INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN Y DESARROLLO DE TECNOLOGÍA DIGITAL

MAESTRÍA EN CIENCIAS CON ESPECIALIDAD EN SISTEMAS DIGITALES

"Técnicas de Procesamiento de Imágenes utilizando un Procesador Digital de Señales"

TESIS QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS
PRESENTA:

Luis Enrique Palafox Maestre

Tijuana, Baja California **Mayo del 2002**

CGPI-14

INSTITUTO POLITECNICO NACIONAL

COORDINACION GENERAL DE POSGRADO E INVESTIGACION

ACTA DE REVISION DE TESIS

En la Ciudad de TIJUANA, B.C. siendo las 12:00 horas del día 26 del mes de ABRIL del 2002 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de CITEDI para examinar la tesis de grado titulada:

TÉCNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES UTILIZANDO UN PROCESADOR DIGITAL DE SEÑALES.

Presentada por el alumno:

PALAFOX	MAESTRE	LUIS ENRIQUE
Apellido paterno	materno	nombre(s)

Con registro: | 9 | 8 | 0 | 7 | 1 | 7 |

aspirante al grado de:

MAESTRO EN CIENCIAS CON ESPECIALIDAD EN SISTEMAS DIGITALES

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACION DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISION REVISORA

Director de tesis

DR. JUAN GARCIA LOPEZ

DR. CIRO ANDRES MARTINEZ GARCIA MORENO

DR. SERGIO ANTONIO HERRERA GARCIA

M.C. JORGE TORRES RODRIGUEZ

M.C. JULIO CESAR ROLON GARRIDO

S. E. P.
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION Y DESARROLLO
DE TECNOLOGIA DIGITAL
DIRECCION

EL PRESIDENTE DEL COLEGIO

DR. ALFONSO ANGELES VALENCIA

"La libertad del hombre radica en su dependencia emocional"

Dedicatoria

A las tres mujeres que me han enseñado a vivir la vida intensamente: mi abuelita Mina, mi mamá y Gabby, la compañera de mi vida.

Agradecimientos

Gracias a ese pedacito de mi vida, que aún sin haber nacido, me ha impulsado a concluir con este trabajo.
A mi papá, que siempre me apoyó para seguir adelante.
A mi hermano Jorge, que siempre ha sido cómplice de todos mis logros.
A mis hermanos Michel y Genaro, por recordarme siempre lo bello que es la infancia.
A Yolanda, por recibirme siempre con los brazos abiertos.
A mi abuelita Lola, por estar siempre conmigo.
A mi abuelita Ofelia, por todo ese cariño.
A mis tíos y tías: José, Jesús, Mina, Malia, Genaro y Eduardo por motivarme a seguir adelante.
A mis suegros y al "Poncho", por abrirme las puertas de su casa.
A mis compañeros de laboratorio: Jaime Esqueda, César Ortega y Julio Mendoza, por su apoyo técnico pero sobre todo por su apoyo humano.
A mis alumnos, porque he aprendido tantas cosas de ellos.
Al Dr. Juan García, por su tiempo y su inagotable paciencia.
Al Comité Revisor: Dr. Ciro Martínez, Dr. Sergio Herrera, M.C. Julio Rolón y M.C. Jorge Torres, por sus valiosas aportaciones a este trabajo.
Al CITEDI y al CONACYT por su apoyo brindado durante la realización de este trabajo.
A mi alma mater, la Universidad Autónoma de Baja California, por hacer esto posible.
Y a todas esas personas que por cuestiones de espacio no he mencionado, pero que sin duda han jugado un papel muy importante en el desarrollo de este trabajo.

CONTENIDO

LISTA DE FIGURAS iv

LISTA DE TABLAS vi

LISTA DE SÍMBOLOS Y ACRÓNIMOS vii

RESUMEN ix

ABSTRACT ix

INTRODUCCIÓN 1

CAPÍTULO I. TÉCNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES 3

I.1 Introducción. 3

I.2 Métodos espaciales. 4
I.2.1 Filtrado de imágenes. 4
I.2.1.1 Filtros bidimensionales. 6
I.2.2 Realce de imágenes. 9
I.2.2.1 Diferenciación 10
I.2.2.2 El Laplaciano 12
I.2.3 Binarización de imágenes. 15
I.2.4 Detección de bordes 18

I.3 Métodos frecuenciales. 19
I.3.1 Compresión de imágenes mediante la Transformada Discreta Coseno. 20
I.3.1.1 Transformada de una matriz. 20
I.3.1.2 Transformada Discreta Coseno. 21
I.3.1.3 Codificación en bloques y cuantización. 22
I.3.1.4 Descompresión de imágenes mediante la TDC. 25
I.3.2 Amplificación de imágenes mediante la Transformada Discreta de Fourier. 28

I.4 Métodos estadísticos 33
I.4.1 Ecualización de imágenes en base a su histograma. 34
I.4.2 Compresión sin pérdidas. 39
I.4.2.1 Codificación Huffman. 41

CAPÍTULO II. LA TRANSFORMADA DE ONDÍCULAS. 48

II.1 Introducción. 48

II.2 Principios de la transformada continua de ondículas. 50
II.2.1 Propiedades de las ondículas. 50

II.3 Ondículas Discretas. 52
II.3.1 Ondículas discretas como un filtro pasa banda. 55
II.3.2 La función de escalamiento. 57
II.3.3 Codificación en sub-bandas. 59

II.4 La transformada discreta de ondículas 61

CAPÍTULO III. CODIFICADOR DE ONDÍCULAS POR ÁRBOLES DE CEROS. 65

III.1 Introducción. 65

III.2 El Árbol de ceros. 66

III.3 Funcionamiento del codificador. 68
III.3.1 El algoritmo. 70

CAPÍTULO IV. IMPLEMENTACIÓN DEL SISTEMA DE PROCESAMIENTO DIGITAL DE IMÁGENES EN EL DSK 6211. 74

IV.1 Introducción. 74

IV.2 Comunicación Anfitrión-Destino 75
IV.2.1 Establecimiento de la comunicación inicial. 78
IV.2.2 Transferencia de imágenes entre el anfitrión y el destino 79

IV.3 Técnicas clásicas de procesamiento digital de imágenes. 80
IV.3.1 Binarización. 80
IV.3.1.1 Ejemplos de binarización. 81
IV.3.2 Detección de bordes. 82
IV.3.2.1 Ejemplos de detección de bordes. 82
IV.3.3 Filtrado bidimensional. 83
IV.3.3.1 Filtro promediador. 84
IV.3.3.2 Filtro definido por el usuario. 86

IV.4 Compresión de imágenes utilizando ondículas 88
IV.4.1 Filtros biortogonales 5.5. 88
IV.4.1.1 Filtro de análisis. 89
IV.4.1.2 Filtro de síntesis. 95
IV.4.1.3 Reconstrucción perfecta. 99
IV.4.1.4 Proceso de compresión–descompresión de imágenes digitales. 102

CONCLUSIONES Y RECOMENDACIONES 107

APÉNDICES

A. Coeficientes de los filtros biortogonales. 109

B. El método de las diferencias perceptibles asociado al sistema visual humano..... 111

C . El PDS TMS320C6211 y sus especificaciones. 117

D. Tiempos de procesamiento estimados en los experimentos realizados. 119

E. Contenido del disco compacto incluido. 123

REFERENCIAS Y BIBLIOGRAFÍA..... 125

LISTA DE FIGURAS

Fig. I.1 Representación gráfica de la convolución bidimensional. 6
Fig. I.2 Construcción de un filtro bidimensional a partir de dos filtros unidimensionales. 7
Fig. I.3 Filtrado de la imagen de Saturno. 8
Fig. I.4 Filtrado de la imagen de la Luna. 9
Fig. I.5 a) Imágenes originales, b) Después de restarles el Laplaciano (realce de bordes) 14
Fig. I.6 Imagen de Saturno procesada mediante el realce de bordes. 15
Fig. I.7 Imagen de la Luna procesada mediante el realce de bordes. 15
Fig. I.8 Imagen de 256 niveles de gris binarizada con un umbral de 64. 17
Fig. I.9 Imagen de 256 niveles de gris binarizada con un umbral de 128. 17
Fig. I.10 Operaciones para la detección de bordes. 18
Fig. I.11 Imagen procesada mediante el algoritmo de detección de bordes utilizando un umbral de 40. 19
Fig. I.12 Imagen de la espina dorsal procesada mediante el algoritmo de detección de bordes utilizando un umbral de 20. 19
Fig. I.13 Líneas de coeficientes con variancia constante en una imagen típica transformada mediante la TDC. 24
Fig. I.14 Asignación en bits de los coeficientes en una codificación por zonas para una imagen de 8x8. 24
Fig. I.15 Asignación en bits de los coeficientes en una codificación por zonas para una imagen de 16x16. 25
Fig. I.16 Proceso de compresión de una imagen. 25
Fig. I.17 Proceso de descompresión de una imagen. 26
Fig. I.18 Imagen de Saturno procesada con la TDC tomando 4 coeficientes (compresión 16:1) con un MSE = 4.11E-4, SNR = 24.68 dB, PSNR = 29.13 dB. 27
Fig. I.19 Imagen de la Luna procesada con la TDC tomando 16 coeficientes (compresión 4:1) con un MSE = 1E-4, SNR = 31.33 dB, PSNR = 32.26 dB. 27
Fig. I.20 Secuencia temporal de 8 muestras tomadas a 1 Hz. 29
Fig. I.21 Espectro de la secuencia de la figura I.20. 30
Fig. I.22 Espectro después de agregar 24 ceros. 30
Fig. I.23 Secuencia obtenida de la interpolación. 30
Fig. I.24 Inserción de ceros en la frecuencia para lograr una interpolación en el tiempo. 31
Fig. I.25 Aspecto que se logra al agregar ceros al espectro de la imagen. 32
Fig. I.26 Imágenes amplificadas mediante el método de inserción de ceros en el dominio de la frecuencia. 33
Fig. I.27 Aumento en el rango dinámico de una imagen usando una función de transformación s=T(r) 34
Fig. I.28 (a) Histograma de una imagen y su correspondiente histograma acumulativo. (b) Histograma deseado y su correspondiente histograma acumulativo. 37
Fig. I.29 Función de transformación e histograma de la imagen procesada. 37
Fig. I.30 Ecualización de una imagen de 240 x 291 píxeles a 256 niveles de gris. 38
Fig. I.31 Ecualización de una imagen de 400 x 378 píxeles a 256 niveles de gris. 39
Fig. I.32 Ejemplo del proceso de codificación Huffman. 42
Fig. I.33 Descorrelación de una imagen para ser comprimida posteriormente. 45

Fig. I.34 Descorrelación de una imagen para ser comprimida posteriormente. 46
Fig. II.1 Localización de las ondículas discretas en el espacio-tiempo-escala en un diagrama diádico. 54
Fig. II.2 Espectros traslapados de las ondículas resultantes del escalamiento de la ondícula madre en el dominio del tiempo. 56
Fig. II.3 Conjunto infinito de ondículas reemplazado por una función de escalamiento. 57
Fig. II.4 División del espectro de una señal por medio de un banco de filtros. 60
Fig. II.5 Implementación de las ecuaciones II.23 y II.24 como una etapa de un banco de filtros iterativo. 64
Fig. III.1 Relación entre los coeficientes en diferentes sub-bandas como árboles cuaternarios. 67
Fig. III.2 (a) Relación entre los coeficientes en diferentes sub-bandas. (b) Orden de rastreo. (c) Símbolos. 69
Fig. III.3 (a) Tipo raster, (b) Tipo Morton. 69
Fig. III.4 Datos del ejemplo presentado. 73
Fig. IV.1 Interconexión anfitrión-destino. 75
Fig. IV.2 Muelle de comunicación anfitrión-destino. 77
Fig. IV.3 Imagen de Lena binarizada. 81
Fig. IV.4 Imagen del camarógrafo binarizada. 81
Fig. IV.5 Detección de bordes de la imagen de Lena. 82
Fig. IV.6 Detección de bordes de la imagen del camarógrafo. 83
Fig. IV.7 Imagen de Shania pasada por el filtro promediador 3x3. 85
Fig. IV.8 Imagen de las flores pasada por el filtro promediador 3x3. 86
Fig. IV.9 Pantalla de captura de los coeficientes del filtro (programa anfitrión). 87
Fig. IV.10 Imagen de Shania procesada con el filtro IV.4. 87
Fig. IV.11 Imagen de las flores procesada con el filtro IV.5. 88
Fig. IV.12 Primer nivel del banco de filtros de análisis implementado. 90
Fig. IV.13 Arreglos hechos a la imagen para obtener la convolución cíclica. 91
Fig. IV.14 Imagen de Lena procesada por el banco de filtros biortogonales 5.5. 94
Fig. IV.15 Banco de filtros de síntesis para un nivel de descomposición. 95
Fig. IV.16 Formato de la imagen obtenida durante la etapa de síntesis. 99
Fig. IV.17 Sistema implementado para la prueba de reconstrucción perfecta. 100
Fig. IV.18 Imagen del camarógrafo procesada por el sistema de reconstrucción perfecta. 101
Fig. IV.19 Imagen de las monedas procesada por el sistema de reconstrucción perfecta. 101
Fig. IV.20 Sistema implementado para la compresión-descompresión de imágenes. 103
Fig. IV.21 Imagen de Lena procesada por compresión-descompresión utilizando dos niveles de descomposición. 104
Fig. IV.22 Imagen de Lena procesada por compresión-descompresión utilizando cuatro niveles de descomposición. 104
Fig. IV.23 Imagen del camarógrafo procesada por compresión-descompresión utilizando tres niveles de descomposición. 105
Fig. IV.24 Imagen del camarógrafo procesada por compresión-descompresión utilizando cinco niveles de descomposición. 106

Fig. B. 1 Sistema visual humano descompuesto en dos subsistemas. 111
Fig. B. 2 Modelo simple del nivel periférico del sistema visual humano..... 112
Fig. B. 3 Dos estímulos utilizados en el experimento de discriminación de intensidad. 113
Fig. B. 4 Gráfica de $\Delta I / I$ en función a I..... 114
Fig. B. 5 Imagen de 512x512 pixeles degradada con ruido blanco Gausiano con media cero..... 115
Fig. B. 6 Los dos estímulos utilizados en el estudio de la sensibilidad a la intensidad.. 116
Fig. B. 7 Gráfica de $\Delta I / I$ como una función de I e I_0..... 116
Fig. C. 1 Fotografía del DSK 'C6211..... 118
Fig. D. 1 Tiempos de procesamiento en el proceso de binarización..... 119
Fig. D. 2 Tiempos de procesamiento en el proceso de detección de contornos. 120
Fig. D. 3 Tiempos de procesamiento en el proceso de filtro promediador..... 121
Fig. D. 4 Tiempos de procesamiento en el proceso de filtro biortogonal 5.5 (Análisis). 121
Fig. D. 5 Tiempos de procesamiento en el proceso de compresión – descompresión de imágenes. 122

LISTA DE TABLAS

Tabla I.1 Comparación de los números de operaciones para diferentes tamaños de bloque para procesar una imagen de 256x256 píxeles. 23
Tabla IV.1 Funciones de la biblioteca dsk6211.dll para el manejo de la interfaz HPI..... 76
Tabla IV.2 Palabras de control..... 77
Tabla A. 1 Coeficientes para los filtros biortogonales 5.5 en la etapa de análisis..... 109
Tabla A. 2 Coeficientes para los filtros biortogonales 5.5 en la etapa de síntesis..... 110

LISTA DE SÍMBOLOS Y ACRÓNIMOS

TDC	Transformada Discreta Coseno
MSE	Error Cuadrático Medio
NMSE	Error Cuadrático Medio Normalizado
SNR	Relación Señal a Ruido
PSNR	Relación Señal a Ruido Pico
TDF	Transformada Discreta de Fourier
TRF	Transformada Rápida de Fourier
fdp	Función de Densidad de Probabilidad
FPA	Función de Probabilidad Acumulativa
p(a)	Probabilidad de que ocurra el evento *a*
TO	Transformada de Ondículas
TCO	Transformada Continua de Ondículas
TDO	Transformada Discreta de Ondículas
$\downarrow 2$	Submuestreo por un factor de 2
$\uparrow 2$	Sobremuestreo por un factor de 2
COAC	Codificador de Ondículas por Árboles de Ceros
JPEG	Formato estándar de Imágenes Comprimidas (Joint Photographic Experts Group)
P	Código Positivo (utilizado en el COAC)
N	Código Negativo (utilizado en el COAC)
T	Código Árbol de Ceros (utilizado en el COAC)
Z	Código Cero (utilizado en el COAC)
SPDI6211	Sistema de Procesamiento Digital de Imágenes para el DSK 6211
DSK	Paquete de Introducción al Procesamiento Digital de Señales (DSP Starter Kit)
PDS	Procesamiento Digital de Señales
PC	Computadora Personal
EMIF	Interfaz de Memoria Externa

	(External Memory Interface)
LPT	Puerto Paralelo de la Computadora Personal
DLL	Biblioteca de Enlace Dinámico
	(Dynamic Link Library)
HPI	Interfaz del Puerto Anfitrión
	(Host Port Interface)
CODEC	Codificador-Decodificador
MIPS	Millones de Instrucciones por Segundo
RAM	Memoria de Acceso Aleatorio
	(Random Access Memory)
ROM	Memoria de Solo Lectura
	(Read Only Memory)
SDRAM	Memoria de Acceso Aleatorio Dinámica Síncrona
	Synchronous Dynamic Random Access Memory

"Técnicas de Procesamiento Digital de Imágenes utilizando un Procesador Digital de Señales"

RESUMEN

Este trabajo muestra algunas de las técnicas de procesamiento digital de imágenes más utilizadas que van desde la teoría y principios básicos hasta la implementación de esas técnicas en el procesador digital de señales TMS320C6211 de Texas Instruments. Las técnicas que fueron implementadas en ésta plataforma son: binarización, filtrado bidimensional, detección de bordes, y compresión de imágenes utilizando bancos de filtros biortogonales 5.5 y el codificador de ondículas por árboles de ceros.

"Digital Image Processing Techniques using a Digital Signal Processor"

ABSTRACT

This work shows some of the most commonly used digital image processing techniques from theory and basic principles through the implementation of these techniques on the TMS320C6211 Digital Signal Processor from Texas Instruments. The techniques that were implemented in this platform are: binarization, bidimensional filtering, edge detection and image compression using biorthogonal 5.5 filter banks and the embedded zero tree wavelet coder (EZW).

INTRODUCCIÓN

El interés por las técnicas de procesamiento digital de imágenes nace a principios de los años 1920's cuando las imágenes digitalizadas de eventos mundiales fueron transmitidas por primera vez entre las ciudades de Nueva York y Londres por medio de un cable submarino [6]. Sin embargo, los conceptos de aplicaciones de procesamiento digital de imágenes no se empezaron a divulgar sino hasta mediados de los años 1960's, cuando la tercera generación de computadoras empezó a ofrecer mayor velocidad y capacidad de almacenamiento, los cuales son requeridos para la implementación práctica de los algoritmos. Desde entonces, esta área ha experimentado un crecimiento considerable ya que ha formado parte de estudios interdisciplinarios e investigaciones en áreas como ingeniería, ciencias computacionales, estadística, física, química, biología y medicina. Los resultados de éstas investigaciones le han sumado importancia a las técnicas de procesamiento de imágenes en una variedad de problemas que van desde la restauración y mejora de imágenes astronómicas hasta el procesamiento de huellas dactilares para transacciones comerciales.

Debido a esa importancia se decidió implementar este trabajo de tesis, el cual muestra la simulación e implementación de diversas técnicas de procesamiento de imágenes digitales en el procesador digital de señales TMS320C6211 de *Texas Instruments*. El trabajo se encuentra organizado de la siguiente manera: en el ***Capítulo I*** se muestran las técnicas clásicas de procesamiento de imágenes digitales, además se presentan ejemplos de simulaciones hechas en Matlab, en el ***Capítulo II*** se muestran aspectos teóricos acerca de la *Transformada de Ondículas*, la cual se utilizará posteriormente en la implementación. Enseguida, en el ***Capítulo III*** se muestra el funcionamiento del *Codificador de Ondículas por Árboles de Ceros*, el cual, al igual que la *Transformada de Ondículas* será utilizado en la implementación práctica en el procesador digital de imágenes. Finalmente en el ***Capítulo IV*** se documenta la implementación del sistema de procesamiento digital de imágenes para el DSK 6211 (SPDI6211), el cual incluye algunas de las técnicas clásicas presentadas en el ***Capítulo I*** como son la binarización, filtrado bidimensional y detección de bordes; adicionalmente se incluyeron los procesos de análisis-síntesis de imágenes por medio de un banco de

filtros de reconstrucción perfecta, además se incluyó el proceso de codificación-decodificación con el codificador presentado en el ***Capítulo III***, en este proceso se logró comprimir la imagen, los resultados obtenidos se muestran en el mismo ***Capítulo IV*** junto con sus ejemplos.

Por último se presentan las conclusiones y recomendaciones que se incluyeron con el objetivo de poderle dar seguimiento al presente trabajo.

I TÉCNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES.

I.1 Introducción.

Se presenta un compendio de las técnicas convencionales de procesamiento digital de imágenes más utilizadas en la actualidad, como punto de partida para poder comprender otros métodos más complejos que se verán en los capítulos posteriores, así como la implementación de algunos de estos métodos en el procesador digital de señales DSK6211.

Se dividieron estos métodos de acuerdo al dominio en el cual se lleva a cabo el procesamiento: métodos espaciales, métodos frecuenciales y métodos estadísticos.

Entre los métodos espaciales podemos encontrar el filtrado de imágenes, el cual consiste en pasar una imagen a través de un filtro (bidimensional) con el fin de eliminar ruidos que pudieran deteriorar la calidad de la imagen. Otro de los métodos espaciales que se presentan es el de realce de imágenes, el cual consiste en darle nitidez a las imágenes, este procesamiento se aplica por lo regular cuando se tienen imágenes borrosas; además se incluye el proceso de binarización de imágenes, el cual consiste en asignarle únicamente dos posibles valores al espacio de la imagen y por último, en esta sección se incluye la detección de bordes, la cual se aplica en reconocimiento de patrones.

La sección de métodos frecuenciales, contiene dos algoritmos de procesamiento de imágenes: compresión por transformada coseno discreto en la cual se explica

brevemente cómo implementar un algoritmo de compresión por este método y se demuestra el efecto que tiene el número de coeficientes que se consideran para dicha compresión. También se observa la amplificación de imágenes por el método de inserción de ceros en el dominio de la frecuencia, la cual es una de las técnicas de amplificación más utilizadas.

En cuanto a los métodos estadísticos se incluye el método de ecualización del histograma, el cual consiste en uniformizar los niveles de gris a lo largo del espacio de la imagen, teniendo como resultado un aumento en el contraste de la misma; y el método de compresión sin pérdidas, el cual, como su nombre lo indica consiste en comprimir una imagen pero con el fin de que al volverla a descomprimir que se tenga la misma imagen original (bit a bit). Este método se aplica también en la compresión de datos ya que para los datos se requiere que no exista pérdida alguna.

Cabe señalar que para cada una de las técnicas de procesamiento digital de imágenes que se incluyen en este capítulo se incluyen ejemplos implementados en Matlab (ver apéndice E) los cuales comprueban la funcionalidad de dichas técnicas.

I.2 Métodos espaciales.

En esta sección se muestran algunos de los principales métodos de procesamiento de imágenes digitales implementados en el dominio del espacio, es decir, operando directamente con cada uno de los pixeles y su respectiva intensidad. Entre los métodos que se mencionarán están los siguientes: filtrado de imágenes, binarización de imágenes y detección de bordes.

I.2.1 Filtrado de imágenes.

Las imágenes están expuestas a diferentes tipos de ruido, algunos de estos ruidos son independientes de la imagen, pero otros no lo son. Cuando una imagen se transmite, el canal introduce ruido que generalmente es independiente de la intensidad de la imagen en la señal. La situación es similar a cuando se digitaliza una señal de televisión. En estos casos la imagen ruidosa puede escribirse como **g=f+v**, donde **v** es el ruido y **f** la imagen original, ambos descorrelacionados uno del otro [6]. En este caso el efecto indeseable

puede ser disminuido aprovechando las características estadísticas de esta clase de ruido (que puede tener una distribución normal o uniforme, con media igual a cero), al utilizar un promediador. Un sencillo pero eficiente promediador puede lograrse al convolucionar una imagen con la plantilla mostrada a continuación [19]:

$$\begin{bmatrix} 1/NM & 1/NM & \cdots & 1/NM \\ 1/NM & 1/NM & \cdots & 1/NM \\ \vdots & \vdots & \ddots & \vdots \\ 1/NM & 1/NM & \cdots & 1/NM \end{bmatrix} \tag{I.1}$$

La plantilla anterior hace las veces de un filtro digital pasa bajas de orden N x M, donde N y M son las dimensiones de la imagen a procesar. Cuando se requiere de un filtrado de orden superior no es práctico utilizar un simple promediador ya que provocaría deterioro de la imagen, en vez de esto se utilizan otros filtros pasa bajas los cuales mantienen las características de las imágenes.

La herramienta matemática para efectuar un procesamiento en el dominio espacial es la convolución bidimensional, para el caso continuo está definida como sigue [16]:

$$g(x,y) = f * h = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\alpha,\beta)h(x-\alpha, y-\beta)d\alpha d\beta, \tag{I.2}$$

para imágenes discretas se efectúa mediante la ecuación siguiente:

$$g(m,n) = \mathbf{f} * \mathbf{h} = \sum_{j=0}^{M_h-1} \sum_{k=0}^{N_h-1} f(m-j, n-k)h(j,k) \tag{I.3}$$

donde **f** representa la imagen a ser convolucionada, y **h** es el filtro. Una representación geométrica de la convolución bidimensional se muestra en la figura I.1.

Cuando **f** tiene dimensiones de (M_f x N_f) y **h** es de dimensiones (M_h x N_h) la imagen resultante de la convolución será de dimensiones

$$[(M_f + M_h - 1) \text{ x } (N_f + N_h - 1)]. \tag{I.4}$$

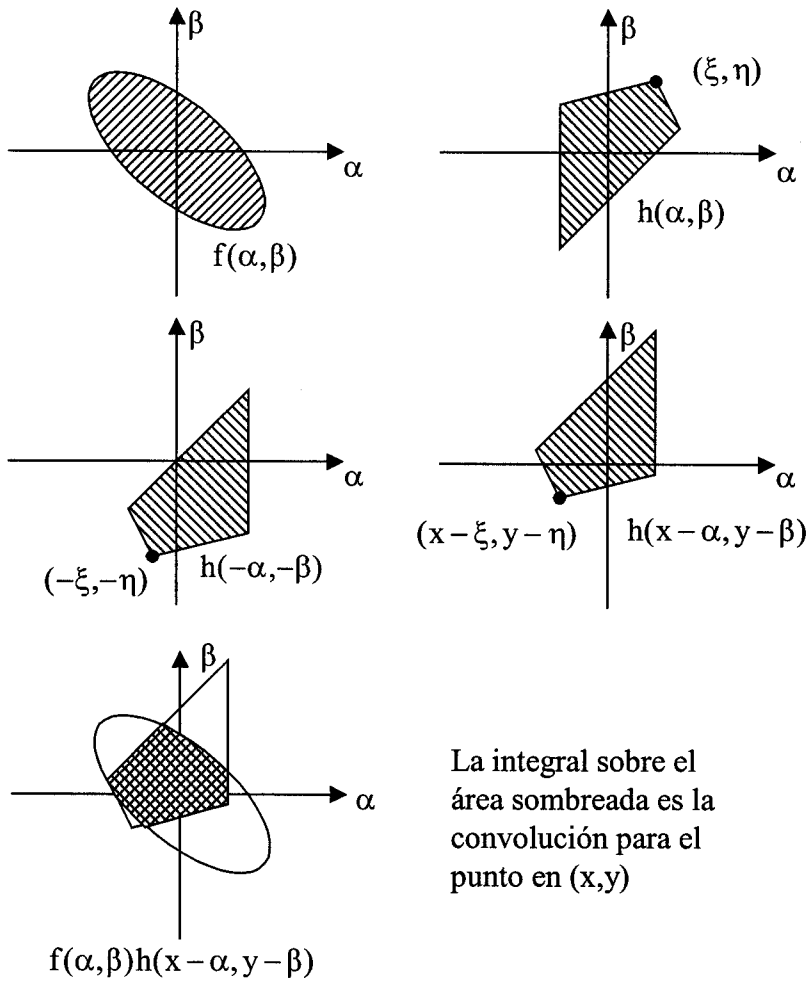


Fig. I.1 Representación gráfica de la convolución bidimensional.

I.2.1.1 Filtros bidimensionales.

La construcción de un filtro bidimensional puede realizarse a partir de dos filtros unidimensionales cada uno con sus respectivas frecuencias de corte, un filtro $\mathbf{h_1}$ para la dirección vertical y un filtro $\mathbf{h_2}$ para la dirección horizontal [9]. Considerando que la convolución de tres matrices $\mathbf{f}$, $\mathbf{h_1}^T$, $\mathbf{h_2}$ es igual a $\mathbf{f}*(\mathbf{h}_1^T*\mathbf{h}_2)$, donde $(\mathbf{h}_1*\mathbf{h}^T{}_2)$ corresponde al filtro bidimensional, es posible construir éste a partir de $\mathbf{h_1}$ y $\mathbf{h_2}$. En la figura I.2 se muestra gráficamente este proceso.

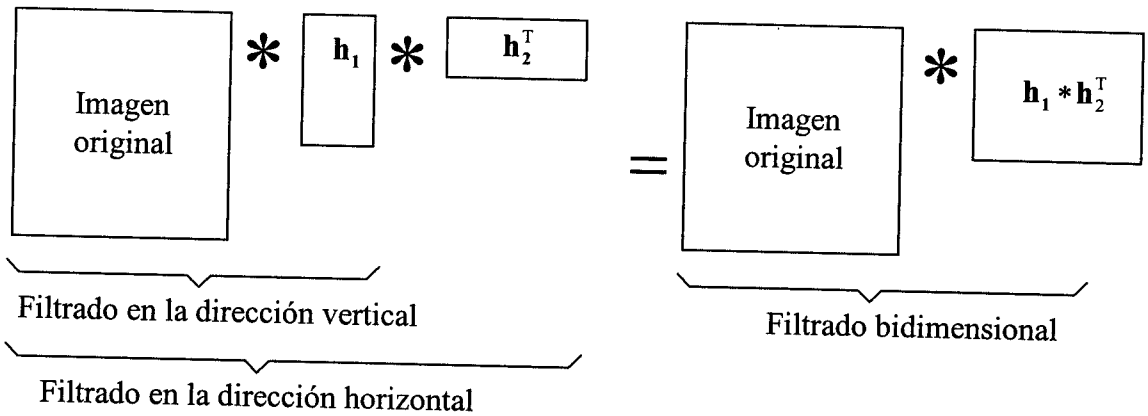


Fig. I.2 Construcción de un filtro bidimensional a partir de dos filtros unidimensionales

A continuación se demuestra que la convolución de dos vectores, uno columna y otro renglón, puede realizarse con una multiplicación matricial.
Considerando $\mathbf{h_1}$ y $\mathbf{h_2}^T$ con dimensiones $(K_a \times 1)$ y $(1 \times K_b)$ respectivamente.
Haciendo $\mathbf{c} = \mathbf{h}_1 * \mathbf{h}_2^T$, donde $\mathbf{c}$ es el filtro bidimensional. De la ecuación I.3 se tiene

$$c(m,n) = \sum_{j=0}^{I-1} \sum_{k=0}^{K_b-1} h_1(m-j, n-k) h_2^T(j,k) \tag{I.5}$$

$$c(m,n) = \sum_{k=0}^{K_b-1} h_1(m-0, n-k) h_2^T(0,k) \tag{I.6}$$

a consecuencia de que $\mathbf{h_1}$ es un vector columna se tiene:

$$h_1(m,x) = \begin{cases} a_x \ ; & \text{para } x = 0 \\ 0 \ ; & \text{para } x \neq 0 \end{cases} \tag{I.7}$$

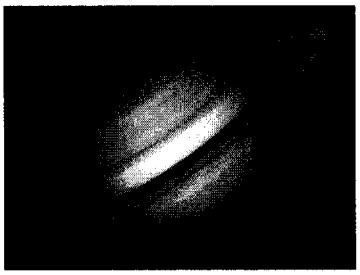
entonces la sumatoria en I.6 tendrá solamente un elemento (cuando k=n), o sea:

$$c(m,n) = h_1(m,0) h_2^T(0,n) . \tag{I.8}$$

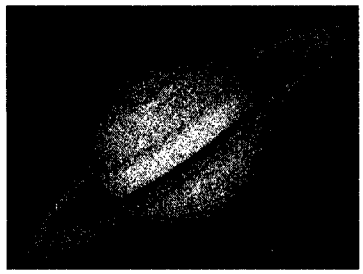
Lo anterior es equivalente a multiplicar un vector columna $\mathbf{h_1}$, por un vector renglón $\mathbf{h_2}^T$.

Ejemplos.

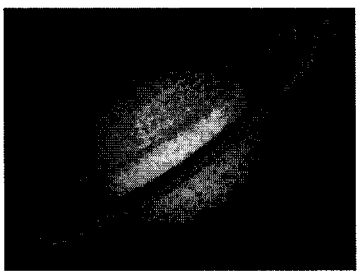
Se desarrollaron dos ejemplos de filtrado de imágenes los cuales consistieron en tomar las imágenes, agregarles ruido Gausiano y pasarlas por un filtro promediador de un orden determinado.



a) Imagen original



b) Imagen con ruido

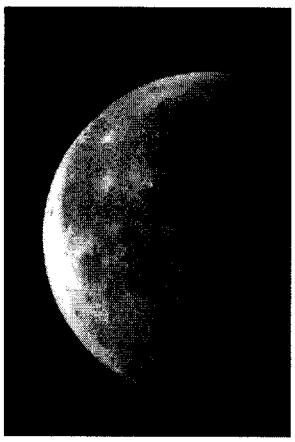


c)Imagen filtrada

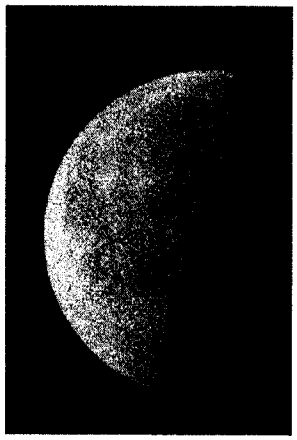
Fig. I.3 Filtrado de la imagen de Saturno.

En este ejemplo a la imagen del planeta Saturno se le agregó ruido Gausiano con media cero y variancia de 0.05, posteriormente se pasó por un filtro promediador de orden 5 x 5 (utilizando la plantilla I.1 con M=5 y N=5), como se puede observar se redujo considerablemente la cantidad de ruido mediante el uso de dicho filtro.

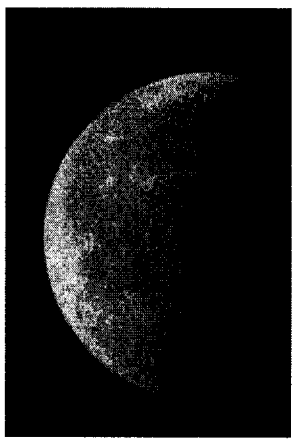
En el segundo ejemplo se procesó la imagen de la Luna agregándosele un ruido con las mismas características que en el ejemplo anterior, en este caso se pasó la imagen por un filtro de orden 3 x 3. Se observa que con un filtro de menor orden se elimina menor cantidad de ruido (ver figura I.4).



a) Imagen original



b) Imagen con ruido



c)Imagen filtrada

Fig. I.4 Filtrado de la imagen de la Luna.

I.2.2 Realce de imágenes.

Una imagen borrosa involucra una operación de promediado ó integración, lo que sugiere que puede ser invertido por diferenciación. En una imagen borrosa sus componentes de alta frecuencia fueron atenuadas por algún proceso, esta imagen puede hacerse más nítida resaltando sus altas frecuencias.

I.2.2.1 Diferenciación

Cualquier derivada $D = \partial^n / \partial x^k \partial y^{n-k}$, es un operador lineal y una combinación de D's también es un operador lineal. Una combinación arbitraria de D's es un operador local, porque su valor para una imagen **f** en un punto (x,y) depende solamente de **f** en una pequeña región. Esto implica que los operadores anteriores son invariantes al desplazamiento. Por lo tanto, se supondrá que todas las derivadas de **f** existen y son continuas [6].

Un operador isotrópico implica que es invariante a la rotación, es decir, al aplicar el operador a la imagen **f** rotada se obtiene el mismo resultado que al aplicar el operador a la imagen **f** original y posteriormente rotar el resultado. Se desean operadores isotrópicos debido a que se quiere resaltar características, tales como bordes y líneas, las cuales tienen una dirección.

La construcción de un operador derivativo isotrópico se ilustra con un ejemplo. Una rotación es definida por las ecuaciones de transformación [9]:

$$x = x'\cos\theta - y'\operatorname{sen}\theta,\ y = x'\operatorname{sen}\theta + y'\cos\theta \tag{I.9}$$

donde (x,y) son las coordenadas sin rotar y (x',y') son las coordenadas rotadas. La primera derivada parcial de **f** en el sistema (x',y') está dada en términos de las primeras derivadas parciales en el sistema (x,y), por:

$$\frac{\partial f}{\partial x'} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x'} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\operatorname{sen}\theta \tag{I.10}$$

$$\frac{\partial f}{\partial y'} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial y'} = -\frac{\partial f}{\partial x}\operatorname{sen}\theta + \frac{\partial f}{\partial y}\cos\theta\,, \tag{I.11}$$

éstas derivadas parciales por sí mismas no son invariantes a la rotación, pero la suma de sus cuadrados sí es invariante, ya que es fácil comprobar que [9]:

$$\left(\frac{\partial f}{\partial x'}\right)^2 + \left(\frac{\partial f}{\partial y'}\right)^2 = \left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2. \tag{I.12}$$

Como se puede observar en las ecuaciones I.10-11, la derivada parcial de una imagen en una dirección arbitraria (por ejemplo $\partial f/\partial x'$) es una combinación lineal de sus derivadas parciales en las direcciones x-y. De hecho, dadas las derivadas en dos

direcciones no colineales, no necesariamente perpendiculares, se puede expresar la derivada en cualquier otra dirección como una combinación lineal de ellas.

Puede encontrarse la dirección en la cual la derivada parcial de **f** tiene un valor máximo, por diferenciación de $\partial f/\partial y'$ con respecto de θ, se obtiene:

$$-\frac{\partial f}{\partial x}\operatorname{sen}\theta + \frac{\partial f}{\partial y}\cos\theta = 0 \tag{I.13}$$

entonces se obtiene la dirección máxima deseada, que es:

$$\theta_n = \tan^{-1}\left(\frac{\partial f}{\partial x}\Big/\frac{\partial f}{\partial y}\right). \tag{I.14}$$

Nótese que $\theta_n+\pi$ también es una solución, en una dirección $\partial f/\partial y'$ es máxima y en la otra es mínima, debido a que se tienen magnitudes iguales pero signos opuestos, es decir, $\partial f/\partial(-y') = -\,\partial f/\partial y'$. Para θ_n se tiene (usando la ecuación I.14):

$$\cos\theta_n = \frac{1}{\sqrt{1+\tan^2\theta_n}} = \frac{\dfrac{\partial f}{\partial x}}{\sqrt{\left(\dfrac{\partial f}{\partial x}\right)^2 + \left(\dfrac{\partial f}{\partial y}\right)^2}} \tag{I.15}$$

así mismo,

$$\operatorname{sen}\theta_n = \sqrt{1-\cos^2\theta_n} = \frac{\dfrac{\partial f}{\partial y}}{\sqrt{\left(\dfrac{\partial f}{\partial x}\right)^2 + \left(\dfrac{\partial f}{\partial y}\right)^2}}, \tag{I.16}$$

entonces el máximo valor de $\partial f/\partial x'$ está dado por:

$$\frac{\partial f}{\partial x}\cos\theta_n + \frac{\partial f}{\partial y}\operatorname{sen}\theta_n = +\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{I.17}$$

El vector con magnitud $\sqrt{(\partial f/\partial x)^2 + (\partial f/\partial y)^2}$, y con dirección θ_n (ó $\theta_n+\pi$ cualquiera que dé la derivada parcial positiva) es llamado el gradiente de **f**. Como se dijo anteriormente, la magnitud del gradiente es un operador derivativo invariante a la rotación. El gradiente se utilizará después en la detección de bordes.

Para imágenes digitales las primeras derivadas en las direcciones de x,y pueden implementarse por las siguientes ecuaciones diferencia:

$$\Delta_x f(i,j) \equiv f(i,j) - f(i-1,j) \tag{I.18}$$

$$\Delta_y f(i,j) \equiv f(i,j) - f(i,j-1) \tag{I.19}$$

Las primeras derivadas en otras direcciones pueden definirse como una combinación lineal de las anteriores, por ejemplo:

$$\Delta_\theta f(i,j) \equiv \Delta_x f(i,j)\cos\theta + \Delta_y f(i,j)\mathrm{sen}\theta \tag{I.20}$$

Usando éstas definiciones, la máxima dirección diferencial también llamada gradiente digital, tiene una magnitud y una dirección dadas respectivamente por:

$$\sqrt{(\Delta_x \mathbf{f})^2 + (\Delta_y \mathbf{f})^2} \quad \text{y} \quad \tan^{-1}\left(\frac{\Delta_y \mathbf{f}}{\Delta_x \mathbf{f}}\right). \tag{I.21}$$

Derivadas de orden superior pueden definirse mediante derivadas sucesivas de la primera derivada, es decir:

$$\begin{aligned} \Delta_x{}^2 f(i,j) &\equiv \Delta_x f(i+1,j) - \Delta_x f(i,j) \\ &= [f(i+1,j) - f(i,j)] - [f(i,j) - f(i-1,j)] \\ &= f(i+1,j) + f(i-1,j) - 2f(i,j) \end{aligned} \tag{I.22}$$

y en forma similar

$$\Delta_y{}^2 f(i,j) = f(i,j+1) + f(i,j-1) - 2f(i,j) \tag{I.23}$$

I.2.2.2 El Laplaciano

El Laplaciano es un operador derivativo lineal y está definido por la siguiente expresión [11]:

$$\nabla^2 \mathbf{f} \equiv \frac{\partial^2 \mathbf{f}}{\partial x^2} + \frac{\partial^2 \mathbf{f}}{\partial y^2}, \tag{I.24}$$

el Laplaciano es invariante a la rotación. Para mostrar su relevancia en el mejoramiento de la nitidez de una imagen, supóngase que una imagen se hace borrosa debido a un proceso de difusión que satisface la ecuación

$$\frac{\partial \mathbf{g}}{\partial t} = k\nabla^2 \mathbf{g}, \tag{I.25}$$

donde **g** es una función de x, y, t (tiempo), con k constante y mayor que cero. En t=0, g(x,y,0) es la imagen no borrosa f(x,y); para un tiempo $t=\tau > 0$, se tiene una imagen borrosa $g(x,y,\tau)$. Si se expande g(x,y,t) en series de Taylor alrededor de $t=\tau$, se tiene

$$g(x,y,0) = g(x,y,\tau) - \tau\frac{\partial g}{\partial t}(x,y,\tau) + \frac{\tau^2}{2}\frac{\partial^2 g}{\partial t^2}(x,y,\tau) - \cdots \tag{I.26}$$

Si se desprecian los términos de orden superior, y se substituye **f** por g(x,y,0) y $k\nabla^2\mathbf{g}$ por $\partial\mathbf{g}/\partial t$, se tiene

$$\mathbf{f} = \mathbf{g} - k\tau\nabla^2\mathbf{g}. \tag{I.27}$$

Con ésta primera aproximación se puede restaurar la imagen **f** no-borrosa restando de **g** un múltiplo positivo de su Laplaciano.

Para una imagen digital, el Laplaciano se puede obtener con la siguiente ecuación [6]:

$$\begin{aligned}\nabla^2 f(i,j) &\equiv \Delta_x{}^2 f(i,j) + \Delta_y{}^2 f(i,j)\\ &= [f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1)] - 4f(i,j)\end{aligned} \tag{I.28}$$

El efecto del Laplaciano digital se puede entender mejor con un ejemplo, considérese una imagen **f** de dimensiones (1xn) que contiene la siguiente secuencia de niveles de gris

..., 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3, ...

el Laplaciano de esta secuencia se obtiene de la segunda derivada respecto a la única variable independiente debido a que la secuencia es unidimensional, se tiene que $\nabla_x^2\mathbf{f}=\partial^2\mathbf{f}/\partial x^2$, la cual se planteó anteriormente como $\nabla_x^2 f(i,j)=f(i+1,j)+f(i-1,j)-2f(i,j)$.

...,0 ,0 ,1 ,0 ,0, 0, 0, -1, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0, -3, 3, 0, 0, 0, ...

Nótese que se tiene un resultado de cero en intervalos que son constantes o rampas, debido al operador diferencial de segundo orden, y un resultado distinto de cero en el inicio y fin de una rampa, también se obtienen resultados diferentes de cero en cada lado (inicio y fin) de un escalón. Si se resta $\nabla_x^2\mathbf{f}$ de **f**, punto por punto, se obtiene:

..., 0, 0, -1, 1, 2, 3, 4, 6, 5, 5, 5, 5, 4, 7, 6, 6, 6, 6, 9, 0, 3, 3, 3, ...

Se ha creado un nivel de gris de sobreimpulso negativo en la parte de abajo de la rampa y en el lado de bajada de los bordes, y un sobreimpulso positivo en la parte de arriba de la rampa y en el lado de subida de los bordes. Lo anterior tiene el efecto de incrementar la pendiente promedio de las rampas e incrementar el contraste en los bordes. De lo anterior se deduce que, en éste caso, la substracción del Laplaciano de la imagen tiene un efecto de aclaramiento.

En dos dimensiones se tiene [6]:

$$f(i,j) - \nabla^2 f(i,j) = 5f(i,j) - [f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1)] \qquad (I.29)$$

Cuando el punto (i,j) se encuentra a la mitad de una rampa ó en una región plana se tiene $\mathbf{f} - \nabla^2 \mathbf{f} = \mathbf{f}$. Si (i,j) está en la parte de abajo de una rampa ó en la bajada de un borde, entonces algunos de los niveles de gris vecinos de f(i,j) son más altos mientras que otros son más bajos, teniendo $(\mathbf{f}-\nabla^2\mathbf{f})<\mathbf{f}$ en (i,j), lo cual produce un sobreimpulso negativo. De forma similar en la parte de arriba de una rampa ó en el lado de subida de un borde ninguno de los vecinos es mayor que f(i,j), y algunos son menores, teniéndose $(\mathbf{f}-\nabla^2\mathbf{f})>\mathbf{f}$ en (i,j), con lo que se produce un sobreimpulso positivo. Un ejemplo sencillo que ilustra este fenómeno se muestra en la figura I.5.

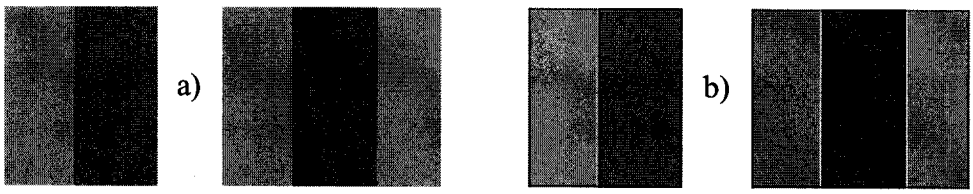


Fig. I.5 a) Imágenes originales, b) Después de restarles el Laplaciano (realce de bordes)

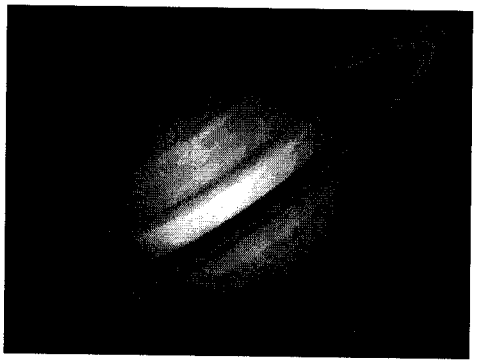
El Laplaciano de una imagen se obtiene aplicando la ecuación I.28 o por convolución al utilizar la plantilla siguiente ($\mathbf{g} = \mathbf{f} * \mathbf{h}$) [6]:

$$\mathbf{h}_L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad (I.30)$$

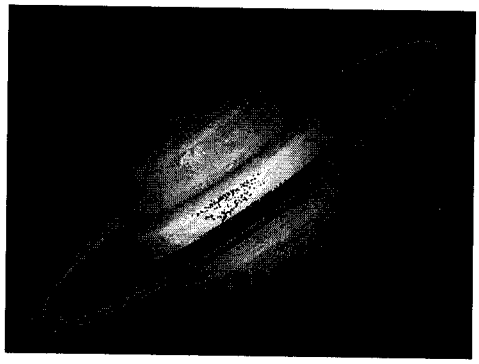
De forma similar (partiendo de la ecuación I.29) se puede calcular $\mathbf{g}=\mathbf{f}-\nabla^2\mathbf{f}$ al convolucionar $\mathbf{f}$ con $\mathbf{h_D}$ definida por [6]:

$$\mathbf{h}_D = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad (I.31)$$

Ejemplos

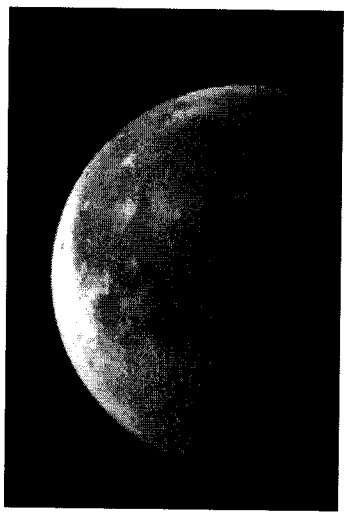


a)Imagen original

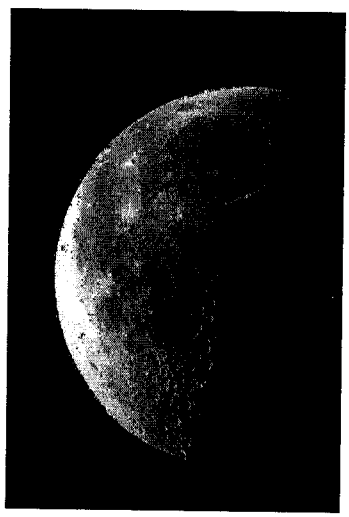


b)Imagen procesada

Fig. I.6 Imagen de Saturno procesada mediante el realce de bordes.



a) Imagen original



b)Imagen procesada

Fig. I.7 Imagen de la Luna procesada mediante el realce de bordes.

I.2.3 Binarización de imágenes.

Una gran cantidad de imágenes están compuestas por dos clases de regiones las cuales ocupan diferentes niveles de gris. Por ejemplo, en una imagen escrita o impresa,

las letras y símbolos generalmente son más obscuros que el papel, así como en imágenes provenientes de alguna fotografía tomada a una tablilla de circuito impreso.

En los casos anteriores, los niveles de gris del histograma de la imagen presentará dos picos los cuales corresponderán a dos niveles de gris en particular (las letras y el papel, las pistas de cobre y el acetato), esto se puede observar en los ejemplos de las figuras I.8b y I.9b. Un histograma que tiene dos picos es llamado bimodal. Cuando se tiene éste tipo de imágenes se utiliza un proceso de binarización.

El proceso de binarización tiene como objetivo convertir una imagen (que contiene múltiples niveles de gris) a una nueva imagen (la cual sólo contendrá dos niveles), esto se logra mediante la comparación de los píxeles de la imagen con un umbral *T*, éste umbral comúnmente se selecciona de tal manera de que divida a los dos picos más altos del histograma de la imagen a procesar.

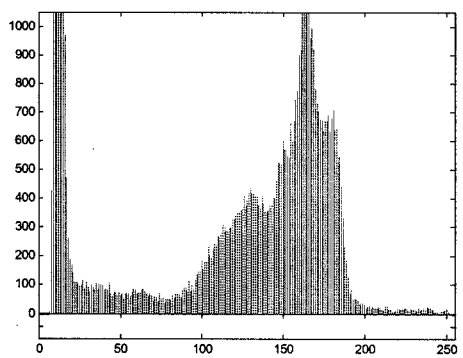
De manera que, si se tiene una imagen original *f* y se pretende obtener una imagen original *g*, se tiene que

$$g(x,y) = \begin{cases} 0 & \text{si } f(x,y) > T \\ 255 & \text{caso contrario} \end{cases} \tag{I.32}$$

El proceso de binarización se utiliza en diversas técnicas de procesamiento, como son: reconocimiento de patrones, detección de áreas en contacto, segmentación, detección de bordes, etc.

Ejemplos

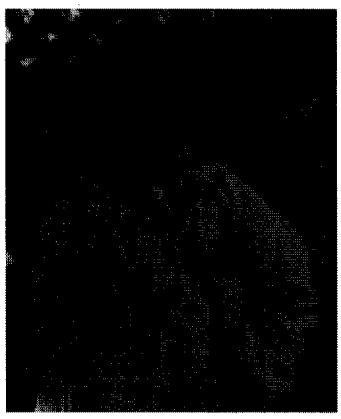
a)Imagen original



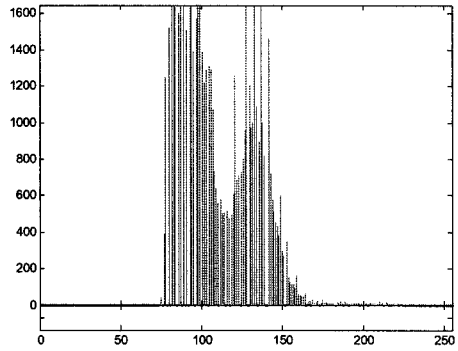
b)Histograma de la Imagen original

c)Imagen binarizada

Fig. I.8 Imagen de 256 niveles de gris binarizada con un umbral de 64.



a)Imagen original



b)Histograma de la Imagen original

c)Imagen binarizada

Fig. I.9 Imagen de 256 niveles de gris binarizada con un umbral de 128.

I.2.4 Detección de bordes

La detección de bordes es una etapa importante en la mayoría de las técnicas de segmentación de imágenes. El propósito de la detección de bordes es identificar las áreas con grandes cambios de intensidad. Estos cambios están asociados con una frontera física (borde) de un objeto en la escena de donde se deriva la imagen. El objetivo final en la segmentación es generar un grupo continuo de píxeles, formando un contorno cerrado alrededor de algún objeto de interés en la imagen.

En la figura I.10 se presenta el diagrama a bloques para un sistema que realiza la detección de bordes [18].

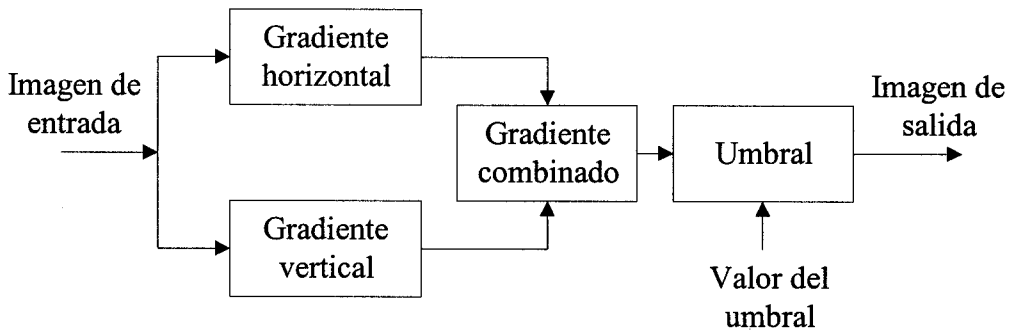
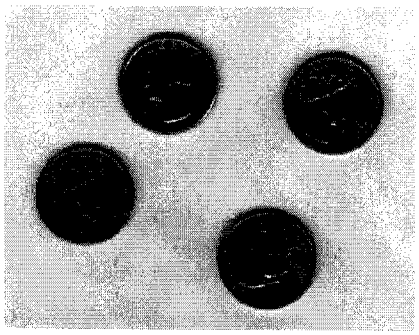


Fig. I.10 Operaciones para la detección de bordes.

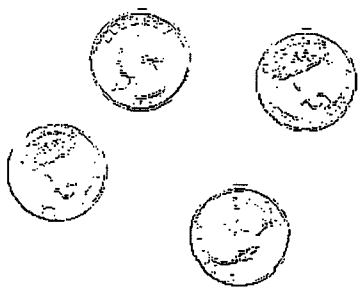
El gradiente horizontal puede ser implementado mediante la ecuación I.18 o convolucionando la imagen con un vector [1 -1], y de igual forma el gradiente vertical puede obtenerse mediante la ecuación I.19 o convolucionando la imagen con el vector $[1\ -1]^T$. Para obtener el gradiente combinado se suman los valores absolutos de los gradientes vertical y horizontal. A continuación se binariza la imagen resultante, obteniéndose una imagen con dos niveles en la cual se identifican fácilmente las áreas que corresponden a los bordes. El criterio que se utiliza para la selección del umbral de binarización se describió en la sección I.2.3.

Si la imagen de entrada contiene ruido es necesario filtrarla mediante un filtro pasa bajas para reducir su efecto, o por el contrario, si en la imagen no se detectan todos los bordes puede ser necesario aplicarle un realce antes de la detección.

Ejemplos

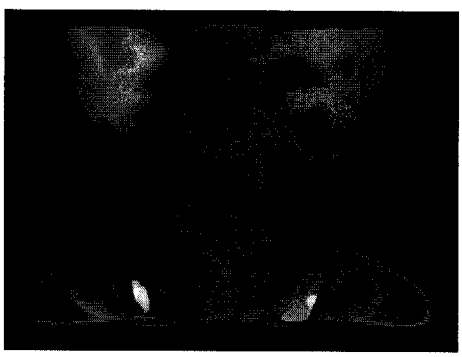


a) Imagen original

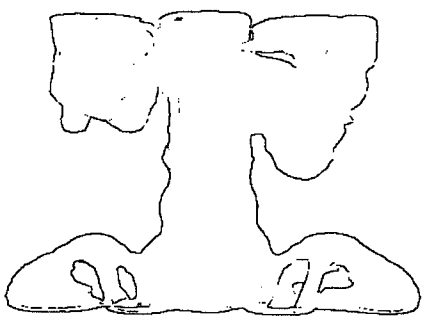


b) Imagen procesada

Fig. I.11 Imagen procesada mediante el algoritmo de detección de bordes utilizando un umbral de 40.



a) Imagen original



b) Imagen procesada

Fig. I.12 Imagen de la espina dorsal procesada mediante el algoritmo de detección de bordes utilizando un umbral de 20.

I.3 Métodos frecuenciales.

En la presente sección se abordarán algunas de las técnicas de procesamiento de imágenes digitales que operan sobre el dominio de la frecuencia de éstas, entre tales técnicas podemos encontrar la compresión de imágenes mediante la Transformada Discreta Coseno, y la amplificación de imágenes mediante la Transformada Discreta de Fourier.

I.3.1 Compresión de imágenes mediante la Transformada Discreta Coseno.

Diferentes métodos de procesamiento se realizan sobre las imágenes, muchos de las cuales eran imposibles anteriormente, bajo éste contexto, surge la necesidad de la compresión de imágenes con el fin de reducir la gran cantidad de información que se genera al digitalizarlas. Como ejemplo, una imagen típica de 512 x 512 píxeles en 256 niveles de gris necesita 262,144 bytes para su representación digital, sin considerar el color (el cual triplica la cantidad de información). Para aplicaciones tales como el almacenamiento o transmisión esta cantidad de información haría ineficiente cualquier sistema, por lo que deben explotarse diferentes características de la imagen para lograr su compresión.

Una forma de lograr la compresión de una imagen es transformarla a un dominio en el cual la energía se encuentre concentrada en regiones, siendo almacenadas después de una cuantización, de forma que sean éstas las que se preserven aún cuando el descartar las regiones de baja concentración de energía introduzca pérdidas. Dentro de éstos métodos se agrupan todas las transformaciones espacio-frecuencia y, dentro de ellas, la transformada discreta coseno (TDC) la cual se abordará en ésta sección.

I.3.1.1 Transformada de una matriz.

En forma general, la transformada de una matriz **f** (representando una imagen) de dimensiones MxN es otra matriz de igual tamaño **F,** definida como el producto de tres matrices [19]:

$$\mathbf{F} = \mathbf{P}\,\mathbf{f}\,\mathbf{Q} \tag{I.33}$$

donde **P** y **Q** son matrices no singulares** de MxM y NxN respectivamente. La transformación estará determinada por la forma en que las matrices **P** y **Q** sean definidas. La transformación **F**, puede ser escrita en forma escalar como sigue [19]:

$$F(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} p(u,m)f(m,n)q(n,v) \tag{I.34}$$

para u=0,1,...,M-1; v=0,1,...,N-1.

** Para una matriz cuadrada existe su inversa si su determinante es diferente de cero, entonces se dice que es no singular.

Dado que las matrices **P** y **Q** no son singulares, existe su inversa. Denotando las matrices inversas de **P** y **Q** por $\mathbf{P}^{-1}$ y $\mathbf{Q}^{-1}$ respectivamente, es posible invertir la transformación con la siguiente ecuación:

$$\mathbf{f} = \mathbf{P}^{-1}\,\mathbf{F}\,\mathbf{Q}^{-1} \tag{I.35}$$

En la expresión anterior, se define la relación inversa y se dice que **f** es la transformada inversa de **F**.

Una matriz es llamada simétrica si la transpuesta es igual a la matriz original, es decir, $\mathbf{T}=\mathbf{T}^{T}$. Cuando la transpuesta de una matriz multiplicada por la original es igual a la identidad, o sea, $\mathbf{T}^{T}\,\mathbf{T} = \mathbf{I}$, se dice que la matriz es ortogonal, entonces si una matriz cuadrada de números reales es simétrica y ortogonal se tiene que $\mathbf{T}=\mathbf{T}^{T}=\mathbf{T}^{-1}$.

I.3.1.2 Transformada Discreta Coseno.

Como ejemplo de transformación ortogonal se puede citar a la transformada discreta coseno, para la TDC se definen las matrices **P** y **Q** como sigue [19]:

$$\mathbf{P} = p(u,m) = \frac{g(u)}{\sqrt{N}}\cos\frac{(2m+1)u\pi}{2N} \tag{I.36}$$

$$\mathbf{Q} = q(n,v) = \frac{g(v)}{\sqrt{N}}\cos\frac{(2n+1)v\pi}{2N} \tag{I.37}$$

para u,v=0,1,...,N-1; m,n=0,1,...,N-1

donde u y v son frecuencias espaciales y g(x) se define como:

$$g(x) = \begin{cases} 1 & \text{si } x = 0 \\ \sqrt{2} & x = 1,2,\ldots,N-1. \end{cases} \tag{I.38}$$

Se puede observar que $\mathbf{Q}^{T}=\mathbf{P}$, la TDC puede ser implementada mediante multiplicaciones matriciales o con la siguiente ecuación escalar (para una imagen de NxN):

$$F(u,v) = \frac{g(u)g(v)}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)\cos\frac{(2m+1)u\pi}{2N}\cos\frac{(2n+1)v\pi}{2N} \tag{I.39}$$

para u,v=0,1,...,N-1

y su inversa se obtiene a través de la ecuación:

$$f(m,n) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} g(u)g(v)F(u,v)\cos\frac{(2m+1)u\pi}{2N}\cos\frac{(2n+1)v\pi}{2N} \qquad (I.40)$$

$$m,n=0,1,\ldots,N-1$$

f(m,n) es la intensidad luminosa del píxel cuyas coordenadas son (m,n).

I.3.1.3 Codificación en bloques y cuantización.

Las transformadas bidimensionales en imágenes cuadradas, tienen un tiempo de computación del orden de N^4, donde N es la longitud de los lados de la imagen, conforme la longitud de uno de los lados de la imagen se incrementa, el tiempo requerido para efectuar una transformada bidimensional se incrementa mucho más rápidamente.

La codificación en bloques divide una imagen en sub-imágenes, entonces, se efectúa una transformada de cada sub-imagen. La tabla I.1 muestra una comparación de las operaciones de transformación para una codificación de imagen completa y la codificación de bloques, para varios tamaños de las sub-imágenes.

Para realizar una transformación a través de la multiplicación de matrices se requieren N^4 multiplicaciones y $(N-1)^4$ sumas [19].

Sin embargo, existe un costo para ésta reducción tan dramática en las operaciones y éste se paga durante el proceso de cuantización. El proceso de cuantización consiste en representar consiste en representar la información original con un menor número de bits de los que se utilizaban originalmente.

Cuando el tamaño del bloque es grande, la degradación durante la cuantización se esparce sobre un espacio grande de la imagen de una forma más o menos pareja. Conforme se reduce el tamaño del bloque, la degradación se vuelve dispareja entre los bloques y los límites de los bloques quedan al descubierto en la imagen recuperada. El sistema de visión humano es más sensible a patrones de interferencia regulares, tales como la de los límites de los bloques que a interferencias aleatorias y dispersas. Como resultado, la calidad de la imagen percibida es menor. Además, debido a que la correlación en imágenes típicas es más alta para píxeles que están más cerca entre sí en la imagen, algo de la independencia de los coeficientes se pierde. Por lo tanto, las medidas cuantitativas de la fidelidad de la imagen también disminuyen.

Tabla I.1 Comparación de los números de operaciones para diferentes tamaños de bloque para procesar una imagen de 256x256 píxeles.

Tamaño del bloque	# de multiplicaciones	# de sumas
64x64	268'435,456	252'047,376
32x32	67'108,864	59'105,344
16x16	16'777,216	12'960,000
8x8	4'194,304	2'458,624
4x4	1'048,576	331,776

Para determinar el tamaño óptimo del bloque, se debe conocer la correlación dentro de los bloques de píxeles adyacentes en la imagen original. En la mayoría de la imágenes típicas existe una correlación significativa para alrededor de 20 píxeles adyacentes. Por lo tanto, conforme el tamaño del bloque se incrementa por encima de 16 x 16, el incremento en la fidelidad es cada vez menos significativo. En los ejemplos que se mostrarán posteriormente se utilizaron sub-imágenes de 8x8.

La razón por la cual los datos de una imagen puedan ser comprimidos en factores grandes, y recuperados después exitosamente con pequeños errores; es la gran cantidad de redundancia en imágenes típicas. Por ejemplo, si un arreglo de números tiene redundancia, es teóricamente posible proporcionar la misma información con menos números. El propósito de llevar a cabo la transformada, es obtener un conjunto de números que representen la imagen pero cuyos valores no están correlacionados (por ejemplo, cada número en el arreglo proporciona nueva información no dada por ningún otro número en el arreglo). Ya que la misma información contenida se va a representar en los arreglos transformado y original, algunos números en el arreglo transformado proporcionan poca información acerca de la imagen original (en la sección I.4.2 se abordará el tema de la cantidad de información), y pueden ser descartados.

La verdadera medida de la utilidad de un coeficiente dado, es su variación sobre un conjunto de imágenes. Si un coeficiente (en el renglón 45 y la columna 99 en una imagen transformada de 256 x 256) mantiene el mismo valor sobre un conjunto de imágenes, entonces no contiene mucha información acerca de la diferencia en las imágenes de este conjunto. Y puede por tanto, ser reemplazado con una constante sin dañar la fidelidad. De forma contraria, si un coeficiente tiene gran variación sobre el

conjunto, no puede ser descartado sin ocasionar serias consecuencias en la imagen recuperada.

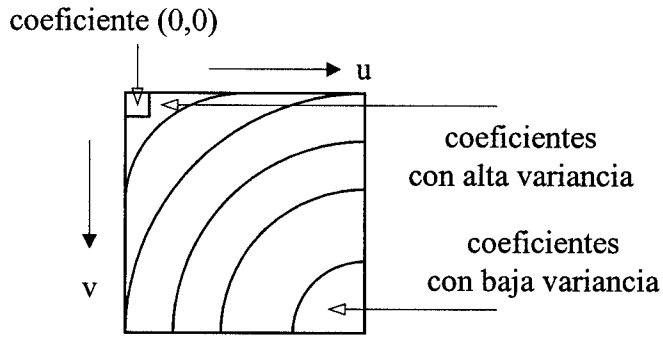


Fig. I.13 Líneas de coeficientes con variancia constante en una imagen típica transformada mediante la TDC.

En la transformada discreta coseno, las variaciones sobre conjuntos típicos de imágenes tienden a tener bordes con variación constante, como se muestra en la figura I.13 los coeficientes con variación alta tienden a estar cerca del origen y de los ejes u y v. Una manera de usar ésta característica, es la de asignar un número de niveles de cuantización, basándose en la variación de los coeficientes. Esto es llamado codificación por zonas, y una asignación típica en una imagen transformada de 8 x 8 y de 16 x 16 se muestran en las figuras I.14 y I.15 respectivamente [19]. La asignación de bits mostrada en la figura I.15 reduce el número de bits requeridos para representar la imagen, de 8 bits por píxel a 1.5 bits por píxel (396 bits/256 píxeles).

En las figuras I.16 y I.17 se muestra el proceso de compresión y descompresión respectivamente.

8	8	4	4	4	4	4	4
8	8	4	4	4	4	0	0
4	4	4	4	0	0	0	0
4	4	4	4	0	0	0	0
4	4	0	0	0	0	0	0
4	4	0	0	0	0	0	0
4	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

Fig. I.14 Asignación en bits de los coeficientes en una codificación por zonas para una imagen de 8x8.

8	8	8	7	7	7	5	5	4	4	4	4	4	4	4	4
8	8	7	6	5	5	3	3	3	3	3	3	2	2	2	2
8	7	6	4	4	4	3	3	2	2	2	2	2	2	2	2
7	6	4	3	2	2	2	2	1	1	1	1	0	0	0	0
7	5	4	2	2	2	2	1	1	1	0	0	0	0	0	0
7	5	4	2	2	2	1	1	0	0	0	0	0	0	0	0
5	3	3	2	2	1	1	0	0	0	0	0	0	0	0	0
5	3	3	2	1	1	0	0	0	0	0	0	0	0	0	0
4	3	2	1	1	0	0	0	0	0	0	0	0	0	0	0
4	3	2	1	1	0	0	0	0	0	0	0	0	0	0	0
4	3	2	1	0	0	0	0	0	0	0	0	0	0	0	0
4	3	2	1	0	0	0	0	0	0	0	0	0	0	0	0
4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. I.15 Asignación en bits de los coeficientes en una codificación por zonas para una imagen de 16x16.

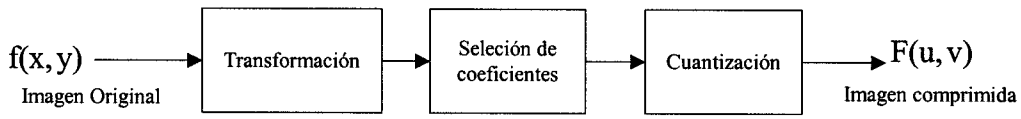


Fig. I.16 Proceso de compresión de una imagen.

Resumiendo hasta este punto, la imagen original fue transformada mediante la TDC para descorrelacionar los datos, posteriormente, se realiza una selección de los coeficientes más significativos en base a su variancia, después de esto, los coeficientes fueron cuantizados. En realidad la compresión se logra al descartar coeficientes y cuantizar a menos bits, por supuesto con una pérdida de calidad en la imagen.

I.3.1.4 Descompresión de imágenes mediante la TDC.

Como se observa en la figura I.17, para recuperar una imagen (una aproximación de la imagen), se requiere invertir el proceso de compresión. Primeramente todos los coeficientes son recuantizados a un mismo número de bits, posteriormente se suponen los coeficientes faltantes, los que fueron descartados, normalmente con ceros para conseguir toda la imagen transformada, a continuación se le aplica la transformada inversa, que como ya se explicó anteriormente consiste en multiplicar por la derecha y por la izquierda con las inversas de las matrices de transformación.

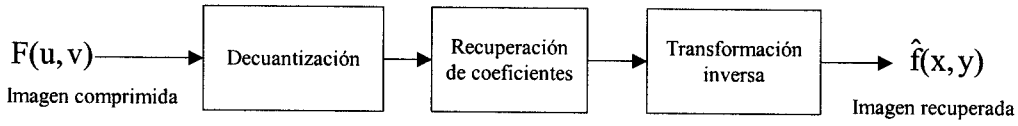


Fig. I.17 Proceso de descompresión de una imagen.

Un método de compresión requiere de evaluaciones subjetivas (comparar visualmente las dos imágenes), y evaluaciones objetivas. Para evaluar objetivamente la calidad de una imagen reconstruida después de haberla comprimido, se utiliza el error cuadrático medio (MSE), el error cuadrático medio normalizado (NMSE), la relación señal a ruido (SNR) y la relación señal a ruido pico (PSNR).

Sean las matrices $\mathbf{X}=[x_{m,n}]$ y $\hat{\mathbf{X}}=[\hat{x}_{m,n}] \in \mathbf{R}^{MxN}$ la imagen original y la imagen reconstruida respectivamente, donde $x_{m,n}$ y $\hat{x}_{m,n} \in \mathbf{R}$ $\forall$ m=0,1,..,M-1; n=0,1,..,N-1 son los mn-ésimos píxeles correspondientes a las imágenes, M es el número de renglones y N el número de columnas de la imagen, L es el máximo nivel de gris en la imagen.
Entonces:

$$\mathrm{MSE} = \frac{1}{\mathrm{NM}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (x_{m,n} - \hat{x}_{m,n})^2 , \tag{I.41}$$

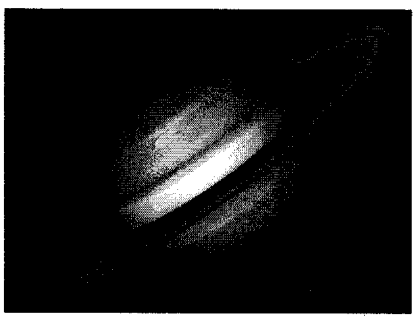
$$\mathrm{NMSE} = \frac{\mathrm{MSE}}{\frac{1}{\mathrm{NM}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n}^2} , \tag{I.42}$$

$$\mathrm{SNR} = 10 \cdot \log_{10}\left(\frac{1}{\mathrm{NMSE}}\right) , \tag{I.43}$$

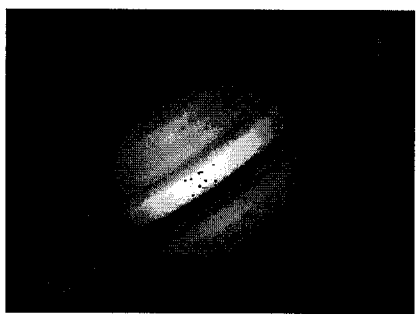
$$\mathrm{PSNR} = 20 \cdot \log_{10}\left(\frac{\mathrm{L}}{\sqrt{\mathrm{MSE}}}\right) . \tag{I.44}$$

En teoría, mientras mayor sea el PSNR, menor es el error entre las imágenes. El PSNR mínimo aceptable al ojo humano para una buena calidad es de 30 dB, lo anterior no siempre es cierto, pues una imagen puede parecer muy buena visualmente y no tener un PSNR muy alto. La evaluación objetiva no puede, por sí misma, determinar completamente la validez de una herramienta para la compresión, pero sí puede dar una idea de cuánto se afectó la imagen reconstruida. En las figuras I.18 y I.19 se muestran dos ejemplos de este tipo de procesamiento.

Ejemplos



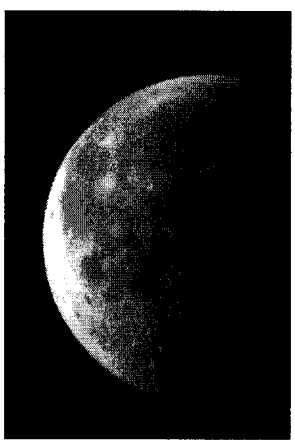
a) Imagen original



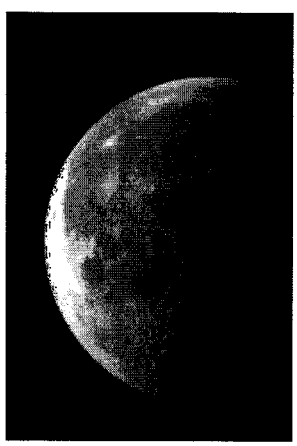
b) Imagen procesada

Fig. I.18 Imagen de Saturno procesada con la TDC tomando 4 coeficientes (compresión 16:1) con un MSE = 4.11E-4, SNR = 24.68 dB, PSNR = 29.13 dB.

En el capítulo IV se presentan ejemplos usando otro esquema de compresión y cuya teoría se expone en el capítulo II (Transformada de Ondículas) y en el capítulo III (Codificador de Ondículas por Árboles de Códigos de Ceros).



a) Imagen original



b) Imagen procesada

Fig. I.19 Imagen de la Luna procesada con la TDC tomando 16 coeficientes (compresión 4:1) con un MSE = 1E-4, SNR = 31.33 dB, PSNR = 32.26 dB.

I.3.2 Amplificación de imágenes mediante la Transformada Discreta de Fourier.

En esta sección se expone un método para amplificar imágenes, basado en la interpolación espacial a través del dominio de la frecuencia.

Definiendo una matriz de transformación Φ de JxJ, como sigue [5]

$$W_J^{mn} = \Phi_{m,n} = \frac{1}{J} e^{-j(2\pi/J)mn}, \quad m,n = 0,1,2,\ldots,J-1 \tag{I.45}$$

con

$$\mathbf{P} = \Phi_{M,M} \quad \text{y} \quad \mathbf{Q} = \Phi_{N,N}, \tag{I.46}$$

la transformada discreta de Fourier (TDF) de **f** está dada por

$$\mathbf{F} = \mathbf{P}\,\mathbf{f}\,\mathbf{Q} \tag{I.47}$$

expandiendo esta expresión, mediante la ecuación I.45 y I.47,

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\exp\left[-j2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)\right] \tag{I.48}$$

rescribiendo la ecuación anterior,

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\exp\left[-j2\pi\frac{um}{M}\right]\exp\left[-j2\pi\frac{vn}{N}\right] \tag{I.49}$$

para u=0,1,2,...,M-1;v=0,1,2,...,N-1.

La ecuación anterior puede implementarse con M transformadas discretas de Fourier unidimensionales de N muestras cada una, de la forma

$$A(m,v) = \frac{1}{M}\sum_{n=0}^{N-1} f(m,n)\exp\left[-j2\pi\frac{vn}{N}\right] \tag{I.50}$$

para m=0,1,2,...,M-1;v=0,1,2,...,N-1.

es decir, transformar cada renglón, y posteriormente aplicar N transformadas discretas de Fourier de M muestras cada una (transformar cada columna de **A**), para obtener la transformada bidimensional, o sea

$$F(u,v) = \frac{1}{N}\sum_{m=0}^{M-1} A(m,v)\exp\left[-j2\pi\frac{um}{M}\right] \tag{I.51}$$

para u=0,1,2,...,M-1;v=0,1,2,...,N-1.

Las ecuaciones I.50 y I.51 pueden implementarse mediante algoritmos de la transformada rápida de Fourier (TRF)unidimensional [5].

La matriz inversa de Φ es otra matriz de JxJ dada por,

$$\left[W_J^{mn}\right]^{-1} = \left[\Phi_{m,n}\right]^{-1} = e^{j(2\pi/J)mn}, \quad m,n = 0,1,2,\ldots,J-1 \tag{I.52}$$

lo anterior puede demostrarse multiplicando las dos matrices Φ y Φ^{-1} obteniéndose:

$$\sum_{m=0}^{J-1} \exp\left[\frac{-j2\pi}{J} km\right] \exp\left[\frac{j2\pi}{J} mn\right] = \begin{cases} J, & k = n \\ 0, & k \neq n \end{cases} \tag{I.53}$$

Recordando la ecuación I.46 se tiene que $\mathbf{P}^{-1} = \left[\Phi_{M,M}\right]^{-1}$ y $\mathbf{Q}^{-1} = \left[\Phi_{N,N}\right]^{-1}$, entonces la relación inversa de la ecuación I.48 está dada por,

$$f(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp\left[j2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)\right] \tag{I.54}$$

para m=0,1,...,M-1; n =0,1,..., N-1

que también puede efectuarse mediante antitransformadas de Fourier unidimensionales.

La TDF convierte una secuencia de entrada en el dominio temporal (o espacial) al dominio de la frecuencia. El objetivo de esta sección es mostrar una aplicación de la TDF para el caso bidimensional, realizando una interpolación en el dominio espacial a través del dominio de la frecuencia. La idea principal es sencilla y parte del caso unidimensional que se describe a continuación.

Es posible realizar una interpolación en el dominio del tiempo, para una secuencia unidimensional, utilizando el espectro de frecuencia. Esto se logra insertando ceros en el centro del espectro, lo que provoca una interpolación en el dominio del tiempo, lo anterior se aclara con el siguiente ejemplo: considerando la secuencia temporal dada en la figura I.20 y cuyo espectro es el dado en la figura I.21,

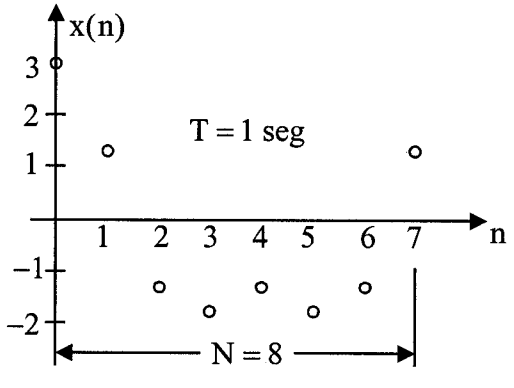


Fig. I.20 Secuencia temporal de 8 muestras tomadas a 1 Hz.

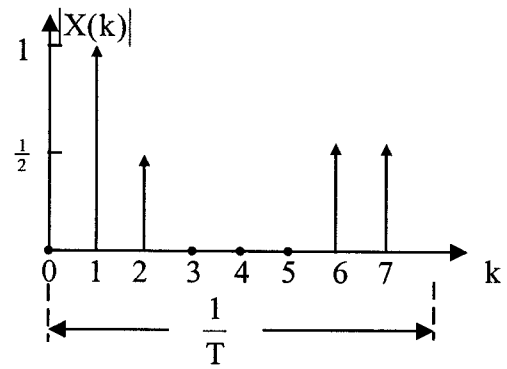


Fig. I.21 Espectro de la secuencia de la figura I.20.

Agregando 24 ceros (para hacer un total de 32 muestras) al espectro X(k) se obtiene: (ver figura I.22)

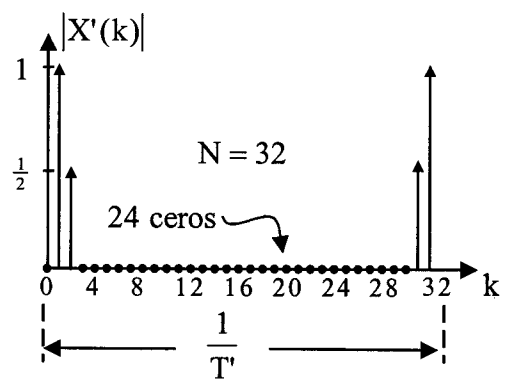


Fig. I.22 Espectro después de agregar 24 ceros.

transformando al dominio del tiempo se obtiene una nueva secuencia x(n') mostrada en la figura I.23

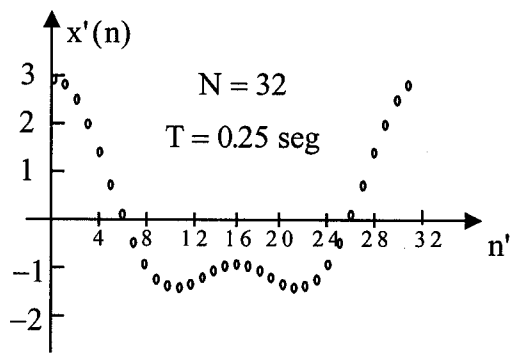


Fig. I.23 Secuencia obtenida de la interpolación.

Nota: las muestras de la nueva secuencia se multiplican por $1/\alpha$, donde α es el factor de interpolación.

Demostración teórica del procedimiento anterior:

Considérese una secuencia x(n) muestreada a una frecuencia 1/T, de la cual se tienen N muestras. Su representación en el dominio de la frecuencia está dada por:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\exp\left[\frac{-j2\pi nk}{N}\right] \tag{I.55}$$

cuando x(n) es real, |X(k)| es simétrico alrededor de N/2, de modo que para lograr una interpolación en el tiempo se insertan (α -1)N ceros al espectro según se muestra en la figura I.24, donde α es el factor de interpolación deseado.

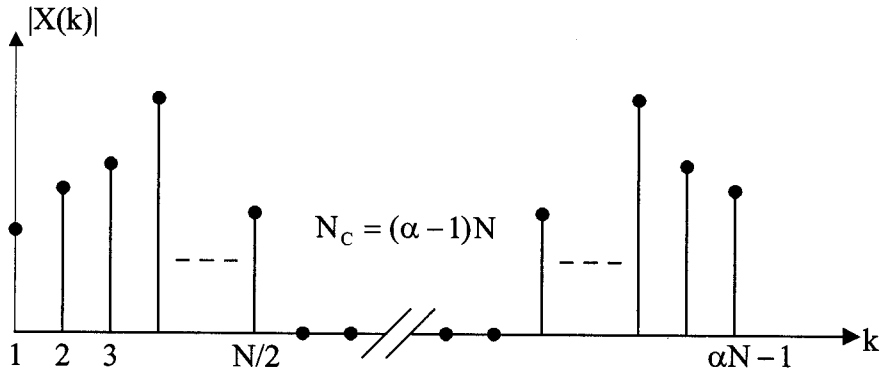


Fig. I.24 Inserción de ceros en la frecuencia para lograr una interpolación en el tiempo.

Entonces antitransformando X(k) se obtiene una nueva secuencia x(n') dada por:

$$x(n') = \frac{1}{\alpha N}\sum_{k=0}^{\alpha N-1} X(k)\exp\left[\frac{j2\pi nk}{\alpha N}\right] \tag{I.56}$$

escribiendo nuevamente la ecuación I.55 y comparando con la antitransformada de X(k) se observa que:

$$x(n') = \frac{1}{\alpha}\frac{1}{N}\sum_{k=0}^{\alpha N-1} X(k)\exp\left[\frac{j2\pi(n/\alpha)k}{N}\right] = \frac{x(n/\alpha)}{\alpha}, \tag{I.57}$$

recordando que x(n) es una forma abreviada de x(nT). Se tiene un escalamiento en el período de muestreo T'=T/α ó un aumento de α veces la frecuencia de muestreo, provocando una interpolación de valores entre las muestras de la secuencia original x(n).

El concepto anterior, puede ser generalizado para secuencias bidimensionales, con lo que se logra la amplificación de una imagen. Debido a que se está utilizando la TRF para pasar de un dominio al otro, la amplificación deberá ser por factores potencias de 2,

así como también el tamaño de la imagen (en píxeles) debe ser de potencias de 2 por lado.

El procedimiento para la amplificación de imágenes es el siguiente [5]:

1. Poner la imagen en una matriz **X** de N x N.
2. Aplicar la TRF a cada renglón de la matriz **X** y almacenar el resultado en una matriz **A** de N x N.
3. Aplicar la TRF a cada columna de la matriz **A** y almacenar el resultado en una matriz **B** de N x N.
4. Insertar $(\alpha\text{-}1)N$ ceros en cada renglón de la matriz **B**, enseguida insertar $(\alpha - 1)N$ ceros en cada columna del resultado anterior.
5. Del paso 4 se obtiene una matriz **B** aumentada (**B'** de dimensiones αN x αN).
6. El siguiente paso es antitransformar la matriz **B'**, primero se aplica la TRF inversa a cada columna de **B'** y después a cada renglón, obteniéndose una matriz **X** aumentada (**X'** de αN x αN).
7. Multiplicar cada elemento de la matriz **X'** por $1/\alpha^2$.

En la figura I.25 se muestra el aspecto geométrico que se logra, al agregar α ceros a los renglones y columnas de la imagen transformada. En la figura I.26 se muestra un ejemplo.

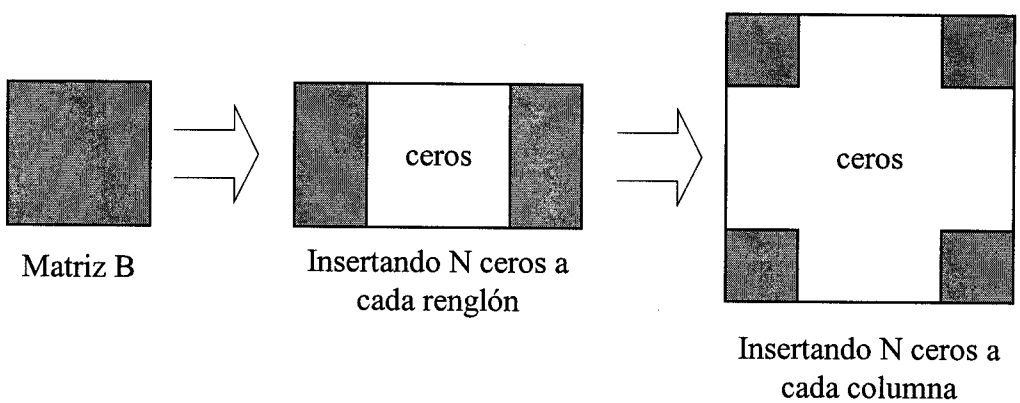


Fig. I.25 Aspecto que se logra al agregar ceros al espectro de la imagen.

Ejemplos

a)Imagen original

b) Imagen amplificada (factor = 2)

c) Imagen amplificada (factor = 4)

Fig. I.26 Imágenes amplificadas mediante el método de inserción de ceros en el dominio de la frecuencia.

I.4 Métodos estadísticos.

En esta sección se abordarán algunas de las técnicas de procesamiento de imágenes digitales en base a datos estadísticos que presentan los pixeles que las conforman, estas técnicas toman parámetros típicos como son: distribución, variancia y desviación estándar. Entre las técnicas que se muestran se pueden encontrar la ecualización de imágenes y la compresión sin pérdidas.

I.4.1 Ecualización de imágenes en base a su histograma.

El ecualizar una imagen consiste en modificar la escala de grises por medio de una función de transformación, que da como resultado el aumento en el rango dinámico de la imagen o contraste. En la figura I.27 se muestra la imagen a procesar, la función de transformación y la imagen procesada. La intensidad luminosa está codificada usando tres bits.

El histograma de una imagen es una función que indica el número de píxeles de la imagen que corresponde a un nivel de gris particular con respecto a ese mismo nivel de gris. Un histograma se puede pensar como la función densidad de probabilidad discreta (fdp) de una imagen individual en la siguiente forma: Cada evento E del experimento probabilístico ε es la selección de un pixel de la imagen al azar, donde el evento E mide el nivel de gris del pixel seleccionado. Cuando la función densidad de probabilidad en cada nivel de gris se multiplica por el número de píxeles en la imagen, se obtiene como resultado el número de píxeles con dicho nivel de gris.

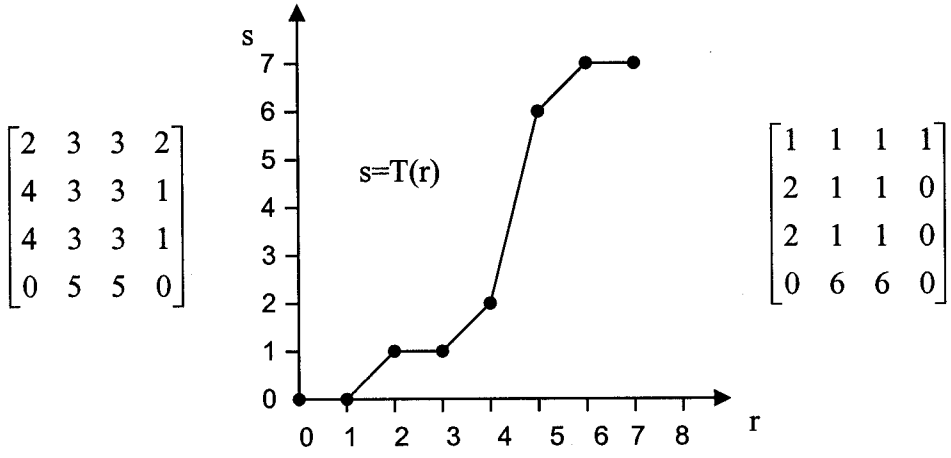


Fig. I.27 Aumento en el rango dinámico de una imagen usando una función de transformación s=T(r)

El histograma no contiene información con respecto a la localización de los píxeles, o de la proximidad de un píxel a otro. Lo que sí contiene, es información de la brillantez aparente y del contraste de una imagen, y son usados en el procesamiento de imágenes para manipular estas mismas características en la imagen. Una vez que se conoce el histograma de una imagen, los niveles de gris en la imagen pueden ser manipulados para cambiar el histograma como se desee. El propósito puede ser el de

mejorar el contraste, cambiar el nivel de brillantez, o hacer corresponder el histograma de una imagen con el de otra.

Cualquier técnica de modificación de niveles de gris, de las cuales la modificación del histograma es un ejemplo, se basa en transformar los niveles de gris de la imagen original a los niveles de gris de la imagen modificada, si los niveles de gris en la imagen original (codificados con 8 bits) se representan por una variable w (que puede tomar valores enteros de 0 a 255), y que los niveles de gris de la imagen modificada se representan por q con el mismo rango de valores, entonces la ecuación de transformación está dada por:

$$q = T(w), \tag{I.58}$$

donde T( ) es algún operador de transformación. Esto significa que si se tiene un píxel en la imagen original con un nivel de gris w, al píxel que se encuentre en la misma posición en la imagen modificada se le dará un nivel de gris T(w).

En el aplanamiento del histograma, el resultado deseado es que la densidad de probabilidad de q (los niveles de gris en la imagen modificada) sea uniforme para todos los valores de q de 0 a 255. La función densidad de probabilidad de la imagen procesada puede ser estimada de los valores del histograma de la imagen original. Por ejemplo, si el nivel de gris w es igual a 120, entonces el valor de $p_w(120)$ es el número de ocurrencias del nivel de gris 120 en la imagen (el histograma de la imagen en 120 o h(120), donde h es el histograma de la imagen original) dividido por el total del número de píxeles en la imagen, es decir:

$$p_w(120) = \frac{h(120)}{N^2} \tag{I.59}$$

donde N^2 es el número de píxeles en la imagen. Se puede demostrar que usando la función de probabilidad acumulativa (FPA) de w (escalada por el máximo nivel de gris) como los valores para q, se obtiene la mayor uniformidad posible de la función de probabilidad. Esta ecuación de transformación es [9]:

$$T(w) = I_{\text{máx}} \sum_{r=0}^{w} p_w(r) \tag{I.60}$$

donde $I_{\text{máx}}$ es el máximo nivel de gris (255 para una imagen de 8 bits). El lado derecho de la ecuación de transformación es la sumatoria de las densidades de probabilidad

discretas desde r=0 hasta w (el nivel de gris de la imagen original), también llamada función de probabilidad acumulativa (FPA) de la variable w. Si se puede encontrar el histograma de la imagen, es claro que también se puede hallar la FPA para cada nivel de gris original y, de está última, se puede determinar el nivel de gris modificado.

Como una revisión de este método, supóngase que la imagen original ya tiene un histograma uniforme. Esto significa que hay un número igual de píxeles para cada nivel de gris individual desde 0 a 255. Si la imagen es de 256 x 256 (se tiene que $N^2 = 65536$), el número de píxeles para cada nivel de gris será 256. Entonces, el valor de $p_w(n)=256/N^2$ para todos los niveles de gris. La imagen con el histograma aplanado usará la FPA en cada valor de nivel de gris, escalada por $I_{máx}$ (que es 255), como el nuevo nivel de gris. La FPA será una función lineal de 0 a 1.0, haciendo que los valores de T(w) sean exactamente los mismos que los niveles de gris originales. Para un histograma deseado (de la imagen procesada), s=T(w) se obtiene de los histogramas acumulativos escogiendo el valor de s para cada valor de r tal que F(s) sea lo más cercano a F(r).

De la discusión anterior, se puede derivar un procedimiento para aplanar el histograma. El primer paso, es determinar el histograma de la imagen original, del cual se pueda derivar el histograma de la imagen procesada. Mediante los histogramas acumulativos, se forma un arreglo de transformación, cuyos índices son los niveles de gris originales, y cuyos valores son los niveles de gris de la imagen modificada.

En la figura I.28 se muestran los histogramas de entrada y salida y sus correspondientes histogramas acumulativos, en la figura I.29 se muestra la relación de transformación s=T(r) y el histograma de la imagen procesada.

En las figuras I.30 y I.31 se dan dos ejemplos de este método de procesamiento. Obsérvese el aumento del contraste en la imagen procesada.

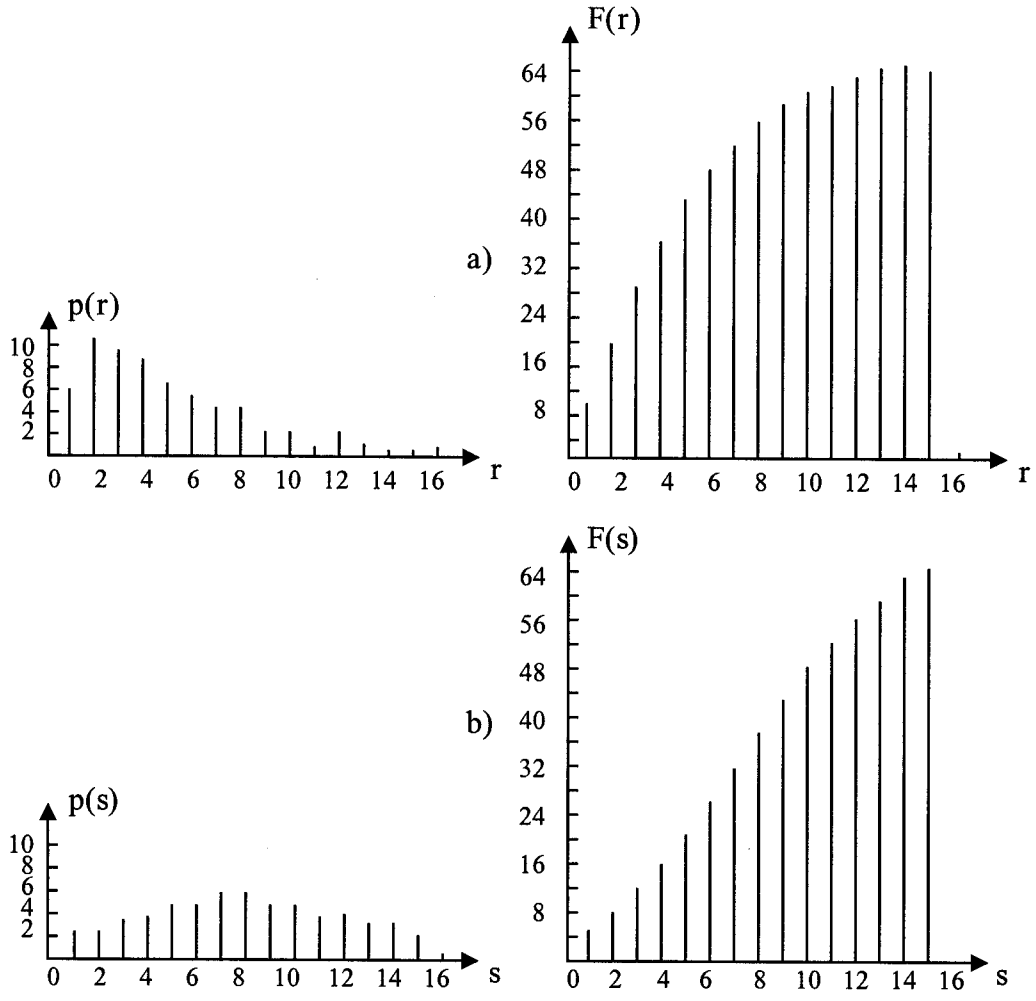


Fig. I.28 (a) Histograma de una imagen y su correspondiente histograma acumulativo.
(b) Histograma deseado y su correspondiente histograma acumulativo.

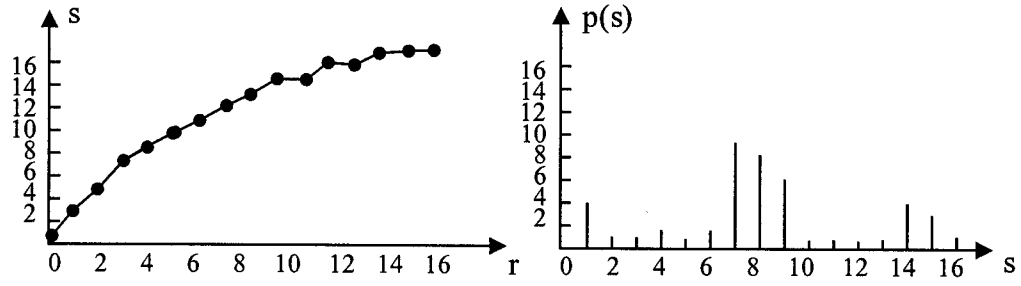
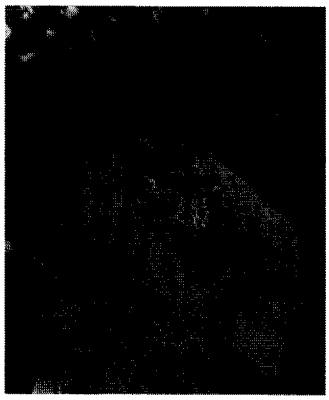
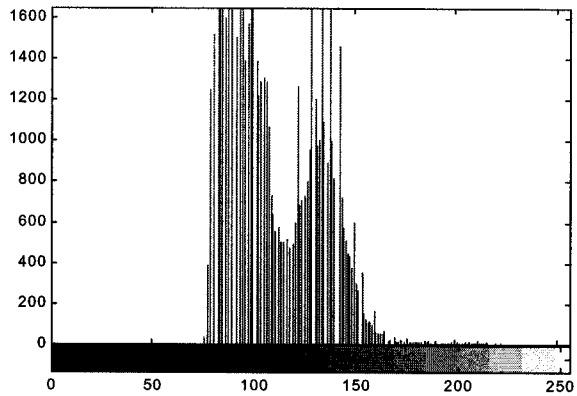


Fig. I.29 Función de transformación e histograma de la imagen procesada.

Ejemplos

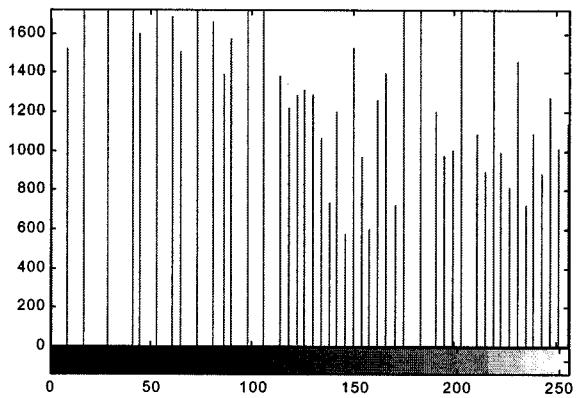


a) Imagen original



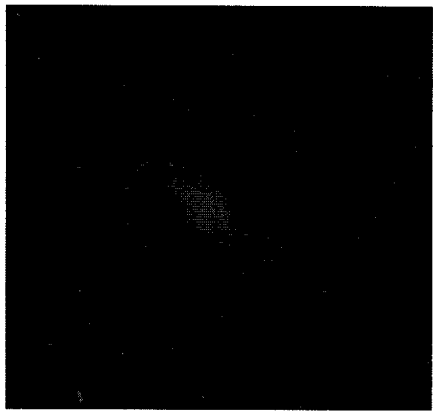
b) Histograma de la imagen original.

c) Imagen ecualizada

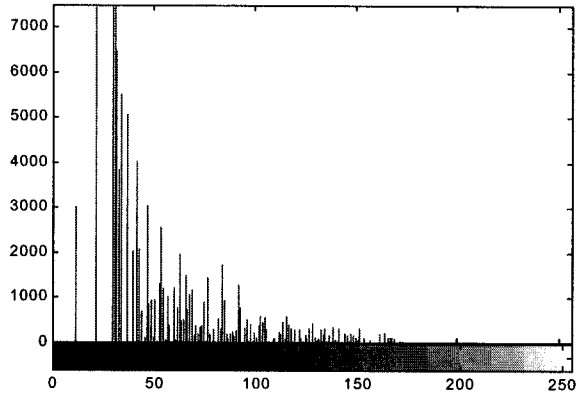


d) Histograma ecualizado de la imagen.

Fig. I.30 Ecualización de una imagen de 240 x 291 píxeles a 256 niveles de gris.

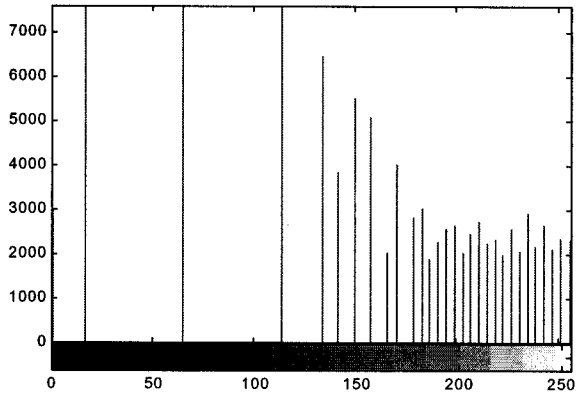


a) Imagen original.



b) Histograma de la imagen original.

c) Imagen ecualizada.



d) Histograma ecualizado de la imagen.

Fig. I.31 Ecualización de una imagen de 400 x 378 píxeles a 256 niveles de gris.

I.4.2 Compresión sin pérdidas.

Una característica que puede utilizarse para la compresión de imágenes, es la cantidad de información que está contenida en los datos. Suponiendo que la fuente produce un símbolo a_j, es posible conocer la cantidad de información $I(a_j)$ que se recibe la cual está definida por [6]

$$I(a_j) = -\log[p(a_j)] \tag{I.61}$$

donde $p(a_j)$ es la probabilidad de que el evento a_j ocurra.

En base a lo anterior, se puede notar que la cantidad de información que aporta un dato está en función de la probabilidad que le está asociada, es decir, cuanto menor sea la probabilidad de que el dato aparezca, mayor será la información que aporte, tomando como ejemplo $p(a_j)=1$, es decir, que solo puede existir el dato a_j ya que tiene la probabilidad total, la cantidad de información que aporta el dato a_{j+1} es cero, porque ya se sabía a priori que éste dato aparecería (ya que es el único $p(a_j)=1$).

La base del logaritmo en la ecuación (I.61) determina las unidades de información, cuando se usa el logaritmo natural (de base e) las unidades son llamadas nat, y cuando se utiliza logaritmo de base 2 la unidad resultante es bit.

La función $I(a_j)$ definida en I.61 es llamada auto-información. El valor medio (o valor esperado) de la auto-información es llamado entropía de la fuente X y está definida por [6]

$$H(X) = -\sum_{j=1}^{m} p(a_j) \log_2 p(a_j). \qquad (I.62)$$

La entropía es la medida de la cantidad promedio de información que aporta un conjunto de datos. También H(X) puede interpretarse como la certeza a priori, que se tiene de recibir un dato.

La compresión sin pérdidas (conocida en inglés como lossless), se basa en la idea de que en una imagen no todos los niveles de gris ocurren con la misma frecuencia, y en base a esto se asignan símbolos cortos a los niveles con más ocurrencias (un símbolo representa un dato ó nivel de gris). Suponiendo que una imagen contiene K niveles de gris (256 niveles para las imágenes que se manejarán), el tamaño de la palabra necesaria para representar cada nivel es de $\log_2 K$ (8 bits), y suponiendo que el esquema de compresión usado requiere de un promedio de R bits por muestra de la imagen. El grado de compresión logrado es $(\log_2 K)/R$.

Según la teoría de la información, el límite que puede lograrse en la compresión (sin pérdidas) de un conjunto de datos es la cantidad de información que aporta el mismo, definida como entropía. Para lograr la cota inferior de compresión generalmente se emplea el siguiente procedimiento: (1) a partir de la imagen original se genera una nueva representación, en la cual cada uno de sus elementos son menos dependientes que en las muestras de la imagen original; (2) de acuerdo a la probabilidad de ocurrencia de cada

uno de los niveles de gris, se les asigna un código único e irrepetible cuya longitud va en aumento conforme la probabilidad decrece en magnitud.

Para efectuar el primer paso se utiliza alguna de las siguientes ecuaciones

$$\xi(m,n) = f(m,n) - f(m,n-1) \tag{I.63}$$

o también

$$\xi(m,n) = f(m,n) - f(m-1,n) - f(m,n-1) + f(m-1,n-1), \tag{I.64}$$

obteniéndose una imagen $\xi(m,n)$ menos correlacionada que f(m,n). La diferencia de aplicar la ecuación I.63 ó I.64 radica en que la primera requerirá símbolos de un bit más que los necesarios para la imagen original, es decir, si se tiene una imagen con K niveles de gris se necesitarían K símbolos pero con información redundante, si se aplica la ecuación I.63 se necesitarán 2K símbolos. Al utilizar la ecuación I.64 se logra una mejor decorrelación pero ésta vez se requerirán 4K símbolos.

Después de obtener la representación de la imagen en forma menos redundante $\xi(m,n)$, el siguiente paso es asignar una palabra binaria a cada posible valor de $\xi(m,n)$. La forma más eficiente de lograr lo anterior se conoce como codificación Huffman [10], que se verá a continuación.

I.4.2.1 Codificación Huffman.

Para ilustrar la forma en que la codificación Huffman es efectuada, considérese un ejemplo en el cual $\xi(m,n)$ puede tomar solamente ocho valores, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, con las siguientes probabilidades:

$p_1 = p\{q_1\} = 0.40$ $\quad p_5 = p\{q_5\} = 0.12$ $\quad p_2 = p\{q_2\} = 0.08$ $\quad p_6 = p\{q_6\} = 0.08$

$p_3 = p\{q_3\} = 0.08$ $\quad p_7 = p\{q_7\} = 0.04$ $\quad p_4 = p\{q_4\} = 0.20$ $\quad p_8 = p\{q_8\} = 0.00$

donde p_i corresponde a la probabilidad de que ocurra el i-ésimo símbolo. En la práctica, las probabilidades pueden ser estimadas a partir del histograma de $\xi(m,n)$. El procedimiento para la obtención de los símbolos es el siguiente:

1. Encontrar en la lista las dos probabilidades menores. Para el ejemplo corresponden al nivel 7 y 8 con valores de 0.04 y 0 respectivamente.
2. Sumar las dos probabilidades menores y agregar el resultado a la lista, retirando los dos valores, formando una nueva lista con un elemento menos.

3. Cuando existen dos símbolos con igual probabilidad, el que se interpreta como menor es el primero que se encuentra (de izquierda a derecha, según la dirección de la búsqueda).
4. Regresar al paso 1 hasta que la lista contenga únicamente un elemento.

Posibles Valores de $\xi(m,n)$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
probabilidades	0.4	.08	.08	.2	.12	.08	.04	0
							0 | 1	
1° pasada	0.4	.08	.08	.2	.12	.08	.04	
						0 | 1		
2° pasada	0.4	.08	.08	.2	.12		.12	
		1 | 0						
3° pasada	0.4	.16		.2	.12		.12	
					1 | 0			
4° pasada	0.4	.16		.2		.24		
		1 | 0						
5° pasada	0.4		.36			.24		
				0 | 1				
6° pasada	0.4			0.6				
	1 | 0							
7° pasada		1.0						

Fig. I.32 Ejemplo del proceso de codificación Huffman.

En base a los pares de números encontrados se forma un árbol binario, asignando 1 a la ruta que se sigue cuando corresponde al número menor y 0 al siguiente menor, por ejemplo en la pasada 1 el valor 0.04 se formó de la suma de los valores q_8 que fue el menor y q_7 que le seguía en magnitud entonces se asigna 1 a la ruta para q_8 y 0 a la ruta para q_7, después de asignar los valores a cada rama se recorre cada una (de abajo hacia arriba) almacenando su valor, de forma que se obtienen los siguientes símbolos:

$q_1 = 1$ $\quad$ $q_2 = 0011$
$q_3 = 0010$ $\quad$ $q_4 = 000$
$q_5 = 011$ $\quad$ $q_6 = 0100$
$q_7 = 01010$ $\quad$ $q_8 = 01011$

puede observarse que ningún símbolo es prefijo de otro, es decir, no existe un símbolo de N bits que sea igual a los primeros N bits de otro símbolo (los primeros de izquierda a derecha). Como primera impresión, podría pensarse que no existe compresión ya que originalmente la representación de los ocho valores constaban de 3 bits (000 - 111) y entre los nuevos símbolos existen algunos con longitudes de 5 bits.

Es posible calcular la longitud promedio en bits de cada píxel en base a sus probabilidades, si son interpretadas como porcentajes, es decir, suponiendo que se asignan los símbolos: 000-1, 001-2, 010-3, 011-4, 100-5, 101-6 y 111-8, en un imagen que tiene la distribución de probabilidades del ejemplo: el 40% de sus píxeles tendrán el nivel 1 y serán representados por 000, y de forma igual para los demás valores, entonces, multiplicando el porcentaje por la longitud del símbolo se obtiene la longitud promedio en bits para representar cada píxel, o sea:

$$L=3(0.4)+3(0.08)+3(0.08)+3(0.2)+3(0.12)+3(0.08)+3(0.04)+3(0) = 3 \text{ bits/pixel},$$

si se utilizan los símbolos generados por la codificación Huffman se tendrán:

$$L=1(0.4)+4(0.08)+4(0.08)+3(0.2)+3(0.12)+4(0.08)+5(0.04)+5(0) = 2.52 \text{ bits/pixel}.$$

Entonces puede observarse que sí existe un reducción en la cantidad de bits promedio necesarios para representar cada píxel en una imagen. Como dato de comparación puede obtenerse el valor de la entropía:

$$H=-[0.4\log_2 0.4+3(0.08\log_2 0.08)+0.2\log_2 0.2+0.12\log_2 0.12+0.04 \log_2 0.04]=2.42 \text{ bits}$$

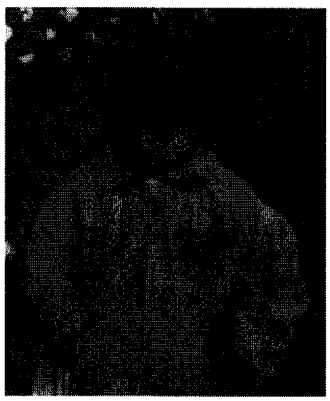
puede observarse que con los símbolos generados por la codificación Huffman casi logran el valor de la entropía.

Como se mencionó, un código Huffman es unívocamente decodificable, ya que una cadena de dígitos binarios solamente puede ser decodificada de una sola forma. Retomando el ejemplo, supóngase que se tiene un patrón binario (de los tres primeros valores de $\xi(m,n)$ de una imagen) 0010101010. Examinando esta cadena de izquierda a derecha, puede notarse que los paquetes 0, 00, ó 001 no corresponden a algún símbolo válido de $\xi(m,n)$. Sin embargo el 0010 corresponde a q_3. Entonces la primera muestra es q_3, lo anterior no es ambiguo ya que 0010 no es prefijo de otro símbolo. El siguiente bit en la cadena es 1, el cual corresponde a q_1. Por lo tanto, la siguiente muestra sin ambigüedad es q_1 , a causa de que 1 no es prefijo de algún símbolo. Examinando el resto

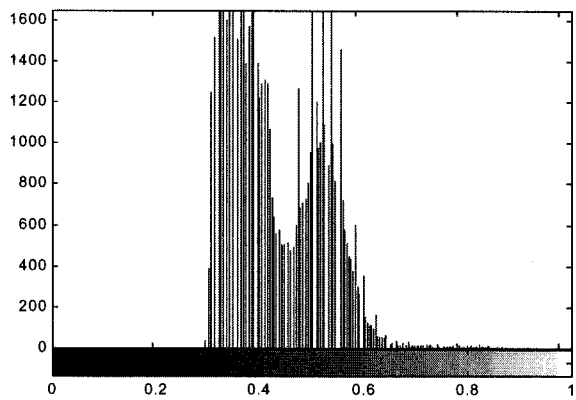
del los bits en la cadena puede notarse que los paquetes 0, 01, 010, ó 0101 no corresponden a algún símbolo válido para $\xi(m,n)$. Pero 01010, corresponde a q_7, entonces, la tercera muestra es igual a q_7. Al realizar la decodificación, los valores obtenidos son exactamente iguales a los que entraron al codificador, por lo tanto no existen pérdidas.

En las figuras I.33 y I.34 se muestran dos ejemplos de la compresión de imágenes sin pérdidas, obsérvese que en cada uno de los ejemplos el histograma se comprimió radicalmente.

Ejemplos

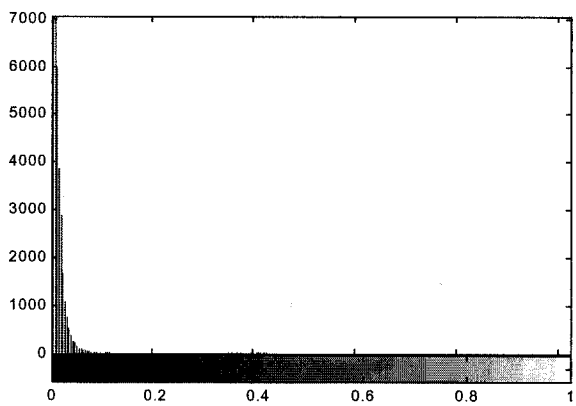


a) Imagen original.



Entropía = 5.7599

b) Histograma de la imagen original.

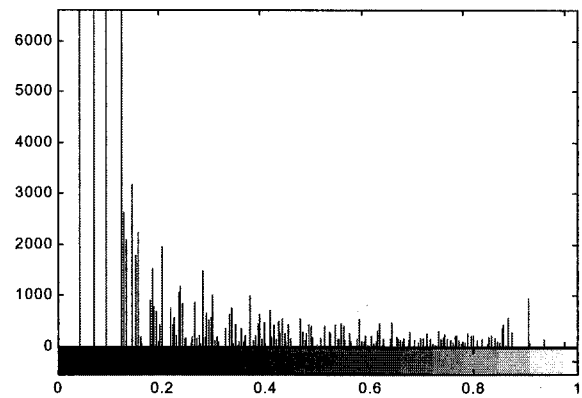


Entropía = 2.1171

c) Histograma de la imagen descorrelacionada.

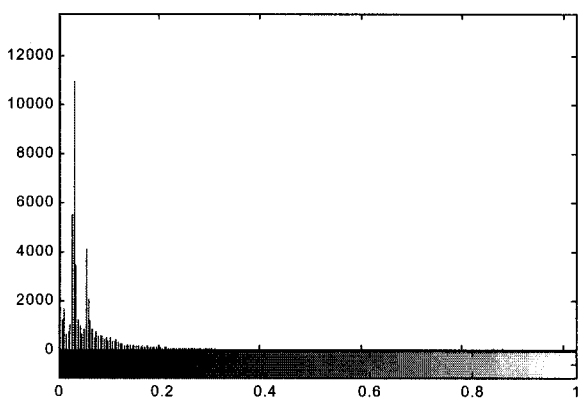
Fig. I.33 Descorrelación de una imagen para ser comprimida posteriormente.

a) Imagen original.



Entropía = 4.7753

b) Histograma de la imagen original.



Entropía = 2.6125

c) Histograma de la imagen descorrelacionada.

Fig. I.34 Descorrelación de una imagen para ser comprimida posteriormente.

En este capítulo se abordaron algunas de las técnicas de procesamiento de imágenes digitales de una manera y se presentaron ejemplos realizados en Matlab, consecuentemente se analizó la factibilidad de implementación en la plataforma que se utilizará en el ***Capítulo IV***. En el siguiente capítulo se hablará acerca de la Transformada de Ondículas, la cual es una de las herramientas utilizadas en las técnicas de procesamiento implementadas en el ***Capítulo IV***.

II LA TRANSFORMADA DE ONDICULAS

II.1 Introducción.

De la teoría de la transformada de Fourier se sabe que una señal puede ser expresada como una suma, posiblemente infinita, de series de senos y cosenos. A esta suma se le conoce también como expansión de Fourier. La gran desventaja de las series de Fourier es que únicamente tienen resolución en la frecuencia, más no en el tiempo. Esto significa que aunque se puedan determinar todas las componentes de frecuencia de una señal, no se podrá determinar cuándo estas componentes están presentes. Para evitar este problema, en las décadas pasadas han sido desarrolladas varias soluciones las cuales cuentan con mayor o menor capacidad de representar una señal en el dominio del tiempo y la frecuencia a la vez.

La idea detrás de estas representaciones en tiempo-frecuencia es la de segmentar la señal de interés en varias partes y analizar estas partes por separado. Está claro que el analizar una señal de esta manera dará más información acerca de cuándo y dónde se presentan los diferentes componentes frecuenciales, pero esto trae un problema fundamental: ¿Cómo se deberá segmentar la señal?. Supóngase que se quieren conocer exactamente todos los componentes presentes en una señal en un tiempo determinado. Lo que se hace comúnmente es cortar la señal en un tiempo corto utilizando una ventana en forma de pulso de ancho τ, y se transforma al dominio de la frecuencia.

El problema aquí, es que el hecho de segmentar la señal corresponde a una convolución entre la señal y la ventana de observación. Y como la convolución en el dominio del tiempo es igual a la multiplicación en el dominio de la frecuencia y como la

transformada de Fourier de un impulso contiene todas las frecuencias posibles, los componentes de frecuencia serán repartidos por todo el espectro. De hecho esta situación es la contraria a la de la transformada de Fourier ya que ahora se tiene resolución en el tiempo pero no en la frecuencia.

El problema mencionado se encuentra descrito en el Principio de Incertidumbre de Heisenberg, el cual, en términos de procesamiento de señales dice que es imposible el conocer la frecuencia exacta al tiempo exacto de la ocurrencia de una frecuencia en una señal. En otras palabras, una señal no puede ser representada como un punto en el espacio de tiempo-frecuencia. El principio de incertidumbre muestra la importancia del cómo se segmenta una señal.

La transformada de ondículas (TO) es probablemente la solución más reciente para resolver los problemas que no son resueltos por la Transformada de Fourier. En el análisis de ondículas el uso de una ventana modulada resuelve el problema de el segmentado de la señal. La ventana es desplazada a lo largo de la señal y para cada posición se calcula el espectro. Después este proceso es repetido varias veces con una ventana más pequeña para cada nuevo ciclo. Al final, el resultado será una colección de representaciones de la señal en el dominio *tiempo-frecuencia*, todas ellas con diferentes resoluciones. Debido a esto al análisis de ondículas se le puede considerar como un análisis multiresolución. En el caso de las ondículas normalmente no se habla de representaciones tiempo-frecuencia sino de representaciones en *tiempo-escala*, siendo *escala* de cierta manera, lo opuesto a la *frecuencia*, ya que el término *frecuencia* es reservado para la transformada de Fourier.

Debido a que en la literatura no se definen claramente los términos de escalas amplias y reducidas, se definirán como sigue: la escala amplia muestra la forma a grandes rasgos, mientras que la escala reducida muestra los detalles. Por lo tanto, el ir de una escala amplia a una escala reducida implica un proceso de acercamiento.

En las siguientes secciones se presentará la transformada de ondículas y se desarrollará una metodología que ayudará considerablemente a implementar la transformada de ondículas de una manera eficiente en cualquier computadora.

II.2 Principios de la transformada continua de ondículas.

El análisis de ondículas descrito anteriormente es conocido como la transformada continua de ondículas (TCO). Y su definición formal es [15]:

$$\gamma(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)dt\,, \qquad \text{(II.1)}$$

donde * denota conjugación compleja. Esta ecuación muestra cómo una función *f(t)* es descompuesta en un conjunto de funciones $\psi_{s,\tau}(t)$, llamadas ondículas y las variables s y τ son las nuevas dimensiones (escala y traslación). Con el objetivo de complementar la ecuación II.1, la ecuación II.2 presenta la transformada inversa de ondículas:

$$f(t) = \iint \gamma(s,\tau)\psi_{s,\tau}(t)d\tau ds\,. \qquad \text{(II.2)}$$

Las ondículas son generadas a partir de una ondícula base $\Psi(t)$, llamada también ondícula madre, por escalamiento y traslación:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right). \qquad \text{(II.3)}$$

En la ecuación II.3 s es el factor de escala, τ es el factor de traslación y el factor $s^{-1/2}$ es para normalizar la energía a lo largo de las diferentes escalas.

Es importante notar que en (II.1), (II.2) y (II.3) las funciones bases de las ondículas no han sido especificadas. Esta es la principal diferencia entre la transformada de ondículas y la transformada de Fourier u otras transformadas.

II.2.1 Propiedades de las ondículas.

Las propiedades más importantes de las ondículas son sus condiciones de admisibilidad y de regularidad, estas propiedades son las que dieron nombre a las ondículas. Se puede demostrar [17] que las funciones integrales cuadráticas $\psi(t)$ que satisfacen la condición de admisibilidad, o sea:

$$\int \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty\,, \qquad \text{(II.4)}$$

se pueden utilizar para analizar y sintetizar una señal sin pérdida de información. En la ecuación II.4 $\Psi(\omega)$ representa la transformada de Fourier de $\psi(t)$. La condición de admisibilidad implica que la transformada de Fourier de $\psi(t)$ desaparece en la frecuencia cero, o sea:

$$\left|\Psi(\omega)\right|^2\Big|_{\omega=0} = 0 . \qquad \text{(II.5)}$$

Esto significa que las ondículas tienen un espectro de tipo pasa-bandas. Esto es una observación muy importante, que después se utilizará para construir una transformada de ondículas eficiente.

Un cero en la frecuencia cero también significa que el valor promedio de las ondículas en el dominio del tiempo deberá ser cero, es decir:

$$\int \psi(t)dt = 0 , \qquad \text{(II.6)}$$

y por lo tanto $\psi(t)$ debe de ser oscilatoria. En otras palabras, $\psi(t)$ debe de ser una onda.

Como se puede observar en la ecuación II.1, la transformada continua de ondículas de una función unidimensional es bidimensional, y la transformada de ondículas de una señal bidimensional será tetra-dimensional. El producto del tiempo con el ancho de banda de la transformada de ondículas es el cuadrado de la señal de entrada. Para la mayoría de las aplicaciones prácticas esta no es una propiedad deseable. Por esta razón se imponen algunas condiciones adicionales en las funciones de ondículas para hacer que la transformada de ondículas decremente rápidamente cuando se decremente el valor de la escala s. Estas son las condiciones de regularidad, las cuales muestran que la función de ondículas debe tener cierto nivel de alisamiento y concentración en los dominios del tiempo y la frecuencia. La regularidad es un concepto más complejo y se tratará de explicar utilizando el concepto de momentos desvanecientes.

Si se expande la transformada de ondículas (II.1) en series de Taylor en $t=0$ hasta un orden n (por simplicidad se hará $\tau = 0$) se obtiene [17]:

$$\gamma(s,0) = \frac{1}{\sqrt{s}}\left[\sum_{p=0}^{n} f^{(p)}(0)\int \frac{t^p}{p!}\psi\left(\frac{t}{s}\right)dt + O(n+1)\right]. \qquad \text{(II.7)}$$

Aquí $f^{(p)}$ representa la *p'ésima* derivada de f y $O(n+1)$ representa el resto de la expansión. Ahora, si se definen los *momentos* de las ondículas como M_p, se tiene:

$$M_p = \int t^p \psi(t)dt\,, \tag{II.8}$$

entonces se puede escribir la ecuación II.7 en forma finita como:

$$\gamma(s,0) = \frac{1}{\sqrt{s}}\left[f(0)M_0 s + \frac{f^{(1)}(0)}{1!}M_1 s^2 + \frac{f^{(2)}(0)}{2!}M_2 s^3 + \ldots + \frac{f^{(n)}(0)}{n!}M_n s^{n+1} + O(s^{n+2}) \right]. \tag{II.9}$$

De la condición de admisibilidad se tiene que $M_0 = 0$ por lo tanto el primer término del lado derecho de la ecuación II.9 es cero. Si también se lograran hacer cero todos los momentos hasta M_n, entonces los coeficientes $\gamma(s,\tau)$ de la transformada de ondículas caerán tan rápido como $s^{n/2}$ para una señal $f(t)$. A esto se le conoce en la literatura como momentos desvanecientes u orden de aproximación. Si una ondícula tiene N momentos desvanecientes, entonces el orden de aproximación de la transformada de ondículas es también N. Los momentos no deberán ser necesariamente cero, un valor pequeño muchas veces es suficiente. De hecho, el estudio experimental mostrado en [3] sugiere que el número de momentos desvanecientes requerido depende de las aplicaciones.

Concretamente, la condición de admisibilidad proporciona la onda (*wave*), la regularidad y los momentos desvanecientes proporcionan el decaimiento (*let*) de la misma [4]. Y como se puede apreciar esto da como resultado el término *wavelet,* del idioma inglés.

II.3 Ondículas Discretas.

La transformada de ondículas como ha sido descrita hasta el momento todavía cuenta con tres propiedades que la hacen difícil de utilizar en forma directa como lo plantea la ecuación II.1. La primera propiedad es la redundancia de la TCO. En la ecuación II.1 la transformada de ondículas es calculada al desplazar continuamente una función (la cual es continuamente escalable) sobre una señal y calculando la correlación entre ambas. Está claro que estas funciones no cuentan con las propiedades de ortogonalidad y por lo tanto, los coeficientes de las ondículas obtenidos serían redundantes. Para la mayoría de las aplicaciones prácticas lo que se requiere es reducir esta redundancia [4].

Aunque no se tuviera redundancia en la TCO , todavía se tendría un número infinito de ondículas en la transformada y lo que se quiere es reducir este número a una cantidad más favorable para la aplicación. Este es el segundo problema que se tiene. El tercer problema es el hecho de que para la mayoría de las funciones, la transformada de ondículas no cuenta con soluciones analíticas y únicamente pueden ser resueltas numéricamente. Se requieren algoritmos rápidos para explotar el potencial de la transformada de ondículas y es, de hecho, debido a estos algoritmos rápidos que la transformada de ondículas ha cobrado un papel importante en áreas científicas y tecnológicas.

A continuación, se empezará por eliminar la redundancia de la transformada de ondículas.

Como se mencionó anteriormente la transformada continua de ondículas mapea una señal unidimensional hacia una representación tiempo-escala bidimensional que es demasiado redundante. El producto entre el tiempo y el ancho de banda de la TCO es igual al cuadrado de la señal, y para la mayoría de las aplicaciones, las cuales requieren representar una señal con el menor número de componentes posibles, esto no es eficiente. Para resolver este problema se ha introducido el término de ondículas discretas. Las ondículas discretas no son continuamente escalables ni trasladables pero sí pueden ser escaladas y trasladadas en pasos discretos. Esto se logra al modificar la representación de las ondículas (ecuación II.3) para llegar según [4] a:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}}\psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j}\right). \qquad \text{(II.10)}$$

Aunque esta es llamada ondícula discreta, normalmente es una función continua. En la ecuación II.10 j y k son enteros y $s_0>1$ es un paso predeterminado de dilatación. El factor de traslación τ_0 depende del paso de dilatación. El paso de discretizar la ondícula consiste en que ahora se muestrea el espacio tiempo-escala en intervalos discretos. Comúnmente se elige $s_0=2$ para que el muestreo con los ejes de frecuencia corresponda al *muestreo diádico*. Esta es una elección muy natural para un sistema computacional, para el oído humano y para la música. Para el factor de traslación usualmente se hace $\tau_0 = 1$ para que también se tenga un muestreo diádico en los ejes del tiempo, ver figura II.1.

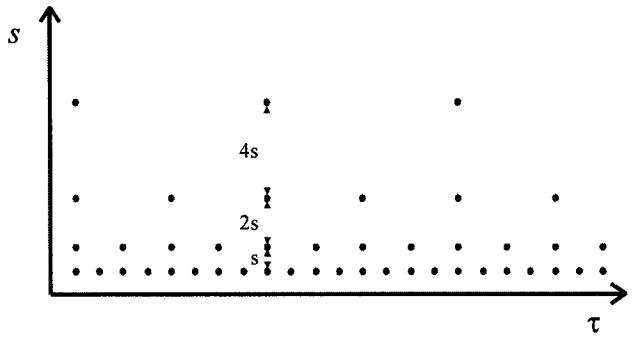


Fig. II.1 Localización de las ondículas discretas en el espacio-tiempo-escala en un diagrama diádico.

Cuando las ondículas discretas se utilizan para transformar una señal continua, esa transformación dará como resultado una serie de coeficientes de ondículas. A esto se le denomina *descomposición de series de ondículas*. Un punto importante en tal esquema de descomposición es el de la reconstrucción. Resulta apropiado el hecho de muestrear la representación tiempo-escala en un diagrama diádico, pero si no fuera posible reconstruir la señal, esto no sería de gran utilidad. De hecho, es posible reconstruir una señal a partir de su descomposición en series de ondículas. En [4] se comprobó que la condición necesaria y suficiente para la reconstrucción estable es que la energía de los coeficientes de las ondículas debe estar dentro de dos límites positivos, que son:

$$A\|f\|^2 \leq \sum_{j,k} \left|\langle f, \psi_{j,k} \rangle\right|^2 \leq B\|f\|^2, \qquad \text{(II.11)}$$

donde $\|f\|^2$ es la energía de f(t), $A>0$, $B<\infty$ y A y B son independientes de *f(t)*. Cuando se satisface la ecuación II.11, a la familia de funciones base $\psi_{j,k}(t)$ con j, $k \in Z$ se le denomina trama con límites A y B. Cuando $A = B$ la trama es cerrada y las ondículas discretas se comportan exactamente como funciones ortonormales. Cuando $A \neq B$ la reconstrucción todavía es posible pero en este caso se requerirá utilizar una *trama dual*. En una transformada de ondículas de trama dual la ondícula de descomposición es diferente que la ondícula de reconstrucción.

El último paso que se tiene que seguir para eliminar toda la redundancia de la transformada de ondículas es la de hacer ortonormales las ondículas discretas. Esto únicamente se le puede hacer para las ondículas discretas. Las ondículas discretas pueden

hacerse ortogonales a sus dilataciones y traslaciones por elecciones especiales de la ondícula base, lo cual significa que:

$$\int \psi_{j,k}(t)\psi^*_{m,n}(t)dt = \begin{cases} 1 & \text{si } j = m \text{ y } k = n \\ 0 & \text{otro caso} \end{cases} \qquad \text{(II.12)}$$

Una señal arbitraria puede ser reconstruida sumando las funciones de la ondículas ortogonales, dándoles el peso de sus respectivos coeficientes [17]:

$$f(t) = \sum_{j,k} \gamma(j,k)\psi_{j,k}(t)\,. \qquad \text{(II.13)}$$

La ecuación II.13 muestra la transformada inversa de ondículas para ondículas discretas.

La ortogonalidad no es esencial en la representación de señales. Las ondículas no necesariamente tienen que ser ortogonales. En algunos casos la redundancia puede ayudar a reducir la sensibilidad al ruido [17] o, incluso, mejorar la invarianza al desplazamiento de la transformada [2]. Esta es una desventaja de las ondículas discretas: la transformada de ondículas resultante ya no es invariante con el desplazamiento, lo cual significa que las transformadas de ondículas de una señal desplazada en el tiempo no son las mismas transformadas desplazadas en el tiempo.

II.3.1 Ondículas discretas como un filtro pasa banda.

Una vez eliminada la redundancia, todavía se tendrían dos problemas para que se pueda utilizar la transformada de ondículas de una manera práctica. Enseguida se tratará de reducir el número de ondículas necesarias en la transformada y se guardará el problema de las soluciones analíticas para después.

Aún con las ondículas discretas, todavía se necesita un número infinito de escalamientos y traslaciones para calcular la transformada de ondículas. La manera más fácil de atacar este problema es simplemente no utilizar un número infinito de ondículas discretas. Desde luego que esto pone en cuestión la calidad de la transformación. ¿Será posible reducir el número de ondículas para analizar una señal y tener un resultado aceptable?

Las traslaciones de las ondículas son limitadas por la duración de la señal que está bajo observación, para que s cuente con un límite superior para las ondículas. Esto trae consigo el problema de dilatación: ¿cuántas escalas se necesitan para analizar una señal? ¿cómo se obtiene un límite inferior? Resulta ser que las respuestas a estas preguntas podrían ser encontradas al ver a la transformada de ondículas de una manera diferente.

En la ecuación II.5 se puede observar que la ondícula tiene un espectro de tipo pasa banda. De la teoría del análisis de Fourier se sabe que la compresión en el tiempo equivale a la expansión del espectro, o sea:

$$F\{f(at)\} = \frac{1}{|a|} F\left(\frac{\omega}{a}\right). \tag{II.14}$$

Esto significa que la compresión de una ondícula en el dominio del tiempo por un factor de 2 provocará una ampliación en el espectro de la ondícula por un factor de 2 y además desplazará todos los componentes de frecuencia por ese mismo factor de 2. Basándose en lo antes mencionado se puede cubrir el espectro finito de la señal con el espectro dilatado de las ondículas, de la misma manera en que se cubrió la señal en el dominio del tiempo utilizando ondículas trasladadas. Para obtener una buena cobertura del espectro de la señal, el conjunto de espectros de las ondículas deben de traslaparse unos con otros (ver figura II.2). Esto se puede lograr diseñando correctamente las ondículas.

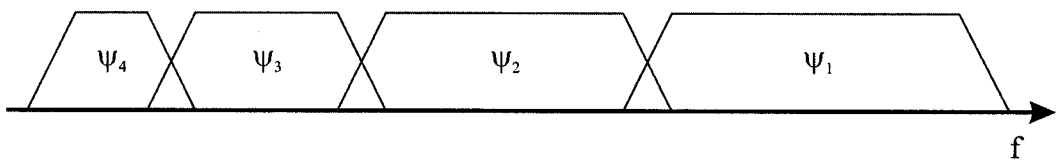


Fig. II.2 Espectros traslapados de las ondículas resultantes del escalamiento de la ondícula madre en el dominio del tiempo.

Concretamente, si una ondícula puede ser vista como un filtro pasa banda, entonces una serie de ondículas dilatadas pueden ser vistas como un banco de filtros pasa banda. Si se observa la razón de la frecuencia central del espectro de la ondícula entre el ancho de este espectro se encontrará que es igual para todas las ondículas. A esta razón se le conoce comúnmente como *factor de fidelidad Q* de un filtro y en el caso de las ondículas se le llama *constante Q* del banco de filtros.

Una restricción importante en el análisis de ondículas es que la señal a analizar debe de tener energía finita. Si se analizara una señal de energía infinita sería imposible cubrir su espectro de frecuencia y su tiempo de duración con ondículas. Formalmente esta restricción es expresada como sigue:

$$\int |f(t)|^2 dt < \infty . \qquad \text{(II.15)}$$

II.3.2 La función de escalamiento.

Otra pregunta que pudiera surgir es: ¿cómo cubrir el espectro hasta llegar a cero? Esto debido a que cada vez que se expanda la ondícula en el dominio del tiempo por un factor de 2, su ancho de banda se reduce a la mitad. Es decir, que, con cada ondícula que se expanda sólo se cubrirá la mitad del espectro restante, lo cual significa que se requerirá de un número infinito de ondículas para cumplir el objetivo.

La solución a este problema es simplemente el no tratar de cubrir todo el espectro hasta llegar a cero con los espectros de las ondículas, en lugar de esto, utilizar una función especial denominada *corcho* para cubrir el vacío restante cuando este sea demasiado pequeño. Esta función corcho tiene un espectro de tipo filtro pasa bajas y pertenece a la llamada *función de escalamiento*. La función de escalamiento fue llamada por primera vez así por Mallat [14], ver figura II.3.

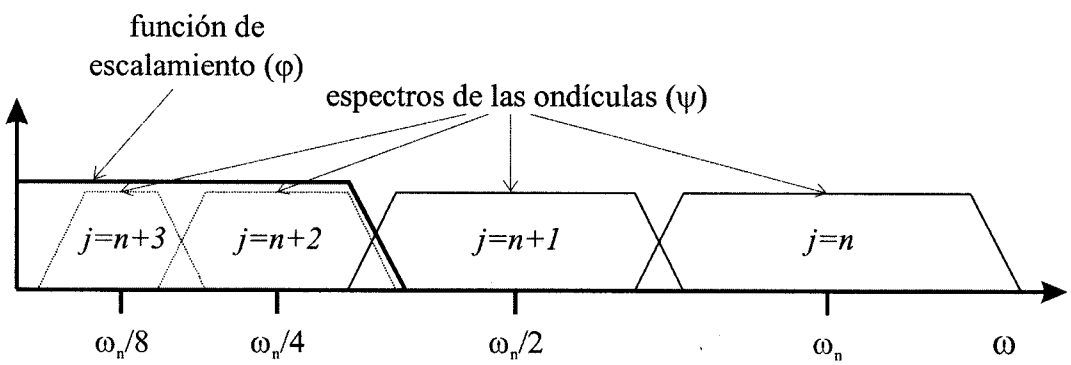


Fig. II.3 Conjunto infinito de ondículas reemplazado por una función de escalamiento.

Debido a la naturaleza pasa bajas del espectro de la función de escalamiento a veces se le conoce también como filtro promediador.

Si se ve a la función de escalamiento como una señal con un espectro de tipo pasa bajas, entonces se puede descomponer en ondículas y expresarse como sigue (de ecuación II.13):

$$\varphi(t) = \sum_{j,k} \gamma(j,k)\psi_{j,k}(t) . \qquad \text{(II.16)}$$

Como se seleccionó la función de escalamiento $\varphi(t)$ de tal manera que su espectro cubriera el espacio dejado por las ondículas, la ecuación II.16 utiliza un número infinito de ondículas hasta una cierta escala j (ver figura II.3). Esto significa que si se analiza una señal utilizando la combinación de la función de escalamiento con las ondículas, la función de escalamiento por sí misma se encarga de cubrir el espectro no cubierto por las ondículas hasta la escala j, mientras que el resto del espectro es cubierto por tales ondículas. De esta manera se ha limitado el número de ondículas (de un número infinito a un número finito).

Al utilizar la función de escalamiento se ha resuelto el problema del número infinito de ondículas y se ha establecido un límite inferior para estas. Desde luego que cuando se usa una función de escalamiento en lugar de ondículas se pierde información. Es decir, desde el punto de vista de la representación de la señal no se pierde información ya que sigue siendo posible reconstruirla, pero desde el punto de vista del análisis de ondículas se está descartando información que pudiera ser valiosa. Por lo tanto, el ancho de banda de la función de escalamiento es un parámetro importante en el diseño de la transformada de ondículas. Mientras menor sea ese ancho de banda, se tendrá un mayor número de coeficientes de ondículas, y en cuanto más coeficientes sean, mayor información. Sin embargo, existen limitaciones prácticas en cuanto al número de ondículas que se pueden manejar. Como se verá después. En la transformada discreta de ondículas, este problema se resuelve automáticamente.

El espectro pasa bajas de la función de escalamiento permite establecer cierta condición de admisibilidad como se hizo en la ecuación II.6, o sea:

$$\int \varphi(t)dt = 1 , \qquad \text{(II.17)}$$

la cual muestra que el momento 0 de la función de escalamiento no se puede desvanecer.

Concretamente, si una ondícula puede ser vista como un filtro pasa banda y una función de escalamiento como un filtro pasa bajas, entonces una serie de ondículas

dilatadas junto con una función de escalamiento pueden ser vistas como un banco de filtros.

II.3.3 Codificación en sub-bandas.

Dos de los tres problemas mencionados anteriormente han sido resueltos, pero aún no se sabe cómo calcular la transformada de ondículas.

Si se ve a la transformada de ondículas como un banco de filtros, entonces se podría considerar que el calcular la transformada de ondículas de una señal sería equivalente a pasar esa señal por este banco de filtros. Las salidas de las diferentes etapas de los filtros serían los coeficientes de las ondículas. El analizar una señal siendo pasada por un banco de filtros no es una técnica nueva, esto ya ha sido utilizado durante varios años y se le conoce como codificación en sub-bandas [15]. Esto se utiliza, por ejemplo, en aplicaciones de visión por computadora.

El banco de filtros necesario para la codificación en sub-bandas puede ser construido por diferentes métodos. Uno de ellos consiste en formar varios filtros pasa bandas para dividir el espectro en bandas de frecuencia. La ventaja es que el ancho de cada banda puede ser elegido arbitrariamente, de tal manera que, el espectro de la señal a analizar quedará cubierto según donde éste sea más importante. La desventaja de este método es que se tiene que diseñar cada filtro por separado y esta es una labor que consume tiempo. Otro método consiste en dividir el espectro en dos partes iguales, una parte pasa bajas y la otra parte pasa altas. La parte pasa altas contiene los detalles más pequeños en los cuales se pudiera estar interesado. Ahora se tienen dos bandas. Sin embargo, la parte pasa bajas todavía contiene algunos detalles de interés por lo cual se procede a dividir la banda de nuevo. Y así sucesivamente, hasta que se esté satisfecho con el número de bandas que se hayan creado. De esta forma se ha creado un *banco de filtros iterativo*. Usualmente el número de bandas está limitado por el número de datos o la capacidad de procesamiento disponible. El proceso de dividir el espectro se muestra gráficamente en la figura II.4. La ventaja de este esquema es que únicamente se tienen dos filtros, la desventaja es que el espectro cubierto de la señal es fijo.

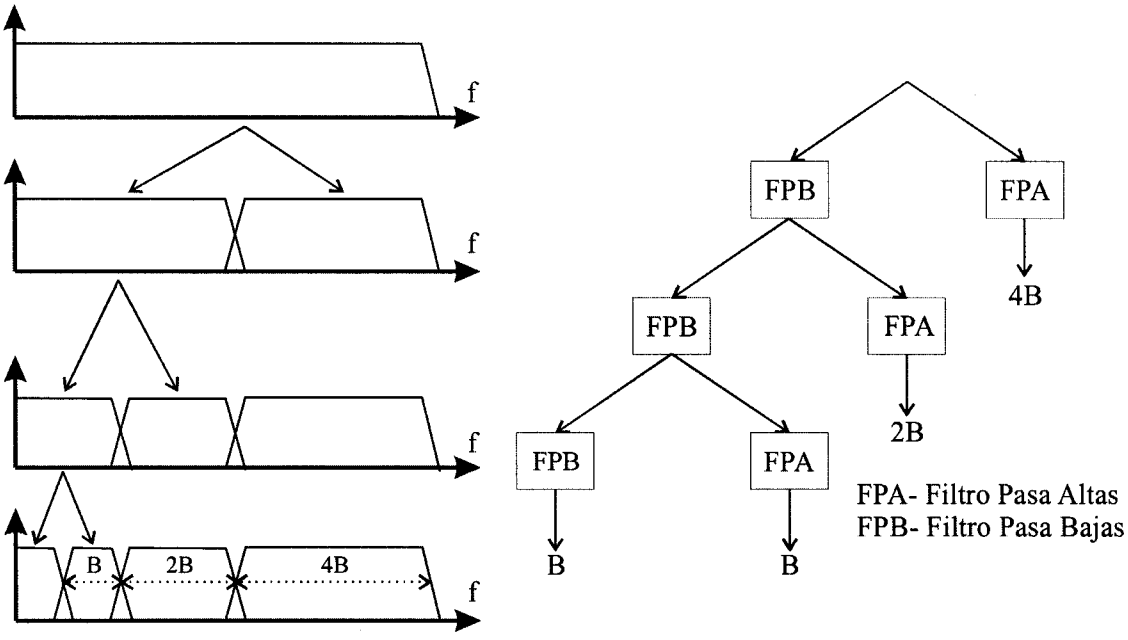


Fig. II.4 División del espectro de una señal por medio de un banco de filtros.

En la figura II.4 se puede observar que lo que queda después de dividir el espectro repetitivamente es una serie de bandas cuyo ancho de banda se dobla y un espectro de tipo pasa bajas (aunque en teoría la primera división dio como resultado un espectro pasa altas y otro espectro de tipo pasa bajas, en realidad, el espectro pasa altas es un espectro pasa bandas debido al ancho de banda limitado de la señal). En otras palabras, se puede desarrollar el mismo análisis de sub-bandas al procesar la señal por un banco de filtros pasa bandas, del cual cada filtro tiene el doble del ancho de banda que su vecino izquierdo, además de otro filtro pasa bajas. Anteriormente se mencionó que esto es lo mismo que aplicar la transformada de ondículas a la señal. Las ondículas daban como resultado bandas cuyo ancho se iba doblando y la función de escalamiento que proveía una banda de baja frecuencia que llegaba hacia el cero. De esto se puede concluir que una transformada de ondículas equivale a un esquema de codificación en sub-bandas utilizando un banco de filtros de Q constante [14]. Generalmente, se referirá a este análisis como un análisis multiresolución.

Concretamente, si se implementa la transformada de ondículas como un banco de filtros en cascada, no se tienen que especificar las ondículas en forma explícita, lo cual facilita la aplicación de éstas.

II.4 La transformada discreta de ondículas.

En algunas aplicaciones, las señales de interés son muestreadas. Para poder utilizar los resultados que se han obtenido en las secciones anteriores con una señal discreta, también se tiene que hacer la transformada discreta. Recuérdese que las ondículas discretas no son discretas en el tiempo, solo la traslación y los pasos de la escala son discretos. Se podría pensar que el hecho de implementar un banco de filtros de ondículas como un banco de filtros de digitales realizaría la transformada discreta, pero esto no es suficiente.

En la ecuación II.16 se estableció que la función de escalamiento podía ser expresada en ondículas desde menos infinito hasta cierta escala j. Si se agrega el espectro de una ondícula al espectro de la función de escalamiento se obtendrá una nueva función de escalamiento, ésta con un espectro el doble de ancho que la primera. El efecto de esta suma de espectros es de que ahora se puede expresar la primera función de escalamiento en términos de la segunda, esto debido a que toda la información necesaria para hacerlo, está contenida en la segunda función de escalamiento. Se puede expresar esto formalmente por medio de la llamada *fórmula multiresolución* [2] o *relación de dos escalas* [17]:

$$\varphi(2^j t) = \sum_k h_{j+1}(k)\varphi(2^{j+1}t - k)\,. \qquad \text{(II.18)}$$

La relación de dos escalas establece que la función de escalamiento a cierta escala puede ser expresada en términos de funciones de escalamiento trasladadas a la siguiente escala menor. Recuérdese que menor escala significa mayor detalle.

La primer función de escalamiento reemplazó a un conjunto de ondículas, por lo tanto, se pueden expresar dichas ondículas en términos de las funciones de escalamiento trasladadas a la siguiente escala. De una manera más específica se puede mencionar que para la ondícula al nivel j:

$$\psi(2^j t) = \sum_k g_{j+1}(k)\varphi(2^{j+1}t - k), \tag{II.19}$$

lo cual es la relación de dos escalas entre la función de escalamiento y la ondícula.

Como la señal *f(t)* puede ser expresada en términos de ondículas dilatadas hasta una escala *j-1*, esto trae como resultado que *f(t)* pueda ser expresada en términos de funciones de escalamiento atrasadas y dilatadas en una escala *j*, es decir:

$$f(t) = \sum_k \lambda_j(k)\varphi(2^j t - k). \tag{II.20}$$

En este caso se puede hablar de funciones de escalamiento discretas ya que únicamente se permiten dilataciones y atrasos discretos.

Si en esta ecuación se decrementa la escala a *j-1*, se tendrán que añadir ondículas para poder mantener el mismo nivel de detalle. Entonces se podría expresar la señal *f(t)* como:

$$f(t) = \sum_k \lambda_{j-1}(k)\varphi(2^{j-1}t - k) + \sum_k \gamma_{j-1}(k)\psi(2^{j-1}t - k). \tag{II.21}$$

Si la función de escalamiento $\varphi_{j,k}(t)$ y las ondículas $\psi_{j,k}(t)$ son ortonormales, entonces los coeficientes $\lambda_{j-1}(k)$ y $\gamma_{j-1}(k)$ son calculados al obtener su producto interno, o sea:

$$\begin{aligned} \lambda_{j-1}(k) &= \langle f(t), \varphi_{j,k}(t) \rangle \\ \gamma_{j-1}(k) &= \langle f(t), \psi_{j,k}(t) \rangle \end{aligned}. \tag{II.22}$$

Si ahora se reemplazan $\varphi_{j,k}(t)$ y $\psi_{j,k}(t)$ en los productos internos por versiones escaladas de las ecuaciones II.18 y II.19 y se manipulan un poco, teniendo en cuenta que el producto interno puede ser escrito como una integración, se llega a este importante resultado [2]:

$$\lambda_{j-1}(k) = \sum_m h(m - 2k)\lambda_j(m); \tag{II.23}$$

$$\gamma_{j-1}(k) = \sum_m g(m - 2k)\gamma_j(m). \tag{II.24}$$

Estas dos ecuaciones establecen que los coeficientes de las funciones de ondículas y de las funciones de escalamiento en una cierta escala, pueden ser encontrados al efectuar una suma escalada de los coeficientes de la función de escalamiento para la escala anterior. Ahora, recuérdese que los coeficientes de la función de escalamiento

provienen de un filtro pasa bajas y recuérdese también cómo se utilizó un banco de filtros iterativo para dividir un espectro de tipo pasa bajas, en una banda pasa bajas y en otra pasa altas. La iteración del banco de filtros empezó con el espectro de la señal; si uno se imagina que el espectro de la señal es la salida del filtro pasa bajas de la escala anterior, entonces se podrían considerar a la señal muestreada como los coeficientes de la función de escalamiento de la escala anterior. En otras palabras, la señal muestreada *f(k)* es igual a $\lambda(k)$ en la escala mayor.

Como se conoce de la teoría de procesamiento de señales: una suma de coeficientes discretos como los que se encuentran en las ecuaciones II.23 y II.24 equivalen a un filtro digital y como se sabe que los coeficientes $\lambda_j(k)$ vienen de la parte pasa bajas del espectro dividido de la señal, los valores de los pesos *h(k)* en la ecuación II.23 deben formar un filtro pasa bajas. Y también, como se sabe que los coeficientes $\gamma_j(k)$ vienen de la parte pasa altas de este espectro dividido, los pesos *g(k)* de la ecuación II.24 deben formar un filtro pasa altas. Esto significa que las ecuaciones II.23 y II.24 forman una etapa del banco de filtros iterativo y a los coeficientes *h(k)* se les conoce como filtro de escalamiento mientras que a los coeficientes *g(k)* se les llama filtro de ondículas.

Hasta ahora ya se ha planteado que se puede implementar la transformada de ondículas con un banco de filtros iterativo y de aquí en adelante se hablará de la transformada discreta de ondículas (TDO).

Además las ecuaciones II.23 y II.24 cumplen con la propiedad de sub-muestreo. Si se observan estas dos ecuaciones se puede ver que los filtros de ondículas y de escalamiento tienen un tamaño de paso de 2 en la variable k. El efecto de esto es que por cada dos elementos de $\lambda_j(k)$ solamente uno es utilizado en la convolución, y así se obtiene que la tasa de entrada es la misma que la tasa de salida, ver figura II.5. Aunque esto no representa una novedad, ha sido utilizado durante años en esquemas de codificación en sub-bandas.

La propiedad de sub-muestreo también resuelve el problema de cómo elegir el ancho del espectro de la función de escalamiento. Debido a esto, en cada iteración del banco de filtros el número de muestras generado se divide a la mitad, para que al final

únicamente quede sólo una muestra (esto en el caso extremo). Debe de quedar claro que aquí es donde la iteración tiene que terminar y ésto es lo que determina el ancho del espectro de la función de escalamiento. Normalmente la iteración se detendrá en el punto donde el número de muestras se haya hecho menor a la longitud del filtro de escalamiento o al filtro de ondículas (el cual sea mayor), entonces se puede decir que la longitud del filtro mayor determina el ancho del espectro de la función de escalamiento.

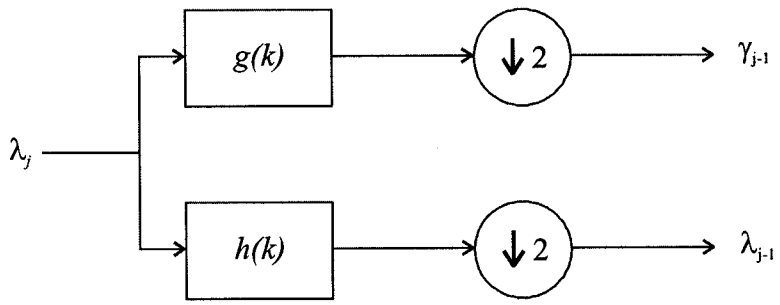


Fig. II.5 Implementación de las ecuaciones II.23 y II.24 como una etapa de un banco de filtros iterativo.

En el siguiente capítulo se estudia la codificación de ondículas por árboles de ceros la cual es una herramienta muy útil en el área de la compresión de imágenes digitales. En el capítulo IV sección IV.4 se presentan ejemplos utilizando este esquema de compresión.

III CODIFICADOR DE ONDÍCULAS POR ÁRBOLES DE CEROS

III.1 Introducción.

Cuando se busca literatura de ondículas para la compresión de imágenes es casi imposible dejar pasar desapercibido el Codificador de Ondículas de Árboles de Ceros de Shapiro [20]. Un Codificador de Ondículas por Árboles de Ceros (COAC) es un codificador especialmente diseñado para utilizarse con la transformada de ondículas (TO), lo cual explica por qué razón se incluye la palabra ondículas en su nombre. El COAC fue originalmente diseñado para operar con imágenes (señales bidimensionales) pero también puede ser utilizado en otro tipo de señales.

El COAC está basado en la codificación progresiva para comprimir una imagen en un flujo de bits con exactitud creciente. Esto significa que cuando más bits sean añadidos al flujo, la imagen decodificada tendrá más detalle, esta propiedad es similar a la de las imágenes codificadas por el estándar JPEG. También es similar a la representación del valor de π en cuanto al hecho de que conforme se agregue un dígito se incrementa la precisión del valor, pero se puede truncar el número de dígitos, dependiendo de la exactitud que se desee. La codificación progresiva también es conocida como codificación incrustada (conocida en inglés como: embedded encoding).

El codificar una imagen utilizando el COAC y el hecho de poder truncar la precisión de la codificación implica una pérdida en la imagen original, lo cual hace a este esquema de codificación una codificación con pérdida (conocida en inglés como: lossy).

III.2 El Árbol de ceros.

El COAC está basado en dos importantes observaciones:

1. Las imágenes naturales generalmente tienen un espectro con características de un filtro pasa bajas [20]. Cuando una imagen es transformada por medio de la TO, la energía en las sub-bandas se decrementa conforme la escala se decrementa (baja escala significa alta resolución), por lo que los coeficientes de las ondículas serán en promedio, menores en las sub-bandas altas que en las sub-bandas bajas. Esto muestra que la codificación progresiva es una elección natural para comprimir imágenes transformadas a ondículas, debido a que las sub-bandas sólo agregan detalle.
2. Los coeficientes grandes son más importantes que los coeficientes pequeños.

Estas dos observaciones son explotadas por el esquema del COAC al codificar los coeficientes en orden decreciente, durante múltiples iteraciones. Para cada iteración un umbral es seleccionado con el cual todos los coeficientes son medidos. Si el coeficiente es mayor que el umbral, éste es codificado y removido de la imagen; si fuera menor, éste es dejado para la siguiente iteración. Cuando todos los coeficientes han sido "visitados" el umbral es decrementado y la imagen es procesada de nuevo para agregar más detalle a la imagen ya codificada. Este proceso es repetido hasta que todos los coeficientes hayan sido codificados en su totalidad o hasta que se haya satisfecho algún otro criterio (por ejemplo, tasa de bits máxima).

Además se utiliza la dependencia entre los coeficientes a lo largo de las diferentes escalas para codificar grandes partes de la imagen, aquí es donde entran los árboles de ceros.

La TO transforma una señal del dominio del tiempo al dominio de escalas de tiempo adjuntas. Esto significa que los coeficientes de las ondículas son bidimensionales. Si se quiere comprimir una señal transformada no sólo se tienen que codificar sus valores, también se tienen que codificar sus posiciones en el tiempo (o en el espacio para el caso de imágenes). Después de haber transformado una imagen por medio de la TO, ésta puede ser representada por árboles debido al submuestreo que es llevado a cabo durante la transformación. Un coeficiente un una sub-banda baja puede ser planteado como si tuviera cuatro descendientes en la siguiente sub-banda superior (ver figura III.1). Los

cuatro descendientes a su vez, tienen cuatro descendientes cada uno de ellos en la siguiente sub-banda superior y de allí surge, como se puede observar, un árbol cuaternario (cada raíz tiene cuatro ramas).

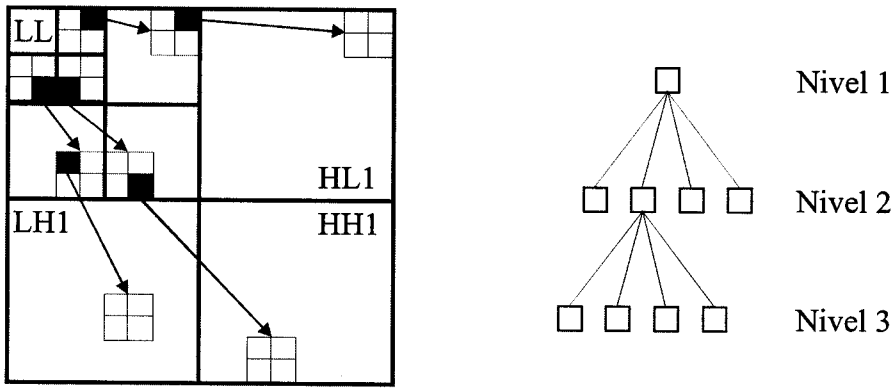


Fig. III.1 Relación entre los coeficientes en diferentes sub-bandas como árboles cuaternarios.

Ahora se puede dar una definición del árbol de ceros. Un árbol de ceros es un árbol cuaternario en el cual todos sus nodos son iguales o menores que su raíz. El árbol es codificado con un solo símbolo y reconstruido por el decodificador como un árbol cuaternario lleno de ceros. Además la raíz tiene que ser menor que el umbral contra el cual los coeficientes son actualmente medidos.

El COAC utiliza a los árboles de ceros basándose en la observación de que los coeficientes decrecen con la escala. Éste supone que habrá una probabilidad muy alta de que todos los coeficientes en un árbol cuaternario serán menores que cierto umbral si la raíz es menor a este umbral. Si éste es el caso, entonces todo el árbol puede ser codificado con un solo símbolo. Ahora, si la imagen es explorada en un orden predefinido, yendo de alta a baja escala, implícitamente muchas posiciones son codificadas a través del uso de símbolos del árbol de ceros. Desde luego, la regla del árbol de ceros es violada frecuentemente, pero generalmente en la práctica, la probabilidad sigue siendo muy alta. El precio que se tiene que pagar por esto, es la adición de nuevos símbolos al código generado.

III.3 Funcionamiento del codificador.

Ahora que se tienen todas las definiciones ya se puede empezar a comprimir. Se empezará por la codificación de los coeficientes en orden decreciente.

Una forma muy directa es la de simplemente transmitir los valores de los coeficientes en orden decreciente, pero esto no es muy eficiente. De ésta manera, muchos bits son desperdiciados en los valores de los coeficientes y no se hace uso del hecho de que se sabe que los coeficientes se encuentran en orden descendente. Una forma más eficiente es la de utilizar un umbral y una señal única si los valores son mayores o menores que el umbral. Si también se transmite el umbral al decodificador, éste puede reconstruir bastante información. Para lograr una reconstrucción perfecta se repite el proceso después de bajar el valor del umbral, se realiza esto hasta que el umbral sea tan pequeño como el coeficiente más pequeño que se desea transmitir. Se puede hacer este proceso aun más eficiente si se resta el umbral a los valores que fueron mayores a éste. Esto resulta en un flujo de bits con precisión ascendente, el cual puede ser reconstruido fácilmente por el decodificador. Si se utiliza una secuencia predeterminada de umbrales, entonces no se necesitan transmitir estos hacia el decodificador, y por lo tanto se ahorraría cierto ancho de banda. Si la secuencia predeterminada fuera una serie de potencias de dos, ésta es llamada codificación del plano de bits, ya que los umbrales en este caso corresponden a los bits en la representación binaria de los coeficientes. La codificación del COAC como se describe en [20] utiliza éste tipo de codificación de los valores de los coeficientes.

De cualquier manera, todavía hace falta un aspecto importante: la transmisión de las posiciones de los coeficientes. De hecho, sin esta información el decodificador no será capaz de reconstruir la señal codificada (aunque puede reconstruir perfectamente el flujo de bits). Es en la codificación de las posiciones donde los codificadores eficientes se separan de los demás. Como se mencionó anteriormente, la codificación del COAC utiliza un orden de rastreo predefinido para codificar la posición de los coeficientes (ver figura III.2).

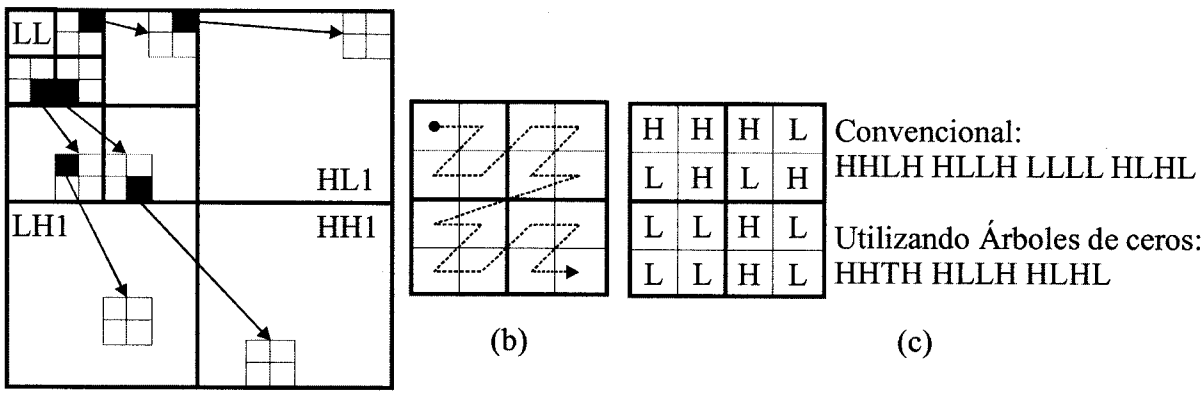


Fig. III.2 (a) Relación entre los coeficientes en diferentes sub-bandas. (b) Orden de rastreo. (c) Símbolos.

La relación entre los coeficientes en diferentes sub-bandas (izquierda), el orden de rastreo, y el resultado de utilizar símbolos de árboles de ceros (T) en el proceso de codificación . Una H significa que el coeficiente es mayor que el umbral y una L significa que es menor. El símbolo de árboles (T) reemplaza a las cuatro L's en la parte inferior izquierda y la L en la parte superior izquierda.

Por medio del uso de árboles de ceros muchas posiciones son codificadas en forma implícita. Además es posible utilizar varios órdenes de rastreo diferentes (ver figura III.3), siempre y cuando las sub-bandas bajas estén completamente rastreadas antes de pasar hacia las sub-bandas altas. En [20] se utiliza un rastreador denominado *raster* (ver figura III.3), mientras que en [1] se mencionan otros tipos ordenes de rastreo.

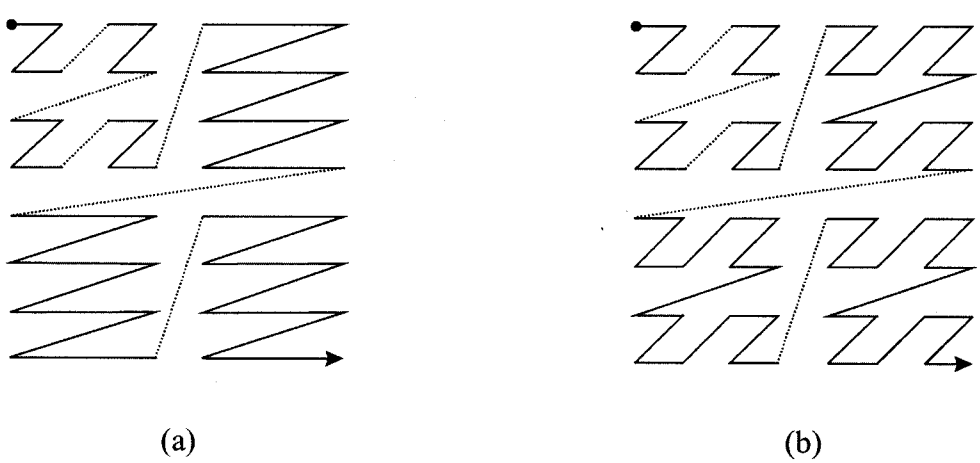


Fig. III.3 (a) Tipo raster, (b) Tipo Morton.

Ahora que se conoce cómo el esquema del COAC codifica los valores y posiciones de los coeficientes, se procederá a ver el algoritmo.

III.3.1 El algoritmo.

El flujo de salida del algoritmo del COAC deberá empezar con alguna información para sincronizar el decodificador. La información mínima requerida por el decodificador es el número de niveles de la transformada de ondículas y el umbral inicial, si se asume que siempre se usará la misma transformada. Además se pueden enviar las dimensiones de la imagen y la media de la misma. Esto puede aumentar la relación señal a ruido pico (PSNR) en forma significativa.

El primer paso en el algoritmo de codificación del COAC es el de determinar el umbral inicial. Si se adopta la codificación del plano de bits, entonces el umbral inicial t_0 será:

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

Donde MAX( ) significa el coeficiente máximo en la imagen y $\gamma(x,y)$ son los coeficientes, con este umbral se entra al ciclo principal del programa:

```
umbral = umbral_inicial;
do
{
        paso_dominante(imagen);
        paso_subordinado(imagen);
        umbral = umbral / 2;
}
while (umbral > umbral_minimo);
```

Como se puede observar en el segmento de código anterior, se requiere de dos pasos para codificar una imagen. En el primer paso, el paso dominante, la imagen es explorada y un símbolo es introducido por cada coeficiente. Si el coeficiente es mayor al umbral un signo **P** (positivo) es codificado, si el coeficiente es menor a la amplitud negativa del umbral, un signo **N** (negativo) es introducido. Si el coeficiente es la raíz de un árbol de ceros entonces se codifica una **T** (árbol de ceros) y por último, si un coeficiente es menor

que el umbral pero este no es la raíz de un árbol de ceros se introduce una **Z** (cero aislado). Esto ocurre cuando existe un coeficiente mayor que el umbral del árbol. El efecto producido al utilizar códigos **N** y **P** es que cuando un coeficiente resulta ser mayor que el umbral (en magnitud) sus dos bits más significativos son extraídos.

Nótese que para determinar si un coeficiente es la raíz de un árbol de ceros o un cero aislado, se debe de explorar todo el árbol. Esto definitivamente consume tiempo. Además, para evitar codificar un coeficiente que ya ha sido identificado en árboles de ceros se deberá llevar registro de éste, lo que, indudablemente implica que se deberá utilizar memoria adicional para llevar estos registros.

Finalmente, todos los coeficientes que son mayores en valor absoluto al umbral actual son extraídos y puestos sin su signo en la lista subordinada y sus posiciones en la imagen son llenadas con ceros. Esto evitará que sean codificados de nuevo.

El segundo paso, el paso subordinado, es el paso de refinamiento. En [20] esto da pie a jugar un poco con intervalos inseguros, pero lo importante es que esto lleva a extraer el siguiente bit más significativo de todos los coeficientes en la lista subordinada. En [20] esta lista está ordenada (de tal manera que el decodificador pueda hacer lo mismo) para que los coeficientes mayores sean transmitidos primero. No se ha implementado este ordenamiento ya que el beneficio-costo es mínimo, según [1].

El ciclo principal termina cuando el umbral llega a un valor mínimo. Para coeficientes enteros este valor mínimo es igual a cero y la división entre dos puede ser substituido por un corrimiento de un bit a la derecha. Si se agrega otra condición de terminación basada en el número de bits que son sacados por el codificador aritmético entonces se podría lograr cualquier tasa de bits sin hacer demasiado procesamiento.

A continuación se resume lo anterior con el siguiente fragmento de código, empezando por el paso dominante.

```
/* Paso dominante */
inicia_fifo( );
while (fifo_no_vacia)
{
        toma_coeficiente_codificado_de_fifo( );
        if coeficiente fue codificado como P, N o Z then
        {
                codifica_coeficiente_siguiente( );
                coloca_coeficiente_codificado_en_fifo( );
```

```
                    if coeficiente fue codificado como P o N then
                    {
                            agrega abs(coeficiente) a lista subordinada;
                            coloca posición de coeficiente en cero;
                    }
            }
    }
```

Aquí se utilizó una estructura de datos tipo FIFO para mantener un registro de los árboles identificados. Si se quiere entrar en este ciclo se deberá inicializar la lista FIFO agregando "manualmente" los coeficientes del primer árbol a la lista. La llamada a la función *codifica_coeficiente_siguiente( )* verifica el siguiente coeficiente sin codificar en la imagen indicado por el orden de rastreo y da salida a una **P**, **N**, **T** o **Z**. Después de codificar el coeficiente este es puesto en la lista FIFO. Esto automáticamente resultará en un rastreo de tipo Morton. Por lo consiguiente, la lista FIFO contiene únicamente coeficientes que ya han sido codificados, en otras palabras, **P**'s, **N**'s, **T**'s o **Z**'s que ya han sido sacadas por estos coeficientes. Por último, si un coeficiente fue codificado como **P** o **N**, éste es removido de la imagen y colocado en la lista subordinada.

Este ciclo tendrá fin siempre y cuando se asegure que los coeficientes en el último nivel (las sub-bandas altas) sean codificados como árboles de ceros.

Al paso dominante le sigue el paso subordinado:

```
/* Paso subordinado */
umbral_subordinado = umbral_actual / 2;
for todos los elementos en la lista subordinada do
{
if coeficiente > umbral_subordinado then
{
        saca un uno;
        coeficiente = coeficiente – umbral_subordinado;
}
else
        saca un cero;
}
```

Si se utilizan umbrales que son potencias de dos, entonces el paso subordinado se reduce a solo algunas operaciones lógicas y este puede ser muy rápido.

Ejemplo

En [20] se muestra un ejemplo no exhaustivo. Aquí se dará el flujo de salida completo del algoritmo de COAC. Los datos del ejemplo se muestran en la figura III.4.

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

Fig. III.4 Datos del ejemplo presentado.

El flujo resultante, dividido en pasos dominante y subordinado por nivel, es:

D1 : *PNZTPTTTTZTTTTTTTPTT*

S1 : *1010*

D2 : *ZTNPTTTTTTTT*

S2 : *100110*

D3 : *ZZZZZPPNPPNTTNNPTPTTNTTTTTTTTPTTTPTTTTTTTTTPTTTTTTTT*

S3 : *10011101111011011000*

D4 : *ZZZZZZZTZTZNZZZZPTTPTPPTPNPTNTTTTTPTPNPPPPTTTTTPTPTTTPNP*

S4 : *110111110110010000011101101000100101011 00*

D5 : *ZZZZZTZZZZZTPZZZTTPTTTTNPTPPTTPTTTNPPNTTTTPNNPTTPTTPPTTT*

S5 : *1011110011010001011111010110110010000000011011011001100011 1*

D6 : *ZZZTTZTTTZTTTTTNNTTT*

El paso subordinado en el último nivel puede ser omitido debido a que el umbral en ese momento ya es cero, por lo tanto no tiene sentido llevarlo a cabo.

El ejemplo presentado anteriormente es puramente demostrativo del codificador de árboles de ceros, por tal motivo no se obtuvo una tasa de compresión, ésta se obtiene en conjunto con la Transformación a Ondículas, la cual se implementará en el siguiente capítulo.

IV IMPLEMENTACIÓN DEL SISTEMA DE PROCESAMIENTO DIGITAL DE IMÁGENES EN EL DSK 6211.

IV.1 Introducción.

El Sistema de Procesamiento Digital de Imágenes para el DSK 6211 (SPDI6211) implementado como parte de éste trabajo, consiste de un sistema integral que fue diseñado con el fin de proporcionar una herramienta didáctica para la evaluación y desempeño de algoritmos de procesamiento digital de imágenes, en la plataforma DSK TMS320C6211 de la compañía Texas Instruments.

Este sistema está constituido por dos subsistemas: el programa anfitrión y el programa destino. El programa anfitrión es el programa implementado para la computadora personal, el cual únicamente se encarga de interactuar con el usuario, éste programa fue realizado con el lenguaje de programación visual *Borland C++ Builder 4*. El programa destino es el programa desarrollado sobre la plataforma DSP (DSK 6211), éste último se encarga de llevar a cabo las labores de procesamiento de las imágenes, el programa destino fue programado utilizando el paquete de desarrollo llamado *Code Composer Studio*. Este paquete se utiliza para implementar aplicaciones para los procesadores digitales de señales de la familia '6000 de Texas Instruments.

En este capítulo se abordarán cada una de las partes que integran este sistema y cómo fueron implementadas. Entre estas partes se pueden encontrar: la comunicación entre el anfitrión y el destino, las técnicas clásicas de procesamiento digital de imágenes (binarización, detección de bordes y filtrado) y la compresión de imágenes utilizando la transformada de ondículas.

IV.2 Comunicación Anfitrión-Destino.

Como se mencionó anteriormente, el SPDI6211 se conforma de dos subsistemas: programa anfitrión y programa destino. Dichos subsistemas requieren de un esquema de comunicación capaz de intercambiar datos y palabras de control entre ellos de una manera eficiente.

El DSK 6211 cuenta con un dispositivo periférico capaz de funcionar como interfaz entre la plataforma destino y la plataforma anfitrión, en este caso, la computadora personal. A dicho dispositivo se le conoce como Puerto Interfaz al Anfitrión (HPI, Host Port Interface), el cual se interconecta directamente hacia el puerto paralelo (LPT) de la computadora personal. En la figura IV.1 se muestra un diagrama general de la interconexión entre las plataformas anfitrión y destino. En esta figura se puede observar además la interfaz EMIF (External Memory Interface) que se encarga de interconectar al procesador digital de señales con la memoria externa incluida en el DSK 6211. El manejo de esta interfaz es transparente para el programador, es decir, no es necesario tomar ninguna consideración adicional para la comunicación con esta interfaz de memoria externa, ya que las bibliotecas de soporte, las cuales se mencionarán a continuación, se encargan del control de la interfaz EMIF.

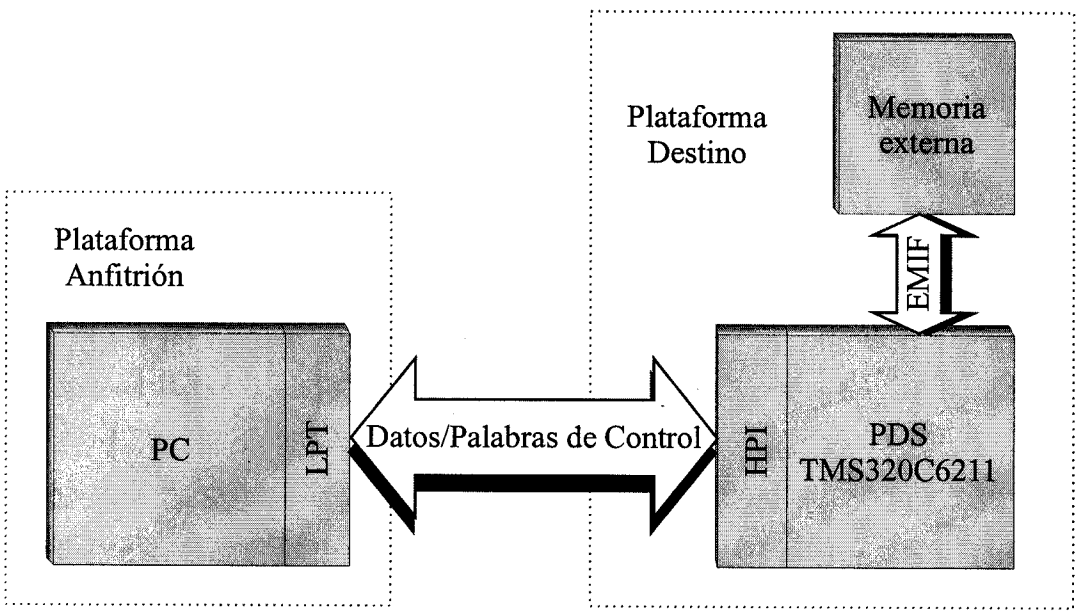


Fig. IV.1 Interconexión anfitrión-destino.

Los fabricantes del DSK 6211 (Texas Instruments) proporcionan a los programadores de aplicaciones de PDS, una librería de enlace dinámico llamada dsk6211.dll (DLL, Dynamic Link Library) la cual provee funciones para escribir y leer datos a/y de la memoria del procesador digital de señales por medio del HPI. En la tabla IV.1 se muestran las funciones contenidas en esa librería que se utilizaron en la implementación del SPDI6211, así como una breve descripción de lo que cada una de estas funciones realiza.

Tabla IV.1 Funciones de la biblioteca dsk6211.dll para el manejo de la interfaz HPI.

Nombre de la función	**Descripción**
BOOL dsk6x_hpi_open(dskHANDLE dskHndl)	Establece una conexión hacia el HPI del DSK 6211.
BOOL dsk6x_hpi_read(dskHANDLEdskHndl, unsigned long *p_buffer, unsigned long *p_length, unsigned long src_addr)	Lee un bloque de datos de 32 bits de la memoria del DSK 6211 por medio de la interfaz HPI.
BOOL dsk6x_hpi_write(dskHANDLE dskHndl, unsigned long *p_buffer, unsigned long *p_length, unsigned long dest_addr)	Escribe un bloque de datos de 32 bits a la memoria del DSK 6211 por medio de la interfaz HPI.
BOOL dsk6x_hpi_close(dskHANDLE dskHndl)	Cierra una sesión con la interfaz HPI.

Los datos que se van a procesar (en el caso del SPDI6211 son imágenes) son capturados desde la plataforma anfitrión (computadora personal) por el usuario, pero se requiere que estos datos sean procesados en la plataforma destino (procesador digital) por lo cual fue necesario implementar un esquema de comunicación que transfiera esos bloques de datos al destino, también es necesario informarle a la plataforma destino qué tipo de procesamiento se desea realizar y además solicitar que el destino proporcione una respuesta sobre su estado actual.

Lo que se implementó para cumplir con este objetivo, fue una muelle de comunicación la cual es un arreglo de seis palabras de 32 bits que van a cumplir con la

labor de transferir palabras de control en forma bidireccional entre las dos plataformas. La definición de la muelle de comunicación anfitrión-destino es la siguiente:

unsigned int handShakingBuffer[HS_BUFFER_LEN];

donde *HS_BUFFER_LEN* es una constante definida con un valor de 6.

En la figura IV.2 se muestra la muelle de comunicación y el uso específico de cada una las posiciones.

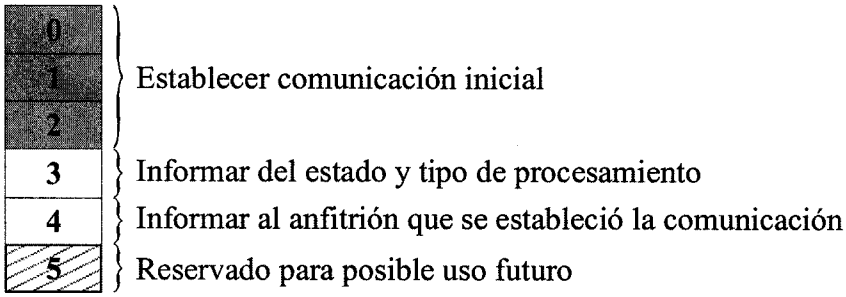


Fig. IV.2 Muelle de comunicación anfitrión-destino.

Cada una de las palabras de control fueron definidas para comunicar una información muy específica. En la tabla IV.2 se muestran las palabras de control utilizadas en el SPDI6211 así como su valor en formato hexadecimal.

Tabla IV.2 Palabras de control.

Nombre	Valor
LISTO	0x0000
BINARIZACION	0x1000
FILTRADO	0x2000
DETECCION	0x3000
FILTRO_PROM	0x4000
FILTRO_DEF	0x5000
BIOR5_5ANALISIS	0x6000
BIOR5_5SINTESIS	0x7000
BIOR5_5ANALSINT	0x8000
HOST_RECEIVE_HAND_SHAKING_INFO	0x55555555

La primera palabra de control (LISTO) se utiliza para que el destino indique al anfitrión que se ha finalizado una sesión de procesamiento y que éste ya está listo para recibir otra palabra de control perteneciente a otra posible técnica de procesamiento. Las siguientes ocho palabras de control que se muestran en la tabla IV.2 le indican al anfitrión qué tipo de procesamiento deberá éste de realizar, y por último, la palabra de control HOST_RECEIVE_HAND_SHAKING_INFO es utilizada al inicio del programa para establecer un enlace entre las plataformas anfitrión-destino y corroborar que ambas estén funcionando adecuadamente al inicio de la aplicación, ésto con el fin de evitar problemas posteriores a la hora de intercambiar datos/palabras de control.

IV.2.1 Establecimiento de la comunicación inicial.

Cuando se inicia la ejecución del SPDI6211 desde la plataforma anfitrión, lo primero que se realiza es una serie de inicializaciones del DSK 6211, ésto se logra ejecutando un conjunto de funciones de la biblioteca ya mencionada, además de inicializarse la plataforma destino es necesario cargar el programa ejecutable para ésta plataforma (archivo llamado 'spdi6211.out'), a continuación se muestra la secuencia de instrucciones requeridas para la inicialización adecuada de la plataforma destino y sus dispositivos periféricos:

```
dsk6x_open(nPort,&hBd);
dsk6x_reset_board(hBd);
dsk6x_reset_dsp(hBd,0,1);
dsk6x_hpi_open(hBd);
dsk6x_coff_load(hBd,"spdi6211.out",0,0,0);
dsk6x_hpi_generate_int(hBd);
```

Una vez que se hayan ejecutado estas instrucciones, el programa destino ya se encontrará en estado de ejecución, pero de cualquier manera el programa anfitrión tendrá que esperar a que el destino llegue a cierto punto de su ejecución para que éste se encuentre listo para procesar datos, esto se logra haciendo esperar al programa anfitrión a que la plataforma destino escriba los identificadores ID_0, ID_1 e ID_2 en las posiciones 0, 1 y 2 respectivamente de la muelle de comunicación anfitrión-destino, por lo tanto, el anfitrión tendrá que estar leyendo continuamente la muelle de comunicación para conocer su estado y así poder detectar cuando el destino escribió los identificadores mencionados, una vez que ésto haya ocurrido, el programa anfitrión escribe la palabra de control

HOST_RECEIVE_HAND_SHAKING_INFO en la posición 4 de la muelle de comunicación, y al detectar esto la plataforma destino, se colocará en el estado de listo para procesar.

IV.2.2 Transferencia de imágenes entre el anfitrión y el destino.

El procesamiento de las imágenes en el SPDI6211 se implementó mediante la definición de dos vectores unidimensionales que tienen una longitud de MAX * MAX, donde MAX es una constante que define las dimensiones máximas permitidas para imágenes procesadas en el sistema, en éste caso se decidió limitar las imágenes a 256 x 256 píxeles, debido a la cantidad de memoria en el DSK 6211 que es de 4 MB. En la plataforma destino se definieron dos imágenes principales:

```
char img1[MAX*MAX];
char img2[MAX*MAX];
```

donde *img1* es la imagen recibida de la plataforma anfitrión e *img2* es el vector donde se colocará la imagen ya procesada por la técnica elegida.

Para transferir las imágenes bidireccionalmente entre las dos plataformas, es necesario que la plataforma anfitrión tenga conocimiento de la dirección de memoria de la plataforma destino donde las imágenes son almacenadas. Para lograr ésto se tuvieron que definir secciones de memoria especiales (en el archivo lnk.cmd) para cada una de estas imágenes donde se les asignó una dirección de memoria fija, y posteriormente por medio de las siguientes directivas:

```
#pragma DATA_SECTION(img1, "img1")
#pragma DATA_SECTION(img2, "img2")
```

se le indica al compilador que las variables *img1* e *img2* deberán de estar ubicadas físicamente en las secciones de memoria creadas con el mismo nombre. Las direcciones de memoria que se utilizaron se almacenaron en las siguientes constantes:

```
#define DIR_IMAGEN        0x80000000
#define DIR_IMAGEN2       0x80010000
```

y una vez hecho esto, ambas plataformas tienen conocimiento de la localidad de memoria donde se encuentran las imágenes.

Una vez establecidos estos parámetros, se procedió a programar las rutinas del programa anfitrión que transfieren la imagen hacia el destino y también la rutina que transfiere la imagen ya procesada desde la plataforma destino de regreso a la plataforma anfitrión.

Cuando el usuario carga una imagen de archivo en el programa anfitrión, esta imagen es transferida en forma automática hacia la plataforma destino por medio de la rutina *subeImagenDSP()*, el código de esta rutina se muestra a continuación:

```
for (unsigned short y = 0; y < (unsigned short)alto; y++)
{
        longitud=ancho;
        ptr = (unsigned char *)Imagen1->Picture->Bitmap->ScanLine[y];
        dsk6x_hpi_write(hBd,(unsigned long *)ptr,&longitud,
         DIR_IMAGEN+(y*ancho));
}
```

de una forma similar, se implementó la rutina bajaImagenDSP() que es utilizada para regresar la imagen ya procesada de la plataforma destino a la plataforma anfitrión:

```
for (unsigned short y = 0; y < (unsigned short)alto; y++)
{
        longitud=ancho;
        ptr = (unsigned char *)Imagen2->Picture->Bitmap->ScanLine[y];
        dsk6x_hpi_read(hBd,(unsigned long *)ptr,&longitud,
        DIR_IMAGEN2+(y*ancho));
}
```

IV.3 Técnicas clásicas de procesamiento digital de imágenes.

En esta sección se presentan las técnicas clásicas de procesamiento digital de imágenes que se implementaron en el SPDI6211 y cómo fueron desarrolladas cada una de ellas.

IV.3.1 Binarización.

Como se mencionó en el capítulo I, la técnica de binarización consiste en convertir una imagen con 256 tonos de gris a una imagen en blanco y negro (1's y 0's).

En el SPDI6211 se logra la técnica de binarización capturando el umbral en la plataforma anfitrión, ese umbral es transmitido directamente a la plataforma destino en una dirección de memoria conocida:

```
#define DIR_UMBRAL          0x80020000
```

una vez que se ha transmitido el umbral de binarización se transmite además la palabra correspondiente a la técnica de procesamiento elegida, en este caso, obviamente es BINARIZACION (ver tabla IV.2). Una vez hecho ésto la plataforma destino inicia la binarización de la imagen ya cargada en memoria. A continuación se presenta el segmento de código crítico de la rutina de binarización implementado en el DSK6211:

```
for (i=0 ; i<(alto*ancho); i++)
{
   if ((unsigned char)ent[i]>um)
     sal[i]=-1;
   else
     sal[i]=0;
}
```

donde *alto* y *ancho* son las dimensiones de la imagen a procesar, *ent* y *sal* corresponde a las imágenes de entrada y de salida respectivamente y *um* representa el valor de umbral dado por el usuario.

IV.3.1.1 Ejemplos de binarización.

En las figuras IV.3 y IV.4 se muestran ejemplos dos de imágenes binarizadas en el SPDI6211.

a) Imagen original.

b) Imagen binarizada.

Fig. IV.3 Imagen de Lena binarizada.

a) Imagen original.

b) Imagen binarizada.

Fig. IV.4 Imagen del camarógrafo binarizada.

En el ***Apéndice D*** se muestran los tiempos de procesamiento de cuatro experimentos realizados utilizando esta técnica de procesamiento.

IV.3.2 Detección de bordes.

La detección de bordes fue implementada binarizando la imagen de entrada utilizando un valor de umbral dado por el usuario, y pasándola posteriormente, por un filtro bidimensional 3x3 el cual es el encargado de calcular la operación del gradiente combinado (mencionado en el capítulo I), para el proceso de filtrado bidimensional se utiliza la rutina *convol2d* la cual se abordará posteriormente en la subsección IV.3.3. En esta técnica, el usuario no tiene que capturar ningún parámetro de entrada en la plataforma anfitrión, únicamente tendrá que seleccionar la técnica de detección de bordes y el programa anfitrión comunicará la palabra de control adecuada (ver tabla IV. 2) y una vez efectuado ésto, se ejecutará el código del algoritmo de detección de bordes, el cual se muestra a continuación:

```
char fil[3][3];
fil[0][0] = 0; fil[0][1] = -1; fil[0][2]= 0;
fil[1][0] = -1; fil[1][1] = 5; fil[1][2]= -1;
fil[2][0] = 0; fil[2][1] = -1; fil[2][2]= 0;
binarizacion(img1,img2,/*umbral*/128);
convol2d(img2,img3,fil);
```

donde *fil* es el filtro 3x3 mencionado previamente, *img1* e *img2* son las imágenes de entrada y de salida respectivamente e *img3* es una imagen de procesamiento intermedio.

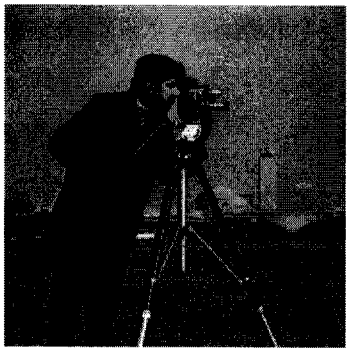
IV.3.2.1 Ejemplos de detección de bordes.

En las figuras IV.5 y IV.6 se muestran dos ejemplos de imágenes procesadas mediante la técnica de detección de bordes en el SPDI6211.

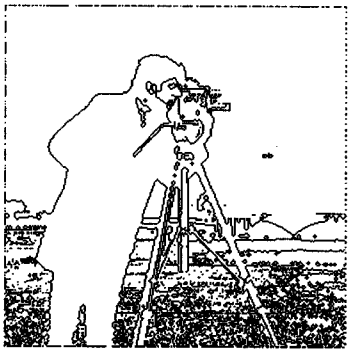
a) Imagen original.

b) Imagen procesada.

Fig. IV.5 Detección de bordes de la imagen de Lena.



a) Imagen original.



b) Imagen procesada.

Fig. IV.6 Detección de bordes de la imagen del camarógrafo.

IV.3.3 Filtrado bidimensional.

El filtrado bidimensional fue implementado en el SPDI6211 con el fin de mostrar los efectos que ocurren al procesar una imagen con un filtro bidimensional, como se sabe conceptualmente, ésto se logra por medio de la operación de convolución bidimensional lineal, la cual se implementó en la plataforma destino (función *convol2d*) y cuyo código se muestra a continuación:

```
void convol2d(char *ent, char *sal, char hn[3][3])
{
 int drows,dcols,normal;
 int i,j,ii,jj,sumval;
 unsigned char *inptr,*outptr;
 char *filptr;
 drows=3/2;
 dcols=3/2;
 normal=0;
 for (i=0;i<3;i++)
  for(j=0;j<3;j++)
    normal+=abs(hn[i][j]);
 if (!normal)
  normal=1;
 for (i=0;i<alto-3+1;i++)
 {
  outptr=(unsigned char *)&sal[(i+drows)*ancho];
  for (j=0;j<ancho-3+1;j++)
  {
   sumval=0;
   for (ii=0;ii<3;ii++ )
   {
    inptr=(unsigned char *)&ent[(ii+i)*ancho];
    inptr=inptr+j;
    filptr=(char *) hn[ii];
    for (jj=0;jj<3;jj++)
      sumval+=(*inptr++) * (*filptr++);
   }
```

```
        outptr[j+dcols]=sumval/normal;
      }
    }
  }
```

donde *ent* y *sal* representan las imágenes de entrada y salida respectivamente y *hn* la matriz del filtro a convolucionar. La función convol2d se utilizó en la técnica de detección de bordes (ver sección IV.3.2), para procesar la imagen con el filtro de detección. Además de ésto, se implementaron dos opciones de procesamiento en el programa anfitrión: filtro promediador y filtro definido por el usuario.

IV.3.3.1 Filtro promediador.

Esta opción de filtrado pasa la imagen a través de un filtro 3x3 del tipo que se muestra en la ecuación I.1., sólo que en este caso, se implementó la rutina para que los filtros tuvieran coeficientes de 8 bits por lo tanto fue necesario escalar estos coeficientes por el valor:

$$\frac{1}{MN} = \frac{1}{(3)(3)} = \frac{1}{9} = 0.\overline{1111} \qquad \text{(IV.1)}$$

se determinaron los coeficientes normalizados de la siguiente manera:

$$h(i,j) = \frac{1}{9}\sum_{i=1}^{3}\sum_{j=1}^{3} h(i,j) = 1 \qquad \text{(IV.2)}$$

por lo tanto, h(m,n) resulta ser una matriz 3x3 con coeficientes unitarios:

$$h(m,n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \text{(IV.3)}$$

y ésto se expresa en el siguiente código:

```
void filtro_prom()
{
 char hn[3][3]={1,1,1,
                1,1,1,
                1,1,1};
 convol2d(img1,img2,hn);
}
```

en este caso *img1* e *img2* son las imágenes de entrada y de salida respectivamente.

Ejemplos

En las figuras IV.7. y IV. 8 se muestran imágenes pasadas por este filtro, a estas imágenes se les agregó ruido de tipo sal y pimienta (conocido en inglés como: salt & pepper). Como se puede observar, esta técnica se puede utilizar para reducir el nivel de ruido de una imagen aunque este tipo de filtro hace que la imagen quede un poco borrosa.

a) Imagen original

b) Imagen con ruido (*variancia=0.01*).

c) Imagen filtrada.

Fig. IV.7 Imagen de Shania pasada por el filtro promediador 3x3.

a) Imagen original

b) Imagen con ruido (*variancia=0.01*).

c) Imagen filtrada.

Fig. IV.8 Imagen de las flores pasada por el filtro promediador 3x3.

IV.3.3.2 Filtro definido por el usuario.

En esta opción del programa anfitrión el usuario tiene la posibilidad de capturar los coeficientes del filtro 3x3 con los que va a procesar su imagen (ver figura IV.9), como ya se mencionó, los coeficientes del filtro deberán ser enteros de 8 bits. Una vez que estos coeficientes hayan sido capturados, se transfieren al programa destino a una variable global definida ahí mismo:

```
char filtro[3][3];
```

en la dirección de memoria dada por:

```
#define DIR_FILTRO          0x80020024
```

una vez que se haya realizado esto, el programa destino inicia la sesión de procesamiento cuyo código se muestra a continuación:

```
convol2d(img1,img2,filtro);
```

y una vez finalizada esta sesión, el programa destino devolverá la imagen procesada al programa anfitrión.

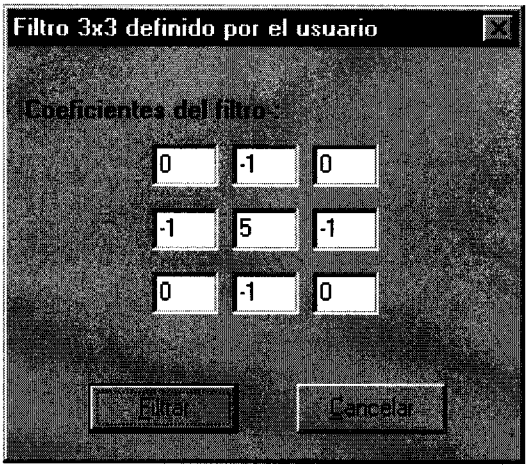


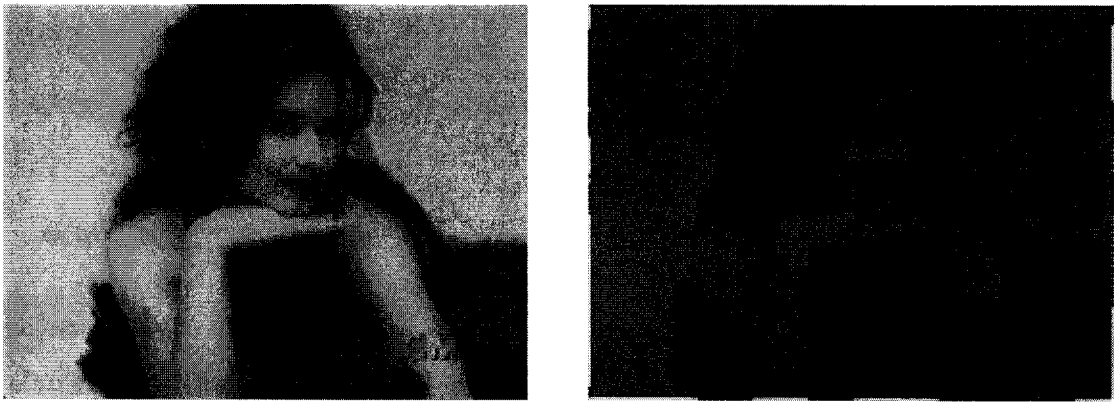
Fig. IV.9 Pantalla de captura de los coeficientes del filtro (programa anfitrión).

Ejemplos

En las figuras IV.10 y IV.11 se muestran ejemplos de imágenes filtradas por medio de filtros específicos, cuyos coeficientes fueron capturados en el programa anfitrión.

La figura IV.10 muestra la imagen filtrada por un filtro cuyos coeficientes se muestran en la ecuación IV.4.

$$h(m,n) = \begin{bmatrix} 4 & -1 & 4 \\ -1 & -5 & -1 \\ 4 & -1 & 4 \end{bmatrix} \qquad \text{(IV.4)}$$



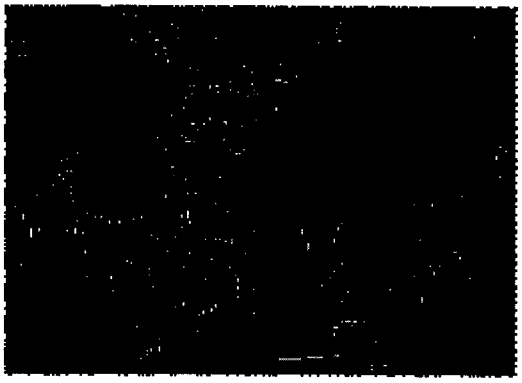
a) Imagen original. b) Imagen filtrada.

Fig. IV.10 Imagen de Shania procesada con el filtro IV.4.

La figura IV.11 muestra la imagen filtrada por un filtro cuyos coeficientes se muestran en la ecuación IV.5.

$$h(m,n) = \begin{bmatrix} 4 & 0 & 4 \\ 0 & 8 & 0 \\ 4 & 0 & 4 \end{bmatrix} \qquad \text{(IV.5)}$$

a) Imagen original.



b) Imagen filtrada.

Fig. IV.11 Imagen de las flores procesada con el filtro IV.5.

IV.4 Compresión de imágenes utilizando ondículas.

En esta sección se abordarán las técnicas que se utilizaron para obtener la compresión de imágenes usando la transformada de ondículas.

IV.4.1 Filtros biortogonales 5.5.

Los filtros biortogonales fueron implementados en el SPDI6211 debido a que son los que mejor desempeño ofrecen en la compresión digital de imágenes [13]. En la etapa de análisis y de síntesis de los filtros biortogonales 5.5 se requiere de un filtro pasa bajas y otro filtro pasa altas en cada una de estas etapas. Cada uno de estos filtros cuenta con doce coeficientes y satisfacen las condiciones de reconstrucción perfecta (ver apéndice A).

Para obtener esos coeficientes de los filtros se utilizó el siguiente mando de MatLab:

[LO_D,HI_D,LO_R,HI_R] = wfilters('bior5.5')

La función *wfilters* calcula los coeficientes de los cuatro filtros requeridos para la obtención de las ondículas, como ya se mencionó, los filtros son del tipo biortogonales 5.5 por lo cual hay que enviar como parámetro *'bior5.5'* y esta función devuelve cuatro vectores: en *LO_D* los coeficientes del filtro pasa bajas de análisis, en *HI_D* los coeficientes pasa altas del filtro de análisis, en *LO_R* los coeficientes del filtro pasa bajas de síntesis y en *HI_R* los coeficientes del filtro pasa altas de síntesis.

Se utilizaron tres niveles de descomposición ya que de acuerdo a [17] es el número óptimo de niveles de codificación requerido para obtener un equilibrio entre los resultados de compresión y el tiempo de procesamiento requerido.

IV.4.1.1 Filtro de análisis.

Como ya se mencionó anteriormente, se implementó un banco de filtros de análisis biortogonales de tres niveles de descomposición. Lo que se hace primero antes de pasar la imagen original por el banco de filtros es convertir cada una de las muestras de la imagen original a valores de tipo flotante, ésto debido a que la imagen original proveniente de la plataforma anfitrión viene con muestras de enteros de un byte (valores de -128 a 127), en cambio los coeficientes de los filtros como se puede observar en el apéndice A, son valores flotantes de pequeña magnitud, por esta razón se optó por convertir la imagen original a su equivalente flotante. Por ejemplo: el valor 126 entero se convirtió a 126.0 flotante. A continuación se muestra la rutina para realizar esta tarea:

```
void copia_a_img1f(char *img)
{
 short i,j;
 for (i=0;i<ancho;i++)
  for(j=0;j<alto;j++)
   img1f[i][j]=(float)(img[i+(ancho*j)]);
}
```

Esta rutina copia la imagen de formato entero *img* definida como parámetro a la imagen de formato flotante *img1f*, la cual fue definida de forma global como una matriz de *N*x*N*.

En la figura IV.12 se muestra la primera etapa del filtro de análisis implementado en el sistema. Como se puede observar, la imagen original es pasada por renglones a cada uno de los dos filtros biortogonales h_1 y h_2 cuyos coeficientes se enlistan en el apéndice A, la salida de estos filtros es submuestreada con un factor de dos y posteriormente ya en forma de columnas se procesa en otra cascada de esos mismos filtros, finalmente se repite

el proceso de submuestreo que da como resultado las cuatro sub-bandas de la imagen, de las cuales la sub-banda baja-baja (esquina superior izquierda) es a la que se le vuelve a aplicar este mismo proceso en la segunda etapa de análisis, y este proceso se repite hasta cumplir con las tres etapas propuestas inicialmente.

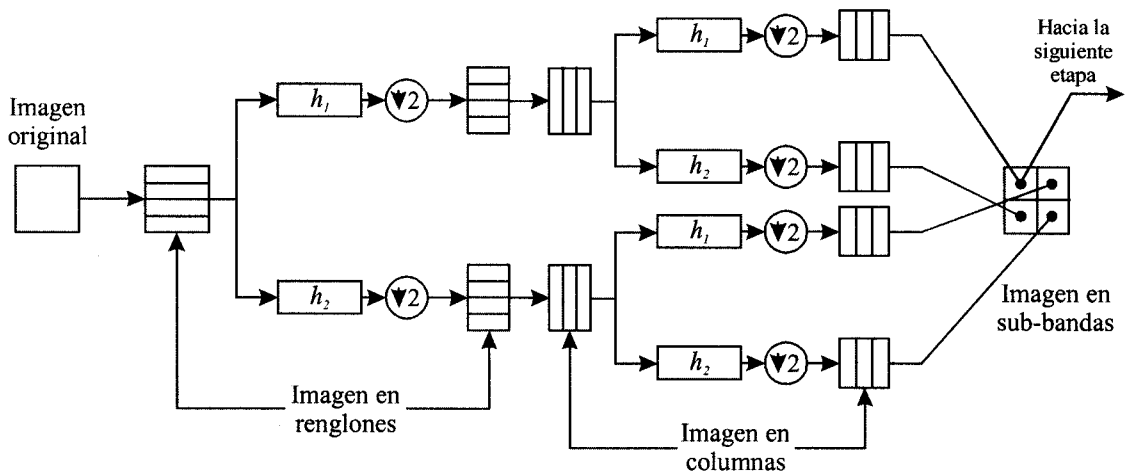


Fig. IV.12 Primer nivel del banco de filtros de análisis implementado.

Se elige la sub-banda baja-baja para ser procesada en las etapas subsecuentes ya que es la que cuenta con la información más relevante de la imagen, y por otro lado, las sub-bandas de altas frecuencias no aportan un gran contenido de información a la imagen.

Para procesar la imagen por el banco de filtros biortogonales se recurrió a la implementación del algoritmo de la convolución cíclica (ver figura IV.13), ésto con el fin de minimizar el error en la imagen filtrada en los bordes de ésta [13].

En la figura IV.13a se observa que se le agregaron las primeras k muestras al final de cada renglón y en la figura IV.13b se observa que se le agregaron k muestras al final de cada columna de la imagen I. Cabe señalar que k corresponde al número de coeficientes del filtro con el que se va a convolucionar la imagen I, en este caso k=12.

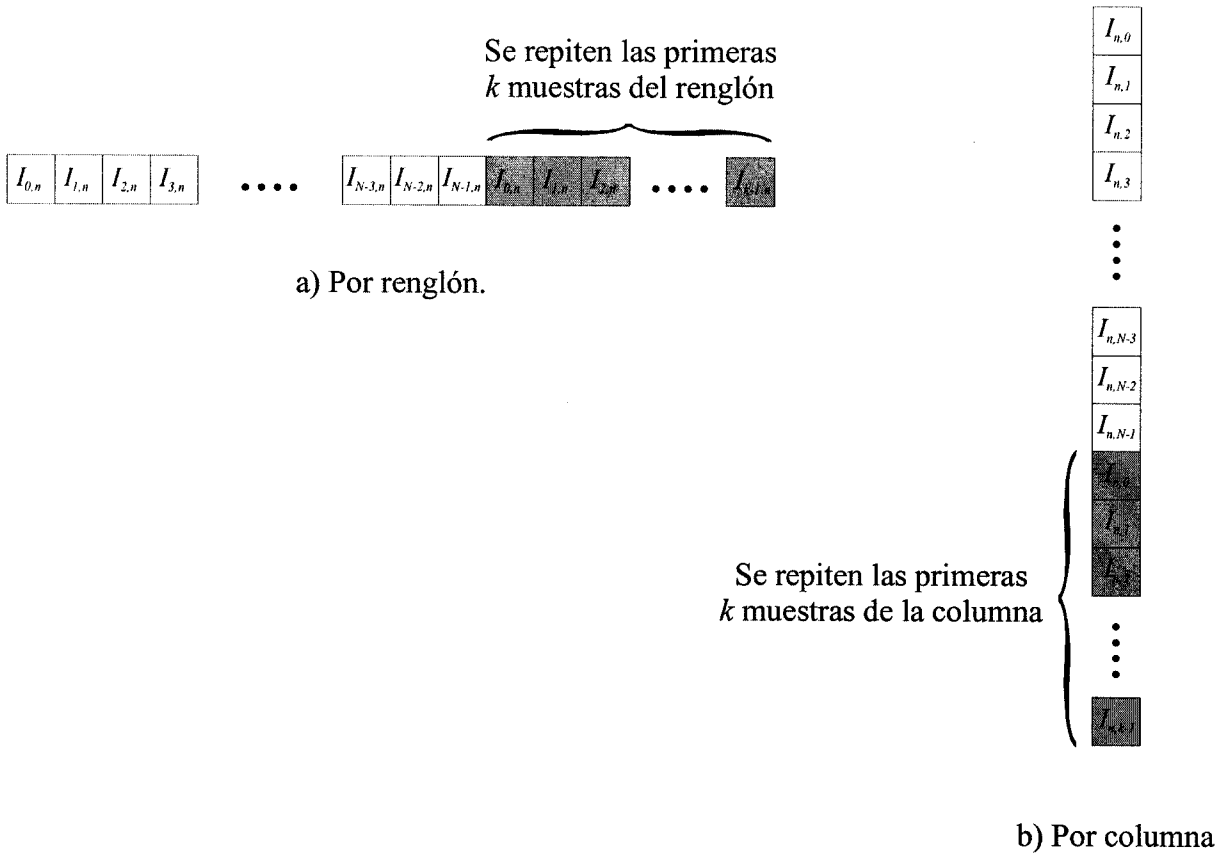


Fig. IV.13 Arreglos hechos a la imagen para obtener la convolución cíclica.

A continuación se muestra la rutina llamada *bior5_5analisis*:

```
void bior5_5analisis(unsigned char num_niv)
{
 float res1,res2;
 short i,j,k,ancho_l,alto_l;
 unsigned char niv;
 alto_l=alto;
 ancho_l=ancho;
 for (niv=0;niv<num_niv;niv++)
 {
        for (j=0;j<alto_l;j++)
        {
    for(i=0;i<ancho_l;i++)
      v1[i]=img1f[i][j];
    for(i=0;i<N;i++)
      v1[i+ancho_l]=img1f[i][j];
  /* Convolucion y diezmado en el mismo ciclo */
  /* Se implemento el filtrado pasabajas y pasa altas en el mismo ciclo!*/
        for (i=1;i<(ancho_l/2)+1;i++)
        {
                res1=res2=0;
                for (k=0;k<N;k++)
```

```
                {
                res1+=fib[k]*v1[k+2*i];
                res2+=fia[k]*v1[k+2*i];
                }
                asi1[i-1][j]=res1;
                asi2[i-1][j]=res2;
        }
        }
        /* Segunda etapa (Parte baja)*/
        for (i=0;i<ancho_l/2;i++)
        {
          for(j=0;j<alto_l;j++)
            v1[j]=asi1[i][j];
          for(j=0;j<N;j++)
            v1[j+alto_l]=asi1[i][j];
        for (j=1;j<(alto_l/2)+1;j++)
        {
                res1=res2=0;
                for (k=0;k<N;k++)
                {
                res1+=fib[k]*v1[k+2*j];
                res2+=fia[k]*v1[k+2*j];
                }
                img1f[i][j-1]=res1;                                    /*division*/
                img1f[i][(j-1)+(alto_l/2)]=res2;  /*division*/
        }
        }
        /* Tercera etapa (parte alta)*/
        for (i=0;i<ancho_l/2;i++)
        {
                for (j=0;j<alto_l;j++)
                  v1[j]=asi2[i][j];
                for (j=0;j<N;j++)
                  v1[j+alto_l]=asi2[i][j];
        for (j=1;j<(alto_l/2)+1;j++)
        {
                res1=res2=0;
                for (k=0;k<N;k++)
                {
                res1+=fib[k]*v1[k+2*j];
                res2+=fia[k]*v1[k+2*j];
                }
                img1f[i+(ancho_l/2)][j-1]=res1;        /*division*/
                img1f[i+(ancho_l/2)][(j-1)+(alto_l/2)]=res2; /*division*/
        }
        }
    ancho_l/=2;
    alto_l/=2;
  }
}
```

En esta rutina la imagen original en formato flotante (*img1f*) es procesada por el banco de filtros mostrado en la figura IV.12, en esta rutina el número de niveles de descomposición

es enviado como parámetro, lo cual da la oportunidad de manejar varios niveles de descomposición, aunque de antemano ya se mencionó que únicamente se manejarían tres. Una vez que la imagen ya fue procesada por el banco de filtros se obtiene la imagen resultante en la misma matriz *img1f*, que como ya se mencionó es de tipo flotante, posteriormente es necesario que la imagen resultante sea de enteros de un byte, debido a que tanto en el algoritmo de Codificación de Árboles de Ceros (COAC) como para desplegar la imagen resultante en la plataforma anfitrión se requiere de este formato. Para realizar esta tarea se desarrolló la rutina *normaliza* la cual lleva a cabo una cuantización vectorial uniforme:

```
void normaliza()
{
 unsigned short i,j;
 factor=127.0/matrizf_abs_max(img1f,ancho,alto);
 for (i=0;i<ancho;i++)
  for (j=0;j<alto;j++)
   img1f[i][j]*=factor;
}
```

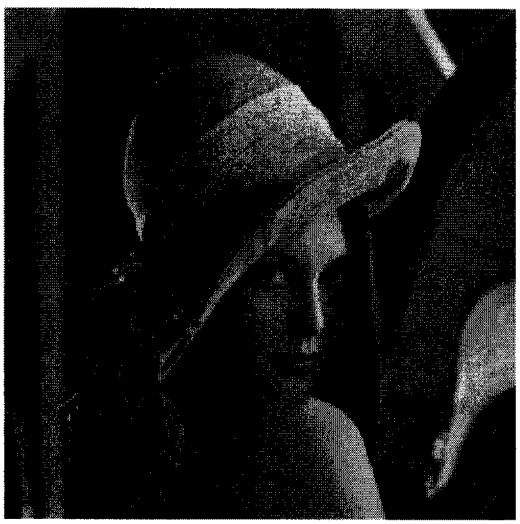
Lo que hace esta rutina es tomar el valor absoluto máximo de la matriz de flotantes y con éste, calcular un valor de escalamiento, posteriormente se multiplica cada una de las muestras de la matriz de flotantes por ese factor, y en ese momento se puede asegurar que cada una de las muestras cae entre –128 y 127 lo cual era el objetivo de este algoritmo. Después de haber cuantizado la matriz se tiene que regresar a entero de un byte, esto se logra con la siguiente rutina:

```
void copia_img1f_a_ent(char *img)
{
 short i,j;
 for (i=0;i<ancho;i++)
  for (j=0;j<alto;j++)
  {
    if (img1f[i][j]<0.0)
      img[i+(ancho*j)]=(char)(img1f[i][j]-0.5);
    else
      img[i+(ancho*j)]=(char)(img1f[i][j]+0.5);
  }
}
```

La imagen en formato entero se guarda en el vector *img* el cual deberá ser de longitud *anchoXalto*. Una vez regresada la imagen original a su formato entero podrá ser devuelta a la plataforma anfitrión para ser visualizada por el usuario o podrá ser codificada por el algoritmo de árboles de ceros el cual se abordará posteriormente.

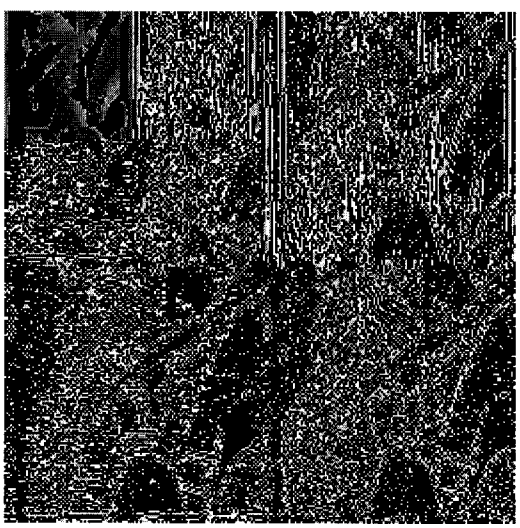
Ejemplos

En la figura IV.14 se muestra la imagen de Lena (256x256) pasada por tres niveles de descomposición a través del banco de filtros biortogonales 5.5 de análisis.

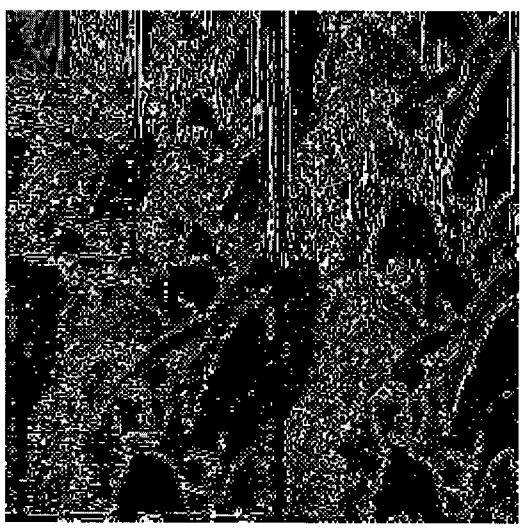


a) Imagen original

b) Imagen después del primer nivel de descomposición.



c) Imagen después del segundo nivel de descomposición.



c) Imagen después del tercer nivel de descomposición.

Fig. IV.14 Imagen de Lena procesada por el banco de filtros biortogonales 5.5.

IV.4.1.2 Filtro de síntesis.

El banco de filtros de síntesis es el encargado de regresar la imagen en sub-bandas a su estado original. Esto se logró implementando dos filtros de síntesis biortogonales 5.5 denominados g_1 y g_2, los coeficientes de este filtro como ya se mencionó se obtuvieron mediante la ayuda de MatLab y tales coeficientes se muestran en el apéndice A. El procedimiento de síntesis al igual que el de análisis, se implementó fijando el número de niveles de descomposición a tres. El procedimiento que se sigue en cada etapa consiste en procesar cada una de las sub-bandas de la imagen por un sobremuestreo con un factor de dos, posteriormente cada una de estas sub-bandas se procesa en un filtro g_1 y g_2 para sub-bandas bajas y sub-bandas altas respectivamente, a la salida de estos filtros, se realiza una sumatoria entre los dos pares de sub-bandas y se sobremuestrea nuevamente la imagen pero esta vez en renglones, una vez realizado este sobremuestreo se vuelve a procesar la imagen en renglones por los filtros g_1 y g_2, finalmente se suman las dos imágenes obtenidas para generar la imagen resultante. Para una explicación gráfica más a detalle ver figura IV.15 donde se muestra un nivel del banco de filtro de síntesis para cuatro sub-bandas.

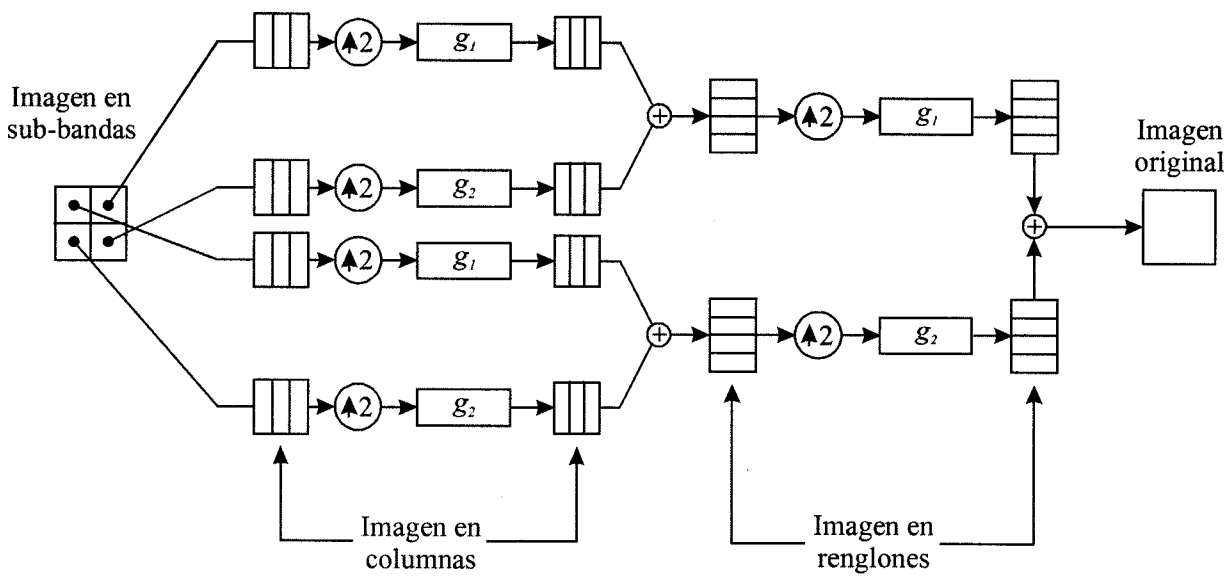


Fig. IV.15 Banco de filtros de síntesis para un nivel de descomposición.

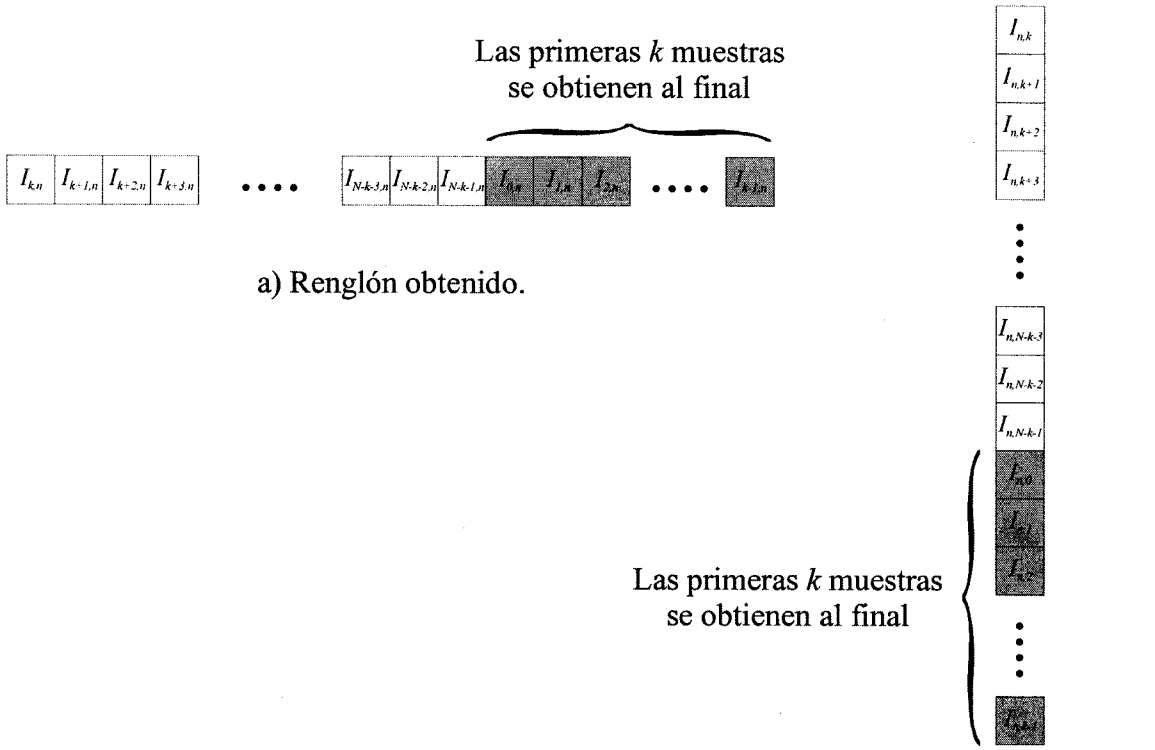
La rutina que se implementó para llevar a cabo la tarea de procesar la imagen en sub-bandas por el banco de filtros de síntesis biortogonales 5.5 se le denominó *bior5_5sintesis*, esta rutina recibe un parámetro entero de 8 bits, el cual indica el número de niveles de descomposición a procesar, como ya se mencionó se trabajó con tres niveles, pero de cualquier manera se dejó abierta la posibilidad de trabajar con cualquier otro número de niveles que se pudieran requerir en un momento dado. La rutina se muestra a continuación:

```
void bior5_5sintesis(unsigned char num_niv)
{
  short i,j,k,ancho_l,alto_l,desp;
 float res1,res2;
 unsigned char niv;
 limpia_vector();
 alto_l=alto;
 ancho_l=ancho;
 desp=0;
 for (niv=0;niv<num_niv;niv++)
 {
  alto_l/=2;
  ancho_l/=2;
 }
 for (niv=0;niv<num_niv;niv++)
 {
  for (i=0;i<ancho_l;i++)
  {
   for(j=0;j<alto_l;j++)
     v1[2*j+1]=img1f[i][j];
   for(j=0;j<N;j++)
     v1[2*j+2*alto_l+1]=img1f[i][j];
   for (j=desp;j<2*alto_l+desp;j++)
   {
    res1=0;
    for (k=0;k<N;k++)
      res1+=fibr[k]*v1[k+j];
    asi1[i][j-desp]=res1;
   }
  }
  for (i=0;i<ancho_l;i++)
  {
   for(j=0;j<alto_l;j++)
     v1[2*j+1]=img1f[i][j+alto_l];
   for(j=0;j<N;j++)
     v1[2*j+2*alto_l+1]=img1f[i][j+alto_l];
   for (j=desp;j<2*alto_l+desp;j++)
   {
    res2=0;
    for (k=0;k<N;k++)
      res2+=fiar[k]*v1[k+j];
    asi2[i][j-desp]=res2;
   }
```

```
 }
 for (i=0;i<ancho_l;i++)
  for (j=0;j<2*alto_l;j++)
    asi1[i][j]+=asi2[i][j];
for (i=ancho_l;i<2*ancho_l;i++)
 {
  for(j=0;j<alto_l;j++)
    v1[2*j+1]=img1f[i][j];
  for(j=0;j<N;j++)
    v1[2*j+2*alto_l+1]=img1f[i][j];
  for (j=desp;j<2*alto_l+desp;j++)
  {
    res1=0;
    for (k=0;k<N;k++)
      res1+=fibr[k]*v1[k+j];
    asi1[i][j-desp]=res1;
  }
 }
 for (i=ancho_l;i<2*ancho_l;i++)
 {
  for(j=0;j<alto_l;j++)
    v1[2*j+1]=img1f[i][j+alto_l];
  for(j=0;j<N;j++)
    v1[2*j+2*alto_l+1]=img1f[i][j+alto_l];
  for (j=desp;j<2*alto_l+desp;j++)
  {
    res2=0;
    for (k=0;k<N;k++)
      res2+=fiar[k]*v1[k+j];
    asi2[i][j-desp]=res2;
  }
 }
 for (i=ancho_l;i<2*ancho_l;i++)
  for (j=0;j<2*alto_l;j++)
    asi1[i][j]+=asi2[i][j];
 for (j=0;j<2*alto_l;j++)
 {
  for (i=0;i<ancho_l;i++)
    v1[2*i+1]=asi1[i][j];
  for (i=0;i<N;i++)
    v1[2*i+2*ancho_l+1]=asi1[i][j];
  for (i=desp;i<2*ancho_l+desp;i++)
  {
    res1=0;
    for (k=0;k<N;k++)
      res1+=fibr[k]*v1[k+i];
    asi2[i-desp][j]=res1;
  }
 }
 for (j=0;j<2*alto_l;j++)
 {
  for (i=0;i<ancho_l;i++)
    v1[2*i+1]=asi1[i+ancho_l][j];
  for (i=0;i<N;i++)
    v1[2*i+2*ancho_l+1]=asi1[i+ancho_l][j];
  for (i=desp;i<2*ancho_l+desp;i++)
```

```
      {
        res2=0;
        for (k=0;k<N;k++)
          res2+=fiar[k]*v1[k+i];
        asi2[i-desp][j]+=(res2);
      }
    }
    for (j=0;j<alto_l*2;j++)
    {
      for (i=0;i<N;i++)
        asi1[i][j]=asi2[i+ancho_l*2-N][j];
      for (i=N;i<ancho_l*2;i++)
        asi1[i][j]=asi2[i-N][j];
    }
    for (i=0;i<ancho_l*2;i++)
    {
      for (j=0;j<N;j++)
        img1f[i][j]=asi1[i][j+alto_l*2-N];
      for (j=N;j<alto_l*2;j++)
        img1f[i][j]=asi1[i][j-N];
    }
    ancho_l*=2;
    alto_l*=2;
  }
}
```

Los coeficientes de los filtros de síntesis pasa bajas y pasa altas se encuentran almacenados en los vectores *fibr* y *fiar* respectivamente, los cuales son vectores de tipo flotante con una longitud de 12. Como se observó en la figura IV.13, antes de procesar la imagen en los filtros de análisis, se realizó el reacomodo de las muestras de cada fila y cada columna. Ahora, al obtener el resultado final de la etapa de síntesis, se obtiene cada fila y cada columna como se ve en la figura IV.16.



b) Columna obtenida

Fig. IV.16 Formato de la imagen obtenida durante la etapa de síntesis.

Por lo tanto, al final de cada nivel de descomposición dentro de la etapa de síntesis es necesario llevar a cabo el proceso inverso, el cual consiste en regresar las últimas k muestras a las primeras k posiciones del vector renglón o vector columna, según sea el caso. Esta labor se llevó a cabo de manera implícita dentro de la misma rutina *bior5_5sintesis*. Una vez realizado esto, se obtiene la imagen resultante, la cual se procura que sea lo más cercana posible a la imagen original (imagen procesada por el banco de filtros de análisis).

IV.4.1.3 Reconstrucción perfecta.

Para medir la confiabilidad de los bancos de filtros implementados, tanto de análisis como de síntesis, se utilizó el principio de reconstrucción perfecta, el cual se expresa de la siguiente manera:

$$I(m,n) = \tilde{I}(m,n), \qquad \text{para } 0 \le m < M, 0 \le n < N \tag{IV.5}$$

lo cual se puede interpretar como: cada una de las muestras de la imagen original I de dimensiones MxN es igual a cada una de las muestras de la imagen resultante $\tilde{I}$ de iguales dimensiones, y se dice que es una reconstrucción perfecta porque de acuerdo a la definición del error cuadrático medio:

$$E = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(I(m,n) - \tilde{I}(m,n)\right)^2 \tag{IV.6}$$

éste resulta ser cero, ya que cada una de las muestras como se mencionó anteriormente son iguales.

Cabe señalar que en este experimento no se aplica ningún esquema de codificación por lo tanto tampoco existe compresión alguna de la imagen original. En la figura IV. 17 se muestra el diagrama a bloques de los módulos utilizados para la realización de estas pruebas.

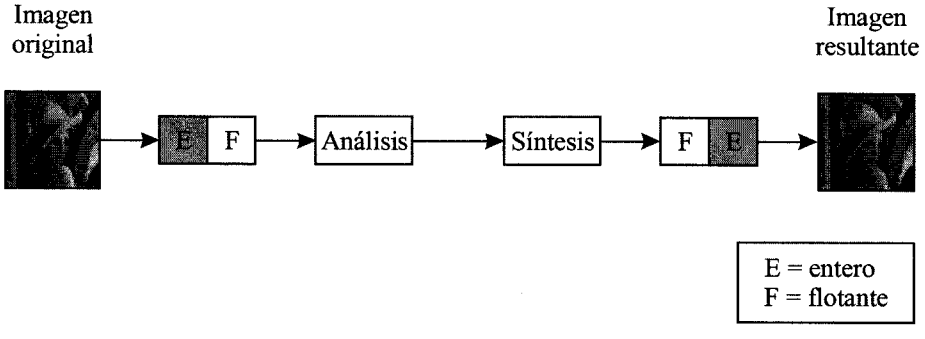


Fig. IV.17 Sistema implementado para la prueba de reconstrucción perfecta.

Como se puede observar en la figura IV.17 la imagen original se convirtió inicialmente a muestras de tipo flotante, posteriormente se procesó por los tres niveles de descomposición de los filtros de análisis, después el resultado de ese procesamiento se introdujo al banco de filtros de síntesis, una vez realizado este procesamiento se convirtieron cada una de las muestras a su valor entero, para así obtener la imagen resultante.

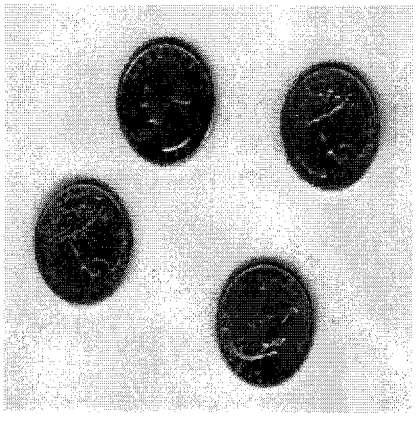
Ejemplos

En las figuras IV.18 y IV.19 se muestran dos ejemplos de pruebas realizadas mediante la implementación del sistema mostrado en la figura IV.17.

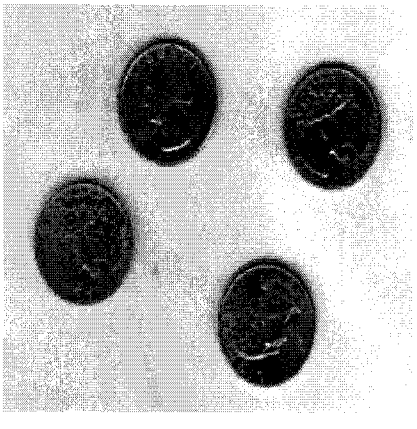
a) Imagen original

b) Imagen resultante

Fig. IV.18 Imagen del camarógrafo procesada por el sistema de reconstrucción perfecta.



a) Imagen original.



b) Imagen resultante.

Fig. IV.19 Imagen de las monedas procesada por el sistema de reconstrucción perfecta.

Como se puede mostrar en los ejemplos de las figuras IV.18 y IV.19, no se alcanza a apreciar diferencia alguna entre las imágenes resultantes y las imágenes, además se obtuvo PSNR $= \infty$ en cada uno de los dos ejemplos anteriores, por lo cual se puede establecer que la implementación tanto de los bancos de filtros de análisis como de los de síntesis es 100% confiable.

IV.4.1.4 Proceso de compresión–descompresión de imágenes digitales.

Una vez establecida la confiabilidad de los filtros biortogonales 5.5 se procedió a implementar el algoritmo de Codificación de Ondículas por Árboles de Códigos de Ceros (COAC) documentado en el Capítulo III, pero como se observó en la figura IV.17 los filtros biortogonales trabajan con datos flotantes mientras el COAC trabaja con valores enteros entre -128 y 127 (un byte) por tal razón fue necesario normalizar los resultados obtenidos a la salida del banco de filtro de análisis para que estos pudieran ser codificados en el COAC, la normalización consiste básicamente en obtener un valor de escalamiento para ajustar que los coeficientes obtenidos de los filtros caigan en el rango deseado, para obtener el factor de escalamiento se utilizó la siguiente relación:

$$f_{esc} = \frac{127}{MAX(|I(x,y)|)} \tag{IV.7}$$

donde f_{esc} es el factor de escalamiento y $MAX(|I(x,y)|)$ representa el máximo valor absoluto de los coeficientes de la imagen obtenidos a la salida del filtro de análisis. Posteriormente se normaliza mediante la siguiente ecuación:

$$I(x,y) = (\text{int}) f_{esc} I(x,y) \tag{IV.8}$$

una vez que se hayan normalizado los coeficientes obtenidos en el banco de filtro de análisis se procede a codificar en el COAC los coeficientes normalizados, y a la salida del codificador se tendrá una representación de la imagen comprimida. Para efectos de evaluar el desempeño del sistema de compresión, se desarrolló el proceso inverso (descompresión) el cual consiste primeramente en decodificar los coeficientes, desnormalizar los coeficientes a la salida del decodificador implementado utilizando el factor de escalamiento obtenido en la ecuación IV.17 y posteriormente procesando los datos ya desnormalizados por el banco de filtros de síntesis, a la salida del banco de filtros de síntesis únicamente resta pasar los coeficientes de flotantes a enteros y así se

llega a la imagen resultante, la cual se debe de aproximar lo más posible a la imagen original. En la figura IV.20 se muestra el diagrama a bloques completo del sistema implementado.

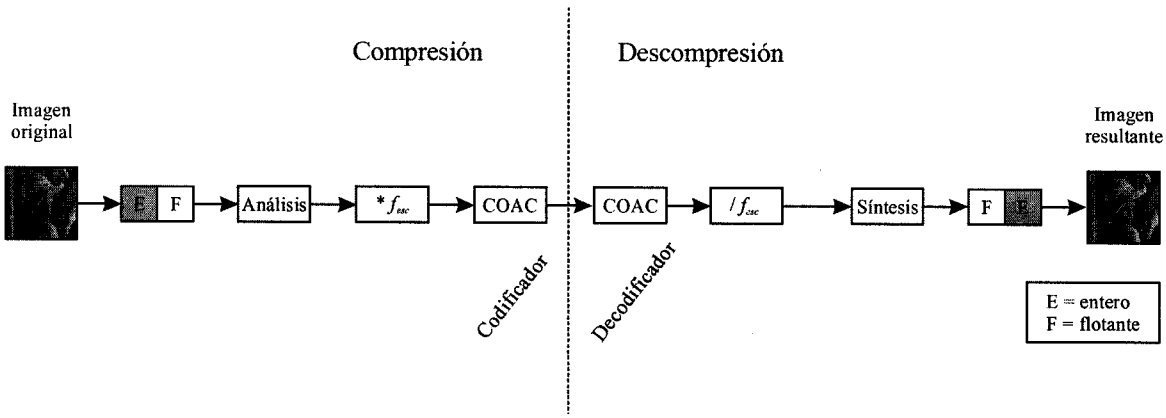


Fig. IV.20 Sistema implementado para la compresión-descompresión de imágenes.

A continuación se presentan ejemplos de imágenes procesadas en el sistema mostrado en la figura IV.20, también se muestran resultados obtenidos de este procesamiento, entre éstos resultados se pueden encontrar: la relación señal a ruido (***SNR***), la relación señal a ruido pico (***PSNR***) y el factor de compresión obtenido en el procesamiento de cada una de las imágenes. La ***SNR*** y la ***PSNR*** se definieron por las ecs. I.42 y I.43 respectivamente.

Ejemplo 1.

Se procesó la imagen de Lena de dimensiones de 256x256 utilizando dos niveles de descomposición. Las imágenes original y procesada se muestran en las figuras IV.21a y IV.21b respectivamente.

a) Imagen original

b) Imagen procesada

Fig. IV.21 Imagen de Lena procesada por compresión-descompresión utilizando dos niveles de descomposición.
(SNR = 34.8dB, PSNR = 61.1dB, Tasa de compresión = 10.96).

Ejemplo 2.

Se procesó la imagen la misma imagen que en el ejemplo anterior, solo que ahora se utilizaron cuatro niveles de descomposición. Las imágenes original y procesada se muestran en las figuras IV.22a y IV.22b respectivamente.

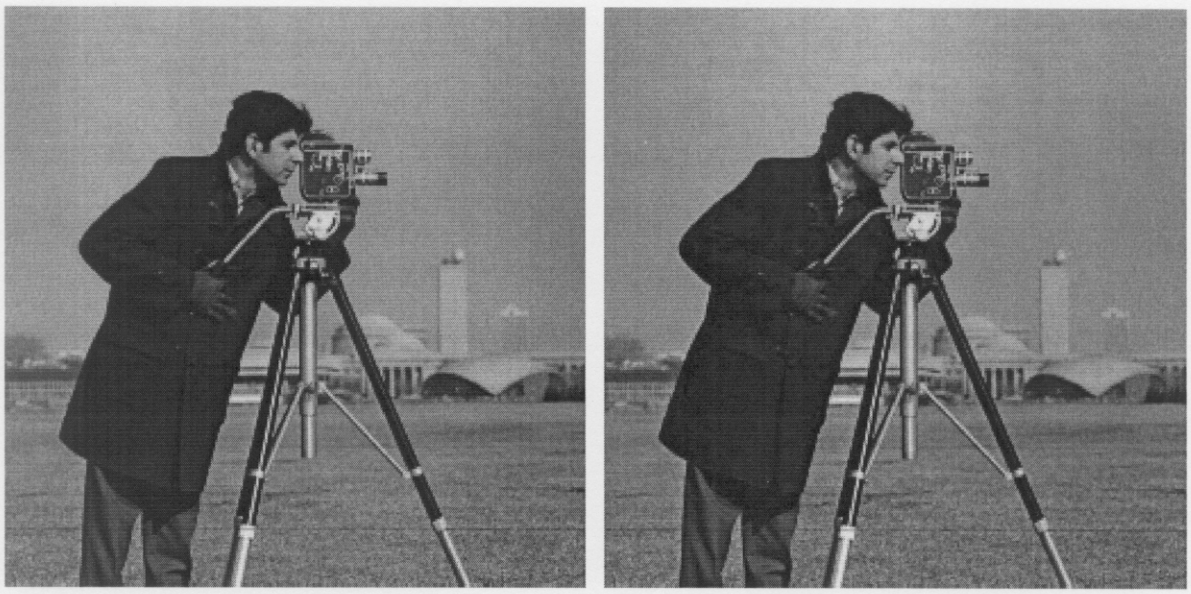
a) Imagen original

b) Imagen procesada

Fig. IV.22 Imagen de Lena procesada por compresión-descompresión utilizando cuatro niveles de descomposición.
(SNR = 20.8dB, PSNR = 47.1dB, Tasa de compresión = 18.47).

Ejemplo 3.

Se procesó la imagen del camarógrafo de dimensiones de 256x256 utilizando tres niveles de descomposición. Las imágenes original y procesada se muestran en las figuras IV.23a y IV.23b respectivamente.

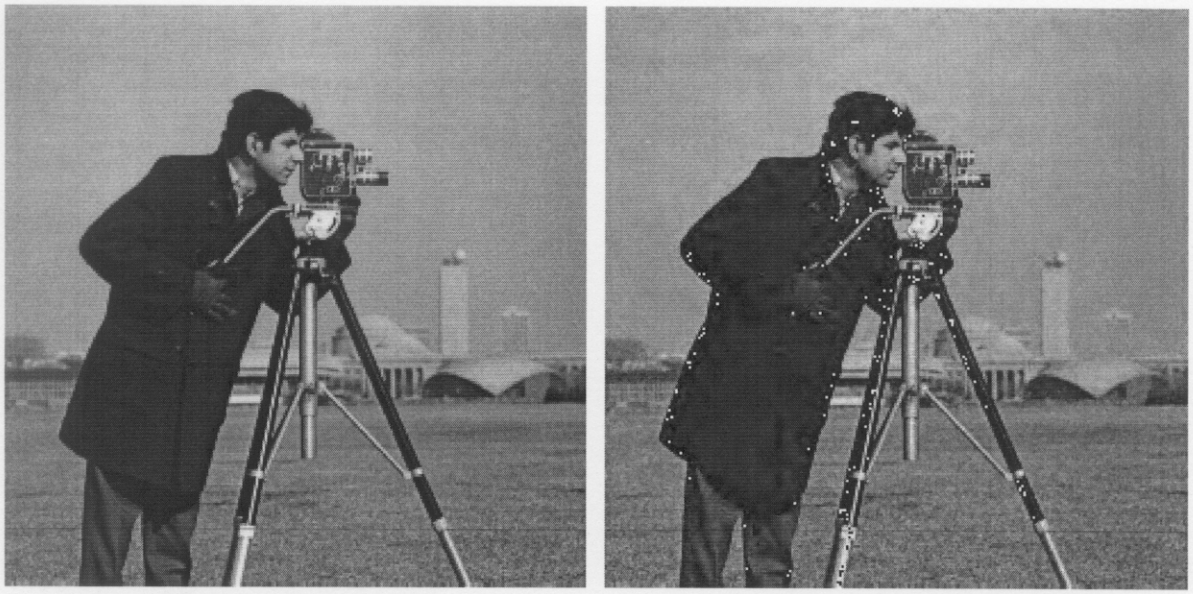


a) Imagen original b) Imagen procesada

Fig. IV.23 Imagen del camarógrafo procesada por compresión-descompresión utilizando tres niveles de descomposición.
(SNR = 31.1dB, PSNR = 59.5dB, Tasa de compresión = 15.14).

Ejemplo 4.

Se procesó la imagen del camarógrafo de dimensiones de 256x256 utilizando cinco niveles de descomposición. Las imágenes original y procesada se muestran en las figuras IV.24a y IV.24b respectivamente.



a) Imagen original b) Imagen procesada

Fig. IV.24 Imagen del camarógrafo procesada por compresión-descompresión utilizando cinco niveles de descomposición.
(SNR = 16.4dB, PSNR = 44.7dB, Tasa de compresión = 25.68).

Como se puede observar de los cuatro ejemplo presentados, conforme se aumenta el número de niveles de descomposición también aumenta la tasa de compresión pero disminuye la relación señal a ruido y la relación señal a ruido pico, para los tiempos de procesamiento ver apéndice D.

Para imágenes fijas una PSNR = 30dB se considera aceptable, para el ojo humano como se mencionó en el capítulo I, existen otras herramientas objetivas para evaluar la calidad de un esquema de compresión tal como el método de las diferencias perceptibles, donde el procesamiento se lleva a cabo en un dominio cercano a donde la visión se lleva a cabo, ver apéndice B.

CONCLUSIONES Y RECOMENDACIONES

Después de la realización del presente trabajo se llegó a las siguientes conclusiones:

- Para el procesamiento de imágenes estáticas: El recurso físico más relevante de la plataforma de procesamiento que se esté utilizando en cualquier técnica de procesamiento es la cantidad de memoria RAM con la que se cuenta, ya que es la que define tanto los límites en la dimensión de la imagen como la cantidad de código que se puede utilizar para dichas técnicas, y, generalmente, las técnicas más complejas son aquellas que requieren una mayor cantidad de código. La velocidad del procesador digital de señales no es trascendental cuando se trata de aplicaciones en las cuales no se pretende llevar a cabo procesamiento en tiempo real. La plataforma DSK6211 cuenta con una cantidad de memoria RAM limitada (ver apéndice C), la cual es insuficiente para desarrollar técnicas de procesamiento más complejas que se presentaron en este trabajo.
- Cuando se utiliza una plataforma de memoria limitada para el procesamiento de imágenes, es necesario llevar a cabo una optimización de código de los algoritmos más exhaustiva que cuando se trabaja con una PC, esta optimización con el fin de minimizar la cantidad de memoria utilizada en código y dejar así más memoria disponible para datos. Aún después de haber optimizado el código implementado, únicamente se lograron procesar imágenes de 256x256 píxeles debido a esta limitante de memoria.
- Se comprobó para la técnica de filtrado de imágenes que la convolución lineal introduce error en los bordes de la imagen [13], por tal razón se implementó la convolución circular, la cual elimina por completo este error.
- En el caso del esquema de compresión de imágenes utilizando la Transformada de Ondículas no se logró la tasa de compresión esperada, ésto debido a que en la etapa de normalización de coeficientes para ser codificados por el COAC hubo muchas pérdidas en la precisión de los coeficientes, esta pérdida en la precisión se debe a que el procesador digital de señales es de punto fijo y no de punto flotante.

- Para los límites en las dimensiones de la imagen que se establecieron (256x256, debido a la cantidad de memoria disponible) se recomienda que los bancos de filtros en los cuales se va a procesar la imagen no excedan de 3 niveles de descomposición, esto debido a que a mayores niveles no se obtiene un aumento significativo en la tasa de compresión, y sin embargo el tiempo de procesamiento aumenta considerablemente.

Con el fin de dar un seguimiento al presente trabajo se hacen las siguientes recomendaciones:

- Migrar las técnicas de procesamiento a una plataforma que cuente con mayores recursos: una mayor cantidad de memoria, procesador de más alta velocidad y de punto flotante, esto permitirá ampliar los límites de las dimensiones de las imágenes procesadas y permitirá además introducir técnicas de procesamiento más complejas como reconocimiento de patrones y procesamiento de imágenes a color.
- Comprobar de una manera práctica que los bancos de filtro biortogonales son los que dan mejores resultados en comparación a otros [13]. No se implementaron filtros adicionales en el SPDI6211 debido a las limitantes de memoria que ya se mencionaron anteriormente.
- Con mayores físicos en la plataforma destino se le podrían agregar al SPDI6211 algunas técnicas de procesamiento para imágenes en movimiento (video), las cuales, inclusive podrían trabajar en tiempo real si las características del nuevo procesador digital de señales lo permitiesen.

A COEFICIENTES DE LOS FILTROS BIORTOGONALES 5.5

En las tablas A.1 y A.2 se muestran los coeficientes de análisis y de síntesis respectivamente para los filtros biortogonales 5.5 obtenidos en Matlab.

Tabla A. 1 Coeficientes para los filtros biortogonales 5.5 en la etapa de análisis.

n	Pasa Bajas (H_0)	Pasa Altas (H_1)
0	0.0	$-1.345670945911872 \times 10^{-2}$
1	0.0	$-2.694966880111507 \times 10^{-3}$
2	$3.968708834740544 \times 10^{-2}$	$1.367065846643292 \times 10^{-1}$
3	$7.948108637240322 \times 10^{-3}$	$-9.350469740093886 \times 10^{-2}$
4	$-5.446378846823691 \times 10^{-2}$	$-4.768032657984843 \times 10^{-1}$
5	$3.456052819560334 \times 10^{-1}$	$8.995061097486484 \times 10^{-1}$
6	$7.366601814282105 \times 10^{-1}$	$-4.768032657984843 \times 10^{-1}$
7	$3.456052819560334 \times 10^{-1}$	$-9.350469740093886 \times 10^{-2}$
8	$-5.446378846823691 \times 10^{-2}$	$1.367065846643292 \times 10^{-1}$
9	$7.948108637240322 \times 10^{-3}$	$-2.694966880111507 \times 10^{-3}$
10	$3.968708834740544 \times 10^{-2}$	$-1.345670945911872 \times 10^{-2}$
11	0.0	0.0

Tabla A. 2 Coeficientes para los filtros biortogonales 5.5 en la etapa de síntesis.

n	Pasa Bajas (F_0)	Pasa Altas (F_1)
0	1.345670945911872 x10^{-2}	0.0
1	-2.694966880111507 x10^{-3}	0.0
2	-1.367065846643292 x10^{-1}	3.968708834740544 x10^{-2}
3	-9.350469740093886 x10^{-2}	-7.948108637240322 x10^{-3}
4	4.768032657984843 x10^{-1}	-5.446378846823691 x10^{-2}
5	8.995061097486484 x10^{-1}	-3.456052819560334 x10^{-1}
6	4.768032657984843 x10^{-1}	7.366601814282105 x10^{-1}
7	-9.350469740093886 x10^{-2}	-3.456052819560334 x10^{-1}
8	-1.367065846643292 x10^{-1}	-5.446378846823691 x10^{-2}
9	-2.694966880111507 x10^{-3}	-7.948108637240322 x10^{-3}
10	1.345670945911872 x10^{-2}	3.968708834740544 x10^{-2}
11	0.0	0.0

Obsérvese que de acuerdo a [5]:

$$F_1(z) = H_0(-z) \text{ ó } f_1(n) = (-1)^n h_0(n) \tag{A.1}$$

$$F_0(z) = -H_1(-z) \text{ ó } f_1(n) = -(-1)^n h_1(n) \tag{A.2}$$

Es posible comprobar que las ecuaciones A.1 y A.2 satisfacen las condiciones de reconstrucción perfecta dadas por:

$$F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0 \tag{A.3}$$

$$F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-D} \tag{A.4}$$

A.1 y A.2 en A.3,

$$-H_1(-z)H_0(-z) + H_0(-z)H_1(-z) = 0,$$

la ecuación A.4 se comprueba como sigue:

$$conv(F_0, H_0) + conv(F_1, H_1) = 2z^{-11}$$

B EL MÉTODO DE LAS DIFERENCIAS PERCEPTIBLES ASOCIADO AL SISTEMA VISUAL HUMANO

El sistema visual humano puede ser representado por dos subsistemas conectados en cascada mostrados en la figura B.1. El primer subsistema, el cual representa la parte periférica del sistema visual, convierte la luz en una señal neuronal, el segundo subsistema, el cual representa la parte central del sistema visual, procesa la señal neuronal y extrae la información.

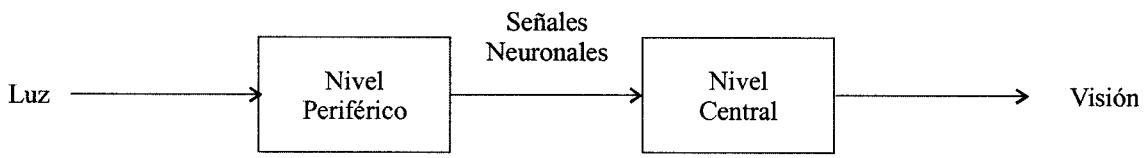


Fig. B. 1 Sistema visual humano descompuesto en dos subsistemas.

A diferencia de la parte central de procesamiento, de la cual se conoce muy poco, la parte periférica es relativamente conocida por la comunidad científica, y se han llevado a cabo muchos intentos por modelarla. Un modelo simple [21] para una imagen monocromática se muestra en la figura B.2. En este modelo, la intensidad de la imagen monocromática *I(x,y)* es modificada por una no-linealidad, tal como una operación logarítmica, eso comprime los niveles altos de intensidad, pero expande los niveles bajos. Posteriormente la salida es filtrada por un filtro pasa banda con una respuesta en frecuencia $H(\Omega_x, \Omega_y)$. La no-linealidad fue adoptada debido a resultados obtenidos por experimentos fisiológicos realizados al ojo humano. El filtro pasa banda fue adoptado debido al tamaño

finito de la pupila, al límite de la resolución impuesto por un número finito de células receptoras de luz y al proceso de inhibición lateral. El tamaño finito de la pupila y el límite de la resolución son responsables de la parte pasa bajas del filtro pasa banda $H(\Omega_x,\Omega_y)$. El proceso de inhibición lateral proviene del hecho de que se ha observado que una fibra neuronal responde a varias células receptoras. La respuesta de una fibra neuronal es la combinación de varias señales provenientes de varias células receptoras de luz. Mientras que unas células contribuyen positivamente, otras lo hacen negativamente (inhiben). Este proceso lateral de inhibición representa la parte pasa altas del filtro pasa banda $H(\Omega_x,\Omega_y)$. Aunque el modelo mostrado en la figura B.2 sea simple y se aplique únicamente a la parte periférica del sistema visual humano, éste es consistente con algunos de los fenómenos visuales que se presentarán posteriormente en este apéndice.

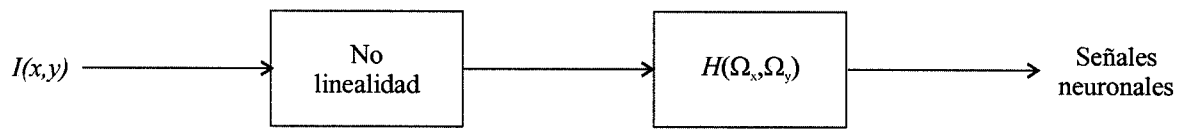


Fig. B. 2 Modelo simple del nivel periférico del sistema visual humano.

Una manera de explotar un modelo como el que se muestra en la figura B.2 es el de procesar una imagen en un dominio más cercano a donde la visión se lleva a cabo. Esto puede ser útil en algunas aplicaciones. En codificación de imágenes, por ejemplo, la información que está en la imagen pero es despreciada por el sistema visual no necesita ser codificada. Al procesar una imagen en un dominio más cercano a donde la visión se lleva a cabo se puede poner más énfasis en lo que es más importante para el sistema visual. Esta es una razón por la cual algunas operaciones de procesamiento de imágenes son desarrolladas en el dominio de intensidad logarítmica en vez del dominio de intensidad lineal.

Además del modelo de la figura B.2, Hall en [8] propuso un modelo no lineal para el sistema visual humano mientras que Granrath en [7] analiza la importancia de estos modelos en el procesamiento de imágenes digitales.

Una manera de cuantificar la habilidad de descifrar dos estímulos visuales que son iguales a excepción de sus intensidades o luminancias, es el medir únicamente las

diferencias apenas perceptibles (jnd, del inglés: just noticeable differences). La diferencia jnd puede ser definida y medida en una variedad de maneras, una manera es a través de un experimento llamado discriminación de intensidad. Supóngase que se presenta a un observador el estímulo mostrado en la figura B.3, la región interna es una imagen monocromática de intensidad uniforme I_{ent}, la cual es aleatoriamente elegida para ser I ó $I + \Delta I$. La región externa es una imagen monocromática de intensidad I_{sal}, la cual deberá ser I cuando I_{ent} sea $I + \Delta I$ ó deberá ser $I + \Delta I$ cuando I_{ent} sea I. Se le pide al observador que elija cuál de las dos intensidades I_{ent} o I_{sal} es más alta. Cuando ΔI es muy grande, el observador dará la respuesta correcta la mayoría de las veces, correcta refiriéndose a que la región con mayor intensidad será elegida. Cuando ΔI es muy pequeña, el observador dará la respuesta correcta aproximadamente la mitad de las veces, a medida de que se vaya reduciendo el valor de ΔI, el porcentaje de las respuestas correctas del observador decrecerá también, y entonces se puede definir una ΔI en que el observador da una respuesta correcta el 75% de las veces como la diferencia jnd de la intensidad I [12].

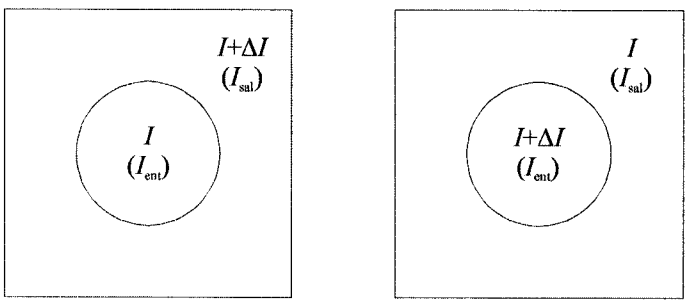


Fig. B. 3 Dos estímulos utilizados en el experimento de discriminación de intensidad.

La figura B.4 muestra la gráfica $\Delta I / I$ como una función de I donde ΔI es la diferencia jnd. De ésta figura, para un amplio rango de I,

$$\frac{\Delta I}{I} \approx \text{constante.} \tag{B.1}$$

Esta relación es llamada Ley de Weber. La Ley de Weber establece que la diferencia jnd ΔI es proporcional a I. A medida que se incrementa I, se requiere de una ΔI mayor para hacer posible el reconocimiento de $I + \Delta I$ con respecto a I. Esta es una manera en la que

el sistema visual permanece sensible a un amplio rango de intensidades. La Ley de Weber se aplica no sólo en la visión, sino también en los demás sentidos humanos como el oído, el gusto, el olfato y el tacto.

A medida de ΔI tiende a 0, la ecuación B.1 puede ser escrita como:

$$\frac{dI}{I} = d(\log I) \approx \text{constante.} \tag{B.2}$$

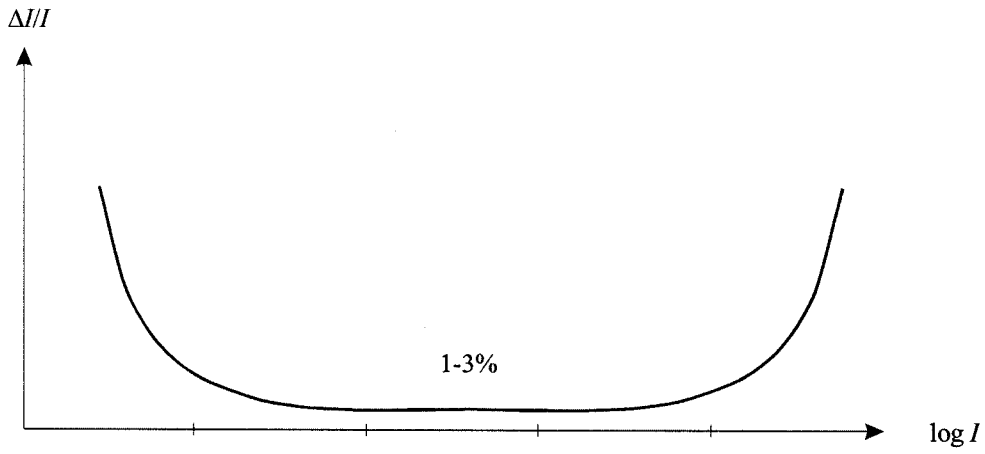


Fig. B. 4 Gráfica de $\Delta I / I$ en función a I.

De la ecuación B.2, la diferencia jnd es constante en el dominio *log I* para un rango amplio de I. Esto es consistente con el hecho de que una operación no lineal tal como el logaritmo es aplicada a la intensidad de la imagen en el modelo simple de la figura B.2. El experimento de discriminación de intensidad involucra una simple tarea por parte del observador, y no es necesario un procesamiento central complicado. Consecuentemente, el resultado del experimento de discriminación de intensidad puede ser relacionado al nivel de procesamiento periférico en el sistema visual.

El resultado del experimento de la discriminación de intensidad establece que la diferencia jnd ΔI aumenta a medida de que aumenta I, esto explica en parte porqué un nivel uniforme de ruido aleatorio es más visible en una región oscura que en una región clara, esto se muestra en la figura B.5. La imagen de la figura B.5 es el resultado de agregar ruido blanco Gaussiano con media cero y una densidad de probabilidad uniforme a una imagen original sin degradación. La apariencia arenosa, la cual se debe al ruido, es más pronunciada en la región del fondo oscuro que en la región del fondo claro. Como la diferencia jnd ΔI es mayor para una I mayor, un nivel más alto de ruido en la región más

brillante es requerido para que sea igualmente visible que en la región más oscura. Como resultado de este experimento, se puede concluir en que es más importante reducir ruido en la regiones oscuras que en la regiones claras.

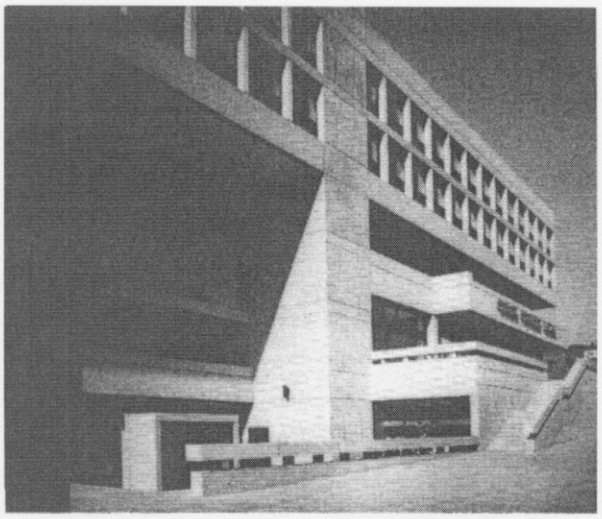


Fig. B. 5 Imagen de 512x512 píxeles degradada con ruido blanco Gausiano con media cero.

Adaptación.

En el experimento de la discriminación de intensidad discutido anteriormente, las intensidades mostradas son I y $I + \Delta I$. Si se supone que el observador tomará más tiempo para tomar una decisión, el resultado es obtenido cuando éste se adapta al nivel de intensidad I. Cuando el nivel de intensidad al cual se adaptó el observador es diferente de I, la habilidad de resolución de intensidad del observador se ve disminuida. Supóngase que se desarrolla el experimento mencionado anteriormente, pero en este caso con una I y una $I + \Delta I$ rodeadas por una región mucho mayor que cuenta con una intensidad I_0, como se muestra en la figura B.6.

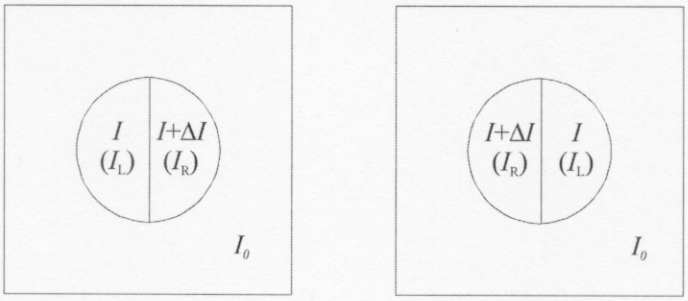


Fig. B. 6 Los dos estímulos utilizados en el estudio de la sensibilidad a la intensidad

El resultado de trazar $\Delta I/I$ como una función de I e I_0 es mostrado en la figura B.7. Cuando I_0 es igual a I, el resultado es el mismo que se ve en la figura B.4. Cuando I_0 es diferente a I, sin embargo, la diferencia jnd ΔI aumenta en relación al caso $I_0=I$, indicando que la sensibilidad del observador disminuye también. El resultado muestra que la sensibilidad a la intensidad es mayor cuando ésta se acerca al nivel de sensibilidad al cual se adaptó el observador. Esta es otra manera en la cual el sistema visual responde a un amplio rango de intensidades en ocasiones diferentes.

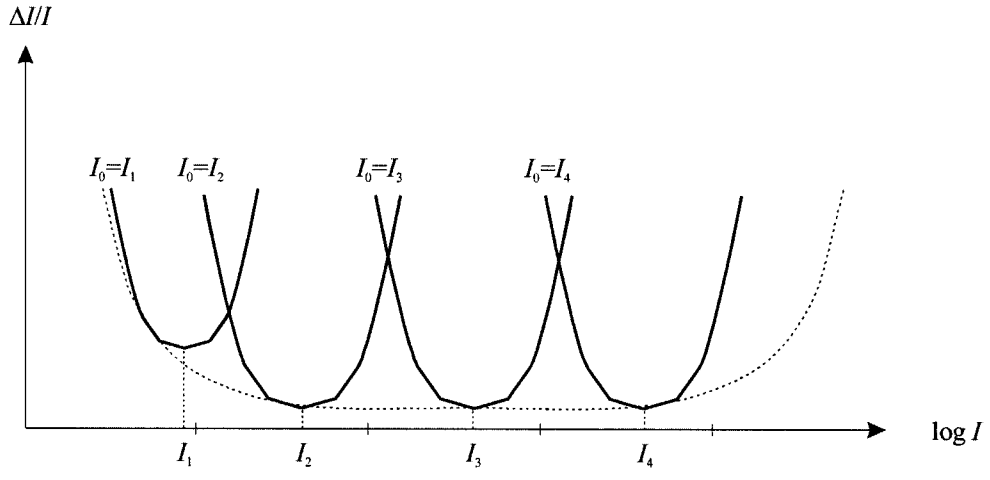


Fig. B. 7 Gráfica de $\Delta I / I$ como una función de I e I_0.

C EL PDS TMS320C6211 Y SUS ESPECIFICACIONES

El sistema de procesamiento sobre el cual funciona el SPDI consiste en una tarjeta basada en el procesador digital de señales (PDS) de punto fijo TMS320C6211 de la compañía *Texas Instruments*, el cual ejecuta 1,200 millones de instrucciones por segundo (MIPS). A este sistema se le conoce como DSK '6211 donde DSK es un acrónimo en inglés de *Digital signal processor Starter's Kit*, que en su traducción al español significa: Paquete de Introducción al Procesamiento Digital de Señales. Este DSK es una plataforma de bajo costo que se comunica a la PC mediante el puerto paralelo, la cual permite desarrollar y probar aplicaciones para el 'C6211.

El procesador digital de señales opera a una frecuencia de reloj de 150 MHz. Posee adicionalmente un segundo reloj que controla la interfaz de memoria externa (*EMIF, External Memory Interface*), el cual trabaja a una frecuencia de 100 MHz. Entre las características más importantes del DSK 'C6211 se pueden mencionar las siguientes [22]:

- Interfaz para la conexión a la computadora (procesador Anfitrión) por el puerto paralelo.
- 4 MBytes de memoria RAM dinámica síncrona (SDRAM) que opera a 100 MHz.
- 128 KBytes de memoria ROM flash programable y borrable.
- Puerto de entrada-salida de 8 bits mapeado en memoria.
- Interfaz de puerto anfitrión (*HPI*) que permite el acceso a toda la memoria del procesador a través del puerto paralelo.
- Convertidor analógico-digital y digital-analógico (CODEC) de 16 bits para aplicaciones de audio, con tasa de muestreo fija de 8 KHz
- Seis indicadores LED (3 de usuario y 3 de propósito específico).

- Utilización externa con fuente de poder y el cable paralelo.

En la figura C.1 se muestra la foto del DSK '6211.

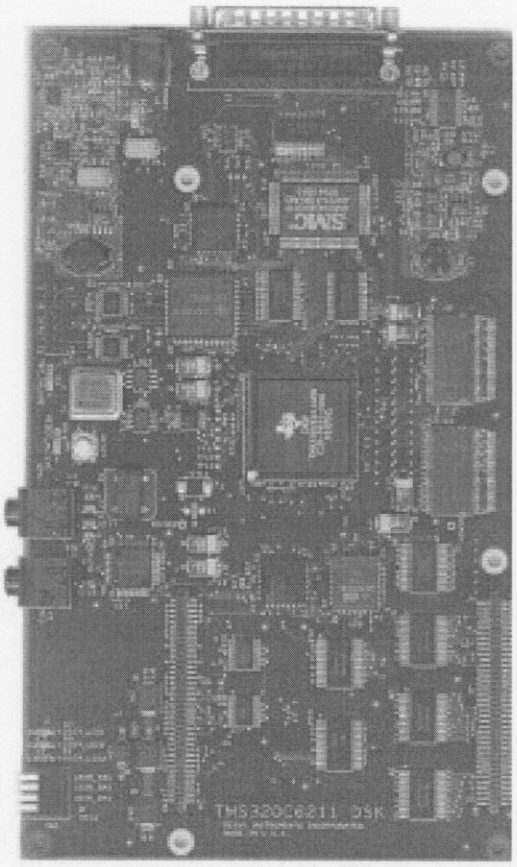


Fig. C. 1 Fotografía del DSK 'C6211.

D TIEMPOS DE PROCESAMIENTO ESTIMADOS EN LOS EXPERIMENTOS REALIZADOS

Durante la elaboración de este trabajo se realizaron diversos experimentos para estimar los tiempos de procesamiento de cada uno de los algoritmos implementados en el SPDI6211, cabe señalar que la resolución de la estimación es de 1 mseg, esto debido a que la arquitectura del procesador no permite una mayor resolución en el tiempo.

En la figura D.1 se muestra la gráfica de tiempos de procesamiento para el proceso de binarización de la imagen de Lena a diferentes dimensiones y con un umbral de 128.

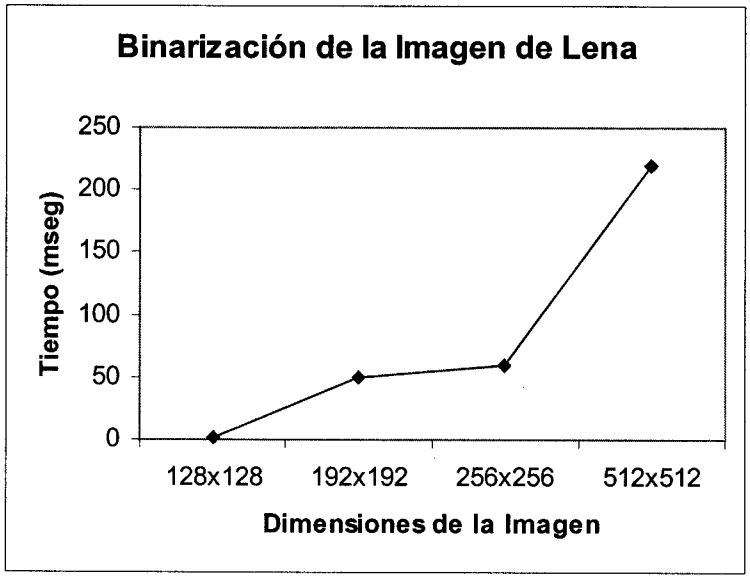


Fig. D. 1 Tiempos de procesamiento en el proceso de binarización.

En el caso del proceso de binarización se decidió utilizar únicamente una imagen (Lena) ya que los tiempos de procesamiento en la binarización no dependen de la información que contenga la imagen sino de las dimensiones de ella.

En la figura D.2 se muestra la gráfica de los tiempos de procesamiento del algoritmo de detección de bordes para dos imágenes con dimensiones diferentes cada una de ellas.

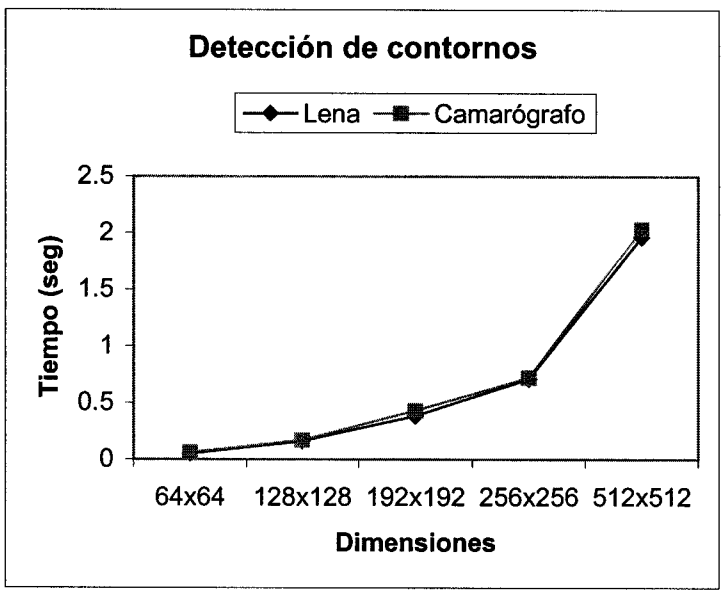


Fig. D. 2 Tiempos de procesamiento en el proceso de detección de bordes.

Como se puede observar en la gráfica anterior los tiempos de procesamiento varían ligeramente entre las imágenes de Lena y del camarógrafo; en algunos puntos la del camarógrafo tomó un poco más de tiempo que la imagen de Lena, ésto debido a que la imagen del camarógrafo cuenta con una mayor cantidad de bordes que la imagen de Lena.

En la figura D.3 se muestran los tiempos de procesamiento para dos imágenes procesadas por el filtro promediador 3x3.

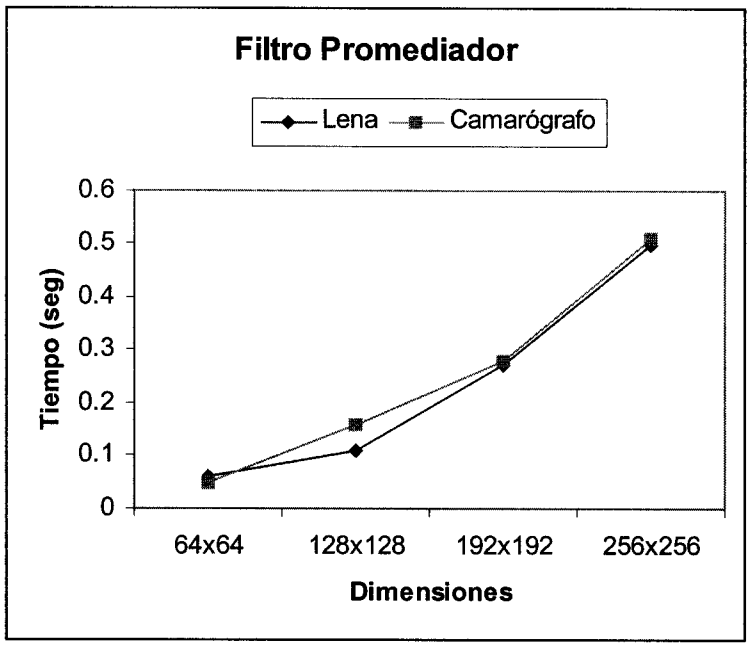


Fig. D. 3 Tiempos de procesamiento en el proceso de filtro promediador.

En la figura D.4 se muestra la gráfica de los tiempos de procesamiento de dos imágenes obtenidos en el proceso de análisis de filtros biortogonales 5.5, éstos experimentos se realizaron para diferentes niveles de codificación.

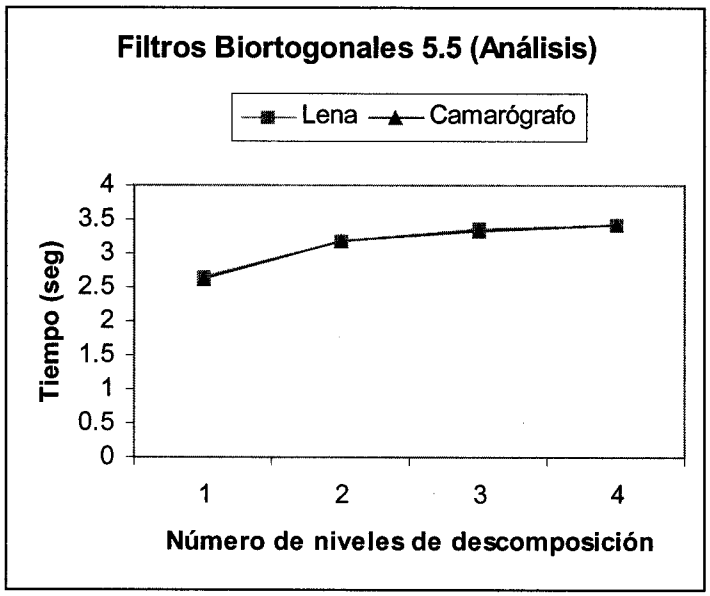


Fig. D. 4 Tiempos de procesamiento en el proceso de filtro biortogonal 5.5 (Análisis).

En la figura anterior se graficaron los resultados obtenidos después de procesar la imagen de Lena y del camarógrafo con una dimensión de 256x256 píxeles cada una de ellas.

En la figura D.5 se muestran los tiempos de procesamiento estimados en el proceso de compresión-descompresión de imágenes digitales, este proceso se describe a detalle en la sección IV.4.2.

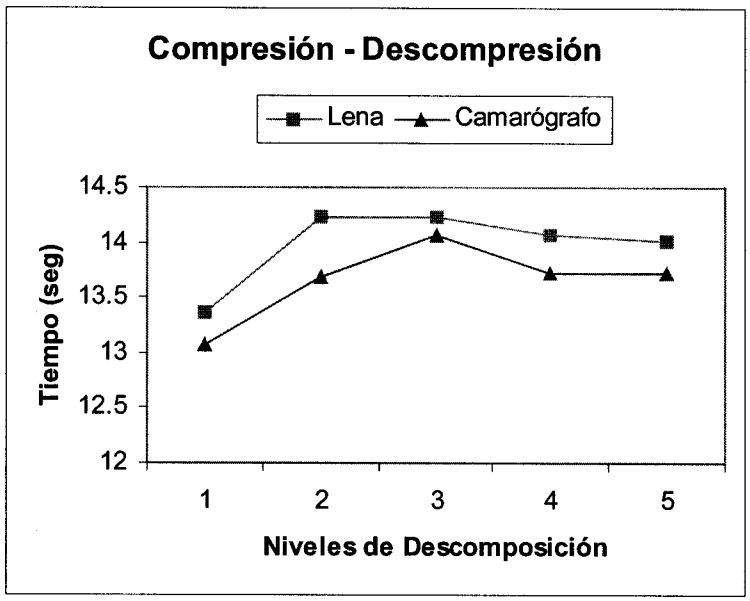


Fig. D. 5 Tiempos de procesamiento en el proceso de compresión – descompresión de imágenes.

Como se puede observar, el tiempo de procesamiento del nivel 1 al nivel 2 aumenta considerablemente, mientras que curiosamente disminuye conforme se va avanzando a los niveles posteriores, ésto se debe a que entre más niveles de descomposición se utilicen, el número de ceros generado por el banco de filtros biortogonales 5.5 es mayor, y el hecho de que este número de ceros aumente favorece a la simplicidad de los cálculos en el algoritmo de Codificación de Ondículas por Árboles de Ceros (COAC).

E CONTENIDO DEL DISCO COMPACTO INCLUIDO

En el presente trabajo se incluye un disco compacto con el siguiente contenido:

- Simulaciones. Se incluye el código fuente de las simulaciones hechas en MatLab y presentadas en el Capítulo I. Esto se puede localizar dentro del directorio '*Simulaciones*' del disco compacto.
- Imágenes utilizadas. Se incluyen además las imágenes utilizadas tanto en las simulaciones como en los ejemplos mostrados de la implementación del SPDI6211, los archivos de imagen se encuentran en formato de mapa de bits (BMP) a 256 tonalidades de gris. Estos archivos se encuentran dentro del directorio '*Imagenes*'.
- Programa Instalador del SPDI6211. Este programa se encarga de instalar en forma automatizada el SPDI6211 en una computadora personal con sistema operativo Windows, además copia también los archivos de imágenes mencionados en el punto anterior. El programa de instalación del SPDI6211 se encuentra en el directorio '*SPDI6211\Instalador*' dentro del disco compacto.
- Código Fuente del SPDI6211. Se incluye además lo que quizás sea la aportación más importante en este trabajo, que es el código fuente del SPDI6211, el cual se divide en dos partes: el programa anfitrión desarrollado en *C++ Builder 4* que se incluye en el directorio '*SPDI6211\Fuente\Host*' y el programa destino que fue desarrollado en *Code Composer Studio v1.23* y cuyo código se encuentra en el directorio '*SPDI6211\Fuente\Target*'.

- Documento Electrónico. Se incluye en el directorio raíz del disco compacto el presente documento en formato PDF llamado '*tesis.pdf*'.

REFERENCIAS Y BIBLIOGRAFÍA.

[1]. *Algazi, V.R. / Estes, R.R.* **Analysis based coding of Image Transform and Subband Coefficients**. Proceedings of the International Society for Optical Engineering Conference on Signal Processing. Vol. 2564, pp. 11-21. 1995.

[2]. *Burrus, C.Sydney. / Gopinath, Ramesh A. / Guo, Haitao.* **Introduction to Wavelets and Wavelet Transforms: A Primer**. Prentice Hall. 1997. ISBN 0-13-489600-9.

[3]. *Calderbank, A.R. / Daubechies, I. / Sweldens, W. / Yeo, B.L.* **Wavelet Transforms that Map Integers to Integers**. Applied and Computational Harmonic Analysis. Vol. 5, No. 2, pp.332-369. 1998

[4]. *Daubechies, I.* **Ten Lectures on Wavelets**. Second Edition. SIAM. 1992. ISBN 0-89-871274-2.

[5]. *García López, Juan.* **Procesamiento Digital de Señales**. Libro en proceso de impresión IPN, 1999.

[6]. *González, Rafael C. / Woods, Richard E.* **Digital Image Processing**. Second Edition. Addison-Wesley. 2001. ISBN 0-20-118075-8.

[7]. *Granrath, D.J.* **The role of human visual models in image processing**. Proceedings of the IEEE Vol. 69, No. 5, pp. 552-561. 1981.

[8]. *Hall, C.F. / Hall, E.L.* **A nonlinear model for the special characteristics of the human visual system**. IEEE Transactions of Man and Cybernetics. Vol. 7, No. 3 1977.

[9]. *Herrera Rosales, Ramsés.* **Técnicas para el Procesamiento Digital de Imágenes.** Tesis de Maestría. Instituto Tecnológico de la Laguna. 1998. ISSN 0188-9060.

[10]. *Huffman, D. A.* **A Method for the Construction of Minimum Redundancy Code**. Proceedings of the Institute of Radio Engineers. Vol.40, No.9, pp. 1098-1101. 1952.

[11]. *Leithold, Louis.* **El Cálculo con Geometría Analítica**. Sexta Edición. HARLA. 1990. ISBN 970-613-040-3.

[12]. *Lim, Jae S.* **Two Dimensional Signal and Image Processing.** Prentice Hall. 1990. ISBN 0-13-935322-4.

[13]. *Lugo Pérez, Eduardo.* **Diseño de Bancos de Filtros para la compresión de Imágenes Digitales**. Tesis de Maestría. Centro de Investigación y Desarrollo de Tecnología Digital. Instituto Politécnico Nacional. 1999.

[14]. *Mallat, Stéphane.* **A Theory for Multiresolution Signal Decomposition: The Wavelet Representation**. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.11, No.7, pp. 674-693. 1989.

[15]. *Mallat, Stéphane.* **A Wavelet Tour of Signal Processing**. Second Edition. Academic Press. 1999. ISBN 0-12-466606-X.

[16]. *Oppenheim, Alan V. / Schafer Ronald W.* **Digital Signal Processing**. Prentice Hall. 1975. ISBN 0-13-214635-5.

[17]. *Poularikas, Alexander.* **The Transforms and applications handbook**. CRC Press. 1995. ISBN 0-84-938342-0.

[18]. *Ramesh, Jain.* **Machine Vision**. Mc Graw Hill. 1994. ISBN 0-07-032018-7.

[19]. *Rosenfeld, Azriel / Kak, Avinash C.* **Digital Picture Processing**. Second Edition. Academic Press. 1982. ISBN 0-12-597301-2.

[20]. *Shapiro, Jerome M.* **Embedded Image Coding Using Zerotrees of Wavelet Coefficients**. IEEE Transactions on Signal Processing, Vol. 14, No. 12, 3445-3462. 1993.

[21]. *Stockman, T.G. Jr.* **Image Processing in the context of a visual model**. Proceedings of the IEEE Vol. 60, No. 7, pp. 828-842. 1972.

[22]. *Texas Instruments.* **TMS320C6000 Peripherals. Reference Guide**. Digital Signal Processing Solutions. 1999.