] sobre una PC con una CPU Intel® Core(TM) QUAD Q6600 de 4 procesadores de 2.40 GHz y 4 GB RAM.

Tabla C.2

Técnicas		Métricas			
		CC	TEUCP	ECMA	ECMa
Conexionistas	AOK	O(*nc**X***^2* *nf**X***)	2.1134	456.8765	232.9000
	AHG	O(*nc**X***^2* *nf**X***)	3.7969	1.3337	2.4692e-005
	AHK	O(*nc**X***^2* *nf**X***)	4.7863	3.1112	2.8765e-001
	AHK+MDE	O(*nc**X***^2* *nf**X***)	10.6709	4.2777	4.4543e-001
No conexionistas	ACP kernelizado	O(*nf**X***^3)	5.1122	0.2314	1.6354e-007
	ACP rápido	O(*nf**X***^2* *nc**X***)	0.1216	768.5327	732.0112
	ACP recursivo	O(*nf**X***^2* *nc**X***)	0.2298	645.4211	566.3211
	ACP sistólico	O(*nc**X***^2* *nf**X***)	0.3119	636.9776	542.0001
	ACP kalmaniano	O(*nf**X***^2* *nc**X***+ *nc**X***^3)	2.2980	0.4675	2.8532e-005
	AGE	O(*nc**X***^2* *nf**X***)	0.3645	671.7361	571.0005

Taxonomía C.1

Técnicas		Métricas					
		DC	Comp	E&C	CP	ELE	SCI&CEP
Conexionistas	AOK	Baja	Si	Buena	Si	Si	Baja
	AHG	Baja	Si	Pobre	Si	No	Baja
	AHK	Media	Si	Buena	Si	No	Media
	AHK+MDE	Alta	Si	Buena	Si	No	Alta
No conexionistas	ACP kernelizado	Media	No	Muy Buena	Si	Si	Nula
	ACP rápido	Baja	No	Muy Buena	No	Si	Nula
	ACP recursivo	Media	No	Muy Buena	No	Si	Media
	ACP sistólico	Media	No	Muy Buena	No	Si	Nula
	ACP kalmaniano	Media	No	Muy Buena	Si	No	Baja
	AGE	Media	No	Muy Buena	No	Si	Baja

C.3 Conclusiones del apéndice

En este apéndice se desarrollaron en detalle los métodos aproximados y rápidos para la implementación de ACP. Como se mencionó en la Sección 1.2.5.2 dichos métodos en ningún caso igualan en precisión, o eficiencia de decorrelación a ACP. De hecho, en algunos casos y más allá de la CC declarada por los autores en cuestión son bastante más lentos que la misma ACP.

Apéndice D

D. Formatos JPEG y JPEG2000

D.1. Introducción

En este apéndice se verán en detalle los algoritmos de JPEG y JPEG2000 con los cuales se medirán en el Capítulo 4 (Simulaciones computacionales) las técnicas propuestas en el Capítulo 3.

D.2. Formatos JPEG y JPEG2000

El formato de archivo gráfico conocido por sus siglas en inglés JPEG (Joint Photographic Experts Group) es actualmente el más conocido, diseminado y utilizado en la industria de las imágenes electrónicas. JPEG2000 es una versión posterior y mejorada que (entre otras cosas) incorpora la tecnología de onditas, a los efectos de aumentar dramáticamente la tasa de compresión. En el presente apéndice, se verán ambos en detalle.

D.2.1. Formato JPEG

D.2.1.1. Requerimientos y proceso de selección

El objetivo de JPEG ha sido desarrollar un método para compresión de imágenes de tono continuo que cumpla los siguientes requisitos:

1. estar en o cerca del estado del arte con respecto a la tasa de compresión y acompañando la fidelidad de la imagen, a lo largo de una amplia gama de métricas de calidad de la imagen y, especialmente, en el rango, donde la fidelidad al original se caracteriza como "muy buena" a "excelente"; también, el codificador debería ser parametrizable, a fin de que la aplicación (o usuario) pueda ajustar el compromiso deseado de compresión/calidad;

2. ser aplicable a prácticamente cualquier clase de imagen fuente digital de tono continuo (i.e. para propósitos más prácticos no estar restringido a imágenes de ciertas dimensiones, espacios de color, tasas de aspecto de pixel, etc.) y no estar limitada a cierta clase de imagenología con restricciones sobre el contenido de una escena, tal como complejidad, rango de colores, o propiedades estadísticas;

3. tenga una CC tratable, para hacer factible implementaciones en software de una performance viable sobre una gran variedad de CPU's, tan bien como las implementaciones de hardware con un costo viable para aplicaciones que requieren alta performance;

4. tenga los siguientes modos de operación:
   - Codificación Secuencial: cada componente de la imagen es codificada en una única exploración izquierda-a-derecha, arriba-a-abajo;

   - Codificación Progresiva: la imagen es codificada en múltiples exploraciones para aplicaciones en las cuales el tiempo de transmisión es largo, y el usuario prefiere mirar la imagen a crearse en múltiples pasos rústico-a-claro;

- Codificación sin pérdidas: la imagen es codificada para garantizar la exacta recuperación de los valores de los pixeles de cada imagen fuente (aunque el resultado sea una compresión baja comparada a los modos con pérdidas);

- Codificación Jerárquica: la imagen es codificada a múltiples resoluciones a fin de que puedan ser accedidas versiones de baja resolución sin tener que primero descomprimir la imagen en toda su resolución.

En Junio de 1987, JPEG condujo un proceso de selección basado en una evaluación a ciegas de la calidad subjetiva de una foto, y se redujo de doce métodos a tres. Tres grupos de trabajo informal formados para refinarlos, y en Enero de 1988, un segundo proceso de selección más riguroso [469] reveló que la propuesta conocida como TDC Adaptativa (TDCA) [470], basada en TDC de 8x8, produjo la mejor calidad de imagen.

Al momento de su selección, el método basado en TDC fue solo parcialmente definido para algunos de los modos de operación. De 1988 a 1990, JPEG emprendió la importante tarea de definir, documentar, simular, probar, validar, y simplemente llegar a un acuerdo sobre la plétora de detalles necesarios para una verdadera interoperabilidad y universalidad. Más detalles sobre la historia de los esfuerzos del JPEG pueden verse en [471-474].

D.2.1.2. Arquitectura del estándar propuesto

El estándar propuesto contiene los cuatro “modos de operación” previamente identificados. Para cada modo, uno o más codecs distintivos son especificados. Los codecs difieren en un modo de acuerdo a la precisión de las muestras de la imagen fuente que ellos pueden manejar o el método de codificación de entropía que ellos usan. Aunque la palabra codec (codificador/decodificador) es usada frecuentemente en esta tesis, no existe ningún requisito acerca de que las implementaciones deben incluir ambos, es decir, un codificador y un decodificador. Muchas aplicaciones tendrán sistemas o dispositivos que requieren exclusivamente una o la otra.

Los cuatro modos de operación y sus varios codecs han resultado del objetivo de JPEG de ser genéricos y de la diversidad de los formatos de imagen a través de las aplicaciones. Las piezas múltiples pueden dar la impresión de una complejidad indeseable, pero en realidad debería ser considerada como una "caja de herramientas" que pueden extenderse a una amplia gama aplicaciones de imágenes de tono continuo. Es poco probable que muchas implementaciones utilicen todas las herramientas - de hecho, la mayoría de las primeras implementaciones - ahora en el mercado (incluso antes de finalizada la aprobación ISO) - han aplicado sólo la secuencia de referencia del codec.

El codec secuencial de referencia es inherentemente un rico y sofisticado método de compresión el cual será suficiente para muchas aplicaciones. Obtener esta mínima capacidad del JPEG implementada apropiadamente y su interoperabilidad proporciona a la industria una importante capacidad inicial para el intercambio de imágenes a través de los proveedores y las aplicaciones

D.2.1.3. Pasos del procesamiento para una codificación basada en la TDC

Las Figuras D.1 y D.2 muestran los principales pasos de procesamiento los cuales son el corazón de los modos de operación basados en TDC. Estas figuran ilustran el caso especial de compresión de imágenes de componente única (escala de grises). El lector puede captar la esencia de la

compresión basada en TDC pensando esencialmente en una compresión de cadenas de bloques de 8x8 pixeles de la imagen en escala de grises. La compresión de una imagen a color consiste en transformar las componentes RGB (azul, verde y rojo; fuertemente correlacionadas) al espacio de componentes YIQ (espacio de color usado en TV a colores; fuertemente descorrelacionadas) y cada una de estas tres bandas se comprime como corresponde con diferentes criterios.

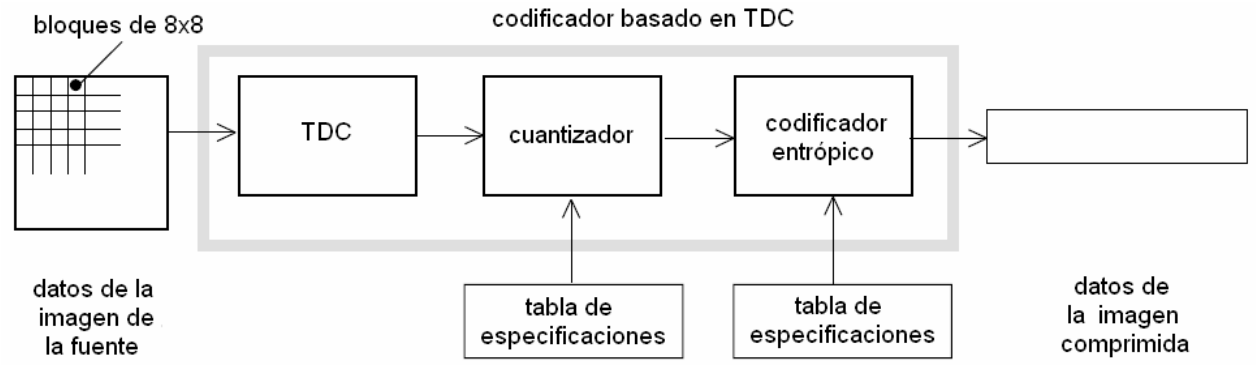


Figura D.1: Pasos de procesamiento del codificador basado en TDC.

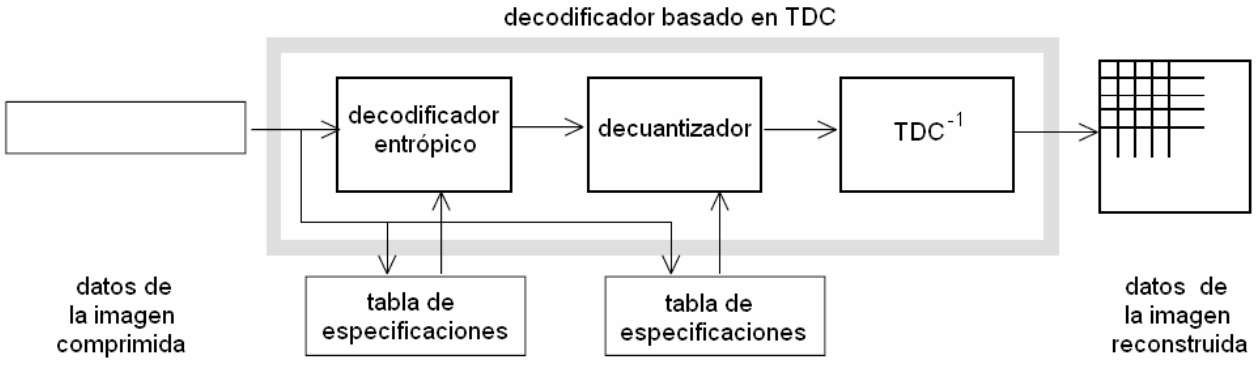


Figura D.2: Pasos de procesamiento del decodificador basado en TDC.

Para los codecs de TDC de modo secuencial, los cuales incluyen el codec secuencial de referencia, el diagrama simplificado indica cómo trabaja la compresión de componente única de una forma bastante completa. Cada bloque de 8x8 es una entrada, hace su camino a través de cada paso del proceso, y genera la salida en forma comprimida en la cadena de datos. Para los codecs TDC de modo progresivo, existe una imagen de reserve antes del paso de codificación entrópica, de manera que una imagen se puede almacenar y, a continuación, separar en múltiples exploraciones, sucesivamente, con una mejora sucesiva de la calidad. Para el modo de operación jerárquico, los pasos mostrados son usados para construir bloques en contexto mucho mayor.

D.2.1.3.1. TDC y TDC^{-1} para bloques de 8x8 pixeles

Como se vio en detalle en el Capítulo 2, solo hay que considerar a la Ecuaciones (2.4) y (2.5) para bloques de 8x8 pixeles para poder ser aplicados en JPEG.

D.2.1.3.2. Cuantización

Como se vio en detalle en el Apéndice B, la cuantización considerada en JPEG será la del tipo escalar.

D.2.1.3.3. Codificación DC y secuencia zig-zag

Luego de la cuantización, el coeficiente DC es tratado separadamente de los 63 coeficientes AC. El coeficiente DC es una medida del valor promedio de los 64 pixeles del bloque. Dado que existe usualmente una fuerte correlación entre los coeficientes DC de los bloques 8x8 adyacentes, el coeficiente DC cuantizado es codificado como la diferencia del término DC del previo bloque en el orden codificado (definido a continuación), como se muestra en la Fig.D.3. Este tratamiento especial vale la pena, dado que los coeficientes DC contienen frecuentemente una fracción significativa del total de la energía de la imagen.

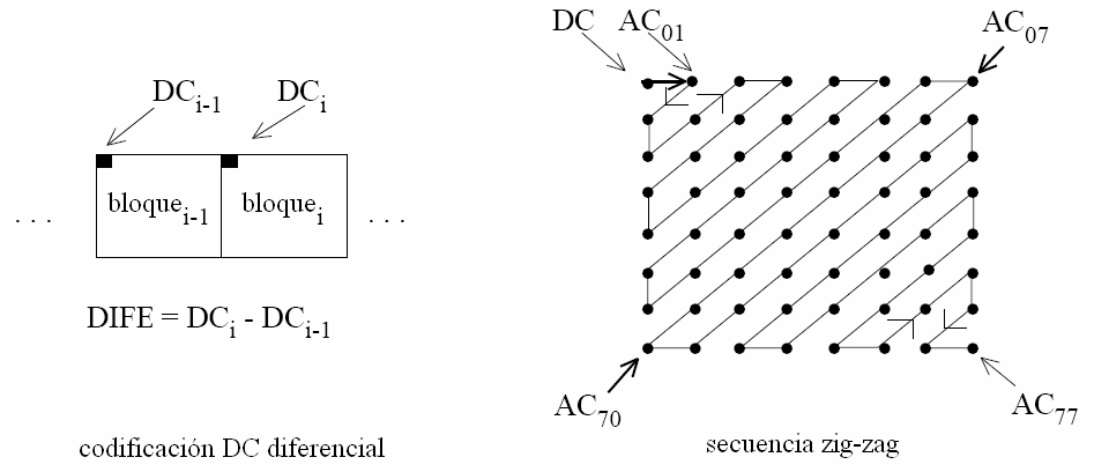


Figura D.3: Preparación de los coeficientes de cuantizado para la codificación entrópica.

Finalmente, la totalidad de los coeficientes cuantizados son ordenados en la secuencia “zig-zag” (vista en detalle en el Apéndice A), de frecuencias menores (mayor energía) a frecuencias mayores (menor energía) también mostrada en la Fig.D.3, lo cual, en rigor, constituye un criterio eurístico, no un ordenamiento. Este ordenamiento ayuda a facilitar la codificación de la entropía al poner los coeficientes de baja frecuencia (los cuales tienen más probabilidad de ser distintos de cero) antes de los coeficientes de alta frecuencia.

D.2.1.3.4. Codificación entrópica

Como se vio en detalle en el Apéndice B, la codificación entrópica la emplearemos previa al canal o bien al proceso de almacenamiento.

D.2.1.3.5. Compresión y calidad de la imagen

Para imagines color con escenas moderadamente complejas, todos los modos de operación basados en TDC producen típicamente los siguientes niveles de calidad de imagen para los rangos indicados de compresión. Estos niveles son simplemente una guía – la calidad y la compresión pueden variar significativamente de acuerdo a las características de la imagen fuente

y el contenido de la escena. (Las unidades “bits/pixel” significan aquí el número total de bits en la imagen comprimida – incluyendo las componentes de crominancia – dividido por el número de muestras en la componente de luminancia.)

• 0.25-0.5 bits/pixel: moderada a buena calidad, suficiente para algunas aplicaciones;

• 0.5-0.75 bits/pixel: buena a muy buena calidad, suficiente para muchas aplicaciones;

• 0.75-1/5 bits/pixel: excelente calidad, suficiente para la mayoría de las aplicaciones;

• 1.5-2.0 bits/pixel: usualmente indistinguible de la original, suficiente para la mayoría de las aplicaciones demandadas.

D.2.1.4. Pasos del procesamiento para un codificador predictivo sin pérdidas

Después de su selección de un método basado en TDC en 1988, JPEG descubrió que los modos sin pérdidas basados en TDC eran difíciles de definir como una práctica estándar en contra de los codificadores y decodificadores que podrían ser aplicados independientemente, sin imponer importantes limitaciones en las implementaciones de ambos codificador y decodificador.

JPEG, con el objeto de cumplir con su requisito de un modo de operación sin pérdidas, ha optado por un método predictivo simple el cual es totalmente independiente del procesamiento de la TDC descrito anteriormente. La selección de este método no fue el resultado de una evaluzación competitiva rigurosa como fue el caso del método basado en TDC. Sin embargo, el método sin pérdidas JPEG produce resultados que, a la luz de su sencillez, se hallan sorprendentemente cerca del estado del arte para la compresión sin pérdidas de tono continuo, como se indica en un reciente informe técnico [475].

La Fig.D.4 muestra los pasos principales del procesamiento para una imagen de componente única. Un predictor combina los valores de hasta tres muestras de vecinos (A, B, y C) para realizar una predicción de la muestra indicada por X en Fig.D.5. Esta predicción es entonces sustraida del valor actual de la muestra X, y la diferencia es codificada sin pérdida por algunos de los métodos de codificación entrópica - Huffman o Aritmético. Pueden ser usados cualquiera de los ocho predictores listados en la Tabla.D.I (bajo “valor de selección”).

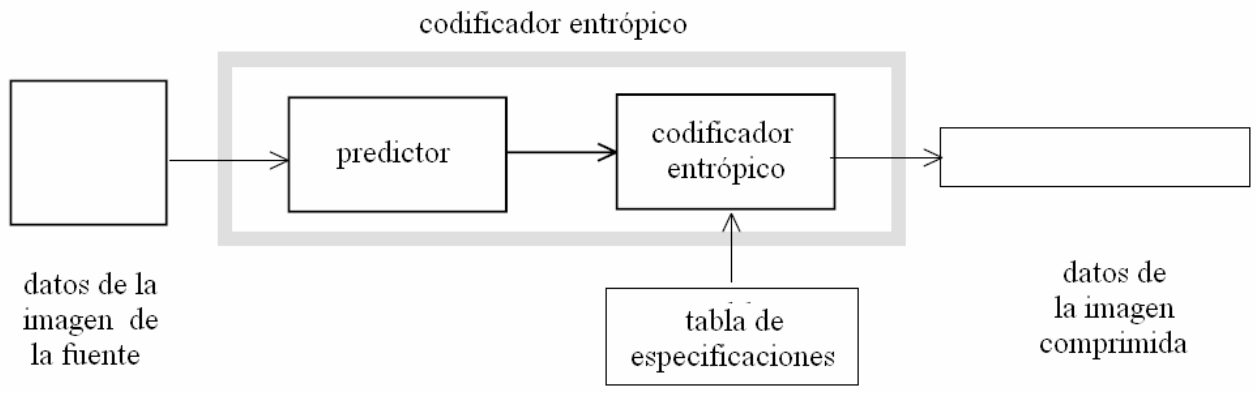


Figura D.4: Pasos de procesamiento del codificador en modo sin pérdidas.

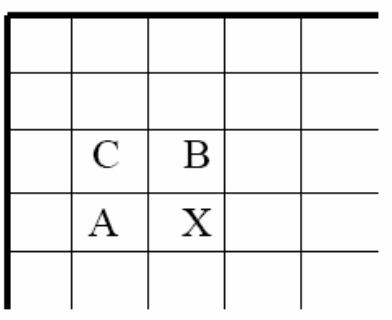


Figura D.5: Vecindario de predicción de 3-muestras.

Tabla.D.I: Predictores para codificación sin pérdida.

Valor de selección	Predicción
0	Sin predicción
1	A
2	B
3	C
4	A+B-C
5	A+((B-C)/2)
6	B+((A-C)/2)
7	(A+B)/2

Las selecciones 1, 2, y 3 son predictores uni-dimensionales y las selecciones 4, 5, 6 y 7 son predoctores bi-dimensionales. La selección-valor 0 puede ser usada solamente para codificación diferencial en el modo de operación jerárquico. La codificación entrópica es casi idéntica a la usada para el coeficiente DC descripto en la Sección D.2.1.6.1. (para codificación de Huffman).

Para el modo de operación sin pérdidas, son especificados dos diferentes codecs – uno por cada método de codificación entrópica. Los codificadores pueden usar cualquier precisión de imagen fuente de 2 a 16 bits/muestra, y puede usar cualquiera de los predictores excepto el de valor 0 de selección. Los decodificadores deben manejar cualquiera de las precisiones de muestra y cualquiera de los predictores. Los codificadores sin pérdidas producen típicamente una compresión de alrededor de 2:1 para imágenes color con escenas moderadamente complejas.

D.2.1.5. Imágenes de múltiples componentes

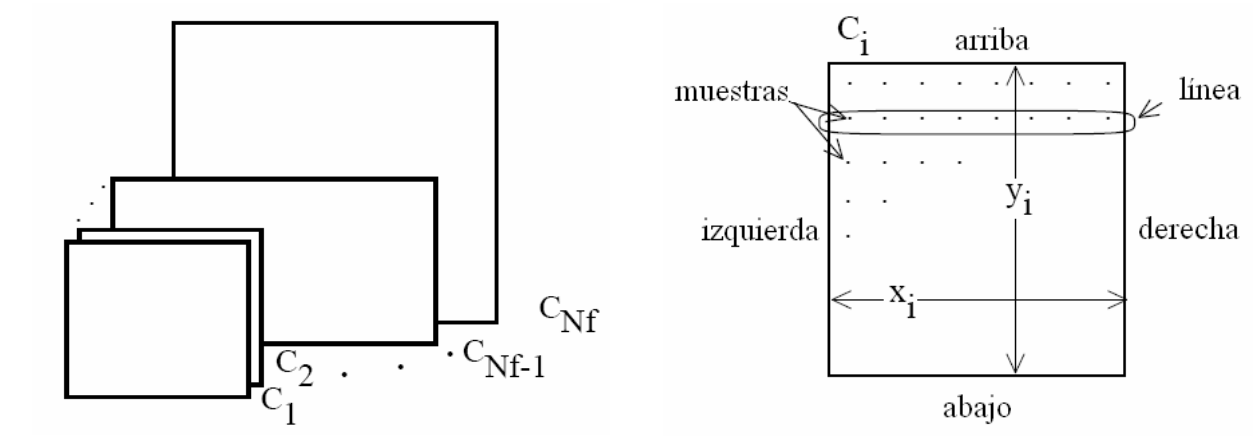
En las secciones previas se discutieron los pasos principales de los codecs basados en TDC y predictivos sin pérdidas para el caso de imágenes fuente de componente única. Estos pasos logran la compresión de los datos de la imagen. Pero una buena parte de la propuesta JPEG también se ocupa de la manipulación y el control del color (o dicho de otra forma) imágenes con múltiples componentes. El objetivo de JPEG para un estándar de compresión genérica requiere su propuesta para dar cabida a una variedad de formatos de imagen fuente.

D.2.1.5.1. Formatos de imagen de fuente

El modelo de imagen fuente usado en la propuesta JPEG es una abstracción de una variedad de tipos de imágenes y aplicaciones, y consiste de solo lo que es necesario para comprimir y

reconstruir los datos de la imagen digital. El lector debería reconocer que el formato de datos comprimidos JPEG no codifica suficiente información para servir como una representación completa de la imagen. Por ejemplo, JPEG no especifica o codifica cualquier información sobre tasas de aspecto s de pixeles, espacio de color, o características de adquisición de la imagen.

La Fig.D.6 ilustra el modelo de imagen fuente JPEG. Una imagen fuente contiene de 1 a 255 componentes de la imagen (Nf), algunas veces llamado color o bandas espectrales o canales. Cada componente consiste en un arreglo rectangular de muestras. Una muestra se define para ser un entero sin signo con una precisión de *P* bits, con cualquier valor en el rango [0, 2^P-1]. Todas las muestras de todos los componentes en la misma imagen fuente debe tener la misma precisión *P*. *P* puede ser 8 o 12 para codecs basados en TDC, y 2 a 16 para codecs predictivos.



(a) Imagen fuente con múltiples componentes. (b) Características de una componente de imagen.

Figura D.6: Modelo de imagen de fuente JPEG.

La *i*-ésima componente tiene las dimensiones de la muestra x_i por y_i. Para acomodar los formatos en los cuales algunas componentes de la imagen son muestreadas a diferentes tasas que otras, componentes pueden tener diferentes dimensiones. Las dimensiones deben tener un relación integral mutua definida por H_i y V_i, los factores de muestreo relativo horizontal y vertical, el cual debe ser especificado para cada componente. Las dimensiones totales de la imagen *X* e *Y* están definidas como el máximo x_i e y_i para todas las componentes en la imagen, y pueden ser cualquier número hasta 2^{16}. Para *H* y *V* están permitidos solo los valores enteros 1 a 4. Los parámetros codificados son *X*, *Y*, y H_is y V_is para cada componente. El decodificador reconstruye las dimensiones x_i e y_i para cada componente, de acuerdo a las siguientes relaciones que muestra la Ecuación D.1:

$$\boldsymbol{x_i} = \left\lceil \boldsymbol{X} \times \frac{\boldsymbol{H_i}}{\boldsymbol{H}_{\max}} \right\rceil$$
$$\boldsymbol{y_i} = \left\lceil \boldsymbol{Y} \times \frac{\boldsymbol{V_i}}{\boldsymbol{V}_{\max}} \right\rceil \qquad \text{(D.1)}$$

donde $\lceil \ \rceil$ es la función “pasa al entero mayor más próximo” .

D.2.1.5.2. Orden de codificación e entrelazado

Un estándar de compresión práctico de imagines debe abordar como los sistemas necesitarán manejar los datos durante el proceso de descompresión. Muchas aplicaciones necesitan vehiculizar el proceso de mostrar e imprimir imágenes de múltiples componentes en paralelo con el proceso de descompresión. Para muchos sistemas, esto solo es posible si los componentes son intercalados junto con la cadena de datos comprimidos.

Para hacer el mismo mecanismo de entrelazado aplicable a ambos codecs: basado en TDC y predictivo, la propuesta JPEG ha definido el concepto de "unidad de datos." Una unidad de datos es una muestra en codecs predictivos y un bloque 8x8 de muestras en codecs basados en TDC.

El orden en el cual las unidades de datos comprimidas tienen lugar en la cadena de datos comprimidos es una generalización del orden de exploración por filas. Generalmente, las unidades de datos son ordenadas de izquierda-a-derecha y de arriba-a-debajo de acuerdo a la orientación mostrada en la Fig.D.6. (Es responsibilidad de las aplicaciones definir cuales bordes de la imagen fuente son arriba, abajo, izquierda y derecha.) Si una componente de imagen es no-entrelazada (i.e., comprimida sin ser entrelazada con otras componentes), las unidades de datos comprimidas están ordenadas una exploración de raster pura como muestra la Fig.D.7.

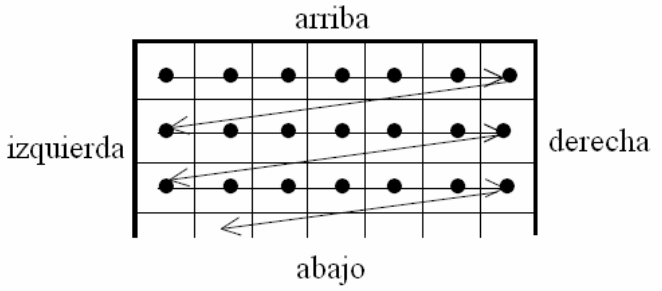


Figura D.7: Ordenamiento no-entrelazado de los datos.

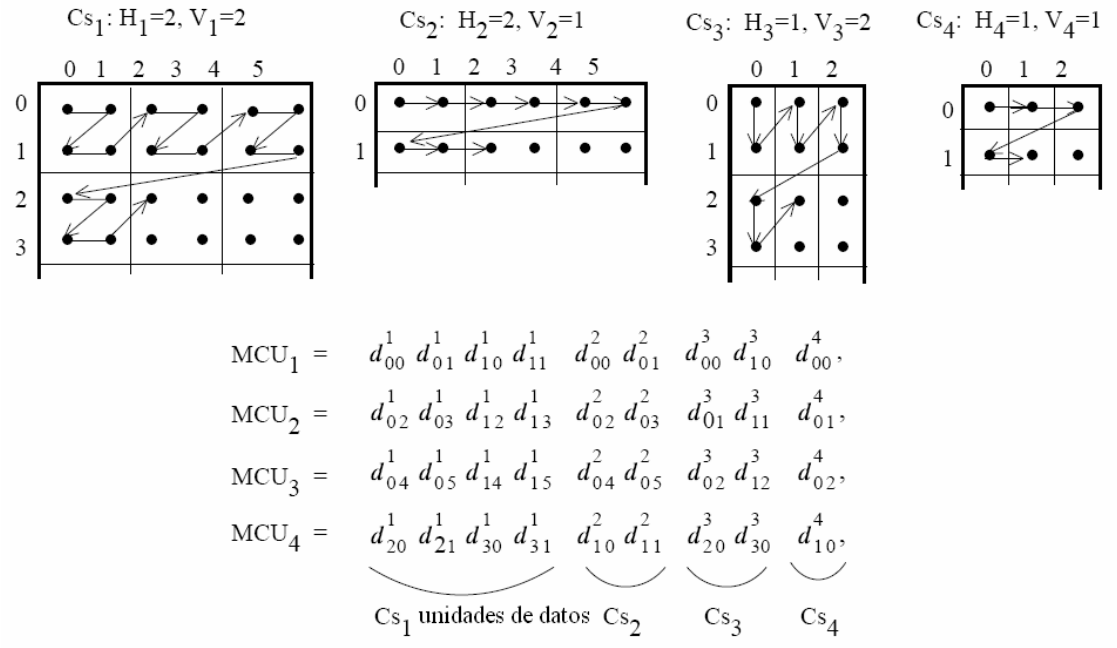


Figura D.8: Ejemplo de ordenamiento entrelazado de los datos.

Cuando dos o más componentes están entrelazados, cada componente C_i es particionada en regiones rectangulares de H_i por V_i unidades de datos, como se muestra en el ejemplo generalizado de la Fig.D.8. Las regiones están ordenadas en una componente de izquierda-a-derecha y arriba-a-abajo, y en una región, unidades de datos son ordenados de izquierda-a-derecha y arriba-a-abajo. La propuesta JPEG define el término Minimum Coded Unit (MCU, unidad de codificado mínimo) a ser el grupo más pequeño de unidades de datos entrelazados. Para el ejemplo mostrado, MCU1 consiste de unidades dato que toman primero de la mayoría de las regiones arriba-izquierda de C_1, seguido por las unidades de datos de la misma región de C_2, y asimismo para C_3 y C_4. MCU2 continúa el patrón mostrado.

Entonces, el dato entrelazado es una secuencia ordenada de MCUs, y el número de unidades dato contenida en un MCU es determinado por el número de componentes entrelazados y sus factores de muestreo relativo. El número máximo de componentes los cuales pueden ser entrelazados es 4 y el número máximo de unidades dato en un MCU es 10. La última restricción es expresada como se muestra en la Ecuación (D.2), donde la sumatoria se realiza sobre las componentes entrelazadas:

$$\sum_{\substack{\textbf{todos los } i \textbf{ en} \\ \text{int } \textbf{erlaceado}}} \boldsymbol{H_i} \times \boldsymbol{V_i} \leq 10 \qquad \text{(D.2)}$$

Debido a esta restricción, no todas las combinaciones de 4 componentes las cuales pueden ser representadas en un orden no-entrelazado en una imagen comprimida JPEG son permitidas para ser entrelazadas. Además, nótese que la propuesta JPEG permite algunos componentes ser entrelazados y algunos ser no-entrelazados en la misma imagen comprimida.

D.2.1.5.3. Tablas múltiples

En adición al control de entrelazado discutido previamente, los codecs JPEG que deben controlar la aplicación de la tabla de datos propia para los componentes propios. La misma tabla de cuantización y la misma tabla de codificación entrópica (o conjunto de tablas) deben ser usadas para codificar todas las muestras en un componente.

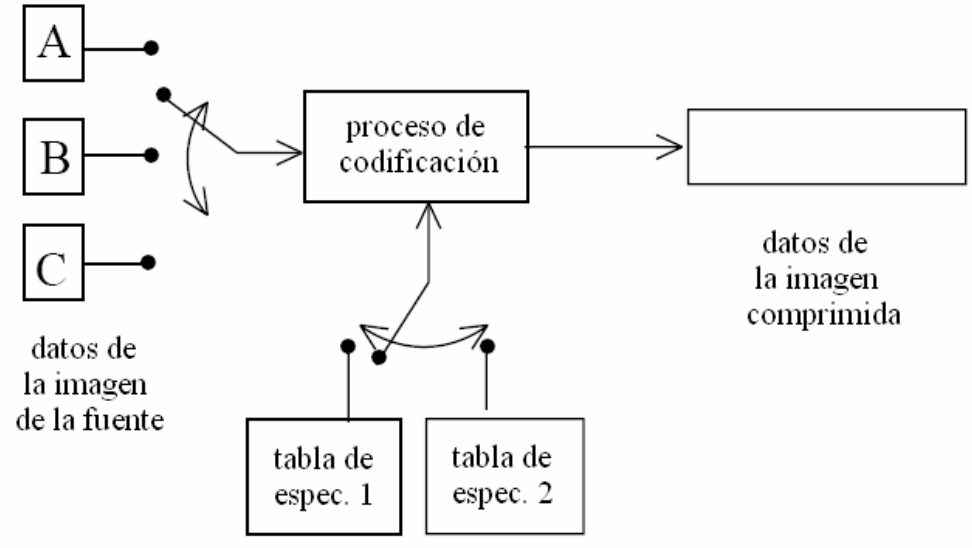


Figura D.9: Componente entrelazado y control de selección de tabla.

Los decodificadores JPEG pueden almacenar hasta 4 diferentes tablas de cuantización y hasta 4 diferentes (conjuntos de) tablas de codificación entrópica simultáneamente. (El decodificador secuencial de referencia es la excepción; dado que puede almacenar solo hasta 2 conjuntos tablas de codificación entrópica.) Esto es necesario para conmutación entre diferentes tablas durante la descompresión de una exploración conteniendo múltiples (entrelazados) componentes, en función de aplicar la tabla propia para la componente propia. (Las tablas no pueden ser cargadas durante la descompresión de una exploración.) La Fig.D.9 ilustra el control de conmutación de tabla que debe ser manejado en conjunción con el entrelazado de componentes múltiples para el lado del codificador. (Esta visión simplificada no distingue entre cuantización y tablas de codificación entrópica.)

D.2.1.6. Referencia y otros codificadores TDC secuenciales

El modo de operación secuencial TDC consiste de los pasos de TDC y cuantización de la Sección D.2.1.3.2, y el control de componente múltiple de la Sección D2.1.5.3. En adición al codec secuencial de referencia, otros codecs secuenciales TDC están definidos para acomodar las dos precisiones de muestras diferentes (8 y 12 bits) y los dos diferentes tipos de métodos de codificación entrópica (Huffman y Aritmético).

La codificación secuencial de referencia es para imágenes con 8-bit por muestra y usa codificación de Huffman exclusivamente. Esto difiere de los codecs de TDC secuencial en que su decodificador puede almacenar solo dos conjuntos de tablas de Huffman (una tabla AC y tabla DC por conjunto). Esta restricción significa que, para imagines con tres o cuatro componentes entrelazados, al menos un conjunto de tablas de Huffman deben ser compartidas por dos componentes. Esta restricción no plantea ninguna limitación en todos los componentes no-entrelazados; un nuevo conjunto de tablas puede ser cargado en el decodificador antes de que comience la descompresión de componentes no-entrelazados. Para muchas aplicaciones las cuales necesitan entrelazar tres componentes de color, esta restricción es una dura limitación en todos. Los espacios de color (YUV, CIELUV, CIELAB, y otros) los cuales representan la información cromática (‘‘color’’) en dos componentes y la información acromática (‘‘escala de grises’’) en una tercera son más eficientes para la compresión que los espacios tales como RGB [476]. Un conjunto de tablas de Huffman puede ser usado para la componente acromática y una para las componentes cromáticas. La estadística de los coeficientes TDC son similares para las componentes de crominancia de la mayoría de las imágenes, y un conjunto de tablas de Huffman pueden codificar ambos al menos tan óptimamente como dos.

El comité consideró también que la pronta disponibilidad de implementaciones en un único chip a precios accesibles animaría la pronta aceptación de la propuesta JPEG en una variedad de aplicaciones. En 1988 cuando el secuencial de referencia fue definido, el comité de expertos de VLSI consideró que la actual tecnología permitió la agrupación de cuatro tablas cargadas de Huffman – en adición a cuatro conjuntos de tablas de cuantización - sobre un único chip de codec a precio accesible era una propuesta arriesgada. Los pasos de procesamiento: TDC, cuantización, diferenciación de DC, y el ordenamiento zig-zag para el codec secuencial de referencia que acaba de proceder, se describen en la Sección D.2.1.3. Antes de la codificación entrópica, usualmente hay pocos coeficientes distintos de cero y muchos ceros. La tarea de la codificación entrópica consiste en codificar estos pocos coeficientes eficientemente. La descripción de la codificación entrópica secuencial de referencia está dada en dos pasos: la conversión de los coeficientes TDC cuantizados en una secuencia de símbolos y un cesión de códigos de longitud variable para los símbolos.

D.2.1.6.1. Representaciones de codificación entrópica intermedia

En la secuencia de símbolos intermedia, cada coeficiente AC distinto de cero es representado en combinación con la ‘‘longitud de la corrida’’ (número consecutivo) los coeficientes AC de valor cero los cuales preceden en la secuencia zig-zag. Cada una de esas combinaciones de coeficientes de longitud de corrida distinta de cero es (usualmente) representada por un par de símbolos:

símbolo-1	símbolo-2
(LONGITUD de la CORRIDA, DIMENSION)	(AMPLITUD)

El símbolo-1 representa dos piezas de información, LONGITUD de la CORRIDA y DIMENSION. El símbolo-2 representa una pieza única de información designada AMPLITUD, la cual es simplemente la amplitud de los coeficientes AC distintos de cero. La LONGITUD de la CORRIDA es el número de coeficientes AC cero consecutivos en una secuencia zig-zag precediendo al coeficiente AC distinto de cero representado. La DIMENSION es el número de bits usados para codificar AMPLITUD – es decir, para codificar el símbolo-2, por la codificación d eentero con signo usada con el método particular de JPEG de codificación de Huffman. La LONGITUD de la CORRIDA representa los ceros corridos de longitud 0 a 15. En este punto los ceros corridos en la secuencia zig-zag pueden ser más grandes que 15, así como el valor del símbolo-1 (15, 0) es interpretado como el símbolo extensión con *longituddelacorrida* = 16. Puede ser de tres extensiones consecutivas (15, 0) antes la terminación del símbolo-1 cuyo valor de LONGITUD de la CORRIDA completa la actual *longituddelacorrida*. La terminación del símbolo-1 es siempre seguida por un único símbolo-2, excepto para el caso en el cual la última corrida de ceros incluye los últimos coeficientes AC (63d). En este caso frecuente, el valor especial del símbolo-1 (0,0) significa EOB (end of block, fin de bloque), y puede ser visto como un símbolo de ‘‘escape’’ el cual finaliza el bloque de 8x8 muestras. Entonces, para cada bloque de 8x8 muestras, la secuencia zig-zag de 63 coeficientes AC cuantizados se representa como una secuencia de pares de símbolos símbolo-1, símbolo-2, aunque cada uno ‘‘par’’ puede tener repeticiones de símbolo-1 en el caso de una gran *longituddelacorrida* o solo un símbolo-1 en el caso de un EOB. El rango posible de coeficientes AC cuantizados determina el rango de valores en el cual ambos, la información de AMPLITUD y de la DIMENSION deben ser representados. Un análisis numérico de la ecuación de la TDC para 8x8 muestra que, si el punto-64 (bloque 8x8) de la señal de entrada contiene N-bits enteros, entonces la parte no-fraccional de los números de salida (coeficientes TDC) pueden crecer en más de tres bits. Esta es también la mayor dimensión posible de un coeficiente TDC cuantizado cuando su dimensión de paso de cuantizado tiene valor entero 1. El secuencial de referencia tiene muestras de fuente enteras de 8-bits en el rango $[-2^7, 2^7-1]$, así las amplitiudes del coeficiente AC cuantizado están cubiertos por enteros en el rango $[-2^{10}, 2^{10}-1]$. La codificación de entero con signo usa símbolo-2 con AMPLITUD de codificado de 1 a 10 bits de longitud (así la DIMENSION también representa valores desde 1 a 10), y la LONGITUD de la CORRIDA representa valores de 0 a 15 como se discutió previamente. Para los coeficientes, la estructura de las representaciones intermedias de los símbolo-1 y símbolo-2 se ilustra en las Tablas D.II y D.III, respectivamente.

Tabla.D.II: Codificación en base de Huffman, estructura del símbolo-1

	DIMENSION					
	0	1	2	…	9	10
LONGITUD de la CORRIDA	0	EOB				
	.	X				
	.	X	valores de la			
	.	X	DIMENSION de la CORRIDA			
	15	ZRL				

Tabla.D.III: Base de codificación entrópica, estructura del símbolos-2

DIMENSION	AMPLITUD
1	-1, 1
2	-3, -2, 2, 3
3	-7.. -4, 4.. 7
4	-15..-8, 8,..15
5	-31..-16, 16,..31
6	-63..-32, 32..63
7	-127 .. -64, 64 .. 127
8	-255 .. -128, 128 .. 255
9	-511 .. -256, 256 .. 511
10	-1023 .. -512, 512 .. 1023

La representación intermedia para un coeficiente DC diferencial del bloque de muestra de 8x8 se estructura en forma similar. El símbolo-1, no obstante, representa solo la información de DIMENSION; el símbolo-2 representa la información AMPLITUD como vimos antes:

símbolo-1	símbolo-2
(DIMENSION)	(AMPLITUD)

Dado que el coeficiente DC se codifica diferencialmente, esta cubierto por el doble de valores, $[-2^{11}, 2^{11}-1]$ como los coeficientes AC, de manera que un nivel adicional debe añadirse a la parte inferior de la Tabla.D.III para los coeficientes DC. El símbolo-1 para los coeficientes DC entonces representa un valor de 1 a 11.

D.2.1.6.2. Codificación entrópica de longitud variable

Una vez que el dato del coeficiente cuantizado para un bloque de 8x8 es representado en la secuencia de símbolos intermedia descrita arriba, se asignan los códigos de longitud variable. Para cada bloque de 8x8, la representación del par símbolo-1 y símbolo-2 para el coeficiente DC es codificada y enviada en primer lugar.

Para ambos coeficientes DC y AC, cada símbolo-1 es codificado con un código de longitud variable (VLC, variable-length code) del conjunto de tablas de Huffman asignado a la componente del bloque de 8x8. Cada símbolo-2 es codificado con un código “entero de longitud variable” (VLI, variable-length integer) cuya longitud en bits esta dada en la Tabla.D.III. Los VLCs y los VLIs son codificados con longitudes variables, pero los VLIs no son códigos de Huffman. Una importante distinción es que la longitud de un VLC (código de Huffman) no es conocido hasta que es decodificado, pero la longitud de un VLI es almacenada en su precedente VLC. El código de Huffman (VLCs) debe ser especificado externamente como una entrada a los codificadores JPEG. (Nótese que la forma en la cual las tablas de Huffman son representadas en la cadena de datos es una especificación indirecta en la cual el decodificador debe construir las tablas mismas antes de la descompresión.) La propuesta JPEG incluye un conjunto de ejemplo de tablas de Huffman en sus anexos de información, pero dado que ellos son aplicaciones específicas, no especifica ninguno para el uso requerido. En contraste, los códigos VLI están “incorporados” en la propuesta. Esto es apropiado, dado que los códigos VLI son mucho más numerosos, pueden ser calculados más que almacenados, y no se ha mostrados que sean apreciablemente más eficientes cuando son implementados como códigos de Huffman.

D.2.1.6.3. Ejemplo de codificación por referencia

Esta sección da un ejemplo de compresión por referencia y codificación de un único bloque de muestras de 8x8. Nótese que aquí omitimos una buena parte de la operación de un codificador complete de JPEG por referencia, incluyendo la creación de el Formato de Intercambio de información (parámetros, encabezados, cuantización y tablas de Huffman), byte de relleno, relleno para el byte de los límites antes de la creación del código, y otras operaciones principales. Sin embargo, este ejemplo debería ayudar a concretar gran parte de la explicación anterior. La Fig.D.10(a) es un bloque 8x8 de 8-bit por muestra, extraídas arbitrariamente de una imagen real. Las pequeñas variaciones de muestra a muestra indican la predominancia de baja frecuencia espacial. Luego de sustraer 128 de cada muestra para el nivel de desplazamiento requerido, el bloque de 8x8 es una entrada para la TDC FDCT, la Ecuación (D.1). La Fig.D.10(b) muestra (para un lugar decimal) los coeficientes TDC resultantes. Excepto para unos pocos coeficientes de las frecuencias más bajas, las amplitudes son muy pequeñas.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(a) muestras de la imagen de la fuente

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

(b) coeficientes de la TDC

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(c) tabla de cuantización

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(d) coeficientes cuantizados normalizados

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(e) coeficientes cuantizados desnormalizados

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

(f) muestras de la imagen reconstruida

Figura D.10: TDC y ejemplos de cuantización.

La Fig.D.10(c) es el ejemplo de la tabla de cuantización para los componentes de luminancia (escala de grises) incluidos en el anexo de información del borrador del JPEG estándar parte 1 [2]. La cuantización se aplica sobre los coeficientes resultantes de la TDC. Luego de dicha cuantización se podan los decimales resultantes. La Fig.D.10(d) muestra los coeficientes TDC cuantizados, normalizados por sus entradas de la tabla de cuantización, como las especificadas por la Ecuación (B.8). En el decodificador estos números son “desformalizados” de acuerdo a la Ecuación (B.10), y la entrada a la TDC^{-1}, Ecuación (2.5). Finalmente, la Fig.D.10(f) muestra los valores de las muestras reconstruidas, marcadamente similares a los originales en la Fig.D.10(a). Desde luego, los números en la Fig.D.10(d) deben ser codificados de Huffman antes de la transmisión hacia el decodificador. El primer número del bloque a ser codificado es el término

DC, el cual debe ser diferencialmente codificado. Si el término DC cuantizado del bloque previo es, por ejemplo, 12, entonces la diferencia es +3. Entonces, la representación intermedia es (2)(3), para DIMENSION = 2 y AMPLITUD = 3.

A continuación, los coeficientes AC cuantizados son codificados. Seguimos el orden zig-zag, el primer coeficiente distinto de cero es -2, precedido por una corrida de cero de 1. Esto resulta en una representación intermedia de (1,2)(-2). Los siguientes encontrados en el orden zig-zag son los tres distintos de cero consecutivos de amplitud -1. Cada uno es precedido por una corrida de cero de longitud cero, para símbolos intermedios (0,1)(-1). El ultimo coeficiente distinto de cero es -1 precedido por dos ceros, para (2,1)(-1). Dado que este es el último coeficiente distinto de cero, el símbolo representando este bloque de 8x8 es EOB, o (0,0). Entonces, la secuencia intermedia de símbolos para este ejemplo del bloque 8x8 es:

(2)(3), (1,2)(-2), (0,1)(-1), (0,1)(-1), (0,1)(-1), (2,1)(-1), (0,0)

A continuación deben ser asignados los códigos. Para este ejemplo, los VLCs (códigos de Huffman) del anexo de información [477] que usaremos. El diferencial-DC VLC para este ejemplo es:

(2) 011

Los VLCs AC de luminancia para este ejemplo son:

(0,0) 1010
(0,1) 00
(1,2) 11011
(2,1) 11100

Los VLIs especificados en [487] son relativos a la representación complementaria de los dos. Ellos son:

(3) 11
(-2) 01
(-1) 0

Entonces, la cadena de bits para este bloque 8x8 de ejemplo es como sigue. Note que son requeridos 31 bits para representar 64 coeficientes, los cuales logran la compresión de algo menos de 0.5 bits/muestra:

0111111011010000000001110001010

D.2.1.6.4. Otros codificadores TDC secuenciales

La estructura del codec secuencial TDC 12-bit con codificación de Huffman es una simple estén-sión del método de codificación entrópica descrita previamente. Los coeficientes TDC cuanti-zados pueden ser 4 bits mas grandes, de manera tal que la información de DIMENSION y AMPLITUD se extiende en consecuencia. La TD secuencial con codificación Aritmética se describe en detalle en [477].

D.2.1.7. Modo TDC progresivo

El modo de operación TDC progresivo consiste de los mismos pasos de TDC y cuantización (de la Sección D.2.1.3) que son usados para el modo TDC secuencial. La diferencia principal reside en que cada componente de imagen es codificada en múltiples exploraciones más que en una

única exploración. La primera exploración(es) codifica una rústica pero reconocible versión de la imagen la cual puede ser transmitida rápidamente en comparación al tiempo de transmisión total, y es refinado por sucesivas exploraciones hasta alcanzar un nivel de calidad de imagen que fue establecida por las tablas de cuantización. Para lograr esto se requiere la adición de un buffer de memoria de una capacidad similar a la de la imagen a la salida del cuantizador, antes de la entrada al codificador entrópico. La memoria del buffer debe ser de tamaño suficiente para almacenar la imagen así como los coeficientes TDC cuantizados, cada uno de los cuales (si se almacena directamente) es 3 bits más grande que las muestras de la imagen fuente. Después de que cada bloque de coeficientes TDC es cuantizado, es almacenado en el buffer de memoria para coeficientes. Los coeficientes almacenados son entonces parcialmente codificados en cada exploración múltiple.

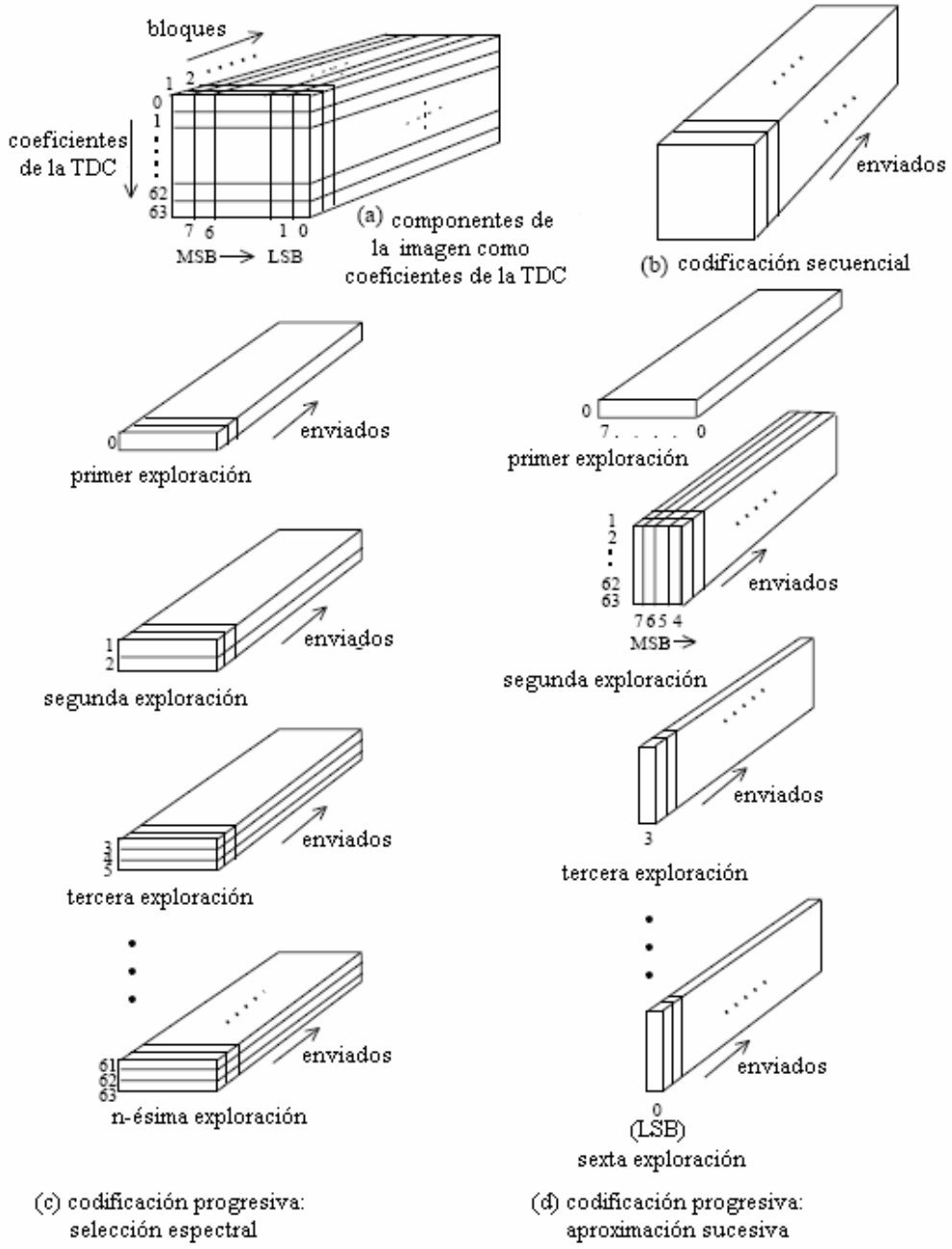


Figura D.11: Selección espectral y métodos de aproximaciones sucesivas de codificación progresiva.

Hay dos métodos complementarios mediante los cuales un bloque de coeficientes TDC cuantizados puede ser parcialmente codificado. En primer lugar, solo una “banda” de coeficientes especificados de la secuencia zig-zag necesita ser codificado en una exploración dada. Este procedimiento se llama “selección espectral,” dado que cada banda contiene típicamente coeficientes los cuales ocupan una parte más baja o más alta del espectro espacial de frecuencias para ese bloque 8x8. En segundo lugar, los coeficientes en la actual banda no necesitan ser codificados a su completa exactitud (cuantizados) en una exploración dada. Tras la primera codificación del coeficiente, los N bits más significativos pueden ser codificados primero, donde N es especificable. En subsecuentes exploraciones, los bits menos significativos pueden entonces ser codificados. Este procedimiento se llama ‘‘aproximación sucesiva.’’ Ambos procedimientos pueden ser usados separadamente, o intercalados en combinaciones flexibles. Alguna intuición para la selección espectral y la aproximación sucesiva pueden ser obtenidas de la Fig.D.11. La información de los coeficientes TDC cuantizados puede ser vista como un rectángulo para el cual los ejes son los coeficientes TDC (en orden zig-zag) y sus amplitudes. La selección espectral secciona la información en una dimensión y la aproximación sucesiva en la otra.

D.2.1.8. Modo jerárquico de operación

El modo jerárquico provee una codificación “piramidal” de una imagen en resoluciones múltiples, cada una difiriendo en resolución de su codificación adyacente por un factor de dos en la dimensión horizontal o vertical o ambas. El procedimiento de codificación se puede resumir así:

1. Filtrar y sub-muestrear la imagen original por el número deseado en múltiplos de 2 en cada dimensión.

2. Codificar esta imagen de dimensión reducida usando uno de los TDC secuenciales, TDC progresivos, o codificadores sin pérdidas como los descritos previamente.

3. Decodificar esta imagen de dimensión reducida y entonces interpolar y sobre-muestrearla por 2 horizontalmente y/o verticalmente, usando el filtro de interpolación idéntico el cual debe ser usado en el receptor.

4. Usar esta imagen sobre -muestreada como una predicción de la original en esta resolución, y codificar la imagen diferencia usando uno de los TDC secuencial, TDC progresivo, o codificadores sin pérdidas descritos previamente.

5. Repetir los pasos 3) y 4) hasta que la resolución completa de la imagen ha sido codificada.

La codificación en los pasos 2) y 4) se debe hacer usando solo procesos basados en TDC, solo los procesos sin pérdidas, o los procesos basados en TDC con un proceso final sin pérdidas para cada componente. La codificación jerárquica es útil en aplicaciones en las cuales se requiere que una imagen de muy alta resolución sea expuesta a un resolución más baja. Un ejemplo de esto es una imagen escaneada y comprimida en alta resolución para una impresora de muy alta calidad, donde la imagen debe ser también expuesta sobre la pantalla de una PC de baja resolución.

D.2.1.9. Otros aspectos de JPEG

Algunos aspectos principales del estándar propuesto solo pueden ser mencionados brevemente. Entre estos están los puntos concernientes a la representación codificada para los datos de la imagen comprimida especificada en adición a los procedimientos de codificación y decodificación.

Lo que es más importante, una sintaxis *formato de intercambio* es especificada la cual asegura que una imagen JPEG comprimida puede ser intercambiada satisfactoriamente entre diferentes ambientes de aplicaciones. El formato es estructurado en una forma consistente para todos los modos de operación. El formato de intercambio siempre incluye todas las tablas de cuantización codificación entrópica las cuales fueron usadas para comprimir la imagen.

La aplicaciones un estándar específico de una de aplicación) son los “usuarios” del estándar JPEG. El estándar JPEG no impone ninguna exigencia que, en un ambiente de aplicación, todo o aún cualquiera de las tablas debe ser codificada con los datos de la imagen comprimidos durante el almacenamiento o la transmisión. Esto deja a las aplicaciones la libertad para especificar la falta o no de tablas referenciadas si ellas son consideradas apropiadas. También les deja la responsabilidad para asegurar que los decodificadores JPEG acordes usados en su ambiente consiguen cargarse con tablas propias al propio tiempo, y que las tablas propias están incluidas en el formato de intercambio cuando una imagen comprimida es “exportada” fuera de la aplicación. Algunas de las aplicaciones importantes que están listas en el proceso de adoptar la compresión JPEG o han declarado su interés en hacerlo así son el lenguaje PostScript de Adobe para sistemas de impresión [478], la porción del contenido tramado del la arquitectura y formato de intercambio de documentos Office ISO [479], el futuro estándar de facsimil color CCITT, y el estándar de videotexto europeo ETSI [480-484].

D.2.2. Formato JPEG2000

D.2.2.1. Pre-procesamiento de los datos

JPEG2000 [485] es un joven estándar para compresión y transmisión de imágenes, el cual puede comprimir eficientemente tanto imágenes de tono continuo como indexadas (ver Sección D.2.2.5) en los modos con y sin pérdidas. La resolución y la escalabilidad de calidad, el bajo error de recuperación, y la codificación de regiones-de-interés son solo unas pocas de sus características. JPEG2000 emplea un número de mecanismos los cuales son considerados como el estado del arte en compresión de imágenes con y sin pérdidas. A diferencia de la arquitectura de JPEG (ver Fig.D.12), el diagrama del JPEG2000 está en el espíritu del descripto en la Sección D.2.1 y se muestra en la Fig.D.13. El primer paso del codificador JPEG2000 es la transformación del color. Usualmente las imágenes a color están almacenadas como una serie en la triada RGB. No obstante, el espacio de color RGB no es un color consistente, i.e. una leve variación en la componente de color de un pixel podría conducir a problemas en los bordes sensibles en las otras componentes de color (como blur o deterioro de los bordes). Existen otros espacios de color equivalentes a RGB en los cuales esto no es cierto. El paso inicial en la transformación del color es entonces realizado para convertir imágenes RGB en alguna otra representación más conveniente. La transformación del color no tiene gran impacto sobre la compresión sin pérdidas dado que el dato de la imagen es restaurado exactamente por el decodificador. Por el contrario, es un paso crucial en el modo con pérdidas. La performance de compresión puede beneficiarse también de la independencia de componentes del espacio de color transformado, dado que permite al usuario dejar el canal de luminancia sin cambios, y sub-muestrar las componentes de color sin ninguna pérdida perceptual (o apenas sensible) en la imagen reconstruida.

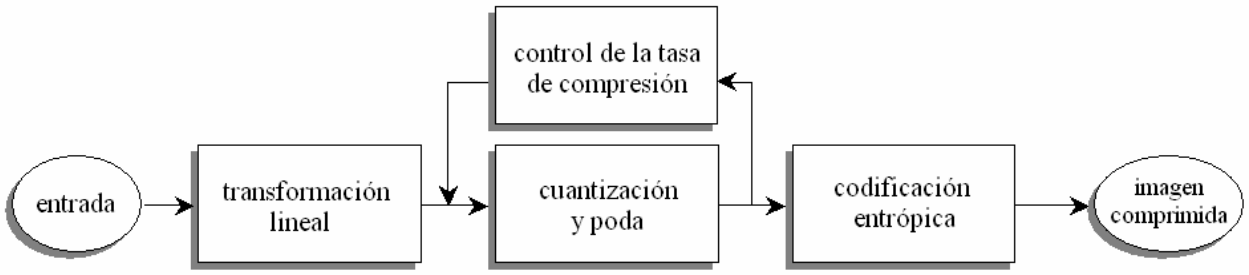


Figura D.12: Esquema común de compresión de imágenes (en el espíritu de JPEG).

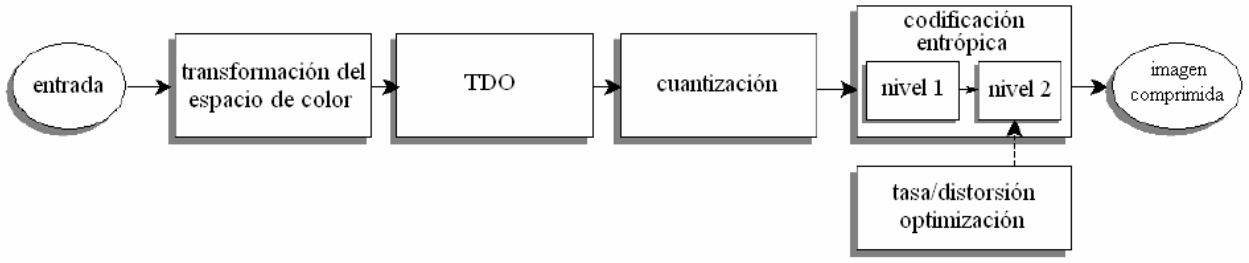


Figura D.13: Esquema de compresión de imágenes JPEG2000.

Los planos de color son codificados separadamente como si ellos fueran imágenes en niveles de grises. Ellos están usualmente divididos en bloques de de igual tamaño llamados mosaicos. Para cada mosaico se genera una cadena de código disjunta usando el mismo esquema. Esto se hace principalmente para mantener baja la carga computacional, y para accesos relativamente aleatorios en el dominio comprimido [486-506].

La transformada de onditas se aplica a los mosaicos antes de la codificación entrópica. El beneficio de emplear la transformada de onditas es que los datos transformados exhiben usualmente una entropía más baja por lo que son más “comprimibles”. En particular, dado que la transformada de onditas obtiene cuatro sub-bandas a partir de un mosaico, el modelado de la fuente puede ser mosaiqueado para cada sub-banda. De hecho, dado que los filtros de onditas están diseñados para almacenar frecuencias diferentes en cada sub-banda (ver Fig.D.14), las sub-bandas exhiben características peculiares las cuales son “capturadas” por el modelador fuente JPEG2000. La Fig.D.15 muestra los efectos del filtrado en frecuencia sobre una imagen en niveles de grises. Un número de filtros de onditas diferente es permitido por el estándar para ambos tipos de compresión con y sin pérdidas. Los filtros con pérdidas dan usualmente mejores resultados pero ellos involucran operaciones de punto flotante. Debido a la aproximación de punto flotante, la correcta reconstrucción de las señales de entrada no esta garantizada usando estos filtros. Estos filtros implican solo operaciones enteras que además son permitidas para superar este problema.

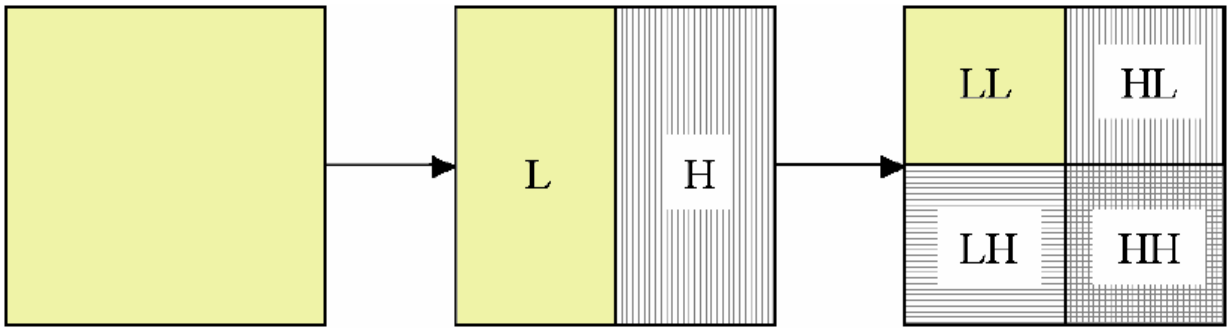


Figura D.14: Transformada de onditas: División en sub-bandas y filtrado en frecuencia.

Figura D.15: Transformada de onditas: efecto del filtrado en frecuencia sobre una imagen en escala de grises.

La transformada de onditas es seguida por un paso de cuantización en la compresión con pérdidas. Ver Apéndice B.

D.2.2.2. EBCOT – Nivel 1

El núcleo de JPEG2000 es su motor de codificación, el Embedded Block Coding with Optimised Truncation (EBCOT, codificación por bloque embebido con truncamiento optimizado) [504, 505]. El algoritmo EBCOT esta conceptualmente dividido en dos capas llamadas niveles. El nivel 1 es responsable por el modelizado de la fuente y la codificación entrópica, mientras que el nivel 2 genera la cadena de salida. La Fig.D.16 muestra una representación esquemática del nivel 1 del EBCOT. Las sub-bandas de onditas son particionadas en pequeños bloques de código (para no

confundirlos con los mosaicos) los cuales a su tiempo son codificados en planos de bits, i.e. bits del coeficiente del mismo orden son codificados juntos (ver Fig.D.17). Los bits más significativos se codifican primero, entonces los bits de orden inferior son codificados en orden descendiente. Cada plano de bits es codificado separadamente como si fuera una imagen de solo dos niveles, excepto por el hecho de que la formación de contexto es guiada no solo por los bits previamente codificados en el mismo plano de bits, sino además porque ha sido visto en el plano de bits de orden más alto. Los planos de bits son además particionados en tres pasos de codificación. Cada paso de codificación constituye una unidad atómica de código, llamada trozo. Los trozos de palabra-código son agrupados en capas de calidad y pueden ser transmitidas en cualquier orden, a condición de que aquellos trozos que pertenecen al mismo bloque de código sean transmitidos en su orden relativo.

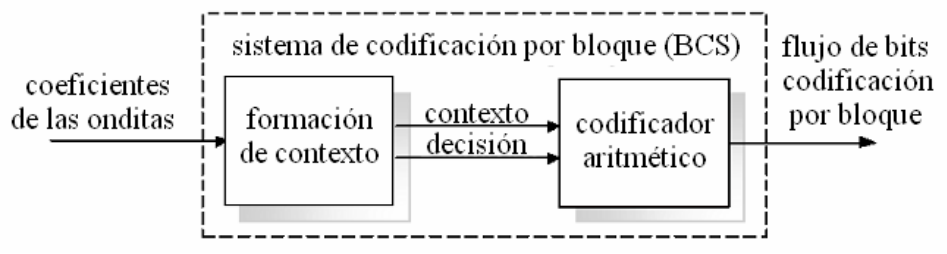


Figura D.16: Representación esquemática del nivel 1 del EBCOT.

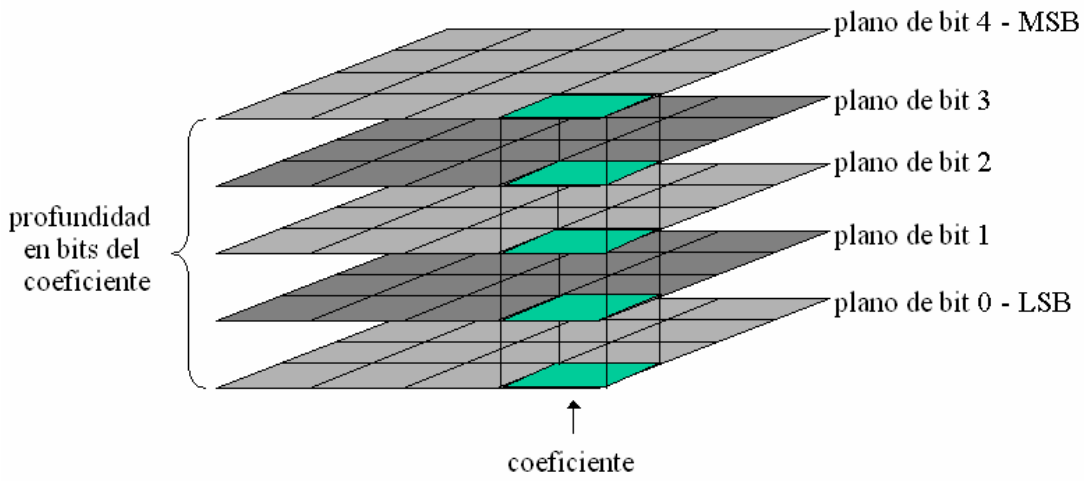


Figura D.17: Codificación por plano de bits: Un ejemplo de codificación por plano de bits de coeficientes onditas con un nivel de profundidad de bits de 5.

Los trozos constituyen puntos de truncamiento válidos, i.e. la palabra-código puede ser truncada al final del trozo sin comprometer la correcta reconstrucción de los datos comprimidos (el particionado es consistente con la sintaxis de la palabra-código). Ambos, el codificador y el decodificador pueden truncar la cadena de bits en correspondencia de un punto de truncado válido para recuperar los datos originales hasta una calidad objetivo. La Fig.D.18 muestra un ejemplo de cadena de código particionada en trozos y capas. La contribución de bloques de código para capas puede ser extremadamente variable y algunos bloques de código no deberían contribuir en todo a algunas capas. Este comportamiento depende de la información contenida en cada trozo, mientras que la contribución de los bloques de código de las capas es requerida para reflejar que tan “útil” es esta información para alcanzar la calidad objetivo dada. Este concepto quedará más claro posteriormente en esta sección.

La cadena de salida es acondicionada por un número de paquetes conteniendo trozos de los bloques de código en una sub-banda dada y pertenencia a la misma capa (i.e. una fila en la Fig.D.18). Dado que la información del encabezado del paquete es altamente redundante, el encabezado mismo emplea técnicas de compresión conocidas para reducir el sobre-encabezado del paquete (ver Sección D.2.2.3).

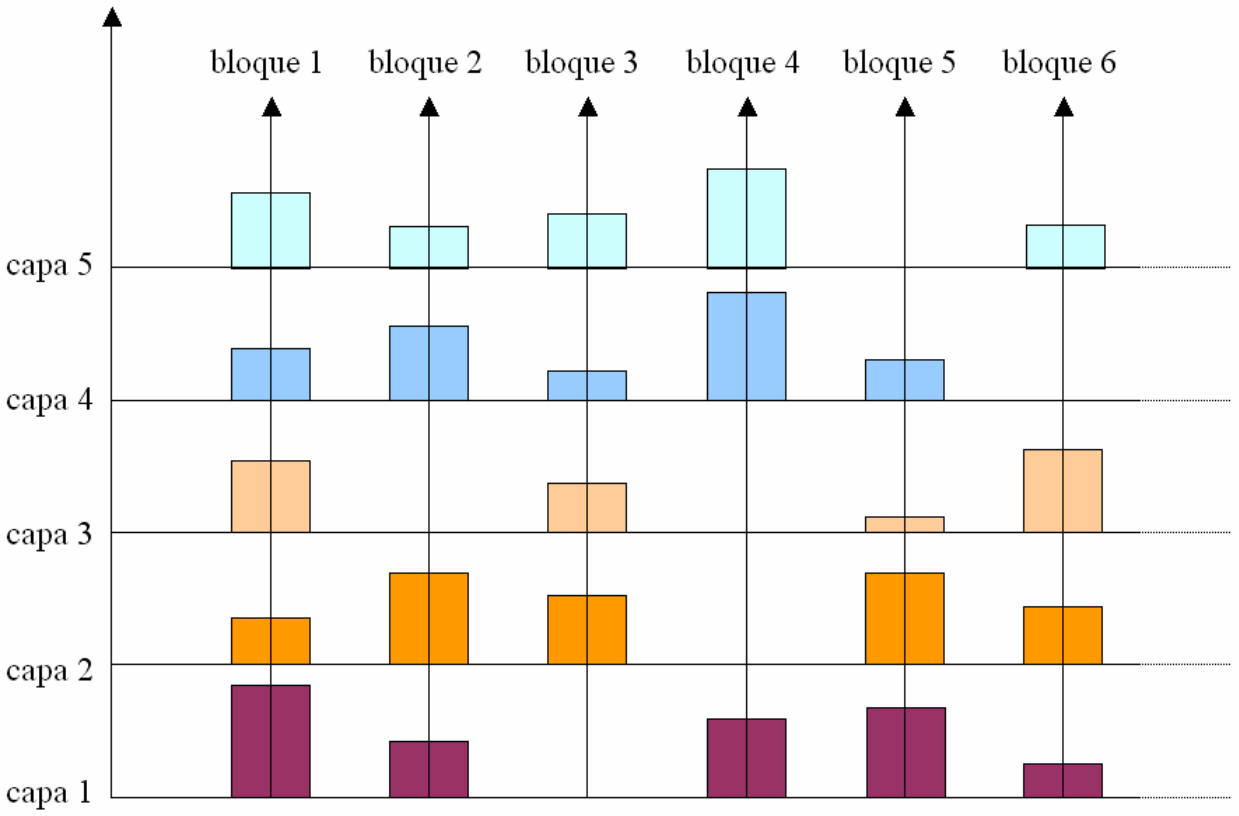


Figura D.18: Partición de la cadena de código: se muestran cinco capas y seis bloques de código. Cada rectángulo representa los trozos de un bloque de código el cual ha sido incluido en la capa relativa. Los colores codifican las capas. Los datos codificados para un bloque se representan como un trozo falso sobre la correspondiente flecha vertical, mientras la cadena de bits comprimida es transmitida de una forma de a capa (es decir, primero los niveles más bajos).

D.2.2.2.1. Modelización de contexto

La modelación del contexto en JPEG2000 es un poco difícil por su complejidad conceptual y es compensada por su impresionante performance de compresión. La modelización de contexto más ampliamente usada es la JPEG-LS [480, 481]. El bit actual (la decisión en la Fig.D.16) es codificado entrópico usando un codificador aritmético cuyos estimados de probabilidad son actualizados usando un autómata con 46 estados. La probabilidad estimada se actualiza separadamente para cada una de las 18 diferentes clases usando contextos separados (ver Fig.D.16). Para entender como trabaja la modelización de contexto de JPEG2000 deben quedar claro tres conceptos.

Paso de Codificación – Una exploración completa del plano de bits actual agrupando juntos bits con similares propiedades estadísticas (ver Fig.D.19). Generalmente un paso de codificación no compromete a todos los bits en el plano de bits actual.

Primitiva de Codificación – Un pequeño conjunto de operaciones implica la actualización del estado de los coeficientes siendo codificados y la selección del contexto más apropiado para codificar el bit actual, dado la información estadística disponible sobre los pixeles vecinos.

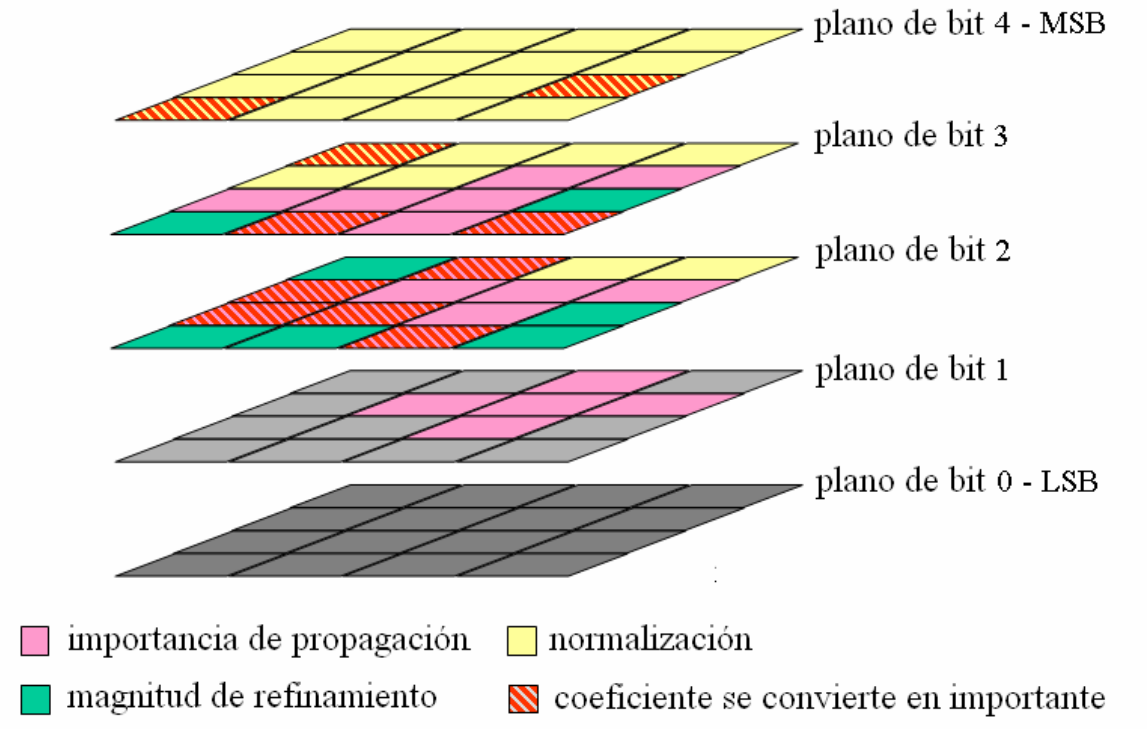


Figura D.19: Paso de codificación: Un ejemplo de codificación con 5 bits de profundidad.

Contexto – Se refiere a toda la información necesaria para mantener la pista del estado actual del autómata finito el cual guía la actualización estadística de cada clase de código. Los contextos están determinados por las primitivas de codificación, las estadísticas de los vecinos, y las propiedades de frecuencia de las sub-bandas.

Son usados tres diferentes pasos de codificado (ver la Fig.D.19): la importancia de la propagación, la magnitud del refinamiento, y la normalización (limpieza). Cada paso es requerido para estimar las probabilidades de los símbolos de una de las tres clases diferentes de bits. Ellos son ejecutados en el orden en el cual son presentados aquí. Cada bit es codificado en uno y solo uno de tres pasos. Las probabilidades de los símbolos son estimadas teniendo en cuenta que ha sido codificado en los planos de bits previos y en los bits vecinos. La vecindad se define como los ocho bits inmediatamente adyacentes al bit actual en el plano de bits actual. Se dice que un coeficiente es significativo si hay al menos un bit no nulo entre sus correspondientes bits codificados en los planos de bits previos. Un coeficiente que no es significativo se llama no-significativo.

El refinamiento de magnitud es usado para refinar la magnitud del coeficiente correspondiente al bit siendo codificado en uno o más bits (el actual). Un bit es codificado en este paso si pertenece a un coeficiente que ya es significativo. La importancia de la propagación es usada para propagar la información importante. Cuando un coeficiente se convierte en significativo, la probabilidad de que sus coeficientes vecinos se conviertan en significativos se hace más y más alta como los beneficios de la codificación. Esto es porque esos bits son codificados usando diferentes estimados de probabilidad. De ahí, la importancia de propagación junta a todos los bits que pertenecen a

los coeficientes no-significativos teniendo una vecindad preferida (i.e. al menos un coeficiente en la vecindad del coeficiente actual es significativo). Finalmente, todos los bits pertenecen a coeficientes no-significativos no teniendo coeficiente significativo en su vecindad que sea codificado en el paso de normalización. Nuevamente, esto se hace porque son usados diferentes estimados de probabilidad dado que es altamente probable que esos coeficientes pertenezcan a áreas homogéneas de imagen. Este procedimiento de particionado de planos de bits es representado en Fig.D.20. Los pasos de codificación, más que proveer una modelización exacta de fuente, da un particionamiento fino de planos de bits en tres conjuntos. En realidad su propósito es principalmente permitir más puntos de truncamiento válidos en la cadena de código comprimido (ver Sección D.2.2.3).

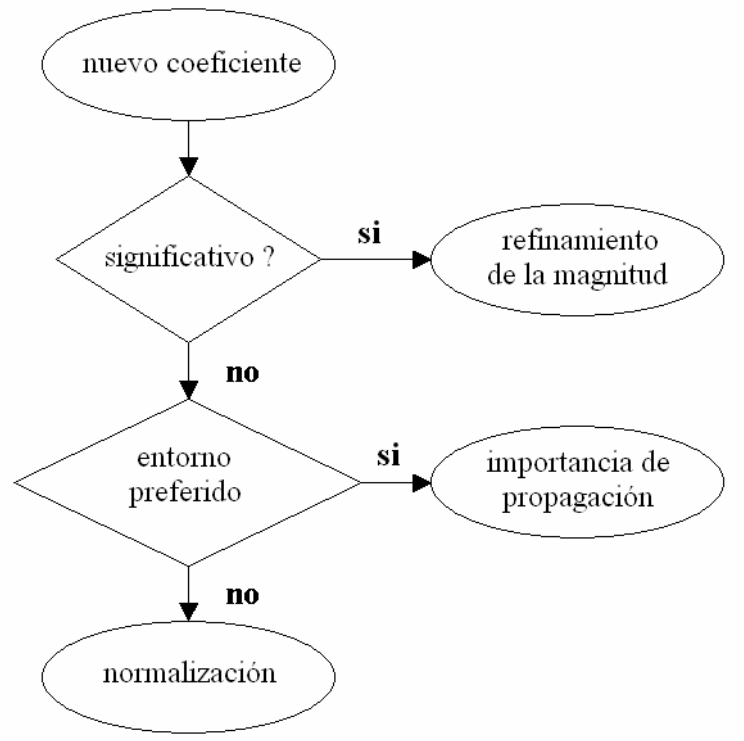


Figura D.20: Diagrama de flujo de la partición de plano de bits.

Las primitivas de codificación son empleadas para obtener una modelización más fina. Ellas son responsables de encontrar el contexto más apropiado para codificar al bit actual. Cuatro primitivas son implementadas: Codificación Cero (CC), Codificación por Longitud de Corrida (CLC), Codificación por Signo (CS), y Refinamiento de Magnitud (RM) (para no confundir con el paso de refinamiento de magnitud). La primitiva CC es ejecutada sobre coeficientes no-significativos. Se puede elegir entre nueve contextos dependiendo de la vecindad de los coeficientes (los vecinos verticales, horizontales, y diagonales están diferenciados) y la sub-banda a la cual pertenece el bit que está siendo codificado. CLC es usada en lugar de CC cuando se codifican grandes áreas homogéneas. Si son encontrados (exactamente) cuatro coeficientes consecutivos no-significativos sin vecindad preferida, solo un bit es codificado para la secuencia entera de cuatro bits para indicar si uno de esos cuatro coeficientes se convertirá en significativo en el actual plano de bits. Entonces un índice adicional de dos bits que indica el primero de ellos es transmitido. Esta primitiva es usada para reducir el número de bits a ser codificados por el codificador entrópico cuando secuencias largas de coeficientes no-significativos son encontradas.

Un contexto particular es reservado para la primitiva CLC. La CS es invocada al menos una vez por coeficiente tan pronto se convierte en significativo, al codificar su signo. La primitiva RM es usada para refinar la magnitud de un coeficiente significativo, para la cual el signo ya ha sido codificado en un plano de bits previo. Se elige el contexto correcto teniendo en cuenta la vecindad del coeficiente. Además, son elegidos diferentes contextos si el RM está siendo ejecutado por primera vez o no sobre el coeficiente actual. Cada paso de codificación emplea estas primitivas para codificar los bits en el plano de bits actual. El paso de propagación de importancia ejecuta un CC por cada bit siendo codificado y un CS por cada bit el cual se convierte en significativo en el plano de bits actual. El paso de normalización tiene el mismo comportamiento excepto para la oportunidad de aplicar CLC en lugar de CC donde sea posible. Finalmente, el paso de refinamiento de magnitud puede llamar solo a la primitiva RM. El símbolo a ser codificado en el contexto relativo es pasado al codificador aritmético, el cual "carga" el estado relativo actual para el contexto dado y lo usa para codificar el símbolo dado. Entonces el estado es actualizado para refinar los estimados de probabilidad para el contexto actual.

D.2.2.2.2. Codificación Aritmética: Codificador MQ

El motor de codificación Aritmética (ver Apéndice B) en JPEG2000 (codificador-MQ) fue desarrollado por el estándar JBIG (Joint Bi-level Image Experts Group, Comité de Grupos de Expertos en Imágenes de dos niveles) para la compresión de imágenes de dos niveles [485]. Es ligeramente diferente de los codificadores aritméticos estándar. Por lo tanto, necesita una corta explicación. El codificador MQ es un codificador entrópico el cual, más que codificar los valores binarios de los símbolos de entrada (el bit actual), codifica una información binaria la cual indica si el símbolo que es codificado es el que esperamos (el símbolo más probable, o MPS) o no (el símbolo menos probable, o LPS). En particular, dos características peculiares del codificador MQ serán discutidas brevemente: la renormalización, y el cambio condicional.

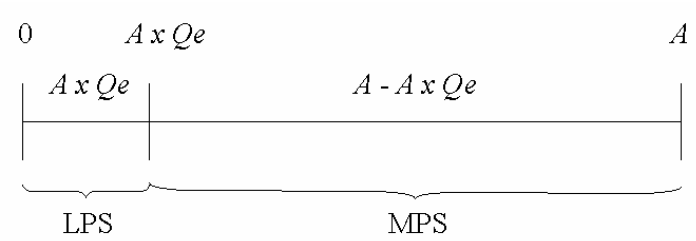


Figura D.21: Intervalo de subdivisión en el codificador MQ.

Recalcamos que en cada paso de la codificación aritmética de una secuencia la sub-secuencia actual es representada como un intervalo en [0, 1]. La Fig.D.21 muestra como se particiona el intervalo actual en un codificador MQ. Siguiendo la notación de [2], ***C*** y ***A*** se refieren a la base y la longitud del intervalo actual, mientras que ***Qe*** es la probabilidad estimada del LPS. Las ecuaciones clásicas de actualización pueden entonces ser re-escritas como siguen.

Codificación MPS:

$C = C + A \times Qe$ (D.3)

$A = A - A \times Qe$ (D.4)

Codificación LPS:

C : *sin cambio* (D.5)

$A = A \times Qe$ (D.6)

Para prescindir de cálculos complejos, se usa un truco para simplificar las ecuaciones. La longitude del intervalo actual se limita a mentir en $\Re = [0.75, 1.5)$. Cuando $\boldsymbol{A}$ cae al mínimo se deja hasta pasado un $\boldsymbol{A} \in \Re$ (renormalización). Con este simple ajuste $\boldsymbol{A} \approx 1$ es siempre verdadero. Las ecuaciones de arriba pueden entonces ser aproximadas estableciendo $\boldsymbol{A} = 1$ en las multiplicaciones sobre el lado derecho de las Ecuaciones (D.3) (D.4) y (D.6).

Codificación MPS:

$C = C + Qe$ (D.7)

$A = A - Qe$ (D.8)

Codificación LPS:

C : *sin cambio* (D.9)

$A = Qe$ (D.10)

El cálculo de las Ecuaciones (D.7), (D.8), y (D.10) es mucho más eficiente que el cálculo de las ecuaciones relativas no-aproximadas dado que no involucran multiplicaciones, las cuales pueden ser costosas incluso en las implementaciones de hardware. Además, puede ser usada también aritmética de punto fijo para llevar a cabo los cálculos. Esta implementación no da lugar a ninguna una pérdida de eficiencia de codificación, dado que no reduce el espacio de codificación (i.e. la longitud de los dos sub-intervalos generados es todavía la longitud del intervalo actual antes de la división sub-intervalo). No obstante, cuando ***Qe*** es cercano a 0.5 y ***A*** es pequeño, puede suceder que el sub-intervalo MPS sea más pequeño que el sub-intervalo LPS. Como un ejemplo, considere la situación: ***A*** = 0.8, ***Qe*** = 0.43. De las Ecuaciones D.8 y D.10 se desprende que $\boldsymbol{A_{MPS}} = 0.37$ y $\boldsymbol{A_{LPS}} = 0.43$, i.e. $\boldsymbol{A_{MPS}} < \boldsymbol{A_{LPS}}$. A los efectos de esquivar tal inversión, es esos casos el intervalo asignado es invertido como se muestra en la Fig.D.22 (cambio condicional). Los algoritmos para codificar MPS y LPS son expresados aquí.

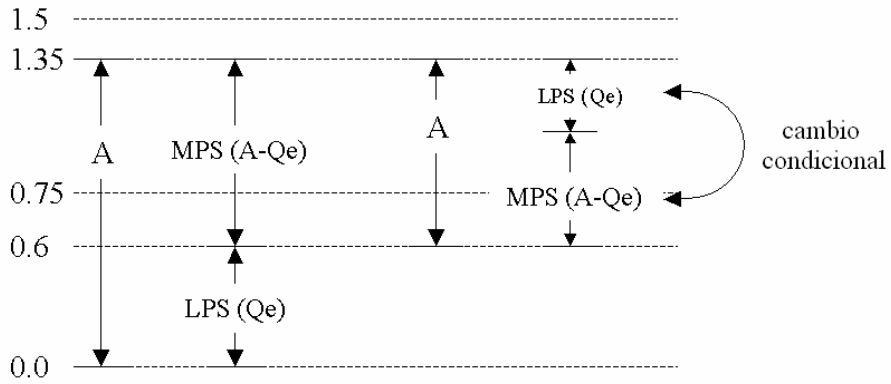


Figura D.22: Cambio condicional.

El proceso de estimación de la probabilidad se basa en un ordenamiento del contador de los límites aproximados de los MPSs por las renormalizaciones del registro ***A***. Luego de cada renormalización el contador se pone a cero y se obtiene un nuevo estimado de ***Qe*** del autómata finito. Si el valor de ***Qe*** es también alto MPS la renormalización es más probable. Esto a su vez, causa que ***Qe*** se fije a un valor más pequeño. Por otro lado, si ***Qe*** es muy pequeño la probabilidad de renormalización de LPS crece y como una consecuencia de ello ***Qe*** tiende aincrementarse. Si la identidad de MPS es errónea ***Qe*** se incrementará hasta aproximadamente 0.5 causando una conmutación de la identidad de MPS.

Algoritmo D.1 Codificación MPS

A = A-Qe	/* Calcular el sub-intervalo MPS */
si A < min	/* Si la renormalización es necesaria */
si A < Qe	/* Si la dimensión del intervalo esta invertida */
A = Qe	/* Poner el intervalo a sub-intervalos LPS */
sino	/* De otra forma */
C = C+Qe	/* Punto a base del intervalo MPS */
fin si	
renormalizar A y C	/* renormalizar */
sino	/* si no renormalización es necesaria **/**
C = C+Qe	/* Punto a base del intervalo MPS */
fin si	

Algoritmo D.2 Codificación LPS

A = A-Qe	/* Calcular el sub-intervalo MPS */
si A < Qe	/* Si la dimensión del intervalo esta invertida */
C = C+Qe	/* Punto a base del intervalo MPS */
sino	/* De otra forma */
A = Qe	/* Poner el intervalo a sub-intervalos LPS */
fin si	
renormalizar A y C	/* renormalización siempre que sea necesaria */

El pseudo-código para la renormalización está dado abajo. Aquí, ***CX*** es el contexto actual como determinado por el modelador de fuente, y ***I*** es el índice del estado actual para ***CX***. NMPS y NLPS indican respectivamente el próximo estado si un MPS o n LPS es encontrado.

Algoritmo D.3 Renormalización MPS

I = indice(CX)	/* Indice actual para el contexto CX */
I = NMPS(I)	/* Nuevo índice para el contexto CX */
Indice(CX) = I	/* Salvar este índice en el contexto CX */
Qe(CX) = valor_Qe(I)	/* Nueva probabilidad estimada para CX */

Algoritmo D.4 Renormalización LPS

I = indice(CX)	/* Indice actual para el contexto CX */
si selección(I) = I	/* Si ocurre un cambio condicional */
MPS(CX) = 1-MPS(CX)	/* Cambio en el sentido MPS */
fin si	/* (1 -> 0 o 0 -> 1) */
I = NLPS(I)	/* Nuevo índice para el contexto CX */
Indice(CX) = I	/* Salvar este índice en el contexto CX */
Qe(CX) = valor_Qe(I)	/* Nueva probabilidad estimada para CX */

D.2.2.3. EBCOT – Nivel 2

La cadena de bits de JPEG2000 es escalable con respecto a la resolución (gracias a la descomposición de onditas) y la calidad, y provee un acceso aleatorio razonable para bloques de imágenes en el dominio comprimido. La escalabilidad de la calidad se obtiene por medio del plano de bits y de la codificación del paso de particionado. Todos los bits codificados para el paso de codificación constituyen una unidad atómica y son transmitidos como bloque único. No obstante, los trozos comprimidos necesitan ser transmitidos solo hasta un número mínimo para alcanzar una

calidad definida por el usuario. Es una especie de cuantización desigual post-compresión la cual usualmente da mejores resultados que los métodos de cuantización estándar, en función de la calidad de la imagen reconstruida. Con el fin de que el usuario pueda interrumpir la cadena hasta la resolución y/o calidad deseadas, o visualizar un área particular de la imagen, la estructura de cadena de bits debe ser cuidadosamente organizada. Esta tarea es llevada a cabo por el nivel 2 del algoritmo EBCOT. La cadena es seccionada en paquetes conteniendo la información del encabezado en adición a la cadena misma. En particular, los encabezados de los paquetes contienen información útil para recuperar los trozos de la palabra de código en la cadena de salida, e.g. si cualquier bloque de una determinada sub-banda de ondita es incluida en el paquete o no, la longitud de la palabra código, y así sucesivamente.

D.2.2.3.1. Puntos de truncado óptimo

La salida del nivel 1 es una serie de palabras código una por cada bloque de código) con una indicación sobre los puntos de truncado válidos. Esta información complementaria es usada para encontrar los puntos de truncado óptimos en un sentido tasa/distorsión, dando una tasa de bits objetivo. El conjunto de puntos de truncado los cuales dan juntos el mejor compromiso entre longitud de palabra código y la cantidad correspondiente de distorsión con respecto a la imagen original (i.e. calidad), es seleccionada para cada bloque de código incluido en el paquete. Encontrar los puntos de truncado óptimo implica la minimización de la relación tasa/distorsión. Indiquemos con $\boldsymbol{R_i^j}$, $\boldsymbol{j} = 1,2,\ldots$ la secuencia válida de puntos de truncado de la palabra código correspondientes al ***i***-ésimo bloque de código, y con $\boldsymbol{D_i^j}$, $\boldsymbol{j} = 1,2,\ldots$ las distorsiones asociadas. Si indicamos con $\boldsymbol{n_i}$ el punto de truncado óptimo para la ***i***-ésima palabra código, la longitud total de la cadena de bits puede ser escrita como $\boldsymbol{R} = \sum_i \boldsymbol{R_i^{n_i}}$ y $\boldsymbol{D} = \sum_i \boldsymbol{D_i^{n_i}}$ es la distorsión global. Los puntos de truncado óptimo $\boldsymbol{n_i}$ son encontrados en correspondencia de las longitudes de las palabras código minimizando la distorsión global. Dada una tasa de bits objetivo $\boldsymbol{R_{max}}$, queremos minimizar la distorsión global $\boldsymbol{D}$ sujeta a las restricciones $\boldsymbol{R} = \boldsymbol{R_{max}}$. Un método bien conocido para hacer el trabajo es el de los multiplicadores de Lagrange para problemas de optimización con restricciones. En nuestra configuración tenemos que encontrar un valor λ el cual minimize la función $\boldsymbol{D} + \lambda \boldsymbol{R}$. De todos modos, dado que existen un gran número finito de puntos de truncado no podemos garantizar que $\boldsymbol{R} = \boldsymbol{R_{max}}$. Esta condición es entonces relajada y se convierte en $\boldsymbol{R} \leq \boldsymbol{R_{max}}$. Por lo tanto, buscamos la longitud del código $\boldsymbol{R}$ más cercano a $\boldsymbol{R_{max}}$ el cual minimiza nuestra función costo. Se puede profundizar acerca de este tema en [502].

D.2.2.3.2. Encabezado del paquete

Dada una tasa de bits objetivo, los puntos de truncado óptimo definen el número de bits que aparecen en una capa dada para cada bloque de código. Dependiendo de su contenido de información, un código bloque dado puede estar completamente incluido en una capa o, en el otro lado, puede no aparecer en absoluto en la capa. Por lo tanto, el encabezado del paquete debería codificar la inclusión de información. Otra información importante contenida en el encabezado del paquete es la longitud de la palabra código para cada bloque de código incluido y el número de los planos de bits nulos más significativos, el cual es útil para eludir el gasto de tiempo y bits para codificar secuencias largas de ceros. Todo en estas tres piezas de información muestra alta redundancia, en ambos casos, dentro de la misma capa y entre diferentes capas (ver [486, 397]). Una estructura de árbol particular, el árbol etiquetado, es adoptada para exhibir esta redundancia para obtener una representación más compacta de los encabezados de los paquetes. A continuación, serán introducidos los árboles etiquetados y se mostrará como son empleados eficiente-

mente para codificar información del encabezado de los paquetes.

D.2.2.3.2.1. Arboles etiquetados

Un árbol etiquetado es una estructura particular de árbol concebida para la codificación eficiente de matrices bi-dimensionales altamente redundantes. La información a ser codificada esta asociada a las hojas. Su estructura se asemeja a un árbol cuaternario. No obstante, ellos tienen dos diferencias principales con respecto a los árboles cuaternarios convencionales: los árboles etiquetados no necesitan estar valuados en forma binaria, y la información en un árbol etiquetado puede ser codificada en cualquier orden, a condición de que el decodificador siga el mismo orden. La Fig.D.23 muestra un ejemplo de un árbol etiquetado.

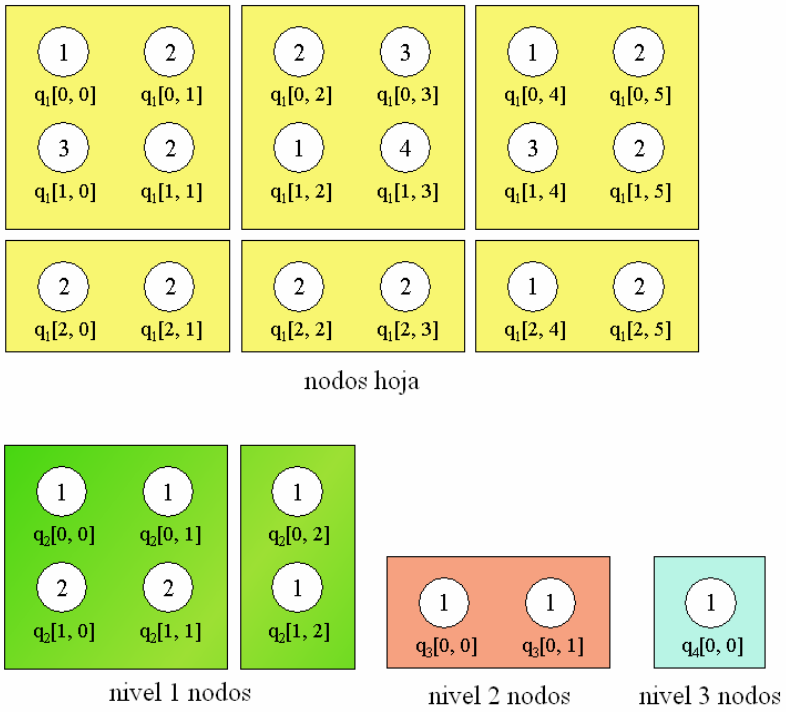


Figura D.23: Un ejemplo de árbol etiquetado.

$q_1[m, n]$ denota un arreglo de cantidades (enteros no-negativos) que nos gustaría representar a través de árboles etiquetados. Asociamos estas cantidades con los nodos hoja del árbol. Los nodos de los próximos niveles son rotulados como $q_2[m, n]$, $q_3[m, n]$, y así sucesivamente. Cada nodo interno está asociado con un bloque de 2×2 de nodos en el nivel más bajo, y contiene el mínimo entre los valores de sus cuatro hijos. La raíz contiene el mínimo de toda la estructura. El árbol etiquetado codifica eficientemente la información $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}] \geq t$, donde $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}]$ es un valor en la matriz de entrada y t e el valor de un umbral. El algoritmo para construer el árbol etiquetado es usualmente expresado como un procedimiento $\boldsymbol{T}(m, n, t)$ el cual codifica la cadena de bits de longitud mínima para indicar si $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}] \geq t'$, $\forall\, t'$: $1 \leq t' \leq t$, dada la información la cual ha sido codificada en llamadas previas. A saber, el algoritmo evita codificar de nuevo la información codificada en previas llamadas para los nodos intermedios. Esta información se mantiene por medio de una variable de estado $t_k[m, n]$, la cual asegura que suficiente información ha sido codificada para saber si $q_1[\overline{\boldsymbol{m}}, \overline{\boldsymbol{n}}] \geq t'$, $t' = 1, 2 \ldots t_k[m, n]$. Es inicializado a cero, dado que la información no ha sido codificada en el comienzo.

Algoritmo D.5 Procedimiento de codificación del árbol etiquetado ($T[m, n, t]$).

(1) $k = K$	/* comenzar en el nodo raíz */
(2) $t_{mín} = 0$	/* se usa para propagar el conocimiento a los descendientes */
(3) $\boldsymbol{m_k} = \left\lfloor \frac{\overline{m}}{2^{k-1}} \right\rfloor, \quad \boldsymbol{n_k} = \left\lfloor \frac{\overline{n}}{2^{k-1}} \right\rfloor$	/* $[\boldsymbol{m_k}, \boldsymbol{n_k}]$ es la ubicación del nodo del ancestro hoja en el nivel k */
(4) si $\boldsymbol{t_k}[\boldsymbol{m},\boldsymbol{n}] < \boldsymbol{t}_{\min}$ hacer $\boldsymbol{t_k}[\boldsymbol{m},\boldsymbol{n}] = \boldsymbol{t}_{\min}$	/* actualizar la variable de estado */
(5) si $\boldsymbol{t} \le \boldsymbol{t_k}[\boldsymbol{m},\boldsymbol{n}]$	
(5.1) si $k = 1$ se realizan (5.2) de otra manera - poner $\boldsymbol{t}_{\min} = \min\{\boldsymbol{t_k}[\boldsymbol{m_k},\boldsymbol{n_k}], \boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]\}$ - decrecer k e ir al paso (3)	/* Hemos alcanzado la hoja. Ha sido codificada suficiente información */
(6) de otra forma (i.e. si $\boldsymbol{t} > \boldsymbol{t_k}[\boldsymbol{m},\boldsymbol{n}]$) - si $\boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}] > \boldsymbol{t_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ emite un bit 0 - sino $(\boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}] = \boldsymbol{t_k}[\boldsymbol{m_k},\boldsymbol{n_k}])$ emite un bit 1 - incrementar $\boldsymbol{t_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ e ir al paso (5)	/* enviar suficiente información para incrementar $\boldsymbol{t_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ */

Algoritmo D.6 Procedimiento de codificación del árbol etiquetado.

(1) $k = K$	/* comenzar en el nodo raíz */
(2) $t_{mín} = 0$	/* se usa para propagar el conocimiento a los descendientes */
(3) $\boldsymbol{m_k} = \left\lfloor \frac{\overline{m}}{2^{k-1}} \right\rfloor, \quad \boldsymbol{n_k} = \left\lfloor \frac{\overline{n}}{2^{k-1}} \right\rfloor$	/* $[\boldsymbol{m_k}, \boldsymbol{n_k}]$ es la ubicación del nodo del ancestro hoja en el nivel k */
(4) si $\boldsymbol{q_k}[\boldsymbol{m},\boldsymbol{n}] < \boldsymbol{t}_{\min}$ hacer $\boldsymbol{q_k}[\boldsymbol{m},\boldsymbol{n}] = \boldsymbol{t}_{\min}$	/* actualizar la variable de estado */
(5) si $\boldsymbol{s_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ es verdadero	/* suficiente información ha sido codificada para el nivel actual */
- poner $\boldsymbol{t}_{\min} = \boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$	/* actualizar $\boldsymbol{t}_{\min}$ para propagar el conocimiento */
- decrecer k e ir al paso (3)	/* y proceder al próximo nivel */
(6) de otra forma ($\boldsymbol{s_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ es falso)	
- si $\boldsymbol{t} \le \boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$	/* hemos alcanzado la hoja. Suficiente información ha sido codificada */
- si $k = 1$ se realizan	
- de otra forma	/* actualizar $\boldsymbol{t}_{\min}$ para propagar el conocimiento sin enviar nuevos bits */
- poner $\boldsymbol{t}_{\min} = \boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$	
- decrecer k e ir al paso (3)	/* leer suficiente información para actualizar el valor de $\boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ */
- de otra forma (i.e. si $\boldsymbol{t} > \boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$)	/* Un 0 significa que puede */
- sea b el próximo símbolo de entrada	/* actualizar nuestro conocimiento de $\boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ */
- si $b = 0$	/* 1 significa que no es necesaria más información para codificar este nodo a partir de ahora */
- incrementar $\boldsymbol{q_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$	
- sino poner $\boldsymbol{s_k}[\boldsymbol{m_k},\boldsymbol{n_k}]$ a verdadero	
- ir al paso (5)	

El algoritmo de codificación se muestra en el Algoritmo D.5. Como un ejemplo, considerar el árbol etiquetado en la Fig.D.23 y suponer que queremos codificar suficiente información para un umbral $t = 1$, usando el Algoritmo D.5.

Comenzamos con $q_1[0, 0]$ y procedemos en un orden por barrido (i.e. de izquierda a derecha, y luego la fila de abajo). Como la prueba $t \leq t_4[0, 0]$ falla en el paso (5) (i.e. no ha sido codificada suficiente información todavia para la raíz) nos vemos impulsados al paso (6), donde t_4 $[0, 0]$ es incrementado y un bit 0 es emitido. Luego se conduciría de nuevo al paso (5).

Esta vez la prueba es satisfactoria por lo que t_{min} es actualizado en el paso (5.2) y pasamos al nivel inferior del árbol. Para cada nivel subsecuente del árbol etiquetado nada es codificado dado que la información reunida ya es suficiente. Finalmente, Acabamos con un único bit 0 a codificar si $q_1[0, 0] \geq 1$. No se necesita información adicional a codificar si $q_1[m, n] \geq 1, \forall\, m, n$ dado que sabemos que la raiz contiene el mínimo del árbol completo, y este valor no es menor que el umbral dado. La Fig.D.24 muestra los bits codificados para el árbol etiquetado incrementando el valor del umbral a dos. Observe que la secuencia de los umbrales utilizados para codificar debe ser conocida por el decodificador. De hecho, el decodificador puede reconstruir los datos originales solo si la secuencia es decodificada en el mismo orden que fue codificada. Por lo tanto, los valores de la Fig.D.24 son válidos solo si el procedimiento de codificación ha sido previamente ejecutado con umbral $t = 1$.

1111	0	10	0	111	0
0	0	1	0	0	0
0	-	0	-	11	0

Figura D.24: Bits codificados para el árbol etiquetado de la Fig.D.23. El valor del umbral se establecido en 2. El procedimiento de codificado ha sido previamente ejecutado con un umbral de $t = 1$.

El algoritmo de decodificado sigue la misma idea general. Dadas las dimensiones de la matriz original y un umbral, se pueden extraer los datos originales al propagar en niveles más bajos la información ya conocida por los niveles superiores. Un pseudocódigo para el decodificador esta dado en el Algoritmo 2. Una variable de estado Booleana adicional, $s_k[m_k, n_k]$, es usada en lugar de la variable de estado $t_k[m, n]$ del procedimiento de codificado para indicar cuando la información suficiente ha sido decodificada para determinar los valores de $q_k[m_k, n_k]$. Cuando $s_k[m_k, n_k]$ se convierte en verdad no mas bits serán decodificados para $q_k[m_k, n_k]$.

D.2.2.3.2.2. Codificando la información del encabezado del paquete

El árbol etiquetado encuentra una aplicación natural en la información del encabezado del paquete de codificado JPEG2000. En particular, son usados para codificar dos campos de cada cuatro. Los valores contenidos en el encabezado del paquete para cada bloque de código son:

- información de inclusión,
- máximo número de bits en la mantisa de los coeficientes del bloque código (i.e. el número de planos de bits nulos más significativos).
- número de pasos nuevos incluidos (determinado como una función de los puntos de truncado óptimo),
- longitud del código de palabra

La información de inclusión es codificada usando un árbol etiquetado para indicar la primera capa en la cual se incluye un código de bloque dado, entonces se usa un único bit para más capas a los efectos de indicar si el bloque de código se incluye o no. Los árboles etiquetados son eficientemente codificados para esta clase de información dado que la vecindad de los bloques de código tienden a aparecer en capas cercanas unas de otras. La misma observación se mantiene verdadera para el número de planos de bits nulos más significativos. El número de pasos de codificación incluidos usan un esquema de codificación diferente, similar a la codificación de Huffman en esos valores bajos (altamente probable) obtiene menos bits que los valores altos (menos probable). Finalmente, la longitud de la palabra código es codificada como un código de dos partes en el cual la primera parte indica la longitud de la segunda. Esto a su vez indica la longitud de la palabra código. En particular, el número de bits en la segunda parte es calculada como $\boldsymbol{n}_{bits} = \lfloor \log_2(\boldsymbol{p}_i) \rfloor$, donde $\boldsymbol{p}_i$ es el número de nuevos pasos de codificación incluidos en el paquete actual. Este delicado esquema se destina como modelo del hecho que la longitud de la palabra código de alguna manera proporcional al número de pasos de codificación incluidos.

La estructura orientada a los paquetes de la cadena comprimida de JPEG2000 permite alta escalabilidad en el dominio comprimido. La progresión del orden de decodificación se determina al colocar adecuadamente los paquetes en la cadena de salida.

D.2.2.4. Resumen del algoritmo JPEG2000

Para tener una idea global del algoritmo JPEG2000, resumimos aquí las operaciones llevadas acabo por el codificador (referentes a la Fig.D.13). La imagen de entrada se somete a una transformación de color preliminar en un espacio de color separable (para la cual no haya correlación mutua entre las bandas como es el caso de RGB). Los planos de color (aparte del plano de luminancia) son posiblemente sub-muestreados (compresión con pérdidas). Entonces, la imagen es particionada en grandes bloques llamados mosaicos. Se aplica la descomposición de onditas a los mosaicos separadamente. Bloques pequeños (posiblemente cuantizados) de coeficientes de onditas son procesados separadamente usando el algoritmo EBCOT (Fig.D.16). En el nivel 1 tienen lugar el modelado de la fuente y la compresión entrópica. El modelado de la fuente se lleva a cabo en un nivel de plano de bits y es impulsado por la información estadística sobre los coeficientes de la vecindad. Su salida es un contexto en el cual ayuda el codificador aritmético para actualizar correctamente las probabilidades de los símbolos. Luego de que ha sido codificado cada bloque en un mosaico, el nivel 2 lo posiciona convenientemente en la cadena de bits de salida comprimida. La información del encabezado es comprimida así por medio de una estructura particular de árbol llamada árbol etiquetado. Una cuantización apropiada puede llevarse a cabo en esta fase de ambos lados, codificador y decodificador al truncar la cadena de bits hasta un punto de truncado óptimo. El decodificador realiza los mismos pasos en orden invertido.

D.2.2.5. Codificación de imágenes indexadas

Cuando la compresión de imágenes muestra un número bajo de colores distintos, se adopta usualmente el esquema de compresión de imágenes indexadas. Aquí la observación reside en que el número de colores en una imagen no puede exceder nunca el número de pixeles, y es usualmente mucho más baja. Entonces, podemos asignar un índice para cada color y almacenar este en lugar del color relativo. Una tabla de color (paleta) almacena la triada RGB correspondiente a las imágenes indexadas. Si el número de colores es lo suficientemente bajo (i.e. la representación de los índices es corta), los costos generales pagados para almacenar la paleta son compensados por la ganancia en la representación de los pixeles. La matriz de indices puede ser comprimida

usando cualquier método genérico de compresión sin pérdidas. Este esquema es muy efectivo cuando el número de colores es bajo, mientras su eficiencia se degrada a medida que el número de colores se hace más grande. Esto, por supuesto, depende de la dimensión de la tabla de colores. Los métodos de compresión de imagines indexadas se comportan bien con imagines de pocos colores, tal como líneas dibujadas, imágenes en blanco y negro y texto pequeño con solo unos pocos pixeles de alto. Por el contrario, la performance de codificación de tales algoritmos para imágenes de tono continuo, tales como fotos, puede ser muy pobre.

JPEG2000 puede ser usado para comprimir imágenes indexadas tan bien como aquellas de tono continuo. Datos experimentales muestran que JPEG2000 puede obtener mejores resultados que GIF o PNG si es elegida una apropiada indexación de color. La idea es suavizar las transiciones de índices en la imagen indexada. Desafortunadamente, dada una imagen $\boldsymbol{I}$ con $\boldsymbol{M}$ colores distintos, el número posible de colores indexados es $\boldsymbol{M!}$. La obtención de un esquema de re-indexado óptimo se dificulta al ser un problema intrínsicamente difícil (NP-completo). Entonces solo los métodos heurísticos pueden escoger un ordenamiento óptimo de los muchos posibles que se conocen. En esta sección revisaremos dos de ellos [503, 506].

Ambas aproximaciones presentarán una recopilación de las estadísticas de adyacencia del color usando una matriz $\boldsymbol{C}$ llamada matriz de co-ocurrencia. Cada entrada $\boldsymbol{C}(i, j)$ reporta el número de veces que el par de símbolos de color vecinos (c_i, c_j) ha sido observado sobre la imagen indexada. $\boldsymbol{C}$ es usada para ponderar los pares de índices para guiar el proceso de re-indexado. La matriz de co-ocurrencia podría ser re-organizada en función del perfil del esquema predictivo particular adoptado por el motor de compresión. Ambos algoritmos construyen una cadena de colores indexados y entonces deriva un simple re-indexado mediante la asignación de índices consecutivos para colores consecutivos. El primer método que presentamos fue propuesto por Zeng *et al* [506] durante el proceso de estandarización de JPEG2000. Es un ambicioso algoritmo iterativo basado en la maximización de la función potencial definida como $\boldsymbol{w}_{N,j} = \log_2\left(1+\frac{1}{\boldsymbol{d}_{N,j}}\right)$, donde $\boldsymbol{d}_{N,j}$ es la distancia del j-ésimo índice desde el fin de la cadena de índices actual. $\boldsymbol{N}$ indica la longitud de la cadena. En cada paso se adiciona un índice de color a uno de los de los criterios de evaluación de la cadena actual al maximizar la función $\boldsymbol{w}_{N,j}$. La cadena es inicializada al adicionar el color c_i maximizando $\sum_{j\neq i}\boldsymbol{C}(\boldsymbol{i},\boldsymbol{j})$.

En la aproximación introducida por Battiato *et al* [503] el problema del re-indexado es mapeado sobre un problema de optimización para un camino Hamiltoniano en un grafo ponderado. Dada la matriz de co-ocurrencias $\boldsymbol{C}$ calculada como se indica arriba, un grafo $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ se define como sigue:

$\boldsymbol{V} = \{ c_1, c_2, \ldots c_M\}$
$\boldsymbol{E} = \boldsymbol{V} \times \boldsymbol{V}$
$\boldsymbol{w} : \boldsymbol{E} \rightarrow \boldsymbol{N}$ donde $\boldsymbol{w}$ se define como

$$\boldsymbol{w}(\boldsymbol{i},\boldsymbol{j}) = \begin{cases} \boldsymbol{C}(\boldsymbol{i},\boldsymbol{j}) + \boldsymbol{C}(\boldsymbol{j},\boldsymbol{i}) & \boldsymbol{i} \neq \boldsymbol{j} \\ 0 & \boldsymbol{i} = \boldsymbol{j} \end{cases}$$

Un re-indexado es encontrado al buscar por el camino Hamiltoniano más pesado en $\boldsymbol{G}$, i.e. una solución para el Problema de Viajante de Comercio (Travelling Salesman Problem, TSP), el cual

es un bien conocido problema NP-Completo. Los autores sugieren una solución muy eficiente para resolver el TSP de una manera aproximada.

Los resultados experimentales muestran que los dos algoritmos presentados son comparables en función de las tasas de compresión, eficiencia computacional, y consumo de memoria.

D.3 Conclusiones del apéndice

En este apéndice se presentaron los dos algoritmos con los cuales confrontaremos las soluciones propuestas, es decir, JPEG y JPEG2000, en el Capítulo 4 (Simulaciones computacionales).

Apéndice E

E Revisión de Álgebra Lineal

E.1 Introducción

La matemática de las onditas depende en gran medida de las ideas fundamentales del álgebra lineal [502, 507, 508]. En este apéndice se examinan algunas ideas importantes.

E.2 Algebra Lineal

E.2.1 Espacio vectorial

El punto de partida para álgebra lineal es la noción de un *espacio vectorial*. Un espacio vectorial (sobre los reales) puede ser vagamente definido como una colección de elementos de V donde

1. Para todo $a, b \in \Re$ y para todo $u, v \in \boldsymbol{V}$, $au + bv \in \boldsymbol{V}$.

2. Existe un único elemento $\boldsymbol{0} \in \boldsymbol{V}$ tal que
   - para todo $u \in \boldsymbol{V}$, $0u = \boldsymbol{0}$, y
   - para todo $u \in \boldsymbol{V}$, $\boldsymbol{0} + u = u$.

3. Otros axiomas (omitidos aquí) mantienen esta verdad, la mayoría de las cuales son necesarias para garantizar que la multiplicación y la adición se comportan como se esperaba.

Los elementos de un espacio vectorial V se denominan vectores, y el elemento 0 es el llamado vector cero. Los vectores pueden ser vectores geométricos, o pueden ser funciones, como es el caso cuando se habla de onditas y el análisis de multiresolución.

E.2.2 Bases y dimensión

Se dice que una colección de vectores $u_1, u_2, \ldots$ en un espacio vectorial V son *linealmente independientes* si

$$\boldsymbol{c}_1\boldsymbol{u}_1 + \boldsymbol{c}_2\boldsymbol{u}_2 + \ldots = 0 \quad \text{si y solo si} \quad \boldsymbol{c}_1 = \boldsymbol{c}_2 = \ldots = 0 \tag{E.1}$$

Una colección $u_1, u_2, \ldots \in \boldsymbol{V}$ de vectores linealmente independientes constituye una *base* para V si cada $v \in \boldsymbol{V}$ puede ser expresado como

$$\boldsymbol{v} = \sum_i \boldsymbol{c}_i \boldsymbol{u}_i \tag{E.2}$$

para algunos números reales $c_1, c_2, \ldots$. Se dice que los vectores en una base para V son *generados* en V. Hablando intuitivamente, la independencia lineal significa que los vectores no son redundantes, y una base consiste en un conjunto de vectores mínimo y completo.

Si una base para *V* tiene un número finito de elementos $u_1, \ldots, u_m$, entonces *V* es *dimensional-finita* y su dimensión en *m*. De otra forma, se dice que *V* es *dimensional-infinito*.

Ejemplo: $\Re^3$ es un espacio tridimensional, y $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ es una base para el mismo.

Ejemplo: El conjunto de todas las funciones continuas sobre [0, 1] es un especio vectorial dimensional-infinito. Llamaremos a este especio *C*[0, 1].

E.2.3 Productos internos y ortogonalidad

Cuando se trata de los vectores geométricos del espacio vectorial $\Re^3$, la operación"producto punto" posee un número de usos. La generalización del producto punto para espacios vectoriales arbitrarios es llamado *producto interno* [509-511]. Formalmente, un producto interno $\langle \bullet \mid \bullet \rangle$ sobre un espacio vectorial *V* es todo mapeo desde *V* x *V* a $\Re$ de la que es

1. simetría: $\langle \boldsymbol{u} \mid \boldsymbol{v} \rangle = \langle \boldsymbol{v} \mid \boldsymbol{u} \rangle$,

2. bilineal: $\langle \boldsymbol{au} + \boldsymbol{bv} \mid \boldsymbol{w} \rangle = \boldsymbol{a}\langle \boldsymbol{u} \mid \boldsymbol{w} \rangle + \boldsymbol{b}\langle \boldsymbol{v} \mid \boldsymbol{w} \rangle$, y

3. definida positiva: $\langle \boldsymbol{u} \mid \boldsymbol{u} \rangle > 0$ para todo $\boldsymbol{u} \neq 0$.

A un espacio vectorial junto con un producto interno se lo conoce como *espacio de producto interno*.

Ejemplo: Es sencillo demostrar que el producto interno en puntos sobre $\Re^3$ está definido por

$$\langle (\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3) \mid (\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3) \rangle = \boldsymbol{a}_1\boldsymbol{b}_1 + \boldsymbol{a}_2\boldsymbol{b}_2 + \boldsymbol{a}_3\boldsymbol{b}_3 \tag{E.3}$$

lo que satisface los requerimientos de producto interno.

Ejemplo: El siguiente producto interno "estándar" sobre *C*[0, 1] juega un rol central en la mayoría de las formulaciones del análisis multirresolución:

$$\langle \boldsymbol{f} \mid \boldsymbol{g} \rangle = \int_0^1 \boldsymbol{f}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})\boldsymbol{dx} \tag{E.4}$$

El producto interno estándar puede ser generalizado también para incluir una función de ponderación positiva *w*(*x*):

$$\langle \boldsymbol{f} \mid \boldsymbol{g} \rangle = \int_0^1 \boldsymbol{w}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})\boldsymbol{dx} \tag{E.5}$$

Uno de los más importantes usos del producto interno es el de formalizar la idea de ortogonalidad. Se dice que dos vectores *u, v* en un espacio de producto interno son *ortogonales* si $\langle \boldsymbol{u} \mid \boldsymbol{v} \rangle = 0$. No

es dificil demostrar que una colección u_1, u_2, … de vectores mutuamente ortogonales deben ser linealmente independientes, sugiriendo que la ortogonalidad es una forma fuerte de independencia lineal. Una *base ortogonal* es aquella que consiste de vectores mutuamente ortogonales.

E.2.4 Normas y normalización

Una *norma* es una función que mide la longitud de la vectores. En un espacio vectorial dimensional finita, típicamente usamos la norma $\|\boldsymbol{u}\| = \langle \boldsymbol{u} \mid \boldsymbol{u} \rangle^{1/2}$. Si estamos trabajando con una espacio funcional tal como $C[0, 1]$, nosotros comúnmente usamos una de las normas L^p, definida como

$$\|\boldsymbol{u}\|_{\boldsymbol{p}} = \left(\int_0^1 |\boldsymbol{u}(\boldsymbol{x})|^{\boldsymbol{p}} \, \boldsymbol{dx} \right)^{1/\boldsymbol{p}} \tag{E.6}$$

En el límite, como p tiende a infinito, obtenemos aquello que es conocido como la *norma-máxima*:

$$\|\boldsymbol{u}\|_{\infty} = \max_{\boldsymbol{x} \in [0,1]} |\boldsymbol{u}(\boldsymbol{x})| \tag{E.7}$$

Aún más frecuentemente utilizada es la norma L^2, la cual también puede ser escrita como $\|\boldsymbol{u}\|_2 = \langle \boldsymbol{u} \mid \boldsymbol{u} \rangle^{1/2}$ si nosotros estamos usando el producto interior estándar.

Un vector u con $\|\boldsymbol{u}\| = 1$ se dice que está *normalizado*. Si tenemos una base ortogonal compuesta de vectores que están normalizados en la norma L^2, la base es llamada *ortonormal*. Dicho de forma concisa, una base u_1, u_2, … es ortonormal si

$$\langle \boldsymbol{u}_i \mid \boldsymbol{u}_j \rangle = \delta_{ij} \tag{E.8}$$

donde a δ_{ij} se lo conoce como delta de Kronecker y se define para ser 1 si $i = j$, y 0 en cualquier otro caso.

Ejemplo: Los vectores $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ forman una base ortonormal para el espacio de producto interno $\Re^3$ dotado con el *producto punto* de la Ecuación (E.3).

E.3 Conclusiones del apéndice

En este apéndice se han expuesto los conceptos mínimos imprescindibles de Álgebra Lineal a los efectos de constituir al presente trabajo en una unidad autosuficiente.

Glosario: Acrónimos

1D	Unidimensional
2D	Bidimensional
ACP	Análisis de Componentes Principales
AGE	Algoritmo de Gradiente Estocástico
AHG	Algoritmo Hebbiano Generalizado
AHK	Algoritmo Hebbiano Kernelizado
AOK	Algoritmo de Oja-Karhunen
ARMA	Auto-Regressive Moving Average: Auto-regresivo de Promedios Móviles
ASI	Agencia Spaziale Italiana
Buffer	Registro de memoria temporal sin direccionamiento
CC	Complejidad Computacional
CIOP	Centro de Investigaciones Opticas
CNEA	Comisión Nacional de Energía Atómica
CNES	Centre National d"Etudes Spatiales
CONAE	Comisión Nacional de Actividades Espaciales
CONICET	Consejo Nacional de Investigaciones Científicas y Técnicas
CPU	Central Processing Unit, Unidad Central de Procesamiento
CSA	Canadian Space Agency
DMA	Desviación Media Absoluta
EBCOT	Embedded Block Coding with Optimised Truncation: Codificación por bloque embebido con Truncamiento Optimizado
ECMa	Error Cuadrático Medio de Autovalores
ECMA	Error Cuadrático Medio de Autovectores
EHKR	Espacio de Hilbert Kernelizado de Reproducción
EOB	End of Block: Fin de Bloque
ETC	Estación Terrena de Córdoba
FDP	Función de Densidad de Probabilidad
FPGA	Field Programmable Gate Array
GIF	Graphics Interchange Format: Formato de Intercambio de Gráficos
INPE	Instituto Nacional de Pesquisas Espaciais
INVAP	Investigación Aplicada S.E.
JBIG	Joint Bi-level Image Experts Group: Comité de Grupos de Expertos en Imágenes de dos niveles
JPEG	Joint Photographic Experts Group: Grupo de Expertos Enrolados en Fotográfía
JPEG2000	Nueva versión de JPEG basada en la TDO
JPL	Jet Propulsión Laboratory
MCU	Minimum Coded Unit: Unidad Mímima de Codificado
MDE	Meta-Descenso Estocástico
MSE	Mean Squared Error: Error Cuadrático Medio
NASA	National Aeronautics and Space Administration
PNG	Portable Network Graphics: Gráficos Portables de Red
RGB	Red Green Blue: Rojo Verde Azul
SMEP	Similitud Morfológica Equi-Posicional
TDC	Transformada Discreta de Coseno

TDCA	Transformada Discreta de Coseno Adaptativa
TDF	Transformada Discreta de Fourier
TDH	Transformada Discreta de Hartley
TDKL	Transformada Discreta de Karhunen-Loève
TDO	Transformada Discreta de Onditas
TEUCP	Tiempo Empleado por la Unidad Central de Procesamiento
TH	Transformada de Haar
ti	triangular inferior
TS	Transformada de Slant
TSP	Travelling Salesman Problem: Problema del Viajante de Comercio
TWH	Transformada de Walsh-Hadamard
VLC	Variable-Length Code: Código de Longitud Variable
VLI	Variable-Length Integer: Entero de Longitud Variable
VLSI	Very Large Scale Integration: Integración de Muy Gran Escala
TSP	Travelling Salesman Problem: Problema del Viajante de Comercio
UNLP	Universidad Nacional de la Plata
YUV	Mapa de color derivado del RGB

Bibliografía

[1] **Tjoa S., Lin W. S. and Ray Liu K. J.**, "Transform coder classification for digital image forensics". Disponible en: http://www.cspl.umd.edu/sig/publications/tjoa_ICIP_200709.pdf

[2] **Preisendorfer R. W.**, Principal Component Analysis in Meteorology and Oceanography, Elsevier, Amsterdam, 1988.

[3] **Jollife I. T.**, Principal Component Analysis, 2/e, Springer-Verlag, New York, 2002.

[4] **Diamantaras K. I. y Kung S. Y.**, Principal Component Neural Networks, Wiley, New York, 1996.

[5] **Rao C. R.**, "The Use and Interpretation of Principal Component Analysis in Applied Research," Sankhya, **26**, 329 (1964).

[6] **Jolicoeur P. y Mosimann J. E.**, "Size and Shape Variation in the Painted Turtle: A Principal Component Analysis," Growth, **24**, 339 (1960).

[7] **Bretherton C. S., Smith C., y Wallace J. M.**, "An Intercomparison of Methods for Finding Coupled Patterns in Climate Data," J. Climate, **5**, 541 (1992).

[8] **Hadi A. S. y Ling R. F.**, "Some Cautionary Notes on the Use of Principal Components Regression," Amer. Statistician, **52**, 15 (1998).

[9] **Naik D. N. y Khattree R.**, "Revisiting Olympic Track Records: Some Practical Considerations in the Principal Component Analysis," Amer. Statistician, **50**, 140 (1996).

[10] **Hotelling H.**, "Analysis of a Complex of Statistical Variables into Principal Components," J. Educ. Psychol., **24**, 417 (1933).

[11] **Hotelling H.**, "The Most Predictable Criterion," J. Educ. Psychol., **26**, 139 (1935).

[12] **Kumar P. S. y Prabhu K. M. M.**, "A special case for the KLT of Markov-1 Process," IEEE Trans. SP (Under review).

[13] **Chan Y. H.**, "On the substitution of the Karhunen-Loeve transform," IEEE Trans. SP (Under review).

[14] **Fleury M., Self R. P., y Downton A. C.**, Development of a Fine-grained Parallel Karhunen-Loève Transform.
Disponible en: http://privatewww.essex.ac.uk/~fleum/jpdsPapv3.pdf

[15] Transformada de Karhunen-Loève (KLT). Disponible en:
http://www.diac.upm.es/acceso_profesores/asignaturas/tdi/tdi/transformadas/pdf/karhunen.pdf

[16] **Dony A. C.**, "Karhunen-Loève Transform", The Transform and Data Compression Handbook, CRC Press, 2001.

[17] **Nakagawa M. y Miyahara M.**, "Generalized Karhunen-Loeve transformation- I: Theoretical Consideration," IEEE Trans. Commun. Vol. C-35, pp. 215-223, Feb. 1987.

[18] **Ziyal N. A. *et al***, "Image representation, recovery and analysis using principal component analysis," ICSPAT 97, San Diego, CA, Sept. 1997.

[19] **Hamilton D. J., Sandham W. A. y Blanco A.**, "Electrocardiogram data compression using non-linear principal component analysis," IEEE SP Letters (in print).

[20] **Kitter J. y Young P. C.**, "A new approach to feature selection based on the Karhunen-Loeve expansion," Pattern Recognition, Vol. 5, pp. 335-352, 1973.

[21] **Chen C. C. T., Chen C. T. y Tsai C. M.**, "Karhunen-Loeve transform for text independent speaker recognition," 1997 Intl. Symp. on Communications, Hsinchu, Taiwan, Dec. 1997.

[22] **Chen C. S. y Huo K. S.**, "Karhunen-Loeve method for data compression and speech synthesis," IEE Proc., Vol. 138, pp. 377-380, Oct. 1991.

[23] **Mudugamuwa D. J. y Bradley A. B.**, "Optimal transform for segmented parametric speech coding," ICASSP 98, pp. 53-56, Seattle, WA, May 1998.

[24] **Chouikha M. F., Gilmore E. T. y Ziyad N.,** "Adaptive principal component extraction (APEX) for image compression," ICSPAT 98, Toronto, Canada, Sept. 1998.

[25] **Tsapatsoulis N., Alexopoulos V. y Kollias S.**, "A vector based approximation of KLT and its application to face recognition," EUSIPCO-98, vol. 3, pp. 1581-1585, Island of Rhodes, Greece, Sept.1998.

[26] **Ziyad N., Gilmore E. T. y Chouikha M. F.**, "Improvements for image compression using adaptive principal component extraction," 32nd Asilomar Conf. on Signals, Systems, and Computers, Pacific Grove, CA, Nov. 1998.

[27] **Swets D. L. y Weng J.**, "Using discriminant eigenfeatures for image retrieval," IEEE Trans. PAMI, vol.18, pp.831-836, Aug. 1996.

[28] **Tang X. y Stewart W. K.**, 'Texture classification using principal component analysis techniques," Proc. of SPIE, vol. 2315, pp. 22-35, 1994.

[29] **Rao K. R.**, Chapter 3 Optimal Decorrelation and the KLT. Disponible en: http://www-ee.uta.edu/Dip/Courses/EE5355/ch3.pdf

[30] **Palacios A., Gunaratne G. H., Gorman M. y Robbins K. A.**, Karhunen-Loève analysis of spatiotemporal flame patterns, Physical Review E Vol 57, Number 5 May 1998, pp.5958-5971.

[31] **Kitajima H. y Shimone T.**, "Some aspects of the fast Karhunen-Loeve transform," IEEE Trans. Commun., Vol. 28, pp. 1773-1776, Sept. 1980.

[32] **Algazi V. R. y Sakrison D. J.**, "On the optimality of Karhunen-Loeve expansion," IEEE Trans. IT, Vol. IT-15, pp.319-321, March 1969.

[33] **Kermani B. G., Schiffman S. S., y Nagle H. T.**, "A Novel Method for Reducing the Dimensionality in a Sensor Array," IEEE Trans. Instr. Meas. **IM-47**, 728 (1998).

[34] **Howard P. G.,** The Design and Analysis of Efficient Lossless Data Compression Systems. PhD thesis, Department of Computer Science, Brown University, 1993.

[35] **Mongkolworaphol S., Rangsanseri Y. y Thitimajshima P.,** Multispectral Image Compression using FCM-Based Vector Quantization. Disponible en: http://www.gisdevelopment.net/aars/acrs/2000/ts9/imgp0011.asp

[36] **Jain A. K.**, *Fundamentals of Digital Image Processing*, Englewood Cliffs, New Jersey, 1989.

[37] **Mastriani M.**, "*Union is Strength in Lossy Image Compression*," International Journal of Signal Processing, Volume 5, Number 2, pp.112-119, 2009. Disponible en: http://www.waset.org/ijsp/v5/v5-2-14.pdf

[38] **Roth P. M. y Winter M,** "Survey of Appearance-Based Methods for Object Recognition", Technical Report ICG-TR-01/08, Graz, January 15, 2008. Disponible en: http://web.mit.edu/~wingated/www/introductions/appearance_based_methods.pdf

[39] MATLAB® R2008a (Mathworks, Natick, MA).
Disponible en: http://www.mathworks.com/access/helpdesk/help/helpdesk.html

[40] **Burl J. B**, "Estimating the basis functions of the Karhunen-Loeve transform," IEEE Trans. ASSP, Vol. 37, pp.99-105, Jan. 1989.

[41] **Musatenko Y. S. y Kurashov V. N.**, "Nonlinear improving of Karhunen-Loeve bases obtained by approximate 2D procedures," IS&T/SPIE's 9th Annual Symp., Electronic Imaging, Vol. 3026, San Jose, CA, Feb. 1997.

[42] **Herrara A., Martinez M. y Sanchez O.**, "An acoustic isolated speech recognition approach using KLT and VQ," ICSPAT 97, San Diego, CA, Sept. 1997.

[43] **Selin I.**, "Detection theory," Princeton, NJ: Princeton University Press, 1965.

[44] **She Z., Bogner R. E. y Gray D. A.**, "An eigenvector approach for inverse synthetic aperture radar (SAR) motion compensation and imaging," TENCON 97, IEEE Region 10 Annual Conf., Brisbane, Australia, Dec. 1997.

[45] **Turk M. y Pentland A.**, "Eigenfaces for recognition," J. Of Cognitive Neuroscience, Vol. 3, pp.73-86, 1991.

[46] **Jain A. K.**, "A fast Karhunen-Loeve transform for recursive filtering of images corrupted by white and colored noise," IEEE Trans. Comput. Vol. C-26, pp. 560-571, June 1977.

[47] **Konstantinides K., *et al***, “Noise using block-based singular value decomposition,” *IEEE Transactions on Image Processing*, 6(3), pp.479-483, 1997.

[48] **Netsch L.**, “A robust telephone-based speaker verification system,” Ph.D. Dissertation proposal, Univ. of Texas at Arlington, Arlington, TX, 1992.

[49] **Chakrabarti N. B., Rao T. V. K. H. y Krishna N. R.,** “A recursive filter implementation of KL transform,” SP, Vol. 44, pp. 269-284, July 1995.

[50] **Borman S. y Stevenson R.,** “Image sequence processing,” Department, Ed. Marcel Dekker, New York, 2003. pp.840-879.
Disponible en: http://www.seanborman.com/publications/EOE_Borman.pdf

[51] **Turk M. y Pentland A.**, “Eigenfaces for Recognition,” J. Cogn. Neurosci., **3**, 71 (1991).

[52] **Belhumeur P. N., Hespanha J. P. y Kriegman D. J.**, “Eigenfaces vs. fisherfaces: recognition using class specific linear projection,” IEEE Trans. PAMI, vol.19, pp.711-720, July1997.

[53] **Faloutsos C. y Lin K-I.**, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," In Proc. of SIGMOD, pp: 163-174, 1995.

[54] **Chandrasekaran S. *et al***, "An eigenspace update algorithm for image analysis," CVGIP: Graphical models and image processing journal, 1997.

[55] **White D. y Jain R.**, "Similarity indexing: Algorithm and performance," In Proc. SPIE Storage and Retrieval for Image and Video Databases, 1996.

[56] **Wang Z. y Arie J. B-**, "3D motion estimation using expansion matching and KL based canonical images," IEEE ICIP, pp. MP11-7, Chicago, IL, Oct. 1998.

[57] **Hjorungnes A. y Ramstad T. A.**, "Minimum mean square error transform coders," IEEE ISPACS'98, pp. 738-742, Melbourne, Australia, Nov. 1998.

[58] **Krot A. M., y Kudryavtsev V. O.**, "Eigen transforms over finite rings in filter bank structures," IEEE ISPACS'98, pp. 738-742, Melbourne, Australia, Nov. 1998.

[59] **Kirby M. y Sirovich L.**, "Application of the Karhunen-Loève procedure for the characterization of human faces," IEEE Trans. Pattern Anal. Machine Intell, vol. 12, pp. 103-108, 1990.

[60] **Shapiro L. S. y Brady J. M.**, "Feature-based correspondence: An eigenvector approach," Image Vision Computing, vol. 10, pp. 283-288, 1992.

[61] **Nefian A. V.**, "Face detection and recognition using hidden Markov models," IEEE 20 ICIP, pp. MA5.05, Chicago, IL, Oct. 1998.

[62] **Yan Y. y Zhang J.**, "Rotation invariant 3D reconstruction for face recognition," IEEE ICIP, pp. MA5.08, Chicago, IL, Oct. 1998.

[63] **Celebi H. y Trusset H. J.**, "Colorimetric restoration of digital images," IEEE ICIP, pp. MA6.10, Chicago, IL, Oct. 1998.

[64] **Nandy D. y Arie J. B-**, "EXM eigen templates detecting and classifying arbitrary junctions", IEEE ICIP, pp. MA7.01, Chicago, IL, Oct. 1998.

[65] **Kirac A. y Vaidyanathan P. P.**, "Optimal nonuniform orthogonal filter banks for subband coding and signal representation," IEEE ICIP, pp.WP5.06, Chicago, IL, Oct. 1998.

[66] **Chan L. A. y Nasrabadi N. M.**, "Wavelet-eigen transformation for automatic target recognition," Photonics West, SPIE, vol. 3647, San Jose, CA, Jan. 1999.

[67] **Liu H.**, "Real-time human face recognition using eigenface-based optical filtering," Photonics West, SPIE, vol. 3645, San Jose, CA, Jan. 1999.

[68] **Phoong S. M. y Lin Y. P.**, "PLT versus KLT," ISCAS99, pp.15-19, Orlando, FL, May-June 1999.

[69] **Lee J.**, "Optimized quadtree for Karhunen-Loeve transform in multispectral image coding," IEEE Trans. IP, vol. 8, pp. 453-461, April 1999.

[70] **Dony R. D. y Haykin S.**, "Optimally adaptive transform coding" IEEE Trans. IP, Vol. 4, pp. 1358-1370, 1995.

[71] **Epsein B. R. *et al.***, "Multispectral KLT-wavelet data compression for Landsat thematic mapper images, DCC, pp. 200-208, March 1992.

[72] **Kongkchandra R., Tamee K., y Kimpan C.**, "Improving Thai isolated word recognition by using Karhunen-Loeve transformation and learning vector quantization", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[73] **Kongkchandra R., Tamee K., y Kimpan C.**, "Using Karhunen-Loeve transformation for feature reduction and tones analysis in Thai harmonic frequency speech", ", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[74] **Lahme B. y Miranda R.**, "Karhunen-Loeve decomposition in the presence of symmetry-part I," IEEE Trans IP, vol. 8, pp. 1183-1190, Sept. 1999.

[75] **Tanaka T. y Yamashita Y.**, "Image coding using vector embedded Karhunen-Loeve transform," IEEE ICIP, Kobe, Japan, Oct. 1999.

[76] **Chitwong S. *et al***, "Enhancement of color image obtained from principal component analysis using local area histogram equalization," IEEE ISPACS, Honolulu, HI, Nov. 2000.

[77] **Bharitkar S. y Kyriakakis C.**, "Eigenfilters for signal cancellation," IEEE ISPACS, Honolulu, HI, Nov. 2000.

[78] **Turk M. y Pentland A.**, "Face processing: models for recognition," Intelligent Robots and Computer Vision VIII, SPIE, Philadelphia, PA, 1989.

[79] **Davila C. E.**, "Blind KLT coding and vector quantization," IEEE 9th DSP Workshop, Hunt, TX, Oct. 2000.

[80] **Dony R. D.**, "Karhunen-Loeve transform," Ch. 1 in the transform and data compression handbook (Eds. K.R. Rao and P.C. Yip), Boca Raton, FL: CRC Press, 2001.

[81] **Kongkchandra R., Tamee K., y Kimpan C.**, "Improving Thai isolated word recognition by using Karhunen-Loeve transformation and learning vector quantization", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[82] **Kongkchandra R., Tamee K., y Kimpan C.**, "Using Karhunen-Loeve transformation for feature reduction and tones analysis in Thai harmonic frequency speech", IEEE ISPACS'99, Pukhet, Thailand, Dec. 1999.

[83] **Ray W. D. y Driver R. M**, "Further decomposition of the Karhunen-Loeve series representation of a stationary random process", IEEE Trans. IT, vol. IT-16, pp. 663-668, Nov. 1970.

[84] **Cendrillon R. y Lovell B.**, "Real-time face recognition using eigenfaces," SPIE/VCIP 2000, vol. 4067, pp.2-7, Perth, Australia, June 2000.

[85] **Akkarakaran S. J. y Vaidyanathan P. P.**, "Existence and optimality of nonuniform principal component filter banks," EUSIPCO2000, Tampere, Finland, Sept. 2000. Disponible en: http://eusipco2000.cs.tut.fi

[86] **Eggers J. J., Su J. K. y Girod B.**, "Public key watermarking by eigenvectors of linear transforms," EUSIPCO2000, Tampere, Finland, Sept. 2000.
Disponible en: http://eusipco2000.cs.tut.fi

[87] **Quddus A. y Gabbouj M.**, "Selection of natural scale in discrete wavelet domain using eigenvalues," EUSIPCO2000, Tampere, Finland, Sept. 2000.
Disponible en: http://eusipco2000.cs.tut.fi

[88] **Yang M-h, Ahuja N., y Kriegman D.**, "Face recognition using kernel eigenfaces," IEEE ICIP, Vancouver, Canada, Sept. 2000.

[89] **Rizvi S. A., Saadawi T. N., y Nasrabadi N. M.**, "A modular clutter rejection technique for FLIR imagery using region-based principal component analysis," IEEE ICIP, Vancouver, Canada, Sept. 2000.

[90] **Han J. K. y Tewfik A. H.**, "Eigen-image based video segmentation and indexing," IEEE ICIP97, vol. 2, pp.538-541, Santa Barbara, CA, Oct. 1997.

[91] **Chang C-Y. *et al***, "Fast eigenspace decomposition of correlated images," Proc. Intl. Conf. on Intelligent Robots and Systems (IROS), vol. 1, pp. 7-12, Victoria, Canada. Oct. 1998.

[92] **Stefanoiu D. y Tabus I.**, "Degenerate eigenvalues – a method to design adaptive discrete time wavelets," EUSIPCO2000, Tampere, Finland, Sept. 2000. Disponible en: http://eusipco2000.cs.tut.fi

[93] **Akkarakaran S. J. y Vaidyanathan P. P.**, "Principal component filter banks: existence issues and application to modulated filter banks," IEEE ISCAS 2000, Geneva, Switzerland, May 2000.

[94] **Kim K. I.**, "Kernel principal component analysis for texture classification," IEEE TENCON, Kuala Lumpur, Malaysia, Sept. 2000.
Disponible en: http://www.cairo.utm.my/TENCON2000

[95] **Davila C. E.**, "Blind KLT coding and vector quantization," IEEE DSP Workshop, Hunt, TX, Oct. 2000.

[96] **Bregles C. y Kao Y. K.**, "Eigenlips for robust speech recognition." IEEE ICASSP, pp.669-672, Adelaide, Australia, 1994.

[97] **Mase K. y Pentland A.**, "Automatic lipreading by optical-flow analysis," Syst. Compt. Japan, vol.2, pp.67-76, Jan.1991.

[98] **Turk M. y Pentland A.**, "Face recognition," J. Cognitive Neurosci., vol.3, pp.71-86, Jan. 1991.

[99] **Sahouria E. y Zakhor A.**, "Content analysis of video using principal components," IEEE Trans. CSVT, vo. 9, pp. 1290-1298, Dec. 1999.

[100] **Xiangdong W.**, "Image segmentation using eigencluster extraction," IEEE ICASSP, Salt Lake City, May 2001.

[101] **Hasan M. y Hasan A.**, "Parallelizable eigenvalue decomposition techniques via the matrix sector function,' IEEE ICASSP, Salt Lake City, May 2001.

[102] **Chen T.**, "Principal component analysis for facial animation," IEEE ICASSP, Salt Lake City, May 2001.

[103] **Gan J-Y., Zhang Y-W. y Mao S-Y.**, "Application of adaptive principal components extraction algorithm in the feature extraction of human face," IEEE ISIMP 20001, Hong Kong, May 2001.

[104] **Flierl M. y Girod B.**, "Video coding with motion compensation for groups of pictures," IEEE ICIP 2002, pp. I-69 – I-72, Rochester, NY, Sept. 2002.

[105] **Ouyang S. y Bao Z.**, "Fast principal component extraction by a weighted information criterion," IEEE Trans. SP, vol. 50, pp. , Aug. 2002.

[106] **Chung K., Kee S. C. y Kim S. R.**, "Face recognition using principal component analysis of Gabor filter responses," Proc. 1999 Intrnl. Workshop on recognition, analysis and tracking of faces and gestures in real-time systems, pp. 53-57, 1999.

[107] **Rahman M. M. y Ishikawa S.**, "Eigenspace tuning for human standing pose detection," IS&T/SPIE's 15 th Annual Symp., vol. 5014, Santa Clara, CA, Jan.2003.

[108] **Li B. y Wei J.**, "Remote sensing image fusion on PCA and WT," IS&T/SPIE's 15 th Annual Symp., vol. 5014, Santa Clara, CA, Jan. 2003.

[109] **Hao P. y Shi Q.**, Reversible integer KLT for progressive-to-lossless compression of multiple component images, *Proceedings of International Conference on Image Processing* (ICIP) 2003, Vol. 1, pp. 633-636, Sep. 14-17 2003, Barcelona, Spain.

[110] **Tsai C., Zhang J., y Janatra I. I.**, "Pictures at an Eigen-Exhibition. Disponible en: http://www.stanford.edu/class/ee368/Project_07/reports/ee368group03.pdf

[111] **Parlett B.**, Symmetric Eigenvalue Problem, Prentice Hall, Upper Saddle River, NJ, 1980.

[112] **Abarbanel H. D. I.**, Analysis of Observed Chaotic Data, Springer-Verlag, New York, 1996.

[113] **Kantz H. y Schreiber T.**, Nonlinear Time Series Analysis, Cambridge Univ. Press, Cambridge, 1997.

[114] -., A. S. Weigend and N. A. Gershenfeld, eds., Time Series Prediction: Forecasting the Future and Understanding the Past Addison-Wesley, Reading, MA, 1994. Disponible en: ftp://ftp.santafe.edu/pub/Time-Series/

[115] **Yang D., Ai H., Kyriakakis C., y Kuo C.-C.**, An inter-channel redundancy removal approach for high-quality multichannel audio compression," in AES 109th convention, AES preprint 5238, (Los Angeles, CA), September 2000.

[116] **Yang D., Ai H., Kyriakakis C., y Kuo C.-C.**, "An Exploration of Karhunen-Loeve Transform for Multi-channel Audio Coding"," in Conference of Electronic Cinema, SPIE's International Symposium on Information Technologies, (Boston, MA), November 5-8 2000.

[117] **Yang D., Ai H., Kyriakakis C., y Kuo C.-C.**, "Adaptive Karhunen-Loeve Transform for Enhanced Multichannel Audio Coding".
Disponible en: http://www.mp3-tech.org/programmer/docs/conf2001_2.pdf

[118] **Zhang D., y Chen S.**, "Fast image compression using matrix K-L transform," Neurocomputing, Volume 68, pp.258-266, October 2005.

[119] -, The transform and data compression handbook, Edited by K.R. Rao, and P.C. Yip, CRC Press Inc., Boca Raton, FL, USA, 2001.

[120] **Semmlow J. L.**, Biosignal and biomedical image processing: MATLAB-Based applications, Marcel Dekker, Inc., New York, 2004.

[121] **Battiato S., Gallo G., Impoco G., y Stanco F.,** A color reindexing algorithm for lossless compression of digital images, 2001.

[122] **Zeng W., Li J., y Lei S.,** An efficient color re-indexing scheme for palette-based compression. In ICIP00, pages 476–479, 2000.

[123] **Ponceleon D. B., *el al***, "Transform coding for low bit rate applications," IS&T/SPIE Symposium on Electronic Imaging: Science & Technology, Vol. 2187, pp. San Jose, CA, Feb. 1994.

[124] **Martinelli G., Ricolti L. P. y Marcone G.**, "Neural clustering for optimal KLT image compression," IEEE Trans. SP, Vol. 41, pp. 1737-1739, April 1993.

[125] **Percival D. J.**, "Compressed representation of a backscatter ionogram data base using Karhunen-Loeve techniques," ICIP, pp.1-22, Edinburgh, U.K. July 1995.

[126] **Lan L. S y Reed I. S.**, "Image compression with the adaptive approximate Karhunen-Loeve transform," SPIE/VCIP, Vol. 2308, pp. , Chicago, IL, Sept. 1994.

[127] **Sikora T., Bauer S. y Makai B.**, "Efficiency of shape-adaptive 2-D transforms for coding of arbitrarily shaped image segments," IEEE Trans. CSVT, Vol. 5, pp. 254-258, June 1995.

[128] **Reed I. S. y Lan L. S.**, "A fast Karhunen-Loeve transform (KLT) for data compression," J VCIR, Vol. 5, pp. 304-316, Dec. 1994.

[129] **Xia X. G. y Suter B. W.**, "On vector Karhunen-Loeve transform and optimal vector transforms," IEEE Trans. CSVT, Vol. 5, pp. 372-374, Aug. 1995.

[130] **Ding W.**, "Optimal vector transform for vector quantization," IEEE SP Letters, Vol. 1, pp.110-113, July 1994.

[131] **Ding R. D. y Haykin S.**, "Optimally adaptive transform coding," IEEE Trans. IP, Vol. 4, pp. 1358-1370, Oct. 1995.

[132] **Huang S. C. y Huang Y. F.**, "Principal component vector quantization," J VCIR, Vol. 4, pp. 112-120, March 1993.

[133] **Huang S. C. y Huang Y. F.**, "A constrained vector quantization scheme for realtime code book retransmission," IEEE Trans. CSVT, Vol. 4, pp. 1-7, Feb. 1994.

[134] **Oja E. y Karhunen J.,** On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1): 69–84, February 1985.

[135] **Oja E.**, "A Simplified Neuron Model as a Principal Component Analyzer", Journal of Mathematical Biology, Vol.15, pp.267-273, 1982.

[136] **Haykin S.**, Neural Networks: A Comprehensive Foundation, Second Edition, Pearson, Delhi, 2005.

[137] **Sanger T. D.**, "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Networks," Vol.2, pp.459-473, 1989.

[138] **Kim K. I., Franz M. O., y Schölkopf B.,** Iterative kernel principal component analysis for image modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(9): 1351–1366, 2005.

[139] **Günter S., Schraudolph N. N., y Vishwanathan S. V. N.**, "Fast Iterative Kernel Principal Component Analysis," Journal of Machine Learning Research 8 (2007) 1893-1918.

[140] **Schraudolph N. N., Günter S., y Vishwanathan S. V. N.,** "Fast iterative kernel PCA". NIPS 2006: 1225-1232.

[141] **Schölkopf B. y Smola A. J.,** *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[142] **Schölkopf B., Smola A. J., y Müller K.-R.,** Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[143] **Sharma A., y Paliwal K. K.**, "Fast principal component analysis using fixed-point algorithm", Pattern Recognition Letters, Vol.28, pp.1151-1155, 2007.

[144] **Principe J. C. *et al***, "Recursive Principal Components Analysis Using Eigenvector Matrix Perturbation," EURASIP Journal on Applied Signal Processing Volume 2004, Issue 13, pp.2034-2041

[145] **Guleryuz O. G. y Orchard M. T.,** "Optimized Nonorthogonal Transforms for Image Compression," *IEEE Transactions on Image Processing*, vol. 6, No. 4, pp. 507-522, April, 1997.

[146] **Guleryuz O. G.**, "Optimal Linear Processing for Image and Video Coding." PhD Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois 1997.

[147] **Zhong J. y Sclaroff S.**, "*Segmenting foreground objects from a dynamic textured background via a robust Kalman filter* ", ICCV03, Vol.6, pp. 44-50, 2003.

[148] **Dehaene J., Moonen M., y Vandewalle J.**, "An improved fully parallel stochastic gradient algorithm for subspace tracking", in *Signal Processing VIII*, (Ramponi G., Sicuranza G.L., Carrato S. and Marsi S., Eds.), Proc. of EUSIPCO-96, Eighth European Signal Processing Conference, Edizione LINT (Trieste, Italy,), 1996, pp. 1373-1376.

[149] **Gonzalez R. C., Woods R. E.**, Digital Image Processing, 2nd Edition, Prentice- Hall, Jan. 2002, pp.675-683.

[150] **Gonzalez R. C., Woods R. E., y Eddins S. L.,** Digital Image Processing using Matlab. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[151] **Jain A. K.**, "A fast Karhunen-Loeve transform for a class of random processes," IEEE Trans. Commun. Vol. COM-24, pp. 1023-1029, Sept. 1976

[152] **Epstein B. R., *et al***, "Multispectral KLT-wavelet data compression for landsat thematic mapper images," In *Data Compression Conference*, pp. 200-208, Snowbird, UT, March 1992.

[153] **Lee J.**, "Optimized quadtree for Karhunen-Loève Transform in multispectral image coding," *IEEE Transactions on Image Processing*, 8(4), pp.453-461, 1999.

[154] **Saghri J. A., *et al***, "Practical Transform coding of multispectral imagery," *IEEE Signal Processing Magazine*, 12, pp.32-43, 1995.

[155] **Kim T-S., *et al***, "Multispectral image data compression using classified prediction and KLT in wavelet transform domain," IEICE Transactions on Fundam Electron Commun Comput Sci, Vol**.** E86-A; No**.**6, pp**.**1492-1497, 2003.

[156] **Saghri J., Tescher A., y Reagan J.**, "Practical Transform Coding of Multispectral Imagery", IEEE Signal Processing Magazine, January 1995.

[157] **Saghri J. y Tescher A.**, "Near-lossless bandwidth compression for radiometric data", Optical Engineering 30, July 1991.

[158] **Hunt B. R. y Kubler O.**, "Karhunen-Loeve multispectral image restoration, part 1: Theory," IEEE Trans. ASSP, Vol. ASSP-32, pp. 592-600, June 1984.

[159] **Saghri J. A., Tescher A. G. y Reagan J. T.**, "Practical transform coding of multispectral imagery," IEEE SP Magazine, Vol. 12, pp. 32-43, Jan. 1995.

[160] **Vaughn V. D. y Wilkinson T. S.**, "System considerations for multispectral image compression designs," IEEE SP Magazine, Vol. 12, pp. 19-31, Jan. 1995.

[161] **Tretter D. y Bouman C. A.**, "Optimal transforms for multispectral and multilayer image coding," IEEE Trans. IP, Vol. 4, pp. 296-308, March 1995.

[162] **Epstein B. R., *et al***, "Multispectral KLT-wavelet data compression for LANDSAT thematic mapper images," Proc. DCC, Mar. 1992.

[163] **Christophe E., *et al***, "Hyperspectral image compression: adapting SPIHT and EZW to anisotopic 3D wavelet coding," submitted to IEEE Transactions on Image processing, pp.1-13, 2006. Disponible en:
http://www.lss.supelec.fr/~publi/UGllcnJlIERVSEFNRUw=_EChristophe-adaptSPIHT.pdf

[164] **Rodríguez del Río L. S.**, "Fast piecewise linear predictors for lossless compression of hyperspectral imagery," Thesis for Degree in Master of Science in Electrical Engineering, University of Puerto Rico, Mayaguez Campus, 2003.

Disponible en: http://grad.uprm.edu/tesis/rodriguezdelrio.pdf

[165] -. *Hyperspectral Data Compression*, Edited by Giovanni Motta, Francesco Rizzo and James A. Storer, Chapter 3, Springer, New York, 2006.

[166] **Mastriani M.**, "Denoising and compression in wavelet domain via projection onto approximation coefficients," International Journal of Signal Processing, Volume 5, Number 1, pp.20-30, 2008.

[167] -, The transform and data compression handbook, Edited by K.R. Rao, and P.C. Yip, CRC Press Inc., Boca Raton, FL, USA, 2001.

[168] **Epstein B. R., *et al***, "Multispectral KLT-wavelet data compression for landsat thematic mapper images," In *Data Compression Conference*, pp. 200-208, Snowbird, UT, March 1992.

[169] **Stollnitz E. J., DeRose T. D., y Salesin D. H.,** Wavelets for computer graphics: A primer, Part I. IEEE Computer Graphics and Applications, 15(3):76–84, May 1995. Disponible en: http://grail.cs.washington.edu/pub/stoll/wavelet1.pdf

[170] **Stollnitz E. J., DeRose T. D., y Salesin D. H.,** Wavelets for computer graphics: A primer, Part II. IEEE Computer Graphics and Applications, 15(4), July 1995. In press. Disponible en: http://grail.cs.washington.edu/projects/wavelets/article/wavelet2.pdf

[171] **Berman D., Bartell J., y Salesin D.,** Multiresolution painting and compositing. In Proceedings of SIGGRAPH 94, pages 85–90. ACM, New York, 1994.

[172] **Beylkin G., Coifman R., y Rokhlin V.,** Fast wavelet transforms and numerical algorithms I. Communications on Pure and Applied Mathematics, 44(2):141–183, March 1991.

[173] **Christensen P. H., Lischinski D., Stollnitz E. J., y Salesin D. H.,** Clustering for glossy global illumination. ACM Transactions on Graphics, Vol. 16, Issue 1, pp. 3–33, January 1997.

[174] **Christensen P. H., Stollnitz E. J., Salesin D. H., y DeRose T. D.,** Wavelet radiance. In G. Sakas, P. Shirley, and S. Müller, editors, Photorealistic Rendering Techniques, pages 295–309. Springer-Verlag, Berlin, 1995.

[175] **Daubechies I.,** Orthonormal bases of compactly supported wavelets. Communications on Pure and Applied Mathematics, 41(7):909–996, October 1988.

[176] **DeVore R., Jawerth B., y Lucier B.,** Image compression through wavelet transform coding. IEEE Transactions on Information Theory, 38(2):719–746, March 1992.

[177] **Finkelstein A. y Salesin D. H.,** Multiresolution curves. In Proceedings of SIGGRAPH 94, pages 261–268. ACM, New York, 1994.

[178] **Gortler S. J. y Cohen M. F.,** Hierarchical and variational geometric modeling with

wavelets. In Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 35–42. ACM, New York, 1995.

[179] **Gortler S. J., Schröder P., Cohen M. F., y Hanrahan P.,** Wavelet radiosity. In Proceedings of SIGGRAPH 93, pages 221–230. ACM, New York, 1993.

[180] **Jacobs C. E., Finkelstein A., y Salesin D. H.,** Fast multiresolution image querying. In Proceedings of SIGGRAPH 95, pages 277–286. ACM, New York, 1995.

[181] **Liu Z., Gortler S. J., y Cohen M. F.,** Hierarchical spacetime control. In Proceedings of SIGGRAPH 94, pages 35–42. ACM, New York, 1994.

[182] **Lounsbery M., DeRose T., y Warren J.**. Multiresolution surfaces of arbitrary topological type. ACM Transactions on Graphics, 1996 (to appear).

[183] **Mallat S.,** A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674–693, July 1989.

[184] **Meyers D.,** Multiresolution tiling.Computer Graphics Forum, 13(5):325–340, December 1994.

[185] **Schröder P., Gortler S. J., Cohen M. F., y Hanrahan P.,** Wavelet projections for radiosity. Computer Graphics Forum, 13(2):141–151, June 1994.

[186] **Stollnitz E. J., DeRose T. D., y Salesin D. H.,** Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann, San Francisco, 1996 (to appear).

[187] **Shen J. y Strang G.**, The zeros of the Daubechies polynomials. *Proc. Amer. Math. Soc.*, 1996.

[188] **Chamberlain N. F.**, Introduction to wavelets v1.7, December 2002.
Disponible en: http://www.sdsmt.edu/syseng/ee/courses/ee624/Wavelets1.7.pdf

[189] **Mallat S. G.**, Multiresolution approximations and wavelet orthonormal bases of L2 (R). *Transactions of the American Mathematical Society*, 315(1), pp.69-87, 1989a.

[190] **Grossman A. y Morlet J.**, Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. *SIAM J. App Math*, 15: pp.723-736, 1984.

[191] **Meyer Y.**, Wavelets: Algorithms & applications, SIAM, 1994, Philadelphia.

[192] **Valens C.**, A really friendly guide to wavelets. Disponible en:
http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html (Actualizado hasta el 12 de Abril de 2001)

[193] **Burrus C. S., Gopinath R. A. y Guo H.**, Introduction to Wavelets and Wavelet Transforms: A Primer. Prentice Hall, New Jersey, 1998.

[194] **Castro L. R. y Castro S. M.**, Wavelets y sus aplicaciones, *Primer Congreso Argentino de Ciencias de la Computación*, Univ. Nac. Del Sur, 5 al 7 de Octubre de 1995, pp.195-204.

[195] **Daubechies I.**, Different Perspectives on Wavelets. *Proceedings of Symposia in Applied Mathematics*, Vol. 47, American Mathematical Society, United State of America, 1993.

[196] **Walker J. S.**, A Primer on Wavelets and their Scientific Applications. Chapman & Hall/CRC, New York, 1999.

[197] **Daubechies I.**, Ten Lectures on Wavelets, SIAM, Philadelphia, 1992.

[198] **Walker J. S.**, Combined image compressor and denoiser based on tree-adapted wavelet shrinkage, Revision of OE 010337. James S. Walker. Department of Mathematics, University of Wisconsin-Eau Claire.
Disponible en: http://www.uwec.edu/walkerjs/media/CompDen.pdf

[199] **Stollnitz E. J., DeRose T. D., y Salesin D. H.**, Wavelets for Computer Graphics (Theory and Applications). Morgan Kaufmann Publishers, San Francisco, 1996

[200] **Kaiser G. A.**, Friendly Guide To Wavelets, Birkhäuser, Glen Allen, Virginia, 1994.

[201] **-.**, Applied and numerical harmonic analysis, Birkhäuser, C. E. D'Attellis, E. M. Fernández-Berdaguer eds., Boston, 1997.

[202] **Rao R. R. y Bopardikar A. S.**, Wavelet transforms: Introduction to theory and applications, Addison-Wesley Longman, Inc., 1998.

[203] **Hubbard B. B.**, The World According to Wavelets (The Story of a Mathema-tical Technique in the Making), A. K. Peter Wellesley, Massachusetts, 1996.

[204] **Andreopoulos Y., Munteanu A., Van der Auwera G., Cornelis J. y Schelkens P.**, Complete-to-Overcomplete discrete wavelet transforms: Theory and Applications, *IEEE Transactions on Signal Processing*, Vol. 53, No. 4, pp. 1398-1412, April 2004.

[205] Ricoh Innovations, Inc., "Compression with Reversible Embedded Wavelets (CREW) Algorithm." http://www.crc.ricoh.com/CREW/ (as of 01.10.02).

[206] **Misiti M., Misiti Y., Oppenheim G., y Poggi J. M.**. (2005, March). Wavelet Toolbox, for use with MATLAB®, User's guide, version 3. Disponible en:
http://www.mathworks.com/access/helpdesk/help/pdf_doc/wavelet/wavelet_ug.pdf

[207] **Strang G.**, Creating and comparing wavelets, *Numerical Analysis:* A. R. Mitchell Birthday Volume, G. A. Watson and D. F. Griffiths, eds., World Scien-tific, 1996.

[208] **Ying L., Ranganath S. y Zhou X.**, Comparison of wavelet and cosine basis for representation of arbitrarily shaped image segments, *Proceedings of Internatio-nal X. Conference on Telecommunications* (ICT 2002), June 2002, Beijing, China.
Disponible en: http://www.itee.uq.edu.au/~zxf/_papers/ICT02.pdf

[209] **Senecal J. G., Duchaineau M. A. y Joy K. I.**, Reversible N-Bit to N-Bit Integer Haar-Like Transforms, *Proceedings of the 7th IASTED International Conference on Computer Graphics and Imaging*, edited by M. H. Hamza. pp 135-140, August 2004. Also available as Lawrence Livermore National Labora-tory technical report UCRL-CONF-202451-REV-1. Disponible en: http://graphics.cs.ucdavis.edu/~jgseneca/Papers/TLHaar_CGIM2004.pdf

[210] **Mulcahy C.**, Plotting and scheming with wavelets, *AMS Mathematics Magazine*, Vol. 69, No. 5, pp.323-343, December 1996.

[211] **Zhong S. y Cherkassky V.**, Image denoising using wavelet thresholding and model selection, *Proc. IEEE Int. Conf. on Image processing*, Vancouver, BC, Canada, November, 2000.
Disponible en: http://www.ece.utexas.edu/~szhong/papers/denoise.pdf

[212] **Darian Muresan D. y Parks T. W.**, Adaptive principal components and image denoising, *IEEE International Conference of Image Processing*, Barcelona, Spain, pp.101-104, 2003.
Disponible en: http://dsplab.ece.cornell.edu/papers/slides/icip2003.pdf

[213] **Donoho D. L., Johnstone I. M., Kerkyacharian G. y Picard D.**, Wavelet shrin-kage: asymptopia, *Journal of Royal Stat. Soc.*, vol. 57, no.2, pp.301-369, 1995.

[214] **Donoho D. L., Johnstone I. M., Kerkyacharian G. y Picard D.**, Density estimation by wavelet thresholding, *Annals of Stat.*, vol. 24, pp. 508-539, 1996.

[215] **Donoho D. L.**, De-Noising by soft-thresholding, *IEEE Trans. on Inf. Theory*, vol. 41, no. 3, pp. 613-627, 1995.

[216] **Donoho D. L. y Johnstone I. M.**, Adapting to unknown smoothness via wavelet shrinkage, *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200-1224, 1995.

[217] **Donoho D. L. y Johnstone I. M**, Ideal spatial adaptation by wavelet shrinkage. *Biometrik*a, 81, 425-455, 1994.

[218] **Coifman R. R. y Donoho D. L.**, Translation-invariant de-noising. Antoniadis, A., & Oppenheim, G. (eds), *Lecture Notes in Statistic*s, vol. 103. Springer-Verlag, pp. 125-150, 1995.

[219] **Gao H. Y. y Bruce A. G.**, WaveShrink with firm shrinkage, *Statistica Sinic*a, 7, 855-874, 1997.

[220] **Adams N., Chen J., Olafsson V. y Yip C.**, Denoising using wavelets: Wavelets & time frequency analysis, December 1991.

[221] **Efromovich S., Lakey J., Pereyra M. C. y Tymes N.**, Data-driven and optimal denoising of a signal and recovery of its derivative using multiwavelets, *IEEE Trans. on Signal Processing*, vol. 52, no. 3, pp.1-8, 2004.

[222] **Chang S. G., Yu B., y Vetterli M.**, Adaptive wavelet thresholding for image denoising and compression, *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp.1532-1546, September 2000.

[223] **Chang S., Yu B., y Vetterli M.**, Spatially adaptive wavelet thresholding with context modelling for image denoising, *IEEE Trans. Image Processing*, vol. 9, pp. 1522-1531, Sept. 2000.

[224] **Choi H. y Baraniuk R.**, Multiscale texture segmentation using wavelet-domain hidden Markov models, in Proc. Int. Conf. Signals, Syst., Comput., vol. 2, 1998, pp. 1692-1697.

[225] **Lang M., Guo H., Odegard J., Burrus C. y Wells R.**, Noise reduction using an undecimated discrete wavelet transform, *IEEE Signal Processing Letters*, vol. 3, no. 1, pp. 10-12., 1996.

[226] **Gopinath R. A., Lang M., Guo H. y Odegard J. E.**, Enhancement of decom-pressed images at low bit rates, *Proceedings of SPIE, Mathematical Imaging: Wavelet Applications is Signal and Image Processing II* (2303-30), San Diego, CA, July 27-29, 1994. Disponible en: http://cmc.rice.edu/docs/docs/Gop1994Jul5Enhanceme.ps

[227] **Gopinath R. A., Lang M., Guo H. y Odegard J. E.**, Wavelet-based post-processing of low bit rate transform coded images, *Rice University CML Technical Report*, 1994. Disponible en: http://cmc.rice.edu/docs/docs/Gop1994Non9Wavelet-Ba.ps

[228] **Crouse M. S, Nowak R. D., y Baraniuk R. G**, Wavelet-based statistical signal processing using hidden Markov models. *IEEE Trans. Signal Processing*, vol 46, no.4, pp.886-902, April 1998.

[229] **Nowak R. D. y Baraniuk R. G.**, Wavelet-based transformations for nonlinear signal processing, IEEE Trans. on Signal Processing, Vol. 47, No. 7, pp.1852-1865, July 1999. Disponible en: http://www.ece.wisc.edu/~nowak/nst.pdf

[230] **Nowak, R. D.**, Wavelet-based Rician noise removal for magnetic resonance imaging, *IEEE Transactions on Image Processing*, Vol. 8, No. 10, pp. 1408-1419, Oct. 1999.

[231] **Nowak R. D. y Baraniuk R. G.**, Wavelet-domain filtering for photon imaging systems, *IEEE Transactions on Image Processing*, Vol. 8, No. 5, pp. 666-678, May 1999.

[232] **Taswell C.**, The what, how, and why of wavelet shrinkage denoising, *Computing in Science and Engineering*, pp.12-19, May/June 2000.

[233] **Abramovich F. y Benjamini Y.**, Adaptive thresholding of wavelet coefficients, *Comput. Statist. Data Anal.*, vol. 22, pp. 351-361, 1996.

[234] **Abramovich F., Sapatinas T. y Silverman B.**, Wavelet thresholding via a Bayesian approach, *J. R. Stat.*, vol. 60, pp. 725-749, 1998.

[235] **Cai Z., Cheng T. H., Lu C. y Subramanian K. R.**, Efficient wavelet based image denoising algorithm, *Electron Lett.*, vol. 37, no. 11, pp. 683-685, May 2001.

[236] **Sylvain D y Nikolova M.**, Restoration of wavelet coefficients by minimizing a specially designed objective function, in *Proc. of IEEE Workshop on Var. and Level Set Meth. in Comp. Vision*, 2003.
Disponible en: http://www.cmla.ens-cachan.fr/Utilisateurs/nikolova/VLSM03.pdf

[237] **Wan S, Raju B. I. y Srinivasan M. A.**, Robust deconvolution of high-frequency ultrasound images using higher-order spectral analysis and wavelets, *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, Vol.50, No.10, pp.1286-1295, October 2003.

[238] **Rangarajan R., Venkataramanan R. y Shah S.**, Image denoising using wavelets: Wavelets & time frequency, December 16, 2002.
Disponible en: http://www-personal.engin.umich.edu/~rvenkata/551_code/Report.pdf

[239] **Pólchlopek W. y Ziólko M.**, Wavelet transform compression and denoising in real-time system, *Proceedings of third international symposium on Communi-cation Systems Networks and Digital Signal Processing*, CSNDSP 2002: 15th-17th July 2002 Staffordshire, UK / ed. R. A. Carrasco. Staffordshire: Sheffield Hallam University Press Learning Centre, pp. 288-291 2002. Disponible en:
http://www.scit.wlv.ac.uk/~jphb/cp4040/rolandonotes/CSNDSP2002/Papers/H2/H2.1.pdf

[240] **Calderbank A. R., Daubechies I., Sweldens W. y Yeo B.-L.**, Wavelet thransforms that map integers to integers, August 1996.
Disponible en: http://cm.bell-labs.com/who/wim/papers/integer.pdf

[241] **Thierschmann M., Martin U. y Rösel R.**, New perspectives on image compression, *Photogrammetric Week '97*, Weichmann Verlag, Heidelberg, pp.189-199, 1997.

[242] **Shapiro J. M.**, Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp.3445-3462, 1993.

[243] **Creusere C. D.**, A new method of robust image compression based on the embedded zerotree wavelet algorithm, *IEEE Transactions on Image Processing*, Vol. 6, No. 10, pp.1436-1442, 1997.

[244] **Algazi V. R. y Estes R. R.**, Analysis based coding of image transform and subband coefficients, *Proceedings of the SPIE*, Vol. 2564, pp. 11-21, 1995.

[245] **Jia W., He X. y Lin Q.**, Echocardiography sequential images compression based on region of interest, *Proceedings of the 2nd International Conference on Information Technology for Application* (ICITA 2004), pp.232-237, 2004. Disponible en:
http://attend.it.uts.edu.au/icita05/CDROM-ICITA04/papers/IC-125.pdf

[246] **Kopp M.**, Lossless Wavelet Based Image Compression with Adaptive 2D Decompo-sition, *Proceedings of the Fourth International Conference in Central Europe on Computer Graphics and Visualization 96* (WSCG96), pages 141-149,edited by Nadia M. Thalmann and Vaclav Skala, Plzen, Czech Republic, February 12-16, 1996.

[247] **Paz Viera J. E.**, Aumento de la tasa de compresión y de la relación señal-ruido en imágenes ruidosas por la combinación de algoritmos de compresión y filtrado,

Informedica 2002: Preparando el Camino para la e-Salud Global, 2do Congreso Virtual Iberoamericano de Informática Médica, Nov. 4-Nov. 30, 2002. Disponible en: http://www.informedicajournal.org/a1n2/files/papers_informedica/paz.pdf

[248] **Kontoyiannis I., Turek J., Castelli V. y Robinson J.**, Multiresolution lossless image compression. Unpublished manuscript, December 1995.
Disponible en: http://www.dam.brown.edu/people/yiannis/PAPERS/MCIA.pdf

[249] **Popat K.**, Lossy Compression of Grayscale Document Images by Adaptive-Offset Quantization, *Proceedings of IS&T/SPIE Electronic Imaging 2001: Do-cument Recognition and Retrieval VIII*, January 2001.
Disponible en: http://www2.parc.com/istl/members/popat/pdf/Popat2001_offset.pdf

[250] **Brooks S. y Tabbara M.**, Wavelet Image Compression: A Heuristic Approach to Space Frequency Quantisation, *Apple University Consortium Conference*, University of Adelaide, 28 September – 1 October, 2003. Disponible en:
http://auc.uow.edu.au/conf/conf03/papers/AUC_DV2003_Tabbara.pdf

[251] **David B. Gerhard y W. Kinsner**, Lossy Compression of Head and Shoulder Images Using Zerotrees of Wavelet Coefficients, *IEEE Canadian Conference on Electrical and Computer Engineering*, Calgary, Alberta. I: 433-437, 1996.
Disponible en: http://www2.cs.uregina.ca/~gerhard/publications/calgary.pdf

[252] **Marcellin M. W., Gormish M. J., Bilgin A. y Boliek M. P.**, An Overview of JPEG-2000, *Proc. Data Compression Conference*, J. A. Storer and M. Cohn, eds., Snowbird, Utah, Mar. 28-Mar. 30, 2000, p. 523-541. Disponible en:
http://www.rii.ricoh.com/%7egormish/pdf/dcc2000_jpeg2000_note.pdf

[253] **Van Otterloo S.**, Amazing Wavelet Image Compression, Utrecht, 2000.
Disponible en: http://www.bluering.nl/sieuwert/research/wavcomp.doc

[254] **Vargic R.**, An Approach to 2D Wavelet Transform and its Use for Image Compression, *Radioengineering*, Vol. 7, No. 4, December 1998.
Disponible en: http://www.ktl.elf.stuba.sk/~vargic/papers/radio98/re98_cr.doc

[255] **Amaratunga K.**, A fast wavelet algorithm for the reduction of color information. Disponible en:
http://citeseer.ist.psu.edu/rd/21750013%2C479595%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/23491/http:zSzzSzwavelets.mit.eduzSzKevinzSzPaperszSzColorQuant.pdf/a-fast-wavelet-algorithm.pdf

[256] **Almeida R., Martinez J. P., Olmos S., Rocha A. P. y Laguan P.**, Automatic delineation of T and P waves using a wavelet-based multiscale approach, *International Congress on Computational Bioengineering (ICCB'03).* pp. 243-247. Zaragoza, 2003. Disponible en:
http://diec.unizar.es/intranet/articulos/uploads/%20Automatic%20delineation%20of%20T%20and%20P%20waves%20using%20a%20wavelet-ased%20multiscale%20approach.pdf

[257] **Cuesta Frau D., Pérez Cortés J. C., Andréu García G., Eck V. y Sastre Mengual C.**, Reducción del ruido en señales electrocardiográficas mediante la transformada wavelet, *Conferencia Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB-2000)*

Cartagena, Spain. September 2000.
Disponible en: http://dcuesta.alc.upv.es/Investigacio/CASEIB2000.pdf

[258] **Cuesta Frau D., Pérez Cortés J. C., Andréu García G. y Novák D.**, Feature extraction methods applied to the clustering of electrocardiographic signals. A comparative study, *International Conference on Pattern Recognition (ICPR-2002)* Quebec, Canada. August 2002.

[259] **Mathias A., Grond F., Guardans R., Seese D., Canela M. y Diebner H. H.**, Algorithms for spectral analysis of irregularly sampled time series, *Journal of Statistical Software*, Vol. 11, No. 2, pp. 1-27, May 2004.

[260] **Paquet A. H., Zahir S. y Ward R. K.**, Wavelet packets-based image retrieval, *IEEE-ICASSP 2002*, Orlando, Florida, May 2002.
Disponible en: http://www.ece.ubc.ca/~alexp/PaperICASSP2002.pdf

[261] **Chandroth G., Sharkey A.J.C. y Sharkey N.E.**, Vibration signatures, wavelets and principal components analysis in diesel engine diagnostics, In: *Marine Technology 99*, Poland, 1999.

[262] **Quian Quiroga R. y García H.**, Single-trial event-related potentials with wavelet denoising, *Clinical Neurophysiology*, 114, pp.376-390, 2003.

[263] **Oweiss K. G. y Anderson D. J.**, Noise reduction in multichannel neural recordings using a new array wavelet denoising algorithm, *Neurocomputing* 38-40, pp.1687-1693, 2001.

[264] **Cárdenas-Barrera J. L., Lorenzo-Ginori J. V. y Rodríguez-Valdivia E.**, A wavelet-packets based algorithm for EEG signal compression, *Medical Informatics*, Vol.29,No.1, pp.15-27, March 2004.

[265] **Mojsilovic A., Popovic M. V., Neskovic A. N. y Popovic A. D.**, Wavelet image extension for analysis and classification of infracted myocardial tissue, *IEEE Trans. on Biomedical Engineering*, Vol.44, No.9, pp.856866, September 1997.

[266] **Angelini E., Laine A., Takuma S. y Homma S.**, Spatio-temporal directional analysis of 4D echocardiography, *Proceedings of SPIE- 45th Annual Meeting*, San Diego CA, pp. 605-614, 2000

[267] **Kolmogorov A. N. y Fomin S. V.**, Elementos de la Teoría de Funciones y del Análisis Funcional, MIR, 1975, Moscú.

[268] **Lawson B. L., Orrison M. E. y Uminsky D. T.**, Spectral analysis of the supreme court.
Disponible en: http://www.math.hmc.edu/~orrison/research/papers/spectral.pdf

[269] **Azar Y., Fiat A., Karlin A. R., McSherry F. y Saia J.**, Spectral analysis of data, *ACM Symposium on Theory of Computing*, pp. 619-626, 2001. Disponible en: http://citeseer.ist.psu.edu/rd/0%2C440568%2C1%2C0.25%2CDownload/http%3AqSqqSqwww.cs.washington.eduqSqhomesqSqsaiaqSqspectral-stoc.ps

[270] **Chan K. W. y So. H. C.**, Accurate frequency estimation for real harmonic sinusoids, *IEEE Signal Processing Letters*, Vol. 11, No. 7, pp. 609-612, July 2004.

[271] **Moody G. B.**, Spectral Analysis of heart rate without resampling, *Computers in Cardiology 1993*, IEEE Computer Society Press, vol. 20, pp. 715-718.
Disponible en: http://www.physionet.org/physiotools/lomb/lomb.pdf

[272] **Barbosa L., Kleisinger G. H., de Valle E. E. y Monzón J. E.**, Implementación de filtros digitales con sistemas computacionales interactivos.
Disponible en: http://www.unne.edu.ar/cyt/2000/8_exactas/e_pdf/e_018.pdf

[273] **Pisarello M. I., Kleisinger G. H. y Monzón J. E.**, Procesamiento digital del ECG con bancos de filtros.
Disponible en: http://www.unne.edu.ar/cyt/2001/8-Exactas/E-006.pdf

[274] **Díaz Calavia E. J., Elizalde Soba P., Berraondo López P. y Tejeira Alvarez J. M.**, Elección de filtros para limpiar el ECG, *XII Física Estadística, FisEs'03*, 23-25 Octubre, Pamplona, Spain, 2003. Disponible en:
http://www.unav.es/ict/empresas/docs/MemoI+D%200304/MemoI+Da%20CCNat.pdf

[275] **Motwani M. C., Gadiya M. C. y Motwani R. C.**, Survey of Image Denoising Techniques, in *Proceedings of GSPx 2004*, September 27-30, 2004, Santa Clara Convention Center, Santa Clara, CA.

[276] **Cernuschi Frias B. y Belausteguigoitía C. F.**, Visión para Computadoras, *EBAI*, 1988.

[277] **Mastriani M.**, Nuevas Técnicas Computacionales para el Mejoramiento de Imágenes Médicas, *31 Jornadas Argentinas de Informática e Investigación Operativa*, Santa Fe, Argentina, 9 al 13 de Septiembre de 2002, pp.262-267.

[278] **Mastriani M.**, Imágenes Médicas sobre TCP/IP y UDP: Su problemática de implementación, *INFORMEDICA'2002 - e-Salud Global*, 30 de Octubre al 30 de Noviembre de 2002.

[279] **Mastriani M.**, New wavelet-based superresolution algorithm for speckle reduction in SAR images, International Journal of Computer Science, Volume 1, Number 4, pp.291-298, 2006.

[280] **Mastriani M. y Giraldez A. E.**, Despeckling of SAR images in wavelet domain, GIS Development Magazine, Sept. 2005, Vol. 9, Issue 9, pp.38-40. Disponible en: http://www.gisdevelopment.net/magazine/years/2005/sep/wavelet_1.htm

[281] **Mastriani M. y Giraldez A. E.**, Kalman's shrinkage for wavelet-based despeckling of SAR images, International Journal of Intelligent Technology, Volume 1, Number 3, pp.190-196, 2006. Disponible en: http://www.enformatika.org/jit/v1/v1-3-21.pdf

[282] **Mastriani M. y Giraldez A.**, Enhanced Directional Smoothing Algorithm for Edge-Preserving Smoothing of Synthetic-Aperture Radar Images, *International Journal of Measurement Science Review*, vol 4, no. 3, pp.1-11, 2004. Disponible en:

http://www.measurement.sk/2004/S3/Mastriani.pdf

[283] **Mastriani M. *et al***, Sistema Inteligente de Adquisición de Señales Biomédicas y su Vinculación con una Historia Clínica Electrónica, *33 Jornadas Argentinas de Informática e Investigación Operativa*, Córdoba, Argentina, 22 al 24 de Septiembre de 2004.

[284] **Mastriani M. *et al***, Evaluation of Children's Anthropometric Features Using Wavelet Decomposition, *9th World Congress of Medicine in Internet (MEDNET 2004) of Society for the Internet in Medicine (SIM)*, (8113), Buenos Aires, Argentina, December 5-7, 2004.

[285] **Jones H. W., Hein D. N. y Knauer S. C.**, "The Karhunen-Loeve, discrete cosine and related transforms via the Hadamard transform," Proc. Intl. Telemeter. Conf., pp. 87-98, Los Angeles, CA, Nov. 1978.

[286] **Watson A. B. and Poirson A.**, Separable two-dimensional discrete Hartley transform, J. Opt. Soc. Am. A, Vol.3; No.12, pp.2001-2004, Decembre 1986. Disponible en: http://vision.arc.nasa.gov/publications/HartleyTransform.pdf

[287] **Khayam S. A.**, The Discrete Cosine Transform (DCT): Theory and Application Technical Report, WAVES-TR-ECE802.602, 2003. Revised February 2005. Disponible en: http://www.niit.edu.pk/~khayam/pdf/DCT_TR802.pdf

[288] **Pennebaker W. B. y Mitchell J. L.**, "JPEG – Still Image Data Compression Standard," New York: International Thomsan Publishing, 1993.

[289] **Strang G.**, "The Discrete Cosine Transform," SIAM Review, Volume 41, Number 1, pp.135-147, 1999.

[290] **Clark R. J.**, "Transform Coding of Images," New York: Academic Press, 1985.

[291] **Radha H.**, "Lecture Notes: ECE 802 - Information Theory and Coding," January 2003.

[292] **Hung A. C. y Meng TH-Y**, "A Comparison of fast DCT algorithms," Multimedia Systems, No. 5 Vol. 2, Dec 1994.

[293] **Aggarwal G. y Gajski D. D.**, "Exploring DCT Implementations," UC Irvine, Technical Report ICS-TR-98-10, March 1998.

[294] **Blinn J. F.**, "What's the Deal with the DCT," IEEE Computer Graphics and Applications, July 1993, pp.78-83.

[295] **Chiu C. T. y Liu K. J. R.**, "Real-Time Parallel and Fully Pipelined 2-D DCT Lattice Structures with Application to HDTV Systems," IEEE Trans. on Circuits and Systems for Video Technology, vol. 2 pp. 25-37, March 1992.

[296] **Haque M. A.**, "A Two-Dimensional Fast Cosine Transform," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-33 pp. 1532-1539, December 1985.

[297] **Vetterli M.**, “Fast 2-D Discrete Cosine Transform,” ICASSP '85, p. 1538.

[298] **Kamangar F. A. y Rao K. R.**, “Fast Algorithms for the 2-D Discrete Cosine Transform,” IEEE Transactions on Computers, v C-31 p. 899.

[299] **Linzer E. N. y Feig E.**, “New Scaled DCT Algorithms for Fused Multiply/Add Architectures,” ICASSP '91, p. 2201.

[300] **Loeffler C., Ligtenberg A., y Moschytz G.**, “Practical Fast 1-D DCT Algorithms with 11 Multiplications,” ICASSP '89, p. 988.

[301] **Vetterli M., Duhamel P., y Guillemot C.**, “Trade-offs in the Computation of Mono- and Multi-dimensional DCTs,” ICASSP '89, p. 999.

[302] **Duhamel P., Guillemot C., y Carlach J. C.**, “A DCT Chip based on a new Structured and Computationally Efficient DCT Algorithm,” ICCAS '90, p. 77.

[303] **Cho N. I. y Lee S. U.**, “Fast Algorithm and Implementation of 2-D DCT,” IEEE Transactions on Circuits and Systems, vol. 38 p. 297, March 1991.

[304] **Cho N. I., Yun I. D., y Lee S. U.**, “A Fast Algorithm for 2-D DCT,” ICASSP '91, p. 2197-2220.

[305] **McMillan L. y Westover L.**, “A Forward-Mapping Realization of the Inverse DCT,” DCC '92, p. 219.

[306] **Duhamel P. y Guillemot C.**, “Polynomial Transform Computation of the 2-D DCT,” ICASSP '90, p. 1515.

[307] **Jain A. K.**, “Fundamentals of Digital Image Processing,” New Jersey: Prentice Hall Inc., 1989.

[308] **Castrillon-Candas J. E. y Amaratunga K.**, “Fast Estimation of Karhunen-Loeve Eigen Function using Wavelets,” Submitted to IEEE Transactions on Signal Processing, 2001.

[309] **Jain A.**, “A Fast Karhunen-loeve Transform for Digital Restoration of Images Degraded by White and Colored Noise,” IEEE Transactions on Computers, vol. 26, number 6, June 1977.

[310] **Gonzalez R. C. y Wintz P.**, “Digital Image Processing,” Reading. MA: Addison-Wesley, 1977.

[311] **Mastriani M.**, Denoising based on wavelets and deblurring via self-organizing map for Synthetic Aperture Radar images, International Journal of Signal Processing, Volume 2, Number 4, pp.226-235, 2005. Disponible en: http://www.enformatika.org/ijsp/v2/v2-4-33.pdf

[312] **Briggs W. L. y Van Emden H.**, The DFT: An Owner’s Manual for the Discrete Fourier Transform, SIAM, 1995, Philadelphia.

[313] **Ras Oliva E.**, Análisis de Fourier y Cálculo Operacional Aplicados a la Electro-tecnia, Marcombo, 1979, Barcelona.

[314] **Echebest N. y Liberman E.**, Resolución numérica de ecuaciones diferenciales parciales por aplicación de la transformada rápida de Fourier, con estimación de errores, Métodos Numéricos en la Mecánica del Continuo, Marshall G. ed., pp. 59-69, 1978, EUDEBA, Buenos Aires.

[315] **Hernández Lerma O.**, Métodos de Fourier en la Física y la Ingeniería, Trillas, 1973, México.

[316] **Hsu H. P.**, Fourier Analysis, Simon & Schuster, 1970, New York.

[317] **Tolimieri R., An M. y Lu C.**, Algorithms for Discrete Fourier Transform and convolution, Springer Verlag, 1997, New York.

[318] **Tolimieri R., An M. y Lu C.**, Mathematics of multidimensional Fourier Transform Algorithms, Springer Verlag, 1997, New York.

[319] **Claveau F. y Poirier M.**, "Real time FFT based cross-covariance method for vehicle speed and length measurement using an optical sensor," ICSPAT 96, pp.1831-1835, Boston, MA, Oct.1996.

[320] **Clarke R. J.**, "Relation between the Karhunen-Loeve and sine transforms," Electron. Lett., Vol. 20, pp. 12-13, Jan. 1984.

[321] **Mastriani M.**, "Denoising based on wavelets and deblurring via self- organizing map for Synthetic Aperture Radar images," ICGST International on Journal of Artificial Intelligence and Machine Learning, Volume 5, 2005.

[322] **Pentland A. P., Pickard R. W. y Scarloff S.**, "Photobook: tools for content based manipulation of image databases," SPIE, Vol. 2185, pp. 34-47, San Jose, CA, 1994 (storage and retrieval for image and video databases)

[323] **Wien M.**, Variable Block-Size Transforms for Hybrid Video Coding, Degree Thesis, Institut für Nachrichtentechnik der Rheinisch-Westfälischen Technischen Hchschule Aachen, February 2004.
Disponible en: http://www.ient.rwth-aachen.de/team/wien/wien_diss.pdf

[324] **Ng R. y Sedighian A.**, "Evaluating multi-dimensional indexing structures for images transformed by principal component analysis," In Proc. SPIE Storage and Retrieval for Image and Video Databases, 1996.

[325] **Sahouria E. y Zakhor A.**," Content analysis of video using principal components," IEEE ICIP, pp. WP1.06, Chicago, IL, Oct. 1998.

[326] **Chang C., Maciejewski A. A. y Balakrishnan V.**, "Eigen-decomposition-based analysis of video images, "Photonics West, SPIE, vol. 3656, San Jose, CA, Jan. 1999.

[327] **Han K. J. y Tewfik A. H.**, "Eigen-image based video segmentation and indexing," ICIP 97, (Vol. II), pp. 538-541, Santa Barbara, CA, OCT. 1997.

[328] **Donoho D. L., y Johnstone I. M.**, "Adapting to unknown smoothness via wavelet shrinkage," Journal of the American Statistical Assoc., vol. 90, no. 432, pp. 1200-1224., 1995.

[329] **Donoho D. L., y Johnstone I. M.**, "Ideal spatial adaptation by wavelet shrinkage," Biometrika, 81, 425-455, 1994.

[330] **Daubechies I.**, "Different Perspectives on Wavelets," in Proceedings of Symposia in Applied Mathematics, vol. 47, American Mathematical Society, USA, 1993.

[331] **Mallat S. G.**, "A theory for multiresolution signal decomposition: The wavelet representation," IEEE Trans. Pattern Anal. Machine Intell., vol. 11, pp. 674–693, July 1989.

[332] **Mallat S. G.**, "Multiresolution approximations and wavelet orthonormal bases of L2 (R)," Transactions of the American Mathematical Society, 315(1), pp.69-87, 1989a.

[333] **Chang S. G., Yu B., y Vetterli M.**, "Adaptive wavelet thresholding for image denoising and compression," IEEE Trans. Image Processing, vol. 9, pp. 1532–1546, Sept. 2000.

[334] **Misiti M., Misiti Y., Oppenheim G., y Poggi J. M.,** (2001, June). Wavelet Toolbox, for use with MATLAB®, User's guide, version 2.1.

[335] **Burrus C. S., Gopinath R. A., y Guo H.**, Introduction to Wavelets and Wavelet Transforms: A Primer, Prentice Hall, New Jersey, 1998.

[336] **Hubbard B. B.**, The World According to Wavelets: The Story of a Mathematical Technique in the Making, A. K. Peter Wellesley, Massachusetts, 1996.

[337] **Grossman A. y Morlet J.**, "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape," SIAM J. App Math, 15: pp.723-736, 1984.

[338] **Walker J. S.**, A Primer on Wavelets and their Scientific Applications, Chapman & Hall/CRC, New York, 1999.

[339] **Stollnitz E. J., DeRose T. D., y Salesin D. H.**, Wavelets for Computer Graphics: Theory and Applications, Morgan Kaufmann Publishers, San Francisco, 1996.

[340] **Mastriani M., y Giraldez A.**, "Smoothing of coefficients in wavelet domain for speckle reduction in Synthetic Aperture Radar images," ICGST International Journal on Graphics, Vision and Image Processing (GVIP), Volume 6, pp. 1-8, 2005. Disponible en: http://www.icgst.com/gvip/v6/P1150517003.pdf

[341] **Mastriani M.**, "Systholic Boolean Orthonormalizer Network in Wavelet Domain for Microarray Denoising," ICGST International Journal on Bioinformatics and Medical Engineering, Volume 5, 2005.

[342] **Mastriani M.**, "Systholic Boolean Orthonormalizer Network in Wavelet Domain for Microarray Denoising," International Journal of Signal Processing, Volume 2, Number 4, pp.273-284, 2005.

[343] **Mastriani M., y Giraldez A.**, "Microarrays denoising via smoothing of coefficients in wavelet domain," International Journal of Biomedical Sciences, Volume 1, Number 1, pp.7-14, 2006.

[344] **Mastriani M., y Giraldez A.**, "Kalman' Shrinkage for Wavelet-Based Despeckling of SAR Images," International Journal of Intelligent Technology, Volume 1, Number 3, pp.190-196, 2006.

[345] **Mastriani M., y Giraldez A.**, "Neural Shrinkage for Wavelet-Based SAR Despeckling," International Journal of Intelligent Technology, Volume 1, Number 3, pp.211-222, 2006.

[346] **Mastriani M.**, "Fuzzy Thresholding in Wavelet Domain for Speckle Reduction in Synthetic Aperture Radar Images," International Journal of Intelligent Technology, Volume 1, Number 3, pp.252-265, 2006.

[347] **Abadir K. M. y Magnus J. R.**, Matrix Algebra, Cambridge University Press, New York, 2005.

[348] **Roman S.**, Advanced Linear Algebra, Third Edition, New York, 2008.

[349] **Meyer C. D.**, Matrix Analysis and Applied Linear Algebra, Siam Press, Philadelfia, 2000.

[350] **Sacchi M. D.**, Statistical and Transform Methods for Geophysical Signal Processing, Chapter 6 KL and Eigen-images, 2001. Disponible en: http://www-geo.phys.ualberta.ca/saig/book/chapter6.pdf

[351] **Freire S. L. M. y Ulrych T. J.**, Application of singular value decomposition to vertical seismic profiling, Geophysics. Vol.53, No.6 (June 1988): pp.778-785

[352] **Cagnoli B., y Ulrych T. J.**, Singular value decomposition and wavy reflections in ground-penetrating radar images of base surge deposits, Journal of Applied Geophysics 48 (2001) 175–182.

[353] **Orfanidis S. J.**, SVD, PCA, KLT, CCA, and All That, Rutgers University, Electrical & Computer Engineering Department, Optimum Signal Processing, 2002–2007. Disponible en: http://www.ece.rutgers.edu/~orfanidi/ece525/svd.pdf

[354] **Winslow P.**, Singular Value Decomposition for Path Reconstruction in High Purity Germanium Detectors for TIGRESS, August 23, 2007. Disponible en: https://gamma.triumf.ca/misc/Student/Peter%20Winslow/SVDReport.pdf/view

[355] **Hoffman K. y Kunze R.**, Álgebra Lineal, Prentice-Hall Hispanoamericana, 1973, México.

[356] **Lang S.**, Introducción al Algebra Lineal, Addison-Wesley Iberoamericana, 1990,

Wilmington, Delaware.

[357] **Leon S. J.**, Linear Algebra with Applications, MacMillan, 1990, New York.

[358] **Fraleigh J. B.**, Algebra Abstracta, Addison-Wesley Iberoamericana, 1987, México.

[359] **Noble B. y Daniel J. W.**, Algebra Lineal Aplicada, Prentice-Hall, 1989, México.

[360] **Marcus M. y Minc H.**, Álgebra Lineal, CECSA, 1969, México.

[361] **Birkhoff G. y MacLane S.**, Algebra Moderna, Vicens-Vives, 1980, Barcelona.

[362] **Grossman S. I.**, Algebra Lineal, McGraw-Hill, 2004, México.

[363] **Nicholson W. K.**, Algebra Lineal con aplicaciones, McGraw-Hill, 2004, México.

[364] **Strang G.**, Álgebra Lineal y sus Aplicaciones, Addison-Wesley Iberoamericana, 1990, México.

[365] **Herstein I. N. y Winter D. J.**, A primer on Linear Algebra, MacMillan, 1990, New York.

[366] **Santalo L. A.**, Vectores y Tensores con sus aplicaciones, EUDEBA, 1981, Buenos Aires.

[367] **Máltsev A. I.**, Fundamentos de Álgebra Lineal, MIR, 1978, Moscú.

[368] **Dubreil P. y Dubreil-Jacotin M. L.**, Lecciones de Algebra Moderna, Reverté, 1975, Barcelona.

[369] **Thompson R. C. y Taqub A.**, Introducción al Álgebra Abstracta y Lineal, Uteha, 1976, México.

[370] **Mastriani M.**, An Application for the Symmetric Functions, *Fifth Society Industrial and Applied Mathematics (SIAM) Conference on Applied Linear Algebra*, Snowbird, Utah, June 15-18, 1994.

[371] **Schalkoff R. J.**, Digital Image Processing and Computer Vision, Wiley, 1989, New York.

[372] **Pajares G. y de la Cruz J. M.**, Visión por computador, Alfaomega-Ra-Ma, 2002, México D. F.

[373] **Pajares G., de la Cruz J. M., Molina J. M., Cuadrado J. y López A.**, Imágenes Digitales: Procesamiento práctico con Java, Alfaomega-Ra-Ma, 2004, México D. F.

[374] **Faúndez Zanuy M.**, Tratamiento digital de voz e imagen y aplicación a la Multimedia, Marcombo, 2000, Barcelona.

[375] **Teuber J.**, Digital Image Processing, Prentice-Hall, 1993, New York.

[376] **Mariño Acebal J. B., Vallverdú Bayés F., Rodríguez Fonollosa J. A. y Moreno Bilbao A.**, Tratamiento digital de la señal, Alfaomega-Edicions UPC, 1999, Barcelona, España.

[377] **Proakis J. G. y Manolakis D. G.**, Tratamiento digital de señales: Principios, algoritmos y aplicaciones, Prentice-Hall, 1998, Madrid.

[378] **Proakis J. G. y Manolakis D. G.**, Introduction to Digital Signal Processing, MacMillan, 1989, New York.

[379] **Kamen E. W.**, Introduction to signals and systems, MacMillan Publishing Company, 1990, New York.

[380] **Kay S. M.**, Fundamentals of Statistical Signal Processing: Estimation Theory, New Jersey, Prentice-Hall, Inc., 1993.

[381] **Oppenheim A. V., Willsky A. S. y Hamid Nawab S.**, Señales y Sistemas, Prentice-Hall, 1998, México.

[382] **Soliman S. S. y Srinath M. D.**, Señales y Sistemas contínuos y discretos, Prentice-Hall, 1999, Madrid.

[383] **Martucci S. A.**, Symmetric convolution and the Discrete Sine and Cosine Transforms: Principles and applications, Doctoral Disertation, Georgia Institute of Technology, May 1993.

[384] **Haykin S.**, An introduction to analog & digital Communications, Wiley, 1989, New York.

[385] **Haykin S. y Van Veen B.**, Señales y Sistemas, Limusa Wiley, 2003, México.

[386] **Haykin S.**, Digital communications, Wiley, 1988, New York.

[387] **Burrus C. S., McClellan J. H., Oppenheim A. V., Parks T. W., Schafer R. W. y Schuessler H. W.**, Ejercicios de Tratamiento de la señal utilizando MATLAB® v.4, Prentice-Hall, 1998, Madrid.

[388] **Papoulis A.**, Sistemas digitales y analógicos, transformadas de Fourier, estimación espectral, Marcombo, 1978, Barcelona.

[389] **Albardar A.**, Procesamiento de señales analógicas y digitales, Thomson Learning, 2002, México.

[390] **Oppenheim A. V. y Schafer R. W.**, Digital Signal Processing, Prentice-Hall, 1975, New York.

[391] **Oppenheim A. V., Schafer R. W. y Buck J. R.**, Tratamiento de señales en tiempo discreto, Prentice-Hall, 2000, Madrid.

[392] **_.**, Adaptive Signal Processing, Sibul L. H. ed., IEEE Press, 1987, New York.

[393] **Couch II L. W.**, Sistemas de Comunicación Digitales y Analógicos, Prentice-Hall, 1998, México.

[394] -., E. F. Deprettere, ed., SVD and Signal Processing, North-Holland, New York, 1988.

[395] -., R. J. Vaccaro, ed., SVD and Signal Processing II, Elsevier, New York, 1991.

[396] **Moonen M. y de Moor B.**, SVD and Signal Processing III, Elsevier, New York, 1995.

[397] **_.**, (2005, March). Image Processing Toolbox, for use with MATLAB®, User's guide, version 5. Disponible en: http://www.mathworks.com/access/helpdesk/help/pdf_doc/images/images_tb.pdf

[398] **_.**, (2005, March). Signal Processing Toolbox, for use with MATLAB®, User's guide, version 6. Disponible en: http://www.mathworks.com/access/helpdesk/help/pdf_doc/signal/signal_tb.pdf

[399] **Samet H.**, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.

[400] **Samet H.**, Applications of Spatial Data Structures – Computer Graphics, Image Processing and GIS, Addison-Wesley, 1990.

[401] **Arnold B.**, An Investigation into using Singular Value Decomposition as a method of Image Compression, University of Canterbury Department of Mathematics and Statistics, September 2000. Disponible en: http://math.ucalgary.ca/~laf/teaching/Material/Arnold.pdf

[402] **Tjoa S. et al**, "Transform coder classification for digital image forensics". Disponible en: http://www.cspl.umd.edu/sig/publications/tjoa_ICIP_200709.pdf

[403] **Goyal V**. (2001). Theoretical foundations of transform coding. IEEE Signal Processing Magazine, 18(5), 9–21.

[404] **Gray R. M., y Neuhoff D. L.** (1998). Quantization. IEEE Transactions on Information Theory, 44(6), 2325–2383.

[405] **Huhle B.**, Kernel PCA for Image Compression. Ph.D., Wilhelm-Schickard-Institut für Informatik Eberhard Karls Universität Tübingen 9. April 2006. Disponible en: http://www.gris.uni-tuebingen.de/people/staff/huhle/publications/huhle-2006-thesis.pdf

[406] **Clarke R. J.**, "Transform coding of images," Orlando, FL: Academic Press, 1985.

[407] **Saha S.**, "Image Compression - from DCT to Wavalets: A Review," *ACM Cross- roads Student Magazine*, vol. 6.3, Spring 2000. http://www.acm.org/crossroads/ Disponible en: http://www.upatras.gr/ieee/skodras/pubs/ans-c37.pdf

[408] **Unser M.**, "Wavelets, filterbanks, and the Karhunen-Loeve transform,"EUSIPCO-98, vol. 3, pp. 1737-1741, Island of Rhodes, Greece, Sept.1998.

[409] **Zhang J. y Walter G.**, "A wavelet based KL-like expansion for wide sense stationary random processes," IEEE Trans. SP, Vol. 42, pp. 1737-1745, July 1994.

[410] **Wornell G. W.**, "A Karhunen-Loeve-like expansion for 1/f processes via wavelets," IEEE Trans. IT, Vol. 36, pp. 859-861, July 1990.

[411] **Hall J. y Crowe J.**, "Ambulatory electrocardiogram compression using wavelet packets to approximate Karhunen-Loeve transform," Int. J. Applied SP, Vol. 3, pp. 25-36, 1996.

[412] **Starck J.-L., y Querre P.**, "Multispectral Data Restoration by the Wavelet-Karhunen-Loève Transform," Preprint submitted to Elsevier Preprint, 2000, pp. 1–29.

[413] **Waldemar P. y Ramstad T.**, "Hybrid KLT-SVD image compression," ICASSP 97, Vol. 4, pp. 2713-2716, Munich, Germany, April 1997.

[414] **Gerbrands J. J.**, "On the Relationships Between SVD, KLT, and PCA," Patt. Recogn., **14**, 375 (1981).

[415] ISO/IEC 15444-1:2000. Information technology – JPEG 2000 image coding system – Part 1: Core coding system, 2000.

[416] **Ruiz V. G.**, Compresión Reversible y Transmisión de Imágenes, Universidad de Almería, Departamento de Arquitectura de Computadores y Electrónica, Tesis Doctoral, Julio 2000. Disponible en: http://www.ace.ual.es/~vruiz/investigacion/tesis.pdf

[417] **Huffman D. A.,** A Method for the Construction of Minimum Redundancy Codes. Proceedings of the Institute of Radio Engineers, 40:1098-1101, 1952.

[418] **Cleary J. G. y Witten I. H.,** Data Compression using Adaptive Coding and Partial String Matching. IEEE Transactions on Communications, 4(32):396-402, 1984.

[419] **Howard P. G. y Vitter J. S.,** Analysis of Arithmetic Coding for Data Compression. Information Processing and Management, 28(6):749-763, 1992.

[420] **Weinberger M. J., Rissanen J., y Arps R. B.,** Applications of the Universal Context Modeling to Lossless Compression of Gray-Scale Images. IEEE Transactions on Image Processing, 5(4):575-586, 1996.

[421] **Knuth D. E.,** Dynamic Huffman Coding. Journal of Algorithms, 6(2):163-180, 1985.

[422] **Nelson M. y Gailly J.,** The Data Compression Book. M&T Books, 1996.

[423] **Ruiz V. G. y García I.,** Compresión de Texto basada en un Modelo Probabilística de Orden 0 y un Codificador de Huffman. Technical Report 1, Depto de Arquitectura de Computadores y Electrónica, Universidad de Almería, 1998.

[424] **Vitter J. S.,** Design and Analysis of Dynamic Huffman Codes. ACM, (4):825-845, 1987.

[425] **Abramson N.,** Information Theory and Coding. New York, McGraw-Hill, 1963.

[426] **Pasco R.,** Source Coding Algorithms for Fast Data Compression. PhD thesis, Stanford University, 1976.

[427] **Langdon G. G. y Rissanen J.,** Compression of Black-White Images with Arithmetic Coding. IEEE Transactions on Communications, 29(6):858-867, 1981.

[428] **Rissanen J. y Langdon G. G.,** Arithmetic Coding. IBM J. Res. Develop., pp.146-162, 1979.

[429] **Guazzo M.,** A General Minimum-Redundancy Source-Coding Algorithm. IEEE Transactions on Information Theory, 26:15-25, 1980.

[430] **Howard P. G. y Vitter J. S.,** Practical Implementations of Arithmetic Coding. In Image and Text Compression, pages 765-779. Kluwer Academic Publishers, 1992.

[431] **Howard P. G. y Vitter J. S.,** Design and Analysis of Fast Text Compression Based on Quasi-Arithmetic Coding. In Data Compression Conference (DCC), pages 89-107, 1993.

[432] **Langdon G. G. y Rissanen J.,** Compression of Black-White Images with Arithmetic Coding. IEEE Transactions on Communications, 29(6):858-867, 1981.

[433] **Mitchell J. L. y Pennebaker W. B.,** Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder. IBM J. Res. Develop, 32:727-736, 1988.

[434] **Pennebaker W. B., Mitchell J. L., Langdon G. G., y Arps R. B.,** An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder. IBM J. Res. Develop, 32:717-726, 1988.

[435] **Rissanen J. y Langdon G. G.,** Arithmetic Coding. IBM J. Res. Develop., pages 146-162, 1979.

[436] **Rissanen J. y Langdon G. G.,** Universal Modeling and Coding. IEEE Transactions on Information Theory, 27:12-23, 1981.

[437] **Rissanen J. J.,** Generalized Kraft Inequality and Arithmetic Coding. IBM J. Res. Develop., 20(198-203), 1976.

[438] **Rissanen J. J.,** A Universal Data Compression System. IEEE Transaction on Information Theory, 29(5):656-664, 1983.

[439] **Rubin F.,** Arithmetic Stream Coding Using Fixed Precision Registers. IEEE Transactions on Information Theory, 25:672-675, 1979.

[440] **Witten I. H., Neal R. M., y Cleary J. G.,** Arithmetic Coding for Data Compression. Communications of the ACM, 30(6):520-540, 1987.

[441] **Arps R. B., Truong T. K., Lu D. J., Pasco R. C., y Friedman T. D.,** A Multi-Purpose VLSI Chip for Adaptive Data Compression of Bilevel Images. IBM J. Res. Develop, 32(6):775-795, 1988.

[442] **Langdon G. G.,** Probabilistic and Q-Coder Algorithms for Binary Source Adaption. In Data Compression Conference (DCC), pages 13-22, 1991.

[443] **Mitchell J. L. y Pennebaker W. B.**, Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder. IBM J. Res. Develop, 32:727-736, 1988.

[444] **Pennebaker W. B. y Mitchell J. L.,** Probability Estimation for the Q-Coder. IBM J. Res. Develop, 32:737-752, 1988.

[445] **Pennebaker W. B. y Mitchell J. L.,** Software Implementation of the Q-Coder. IBM J. Res. Develop, 32:753-774, 1988.

[446] **Osorio R. R.,** Algoritmos y Arquitecturas para la Codificación Aritmética: Explotación de la Localidad Utilizando Memorias Cache. PhD thesis, Depto de Electrónica y Computación, Universidad de Santiago de Compostela, 1999.

[447] **Miano J.,** Compressed Image File Formats: JPEG, PNG, GIF, XBM, BPM. Addison Wesley, 1999.

[448] **Roelofs G.,** PNG: The Definitive Guide. O'Reilly, 1999.

[449] **Wallace G. K.,** The JPEG Still Picture Compression Standard. Communications of the ACM, 34(4):30-40, 1991.

[450] **Karhunen J.,** Optimization criteria and nonlinear PCA neural networks. In *IEEE World Congress on Computational Intelligence*, volume 2, pages 1241–1246, 1994.

[451] **Karhunen J. y Joutsensalo J.,** Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, 7(1):113–127, 1994.

[452] **Boyd S. y Vandenberghe L.,** *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

[453] **Schraudolph N. N.,** Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

[454] **Schraudolph N. N.,** Local gain adaptation in stochastic gradient descent. In *Proc. Intl. Conf. Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London.

[455] **Pearlmutter B. A.,** Fast exact multiplication by the Hessian. *Neural Computation*, 6(1): 147–160, 1994.

[456] **Griewank A.,** *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

[457] **Vishwanathan S. V. N., Schraudolph N. N., y Smola A. J.,** Step size adaptation in reproducing kernel Hilbert space. *Journal of Machine Learning Research*, 7:1107–1133, June 2006.

[458] **Hyvärinen A. y Oja E.,** A Fast Fixed-Point Algorithm for Independent Component Analysis**.** *Neural Computation*, 9(7):1483-1492, 1997.

[459] **Jolliffe I. T.**, Principal Component Analysis, Second Edition, Springer, New York, 2002.

[460] **Dunteman G. H.**, Principal Component Analysis, Sage Publications, Newbury Park, 1989.

[461] -, Handbook of Neural Network Signal Processing, Editado por Yu Hen Hu y Jenq-Neng Hwang, CRC Press, Boca Raton, 2002.

[462] **Björck A.**, Numerical Methods for Least Squares Problems, SIAM Press, Philadelphia, 1996.

[463] **Anderson T. W.**, Introduction to Multivariate Statistical Analysis, 2/e, Wiley, New York, 1984.

[464] **Morrison D. F.**, Multivariate Statistical Methods, 3/e, McGraw-Hill, New York, 1990.

[465] **Wilks D. S.**, Statistical Methods in the Atmospheric Sciences, Academic Press, New York, 1995.

[466] **von Storch H. y Zwiers F. W.**, Statistical Analysis in Climate Research, Cambridge Univ. Press, Cambridge, 1999.

[467] **Gittins R.**, Canonical Analysis, Springer-Verlag, New York, 1985.

[468] **van Huffel S. y Vandewalle J.**, The Total Least Squares Problem, SIAM, Philadelphia, 1991.

[469] **Wallace G. K., Vivian R., y Poulsen H.**, Subjective testing results for still picture compression algorithms for international standardization. In Proceedings of the IEEE Global Telecommunications Conference. IEEE Communications Society, Nov. 1988, pp. 1022-1027.

[470] **Léger, A., Mitchell, M., y Yamazaki, Y**. Still picture compression algorithms evaluated for international standardization. In Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society, Nov. 1988, pp. 1028-1032.

[471] **Hudson, G. P.** The development of photographic videotex in the UK. In Proceedings of the IEEE Global Telecommunications Conference, IEEE Communication Society, 1983, pp. 319-322.

[472] **Hudson, G. P., Yasuda, H., y Sebestyén, I.** The international standardization of a still picture compression technique. In Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society, Nov. 1988, pp. 1016-1021.

[473] **Léger, A.** Implementations of fast discrete cosine transform for full color videotex services and terminals. In Proceedings of the IEEE Global Telecommunications

Conference, IEEE Communications Society, 1984, pp. 333-337.

[474] **Wallace G. K.**, Overview of the JPEG (ISO/CCITT) still image compression standard. Image Processing Algorithms and Techniques, In Proceedings of the SPIE, vol. 1244 (Feb. 1990), pp. 220-233.

[475] **Howard, P.G., y Vitter, J.S.** New methods for lossless image compression using arithmetic coding. Brown University Dept. of Computer Science Tech. Report No. CS-91-47, Aug. 1991.

[476] **Wallace G. K.**, The JPEG Still Picture Compression Standard, Submitted in December 1991 for publication in IEEE Transactions on Consumer Electronics. http://www.ijg.org/files/wallace.ps.gz

[477] Digital Compression and Coding of Continuous tone Still Images, Part 1, Requirements and Guidelines. ISO/IEC JTC1 Draft International Standard 10918-1, Nov. 1991.

[478] Adobe Systems Inc. PostScript Language Reference Manual. Second Ed. Addison Wesley, Menlo Park, Calif. 1990.

[479] Office Document Architecture (ODA) and Interchange Format, Part 7: Raster Graphics Content Architectures. ISO/IEC JTC1 International Standard 8613-7.

[480] **Pennebaker W. B. y Mitchell J. L.**, "JPEG Tech. Specification," Revision 8. Informal Working paper JPEG-8-R8, Aug. 1990.

[481] **Pennebaker W. B. y Mitchell J. L.**, *JPEG: Still Image Compression Standard*. Kluwer Academic Publishers, 1993.

[482] **Wallace G. K.**, The JPEG Still Picture Compression Standard, *Communications of the ACM*, vol. 34, no. 4, pp. 30-44, April 1991.

[483] Digital Compression and Coding of Continuous-tone Still Images, Part 2, Compliance Testing. ISO/IEC JTC1 Committee Draft 10918-2, Dec. 1991.

[484] **Léger, A., Omachi, T., y Wallace, G.** The JPEG still picture compression algorithm. In Optical Engineering, vol. 30, no. 7 (July 1991), pp. 947-954.

[485] ISO/IEC 15444-1 and ITU-T Recommendation T.800, "Information technology {JPEG 2000} image coding system," 2002.

[486] **Taubman D., Ordentlich E., Weinberger M., y Seroussi G.,** Embedded block coding in JPEG 2000. In ICIP00, page TA02.02, 2000.

[487] **Taubman D. S.,** High performance scalable image compression with EBCOT. In IEEE Trans. Image Proc., volume 9, July 2000.

[488] **Christopoulos C. A., Ebrahimi T. y Skodras A. N.**, JPEG2000: The New Still Picture Compression Standard, *Proc. ACM Multimedia 2000, Int. Workshop on Standards,*

Interoperability and Practices, Los Angeles, CA, Nov. 2000.

[489] ISO-14495-1 and ITU-T Recommendation T.87, “Information technology {Lossless and near-lossless compression of continuous-tone still images," 2000.

[490] Oficial JPEG2000 Homepage. http://www.jpeg.org/jpeg2000.html (as of 01.10.02).

[491] University of Southern California, Signal and Image Processing Institute. The USCSIPI Image Database. Disponible en: http://sipi.usc.edu/services/database/Database.html (as of 01.10.02).

[492] ISO/IEC 11544:1993 and ITU-T Recommendation T.82. Information technology – Coded representation of picture and audio information – progressive bi-level image compression, 1993.

[493] ISO/IEC 15444-1:2000. Information technology – JPEG 2000 image coding system – Part 1: Core coding system, 2000.

[494] **Clark R.**, “An Introduction to JPEG2000 and Watermarking." Elysium Ltd. Disponible en: http://www.jpeg.org/public/jpgintro.pdf (as of 01.10.02).

[495] **Taubman D. S. y Marcellin M. W.**, *JPEG2000: Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Boston, 2002.

[496] **Adams M. D.**, “The JPEG-2000 Still Image Compression Standard," Tech. Rep. N2412, ISO/IEC JTC 1/SC 29/WG 1, September 2001.

[497] **Rabbani M. y Santa Cruz D.**, “The JPEG2000 Still-Image Compression Standard." Course given at the 2001 International Conference in Image Processing (ICIP), October 2001. http://jj2000.epfl.ch/jj_publications/papers/011.pdf (as of 01.10.02).

[498] **Christopoulos C. y Skodras A.**, “JPEG2000 - The Next Generation Still-Image Compression Standard." Tutorial given at the IEEE International Conference on Image Processing (ICIP), October 1999. Disponible en: http://www.etro.vub.ac.be/members/christopoulos.charilaos/jpeg2000_cont%ributions.htm (as of 01.10.02).

[499] **Impoco G.**, JPEG2000 - A Short Tutorial, April 1, 2004 Disponible en: http://www.dmi.unict.it/~impoco/files/tutorial_JPEG2000.pdf

[500] ISO/IEC 11544:1993 and ITU-T Recommendation T.82. Information technology – Coded representation of picture and audio information – progressive bi-level image compression, 1993.

[501] ISO/IEC 15444-1:2000. Information technology – JPEG 2000 image coding system – Part 1: Core coding system, 2000.

[502] **Arfken G.,** Mathematical Methods for Physicists. Academic Press, Boston, 3 edition, 1985.

[503] **Battiato S., Gallo G., Impoco G., y Stanco F.,** A color reindexing algorithm for lossless

compression of digital images, 2001.

[504] **Taubman D., Ordentlich E., Weinberger M., y Seroussi G.,** Embedded block coding in JPEG 2000. In ICIP00, page TA02.02, 2000.

[505] **Taubman D. S.,** High performance scalable image compression with EBCOT. In IEEE Trans. Image Proc., volume 9, July 2000.

[506] **Zeng W., Li J., y Lei S.,** An efficient color re-indexing scheme for palette-based compression. In ICIP00, pages 476–479, 2000.

[507] **Moon T. K. y Stirling W. C.**, "Mathematical methods and algorithms," Upper Saddle River, NJ: Prentice Hall, 2000.

[508] **Kreyszig E.**, "Advanced engineering mathematics," 7th Edition, NewYork, NY: John Wiley, 1993.

[509] **Golub G. H. y van Loan C. F.**, Matrix Computations, 3/e, Johns Hopkins University Press, Baltimore, 1996.

[510] **Watkins D. S.**, Fundamentals of Matrix Computations, 2/e, Wiley, New York, 2002.

[511] **Bellman R.**, Introduction to matrix analysis, SIAM, 2000, Philadelphia.

Bibliografía Honomástica

- [15, 114, 119, 165, 167, 201, 392, 394, 395, 397, 398, 461, 477, 479, 483]

Abadir K. M. [347]

Abarbanel H. D. I. [112]

Abramovich F. [234, 234]

Abramson N. [425]

Adams M. D. [496]

Adams N. [220]

Adobe Systems Inc. [478]

Aggarwal G. [293]

Ahuja N. [88]

Ai H. [116, 117, 117]

Akkarakaran S. J. [85, 93]

Albardar A. [389]

Alexopoulos V. [25]

Algazi V. R. [244, 32]

Almeida R. [256]

Amaratunga K. [255, 308]

An M. [317, 318]

Anderson D. J. [263]

Anderson T. W. [463]

Andreopoulos Y. [204]

Andréu García G. [257, 258]

Angelini E. [266]

Arfken G. [502]

Arie J. B-. [56, 64]

Arnold B. [401]

Arps R. B. [420, 434, 441]

Azar Y. [269]

Balakrishnan V. [326]

Bao Z. [105]

Baraniuk R. [224]

Baraniuk R. G. [228, 229, 231]

Barbosa L. [272]

Bartell J. [171]

Battiato S. [121, 505]

Bauer S. [127]

Belausteguigoitía C. F. [276]

Belhumeur P. N. [52]

Bellman R. [511]

Benjamini Y. [233]

Berman D. [171]

Berraondo López P. [274]

Beylkin G. [172]

Bharitkar S. [76]

Bilgin A. [252]

Birkhoff G. [361]

Björck A. [462]

Blanco A. [19]

Blinn J. F. [294]

Bogner R. E. [44]

Boliek M. P. [252]

Bopardikar A. S. [202]

Borman S. [50]

Bouman C. A. [161]

Boyd S. [452]

Bradley A. B. [23]

Brady J. M. [60]

Bregles C. [96]

Bretherton C. S. [7]

Briggs W. L. [312]

Brooks S. [250]

Bruce A. G. [219]

Buck J. R. [391]

Burl J. B. [40]

Burrus C. [225]

Burrus C. S. [193, 335, 387]

Cagnoli B. [352]

Cai Z. [235]

Calderbank A. R. [240]

Canela M. [259]

Cárdenas-Barrera J. L. [264]

Carlach J. C. [302]

Castelli V. [248]

Castrillon-Candas J. E. [308]

Castro L. R. [194]

Castro S. M. [194]

Celebi H. [63]

Cendrillon R. [84]

Cernuschi Frias B. [276]

Chakrabarti N. B. [49]

Chamberlain N. F. [188]

Chan K. W. [270]

Chan L. A. [66]

Chan Y. H. [13]

Chandrasekaran S. [54]

Chandroth G. [261]

Chang C. [326]

Chang C-Y. [91]

Chang S. [223, 222, 333]

Chen C. C. T. [21]

Chen C. S. [22]

Chen C. T. [21]

Chen J. [220]

Chen S. [118]

Chen T. [102]

Cheng T. H. [235]

Cherkassky V. [211]

Chitwong S. [76]

Chiu C. T. [295]

Cho N. I. [303, 304]

Choi H. [224]

Chouikha M. F. [24, 26]

Christensen P. H. [173, 174]

Christophe E. [163]

Christopoulos C. [498, 488]

Chung K. [106]

Clark R. [494]

Clark R. J. [290]

Clarke R. J. [320, 426]

Claveau F. [319]

Cleary J. G. [418, 420]

Cohen M. F. [178, 179, 181, 185]

Coifman R. [172]

Coifman R. R. [218]

Cornelis J. [204]

Couch II L. W. [393]

Creusere C. D. [243]

Crouse M. S. [228]

Crowe J. [411]

Cuadrado J. [373]

Cuesta Frau D. [257, 258]

Daniel J. W. [359]

Darian Muresan D. [212]

Daubechies I. [175, 195, 197, 240, 330]

David B. [251]

Davila C. E. [79, 95]

de la Cruz J. M. [372, 373]

de Moor B. [396]

de Valle E. E. [272]

Dehaene J. [148]

DeRose T. [182]

DeRose T. D. [169, 170, 174, 186, 199, 339]

DeVore R. [176]

Diamantaras K. I. [4]

Díaz Calavia E. J. [274]

Diebner H. H. [259]

Ding R. D. [131]

Ding W. [130]

Donoho D. L. [213, 214, 215, 216, 217, 218, 328, 329]

Dony A. C. [16]

Dony R. D. [70, 80]

Downton A. C. [14]

Driver R. M. [83]

Dubreil P. [370]

Dubreil-Jacotin M. L. [368]

Duchaineau M. A. [209]

Duhamel P. [301, 302, 306]

Dunteman G. H. [460]

Ebrahimi T. [488]

Echebest N. [314]

Eck V. [257]

Eddins S. L. [150]

Efromovich S. [221]

Eggers J. J. [86]

Elizalde Soba P. [274]

Epstein B. R. [71, 152, 162, 168]

Estes R. R. [244]

Faloutsos C. [53]

Faúndez Zanuy M. [374]

Feig E. [299]

Fiat A. [269]

Finkelstein A. [177, 180]

Fleury M. [14]

Flierl M. [104]

Fomin S. V. [267]

Fraleigh J. B. [358]

Franz M. O. [138]

Freire S. L. M. [351]

Friedman T. D. [441]

Gabbouj M. [87]

Gadiya M. C. [275]

Gailly J. [422]

Gajski D. D. [293]

Gallo G. [121, 503]

Gan J-Y. [103]

Gao H. Y. [219]

García H. [262]

García I. [423]

Gerbrands J. J. [414]

Gilmore E. T. [24, 26]

Giraldez A. [282, 340, 343, 344, 345, 280, 281]

Girod B. [104, 86]

Gittins R. [467]

Golub G. H. [509]

Gonzalez R. C. [149, 150, 310]

Gopinath R. A. [193, 226, 227, 335]

Gorman M. [30]

Gormish M. J. [252]

Gortler S. J. [178, 179, 181, 185]

Goyal V. [403]

Gray D. A. [44]

Gray R. M. [404]

Griewank A. [456]

Grond F. [259]

Grossman A. [190, 337]

Grossman S. I. [362]

Guardans R. [259]

Guazzo M. [429]

Guillemot C. [301, 302, 306]

Guleryuz O. G. [145, 146]

Gunaratne G. H. [30]

Günter S. [139, 140]

Guo H. [193, 225, 226, 227, 335]

Hadi A. S. [8]

Hall J. [411]

Hamid Nawab S. [381]

Hamilton D. J. [19]

Han J. K. [90, 327]

Hanrahan P. [179, 185]

Hao P. [109]

Haque M. A. [296]

Hasan A. [101]

Hasan M. [101]

Haykin S. [131, 136, 386, 385, 386, 70]

He X. [245]

Hein D. N. [285]

Hernández Lerma O. [315]

Herrara A. [42]

Herstein I. N. [365]

Hespanha J. P. [52]

Hjorungnes A. [57]

Hoffman K. [355]

Homma S. [266]

Hotelling H. [10, 11]

Howard P. G. [34, 419, 430, 431, 475]

Hsu H. P. [314]

Huang S. C. [132, 133]

Huang Y. F. [132, 133]

Hubbard B. B. [203, 336]

Hudson, G. P. [471, 472]

Huffman D. A. [417]

Huhle B. [405]

Hung A. C. [292]

Hunt B. R. [158]

Huo K. S. [22]

Hyvärinen A. [458]

Impoco G. [121, 499, 503]

Ishikawa S. [107]

ISO/IEC [485, 415]

ISO/IEC 11544:1993 and ITU-T Recommendation T.82. [492, 500]

ISO/IEC 15444-1:2000. [493, 501]

ISO-14495-1 and ITU-T Recommendation T.87 [489]

Jacobs C. E. [180]

Jain A. K. [309, 151, 307, 36, 46]

Jain R. [55]

Janatra I. I. [110]

Jawerth B. [176]

Jia W. [245]

Johnstone I. M. [213, 214, 216, 217, 328, 329]

Jolicoeur P. [6]

Jollife I. T. [3, 459]

Jones H. W. [285]

Joutsensalo J. [451]

Joy K. I. [209]

Kaiser G. A. [200]

Kamangar F. A. [298]

Kamen E. W. [379]

Kantz H. [113]

Kao Y. K. [96]

Karhunen J. [134, 450, 451]

Karlin A. R. [269]

Kay S. M. [380]

Kee S. C. [106]

Kerkyacharian G. [213, 214]

Kermani B. G. [33]

Khattree R. [9]

Khayam S. A. [287]

Kim K. I. [138, 94]

Kim S. R. [106]

Kim T-S. [155]

Kimpan C. [72, 73, 81, 82]

Kinsner G. [251]

Kinsner W. [251]

Kirac A. [65]

Kirby M. [59]

Kitajima H. [31]

Kitter J. [20]

Kleisinger G. H. [272, 273]

Knauer S. C. [285]

Knuth D. E. [421]

Kollias S. [25]

Kolmogorov A. N. [267]

Kongkchandra R. [72, 73, 81, 82]

Konstantinides K. [47]

Kontoyiannis I. [248]

Kopp M. [246]

Kreyszig E. [508]

Kriegman D. [88]

Kriegman D. J. [52]

Krishna N. R. [49]

Krot A. M. [58]

Kubler O. [158]

Kudryavtsev V. O. [58]

Kumar P. S. [12]

Kung S. Y. [4]

Kunze R. [355]

Kuo C.-C. [115, 116, 117]

Kurashov V. N. [41]

Kyriakakis C. [11, 116, 117, 77]

Laguan P. [256]

Lahme B. [74]

Laine A. [266]

Lakey J. [221]

Lan L. S. [126, 128]

Lang M. [225, 226, 227]

Lang S. [356]

Langdon G. G. [427, 428, 432, 434, 435, 436, 442]

Lawson B. L. [268]

Lee J. [153, 69]

Lee S. U. [303, 304]

Léger, A. [470, 473, 484]

Lei S. [122, 506]

Leon S. J. [357]

Li B. [108]

Li J. [122, 506]

Liberman E. [314]

Ligtenberg A. [300]

Lin K-I. [53]

Lin Q. [245]

Lin W. S. [1]

Lin Y. P. [68]

Ling R. F. [8]

Linzer E. N. [299]

Lischinski D. [173]

Liu H. [67]

Liu K. J. R. [295]

Liu Z. [181]

Loeffler C. [300]

López A. [373]

Lorenzo-Ginori J. V. [264]

Lounsbery M. [182]

Lovell B. [84]

Lu C. [235, 317, 318]

Lu D. J. [441]

Lucier B. [176]

Maciejewski A. A. [326]

MacLane S. [363]

Magnus J. R. [347]

Makai B. [127]

Mallat S. G. [183, 189, 331, 332]

Máltsev A. I. [367]

Manolakis D. G. [377, 378]

Mao S-Y. [103]

Marcellin M. W. [252, 495]

Marcone G. [124]

Marcus M. [360]

Mariño Acebal J. B. [376]

Martin U. [241]

Martinelli G. [124]

Martinez J. P. [256]

Martinez M. [42]

Martucci S. A. [383]

Mase K. [97]

Mastriani M. [166, 277, 278, 279, 280, 281, 282, 283, 284, 311, 321, 340, 341, 342, 343, 344, 345, 346, 37, 370]

Mathias A. [259]

MATLAB® [39]

McClellan J. H. [387]

McMillan L. [305]

McSherry F. [269]

Meng TH-Y [292]

Meyer C. D. [349]

Meyer Y. [191]

Meyers D. [184]

Miano J. [447]

Minc H. [360]

Miranda R. [74]

Misiti M. [206, 334]

Misiti Y. [206, 334]

Mitchell J. L. [288, 433, 434, 443, 444, 445, 480, 481]

Mitchell, M. [470]

Miyahara M. [17]

Mojsilovic A. [265]

Molina J. M. [373]

Mongkolworaphol S. [35]

Monzón J. E. [272, 273]

Moody G. B. [271]

Moon T. K. [507]

Moonen M. [148, 396]

Moreno Bilbao A. [376]

Morlet J. [190, 337]

Morrison D. F. [464]

Moschytz G. [300]

Mosimann J. E. [6]

Motwani M. C. [275]

Motwani R. C. [275]

Mudugamuwa D. J. [23]

Mulcahy C. [210]

Müller K.-R. [142]

Munteanu A. [204]

Musatenko Y. S. [41]

Nagle H. T. [33]

Naik D. N. [9]

Nakagawa M. [17]

Nandy D. [64]

Nasrabadi N. M. [66, 89]

Neal R. M. [440]

Nefian A. V. [61]

Nelson M. [422]

Neskovic A. N. [265]

Netsch L. [48]

Neuhoff D. L. [404]

Ng R. [324]

Nicholson W. K. [363]

Nikolova M. [236]

Noble B. [359]

Novák D. [258]

Nowak R. D. [228, 229, 231, 230]

Odegard J. E. [225, 226, 227]

Oficial JPEG2000 Homepage. [490]

Oja E. [134, 135, 458]

Olafsson V. [220]

Olmos S. [256]

Omachi, T. [484]

Oppenheim A. V. [381, 387, 390, 391]

Oppenheim G. [206, 334]

Orchard M. T. [145]

Ordentlich E. [486, 504]

Orfanidis S. J. [353]

Orrison M. E. [268]

Osorio R. R. [446]

Ouyang S. [105]

Oweiss K. G. [263]

Pajares G. [372, 373]

Palacios A. [30]

Paliwal K. K. [143]

Papoulis A. [388]

Paquet A. H. [260]

Parks T. W. [212, 387]

Parlett B. [111]

Pasco R. [426]

Pasco R. C. [441]

Paz Viera J. E. [247]

Pearlmutter B. A. [455]

Pennebaker W. B. [288, 433, 434, 443, 444, 445, 480, 481]

Pentland A. [45, 51, 78, 97, 98]

Pentland A. P. [322]

Percival D. J. [125]

Pereyra M. C. [221]

Pérez Cortés J. C. [257, 258]

Phoong S. M. [68]

Picard D. [213, 214]

Pickard R. W. [322]

Pisarello M. I. [273]

Poggi J. M. [334, 208]

Poirier M. [319]

Poirson A. [286]

Pólchlopek W. [239]

Ponceleon D. B. [123]

Popat K. [249]

Popovic A. D. [265]

Popovic M. V. [265]

Poulsen H. [469]

Prabhu K. M. M. [12]

Preisendorfer R. W. [2]

Principe J. C. [144]

Proakis J. G. [377]

Proakis J. G. [378]

Quddus A. [87]

Querre P. [412]

Quian Quiroga R. [262]

Rabbani M. [497]

Radha H. [291]

Rahman M. M. [107]

Raju B. I. [237]

Ramstad T. [413]

Ramstad T. A. [57]

Ranganath S. [208]

Rangarajan R. [238]

Rangsanseri Y. [35]

Rao C. R. [5]

Rao K. R. [29, 298, 202]

Rao T. V. K. H. [49]

Ras Oliva E. [313]

Ray Liu K. J. [1]

Ray W. D. [83]

Reagan J. [156, 159]

Reed I. S. [126, 128]

Ricoh Innovations, Inc. [205]

Ricolti L. P. [124]

Rissanen J. [420, 427, 428, 432, 435, 436]

Rissanen J. J. [437, 438]

Rizvi S. A. [89]

Robbins K. A. [30]

Robinson J. [248]

Rocha A. P. [256]

Rodríguez del Río L. S. [164]

Rodríguez Fonollosa J. A. [376]

Rodríguez-Valdivia E. [264]

Roelofs G. [448]

Rokhlin V. [172]

Roman S. [348]

Rösel R. [241]

Roth P. M. [38]

Rubin F. [439]

Ruiz V. G. [416, 423]

Saadawi T. N. [89]

Sacchi M. D. [350]

Saghri J. [156, 157, 154, 159]

Saha S. [407]

Sahouria E. [99, 325]

Saia J. [269]

Sakrison D. J. [32]

Salesin D. H. [171, 169, 170, 173, 174, 177, 180, 186, 199, 339]

Samet H. [399, 400]

Sanchez O. [42]

Sandham W. A. [19]

Sanger T. D. [137]

Santa Cruz D. [497]

Santalo L. A. [366]

Sapatinas T. [234]

Sastre Mengual C. [257]

Scarloff S. [322]

Schafer R. W. [387, 390, 391]

Schalkoff R. J. [371]

Schelkens P. [204]

Schiffman S. S. [33]

Schölkopf B. [138, 141, 142]

Schraudolph N. N. [139, 140, 453, 454, 457]

Schreiber T. [113]

Schröder P. [179, 185]

Schuessler H. W. [387]

Sclaroff S. [147]

Sebestyén, I. [472]

Sedighian A. [324]

Seese D. [259]

Self R. P. [14]

Selin I. [43]

Semmlow J. L. [120]

Senecal J. G. [209]

Seroussi G. [486, 504]

Shah S. [238]

Shapiro J. M. [242]

Shapiro L. S. [60]

Sharkey A. J. C. [261]

Sharkey N. E. [261]

Sharma A. [143]

She Z. [44]

Shen J. [187]

Shi Q. [109]

Shimone T. [31]

Sikora T. [127]

Silverman B. [234]

Sirovich L. [59]

Skodras A. N. [498, 488]

Smith C. [7]

Smola A. J. [141, 142, 457]

So. H. C. [270]

Soliman S. S. [382]

Srinath M. D. [382]

Srinivasan M. A. [237]

Stanco F. [121, 503]

Starck J.-L. [412]

Stefanoiu D. [92]

Stevenson R. [50]

Stewart W. K. [28]

Stirling W. C. [507]

Stollnitz E. J. [169, 170, 173, 174, 186, 199, 339]

Strang G. [187, 207, 289, 364]

Su J. K. [86]

Subramanian K. R. [235]

Suter B. W. [129]

Sweldens W. [240]

Swets D. L. [27]

Sylvain D. [236]

Tabbara M. [250]

Tabus I. [92]

Takuma S. [266]

Tamee K. [72, 73, 81, 82]

Tanaka T. [75]

Tang X. [28]

Taqub A. [371]

Taswell C. [232]

Taubman D. [486, 504]

Taubman D. S. [487, 495, 505]

Tejeira Alvarez J. M. [274]

Tescher A. G. [156, 157, 159]

Teuber J. [375]

Tewfik A. H. [327, 90]

Thierschmann M. [241]

Thitimajshima P. [35]

Thompson R. C. [369]

Tjoa S. [1, 402]

Tolimieri R. [317, 318]

Tretter D. [161]

Truong T. K. [441]

Trusset H. J. [63]

Tsai C. [110]

Tsai C. M. [21]

Tsapatsoulis N. [25]

Turek J. [248]

Turk M. [45, 51, 78, 98]

Tymes N. [221]

Ulrych T. J. [351, 352]

Uminsky D. T. [268]

University of Southern California, Signal and Image Processing Institute. [491]

Unser M. [408]

Vaidyanathan P. P. [65, 85, 93]

Valens C. [192]

Vallverdú Bayés F. [376]

Van der Auwera G. [204]

Van Emden H. [312]

van Huffel S. [468]

van Loan C. F. [509]

Van Otterloo S. [253]

Van Veen B. [385]

Vandenberghe L. [452]

Vandewalle J. [148, 468]

Vargic R. [254]

Vaughn V. D. [160]

Venkataramanan R. [238]

Vetterli M. [222, 223, 297, 301, 333]

Vishwanathan S. V. N. [139, 140, 457]

Vitter J. S. [419, 424, 430, 431, 475]

Vivian R. [469]

von Storch H. [466]

Waldemar P. [413]

Walker J. S. [196, 198, 340]

Wallace G. K. [449, 469, 474, 476, 482]

Wallace J. M. [7]

Wallace, G. [484]

Walter G. [409]

Wan S. [237]

Wang Z. [56]

Ward R. K. [260]

Warren J. [182]

Watkins D. S. [510]

Watson A. B. [286]

Wei J. [108]

Weinberger M. [486, 504]

Weinberger M. J. [420]

Wells R. [225]

Weng J. [27]

Westover L. [305]

White D. [55]

Wien M. [323]

Wilkinson T. S. [160]

Wilks D. S. [465]

Willsky A. S. [381]

Winslow P. [354]

Winter D. J. [365]

Winter M. [38]

Wintz P. [310]

Witten I. H. [418, 440]

Woods R. E. [149, 150]

Wornell G. W. [410]

Xia X. G. [129]

Xiangdong W. [100]

Yamashita Y. [75]

Yamazaki, Y [470]

Yan Y. [62]

Yang D. [115, 116, 117]

Yang M-h [88]

Yasuda, H. [472]

Yeo B.-L. [240]

Ying L. [208]

Yip C. [220]

Young P. C. [20]

Yu B. [222, 223, 333]

Yun I. D. [304]

Zahir S. [260]

Zakhor A. [99, 325]

Zeng W. [122, 508]

Zhang D. [118]

Zhang J. [110, 409, 62]

Zhang Y-W. [103]

Zhong J. [147]

Zhong S. [211]

Zhou X. [208]

Ziólko M. [239]

Ziyad N. [24, 26]

Ziyal N. A. [18]

Zwiers F. W. [466]

Publicaciones relativas a esta tesis

[1] M. Mastriani y A. Giraldez, "*Neural Shrinkage for Wavelet-Based SAR Despeckling*," International Journal of Intelligent Technology, Volume 1, Number 3, pp.211-222, 2006.

[2] M. Mastriani, "*Fuzzy Thresholding in Wavelet Domain for Speckle Reduction in Synthetic Aperture Radar Images*," International Journal of Intelligent Technology, Volume 1, Number 3, pp.252-265, 2006.

[3] M. Mastriani, "*New Wavelet-Based Superresolution Algorithm for Speckle Reduction in SAR Images* ," International Journal of Computer Science, Volume 1, Number 4, pp.291-298, 2006.

[4] M. Mastriani, "*Denoising and Compression in Wavelet Domain via Projection onto Approximation Coefficients*," International Journal of Signal Processing, Volume 5, Number 1, pp.20-30, 2008.

[5] M. Mastriani and Marta Mejail, "*Union is Strength in Lossy Image Compression II*," under review in International Journal of Signal Processing.