UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

PROCESAMIENTO DIGITAL DE IMÁGENES USANDO WAVELETS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

P R E S E N T A

ORLANDO CORTAZAR MARTÍNEZ

ASESOR: DR. LUIS ENRIQUE RAMOS VELASCO

PACHUCA DE SOTO, HIDALGO. JUNIO DE 2006

Este trabajo fué parcialmente apoyado por PROMEP, con el número de proyecto 103.5/05/2038 y parcialmente apoyado por la UAEH con el proyecto anual de investigación 2006.

Agradecimientos

A Dios

A mis padres, mi hermana y familiares.

A mi asesor de tesis, el Dr. Luis Enrique Ramos Velasco

A toda mi familia

Familia Cortazar Zavala

Familia Martínez Jiménez

A mis maestros

A mis amigos

Agradecimientos

Primero quiero agradecer a Dios por que el es el único que guía mi camino.

De manera muy especial quiero agradecer a mis padres por el amor que me demuestran día a día. A mi papá por ser la persona a quien mas admiro. A mi mamá por su pasciencia y su apoyo incondicional.

Quiero agradecer a mi hermana por estar siempre conmigo y por apoyarme en todo lo que puede.

A mi familia por la confianza que me han demostrado y por su ayuda en todo momento. Gracias.

Gracias a mi asesor el Dr. Luis Enrique Ramos Velasco por su apoyo, por su tiempo y especialmente por su esfuerzo que puso para que este trabajo pudiese realizarse.

Quiero agradecer a los maestros que tuve durante la licenciatura, no solamente por haberme enseñado conocimientos, sino por que han sido un ejemplo par mi.

Por que me han enseñado muchas cosas, por la confianza que me tienen, por su apoyo, por compartir conmigo alegrías y tristezas, a todos mis amigos les doy las gracias.

Notación empleada en esta tesis

Notación empleada	**Significado**
$\mathbb{R}$	Números reales
$\mathbb{Z}$	Zahlen, números enteros
$\forall$	Para todos los valores de
$\neq$	No es igual a
$\equiv$	Es congruente con
$\in$	Perteneciente a
$\approx$	Se aproxima a
$< , >$	Producto interno
$x(n)$	Función discreta de variable discreta n
$\delta(n)$	Impulso unitario
$u(n)$	Escalón unidad
$u_r(n)$	Rampa unidad
$f(x)$	Función continua de variable real x
$\mathfrak{F}$	Transformada de Fourier
$L^2(\mathbf{R})$	Conjunto de funciones continuas de cuadrado integrable
ψ	Wavelet madre
φ	Función de escala
$\Psi(\omega)$	Transformada de Fourier de $\psi(t)$

Siglas empleadas en esta tesis

Notación empleada	**Significado**
MSI	Escala de integración mediana
LSI	Escala de integración grande
$VLSI$	Escala de integración muy grande
SCR	Rectificador controlado por silicio
$IGBT$	Transistor bipolar de puerta aislada
$MOSFET$	Transistor de efecto de campo de metal óxido semiconductor
DSP	Procesador digital de señales
OLS	Último cuadrado ortogonal
ERR	Taza de reducción del error
$ANOVA$	Análisis de la expansion de la varianza
DWT	Transformada wavelet discreta
CWT	Transformada wavelet continua
CNN	Red de redes neuronales
SAR	Índice de absorción específica
PCA	Proyección de análisis de componentes
MA	Memoria asociativa
OCR	Reconocimiento de caracteres ópticos
AWC	Control wavelet adaptable
ADC	Convertidor análogo digital
DAC	Convertidor digital análogo
DFS	Series de Fourier discretas
$IEEE$	Instituto de ingeniería eléctrica y electrónica
DFT	Transformada discreta de Fourier
FFT	Transformada rápida de Fourier
$STFT$	Transformada de tiempo corto de Fourier
CWT	Transformada wavelet continua
DWT	Transformada wavelet discreta
ME	Equipo móvil
MIT	Instituto tecnológico de Masachuttes
FBI	Agencia federal de investigación

WSQ	Cuantización wavelet escalar
$JPEG$	Grupo unido de expertos fotográficos
IDK	Equipo de desarrollo de imágenes
LPd	Filtro de descomposición pasa bajas
HPd	Filtro de descomposición pasa altas
LPr	Filtro de reconstrucción pasa bajas
HPr	Filtro de reconstrucción pasa altas
ALU	Unidad lógico aritmética
$VLIW$	Palabra de instrucción muy larga
$MIPS$	Millones de instrucciones por segundo
$FLOPS$	Operaciones en punto flotante por segundo
RAM	Memoria de acceso aleatorio
LSB	Bit menos significativo
MSB	Bit más significativo
NaN	No es un número
NMI	Interrupción mascarable
ROM	Memoria de solo lectura
AIC	Chips de interfase analógica
$CMOS$	Computadora de metal óxido semiconductor

Índice general

Índice de Tablas

Índice de figuras

Resumen

Cada vez tiene mayor importancia en la ingeniería el procesado de señales digitales, puesto que es una ciencia que se entrelaza con una gran cantidad de ciencias y puesto que posee grandes características como lo son su alta flexibilidad de diseño, su capacidad de programación y de cambiar esta para obtener mejores resultados en un procesado real, además de que se cuenta con dispositivos digitales de alta velocidad y con un costo mucho mas barato que dispositivos electrónicos análogos o incluso mecánicos, esta ciencia se ha convertido en un pilar fundamental de otras ciencias en cuanto se refiere al tratamiento y representación de las señales.

El procesamiento de imágenes ha tenido una gran evolución. Durante los últimos años ha aumentado significativamente el interés de la morfología de las imágenes, la compresión de estas y el reconocimiento de patrones. Esto representa un gran esfuerzo de modernización. Su interés se deriva en dos áreas principales: La mejora de información pictórica para la interpretación humana y el procesamiento de los datos de la escena para la percepción humana. En este trabajo se usa la segunda area de aplicación más que la primera.

Por lo que este trabajo se enfoca al proceso y representación de imágenes mediante la técnica denominada wavelets, así como también su implementación en un procesador digital de señales denominado por sus siglas en inglés "DSP". Esta técnica se usó para obtener la detección de bordes de imágenes de un sistema péndulo invertido previamente controlado. Aprovechando una de sus características que ofrece esta técnica, como lo es el análisis multiresolución para imágenes en la cual, los componentes de interés localizados a menudo no son enteramente oscilatorios y solo incluyen un ciclo o una parte de un ciclo, como líneas, bordes o puntos.

La técnica de wavelets o también conocida como técnica de ondoletas, en la que la descomposición wavelet de una imagen, se realiza fila por fila y después columna por columna. Con esto se detectan los valores cambiantes dentro de la matriz de la imagen y con ello la detección de bordes mediante la ubicación de valores que cambian bruscamente dentro de una escala de grises aplicada a la matriz de ésta. Con ello se logra un reconocimiento de patrones mediante la detección de bordes. El resultado obtenido se muestra en pantalla mediante sus coeficientes vertical, horizontal y diagonal.

En este trabajo se hace la adquisición de imágenes y su procesamiento mediante procesador digital de señales (DSP) de la serie TMS320C6711 junto con una tarjeta de procesamiento digital de imágenes IDK (Imaging Development Kit). Las imágenes tomadas son de un sistema péndulo invertido que se encuentra bajo control. Todo esto se hace con la finalidad de obtener datos que ayuden en la implementación de controles que hacen uso de una cámara de visión como sensor (este tema esta fuera del alcance de este trabajo).

Capítulo 1

INTRODUCCIÓN

En este capitulo se definen los objetivos a perseguir en este trabajo, así como las metas a alcanzar y sus posibles consecuencias. Además, se da una introducción a lo que es el procesamiento digital de señales. Por otra parte, se da una muy breve semblanza de la historia del procesamiento digital de señales tanto como señales básicas como complejas que para este trabajo se usan las imágenes.

Con este proyecto se procesan las imágenes haciendo uso de la técnica wavelet así como la manipulación de estos en sistemas discretos para su utilización a través de los diferentes tipos de filtros localizados dentro de las tarjetas DSP's con el fin de obtener características de sus imágenes.

1.1. Una breve reseña acerca del procesamiento digital de señales

El procesamiento digital de señales es un área de la ciencia y de la ingeniería que a lo largo de 30 años se ha desarrollado rápidamente [28]. Esto como resultado de los grandes avances tecnológicos tanto en las computadoras digitales como en la fabricación de los circuitos integrados. Durante este periodo el hardware y el software de estas computadoras eran grandes y caros, y como consecuencia tenía aplicaciones limitadas, tanto científicas como comerciales. El gran crecimiento de la tecnología de circuitos integrados, empezando con la integración a media escala (MSI, Médium-Scale Integration) y continuando la integración a gran escala (LSI, Large-Scale Integration), y ahora, la integración a muy grande escala (VLSI, Very Large-Scale integration) de circuitos integrados han tenido como consecuencia el desarrollo de computadoras digitales mas potentes, veloces y económicos para propósito general en tiempo real.

Este tipo de circuitos han hecho posible que sistemas digitales altamente sofisticados, capaces de realizar tareas de procesamiento digital de señales que normalmente serían más difíciles con circuitería o con procesamiento análogo de señales. Por esto que el procesamiento digital que se realizaba comúnmente con el procesado análogo de señales

se realice ahora con hardware digital, más barato, y más confiable.

No por esto se quiere dar a entender que el procesamiento digital de señales sea la única opción para todos los problemas de procesamiento digital de señales. De esta manera, para señales que poseen un gran ancho de banda requiere un procesado en tiempo real. Por lo que un procesado de tipo análogo seria conveniente. Pero, sin embargo, cuando los componentes electrónicos de gran velocidad se encuentran disponibles, normalmente son preferibles.

Los dispositivos digitales no solo dan lugar a los sistemas de procesamiento de señales más baratos y confiables sino que poseen algunas otras ventajas. Principalmente, este tipo de sistemas poseen la característica de permitir operaciones programables. Y por lo tanto por medio del software se pueden modificar los parámetros de estas funciones de procesado de señal para que sean realizadas por medio del hardware existente. Por lo tanto estos dos componentes (Hardware y Software) proveen un mayor grado de flexibilidad en el diseño de los sistemas.

Además, de que normalmente los sistemas con hardware digital y software correspondiente logran obtener una mayor precisión con respecto a los sistemas análogos y el procesado de señales análogo. Por todas estas razones se ha producido una explosión en cuanto a avances en la teoría de procesado digital de señales y sus aplicaciones en los últimos treinta años [28].

Las wavelets fueron desarrolladas independientemente por matemáticos, físicos cuánticos, ingenieros eléctricos y geólogos, pero las colaboraciones entre estos campos durante la década pasada han conducido a los nuevos y variados usos. ¿Que son las wavelets, y porqué pueden ser útiles? La idea fundamental detrás de las wavelets es analizar según la escala. De hecho, algunos investigadores sienten que el usar wavelets de manera que adoptan una nueva tendencia o perspectiva en el proceso de datos. Las wavelets son las funciones que satisfacen ciertos requisitos matemáticos y se utilizan en la representación de datos o de otras funciones. La mayor parte de la teoría básica wavelet ya se ha desarrollado. Las matemáticas han concluido penosamente, y la teoría wavelet ahora está en la etapa del refinamiento. Esto implica el generalizar y extender el dominio de las wavelets, por ejemplo en ampliar las técnicas de paquetes wavelet. El futuro de las wavelet recaen en como-todavía el territorio de usos desconocidos. Las técnicas wavelet no se han trabajado a fondo en usos tales como análisis de datos prácticos, donde, por ejemplo, los datos discretamente muestreados en series de tiempo necesitan ser analizados. Tales usos ofrecen avenidas emocionantes para la exploración [15].

1.1.1. Relación del procesamiento de señales con otras ciencias

El procesamiento digital de señales es una ciencia que se entrelaza con un gran conjunto de ramas entre la que podemos mencionar las telecomunicaciones, el control, la exploración del espacio, la medicina, la arqueología y ciencias de la computación por mencionar algunas. Esto hoy en día es cada vez mas cierto, con la televisión digital y de alta definición, los sistemas de información, de multimedia, y de telefonía celular. Por

lo que, a medida que estos sistemas se van convirtiendo en sistemas de comunicación sin hilos, móviles y multifunción, la importancia de un procesamiento digital de señales mas sofisticadas se hace mas importante y relevante.

Esta ciencia trata de la representación, transformación y manipulación de señales. Cuando se refiere a este proceso se refiere a la representación de las señales mediante secuencias de números finitos y el procesado que se lleva acabo en un sistema digital.

Por lo regular se desea que estos sistemas funcionen en tiempo real, lo que significa que en un sistema de tipo discreto las muestras de salida se calculan a la misma velocidad que la señal en tiempo continuo que tiene en la entrada, la cual puede generar muchas aplicaciones. El procesamiento de estos sistemas en tiempo discreto y en tiempo real de señales en tiempo continuo es una práctica común en sistemas de control, comunicaciones, radar, sonar, codificación y realce de voz y vídeo, ingeniería biomédica, etcétera.

Esta ciencia ha avanzado a pasos gigantescos a lo largo del tiempo. En los años cincuenta el tratamiento de señales se realizaba mediante circuitos electrónicos e incluso con dispositivos mecánicos. Aunque las computadoras digitales ya estaban disponibles en los ámbitos empresariales y en los laboratorios científicos, estos eran costosos y abarcaban cuartos enteros además de que poseían una capacidad relativamente limitada. Una de sus primeras aplicaciones en el procesamiento digital de señales fue en la prospección petrolífera. Se grababan los datos sísmicos y se guardaban en cintas magnéticas para su procesado posterior. Pero este tipo de tratamiento no se podía realizar en tiempo real y aunque el procesamiento digital de señales en computadoras digitales ofrecía una gran ventaja de flexibilidad esta no se podía realizar en tiempo real [23].

Por sólo hablar de las aplicaciones industriales que se sustentan en procesamiento digital de señales se podrían citar algunas tales como [27]:

- Instrumentación electrónica:
  a) Filtrado de señales.
  b) Osciloscopios digitales.
  c) Analizadores de espectro.
- Electrónica de Potencia:
  a) Señales de disparo sobre SCR's, IGBT's, MOSFET entre otros.
- Control:
  a) Reguladores discretos.
  b) Controladores de robots.

- Procesamiento de imágenes:

  a) Filtrado de imágenes.
  b) Reconocimiento de formas.
  c) Compresión y descompresión de imágenes.

- Procesamiento de sonido:

  a) Identificación de fonemas.
  b) Voz sintética.

En cuanto a lo que se refiere en procesamiento de imágenes, son muy amplias las aplicaciones que esta tiene con respecto a otras ciencias como lo es en la ciencia espacial la cual sirvió para realzar y mejorar las imágenes de la Luna enviadas por la misión Surveyor, las misiones a Marte, y los vuelos tripulados Apolo a la una. En medicina, los procedimientos realzan el contraste o codifican los niveles de intensidad en los colores para facilitar la interpretación de las señales de los rayos X o de otras señales biomédicas. Los geógrafos emplean técnicas similares para analizar los patrones de polución a partir de imágenes aéreas o de satélite. En la arqueología, los métodos de procesamiento digital de señales han servido para restaurar imágenes borrosas que eran únicos registros de objetos que fueron dañados o perdidos después de haberlas fotografiado. En la física y en campos de estudio similares, estas técnicas realzan las imágenes de experimentos en los plasmas de alta energía y los microscopios electrónicos. De esta forma, los conceptos de procesamiento de señales se aplican de la misma forma en ciencias como la astronomía, la biología, investigaciones policíacas y aplicaciones industriales [10].

1.2. Planteamiento del problema

Procesamiento de imágenes es el término usado para denominar las operaciones desarrolladas sobre un conjunto de datos de imagen para mejorarlas de alguna forma, para ayudar a su interpretación o para extraer algún tipo de información útil de ella. Es obvio que el procesamiento de imágenes no puede producir información a partir de la nada. Si en el conjunto de datos no existe información concerniente a una aplicación o interpretación en particular, entonces no importa que cantidad de complicadas rutinas de procesamiento apliquemos, no se podrá obtener información [7].

El interés del procesamiento digital de imágenes recae en dos áreas principales de operación:

1. El mejoramiento de la representación pictórica para la representación humana.

2. El procesamiento de datos de la imagen para la interpretación de una maquina.

Las mejoras en los métodos de procesamiento de señales que en nuestro caso se tratan de imágenes han estado evolucionando a través de las décadas pasadas pero realmente fueron las computadoras de gran potencia lo que permitió el avance del procesamiento digital de imágenes [10].

Mediante este trabajo lo que se pretende es usar técnicas y algoritmos computacionales para mejorar las imágenes recibidas desde una cámara digital conectada en un procesador digital de señales (DSP). Para realzar y restaurar la información de las imágenes para la interpretación y el análisis humano [6].

Puesto que la transformada de Fourier y demás transformadas muestran una cierta limitación en este trabajo se implementará la técnica llamada wavelets, que son ondas de limitada duración.

La representación wavelet que usa bases ortonormales wavelet ha recibido la atención extensa. Recientemente se han construido wavelet de bases ortonormaleas de M-bandas y wavelets de M-bandas compactamente apoyadas han sido parametrizadas. Dada la teoría y los algoritmos para obtener el análisis óptimo del multi resolución wavelet para la representación de una señal dada en una escala predeterminada en una variedad de normas de error. Por otra parte, para las clases de señales, se da la teoría y los algoritmos para diseñar el análisis de multi resolución wavelet que reduce al mínimo el peor caso de aproximación del error entre todas las señales. Todos los resultados se derivan para el análisis general de multi resolución. Un esquema numérico eficiente es descrito para el diseño del análisis óptimo de multi resolución wavelet. La teoría wavelet introduce el concepto de la escala que es análoga al concepto de la frecuencia en el análisis de Fourier. Este trabajo introduce esencialmente señales de escala limitada y demuestra que las señales de banda limitada son esencialmente de escala limitada, y da el teorema del muestreo wavelet, la cual indica que los coeficientes de expansión de la función de escalamiento de una función con respecto a una wavelet base, a cierta escala (y mas) especifica totalmente una señal de banda limitada (es decir, se comporta como muestras de la taza de Nyquist (o más alta)) [13].

1.3. Objetivos de esta tesis

OBJETIVO GENERAL

Mostrar y explicar el funcionamiento y la secuencia de un procesado de imágenes a través de un procesador digital de señales (DSP), empleando la técnica de wavelets.

OBJETIVOS PARTICULARES

1. Explicar y aplicar las formas más difundidas de representación de imágenes en los dominios temporal y frecuencial, y las técnicas modernas del análisis de estas en esos dominios.

2. Describir e implementar el uso de la técnica denominada wavelets para el procesamiento digital de imágenes.

3. Explicar y aplicar los principales métodos de síntesis y codificación de imágenes.

4. Describir en términos generales la problemática del reconocimiento automático de datos a través de los DSP's así como los distintos paradigmas propuestos para la implementación de estos sistemas.

5. Describir en forma general la problemática de la Identificación/ Verificación del procesamiento de imágenes.

1.4. Justificación

Cada vez tiene mayor importancia en la ingeniería el procesado de señales digitales, puesto que es una ciencia que se entrelaza con una gran cantidad de ciencias y puesto que posee grandes características como lo son su alta flexibilidad de diseño, su capacidad de programación y de cambiar esta para obtener mejores resultados en un procesado real, además de que se cuenta con dispositivos digitales de alta velocidad y con un costo mucho más barato que dispositivos electrónicos análogos o incluso mecánicos, esta ciencia se ha convertido en un pilar fundamental de otras ciencias en cuanto se refiere al tratamiento y representación de las señales que se encuentran dispersas en este mundo y que por lo regular se encuentran de manera análoga o mejor dicho de forma continua en el tiempo.

Los problemas en procesamiento digital de señales no solo están limitados a señales unidimensionales (por ejemplo la voz) también se implican señales multidimensionales (por ejemplo imágenes). Y aunque existen diferencias importantes en el tratamiento de señales unidimensionales y multidimensionales la gran parte de este trabajo trata de sistemas multidimensionales. Entre las que se destaca las aplicaciones de procesamiento de imágenes digitales.

De esta misma manera, esta ciencia trata de aplicar las mismas características en cuanto se refiere al procesado de imágenes e incluso video que hoy en día son parte fundamental en los sistemas digitales que se usan en la actualidad tales como sistemas de comunicación móvil, Internet, televisión digital, por mencionar algunas. Estas necesitan tener un proceso digital de señales muy avanzado y en algunos casos en tiempo real.

Por lo que en este trabajo nos enfocaremos al proceso y representación de imágenes mediante la técnica denominada wavelets implementado en un procesador digital de señales denominado por sus siglas en inglés "DSP".

Esta técnica nos permite percibir los rasgos de una imagen predeterminada mediante funciones de escala para un análisis de multi resolución óptima. Por lo que la detección de bordes mediante esta técnica es la respuesta que se requiere para una gran variedad de aplicaciones concretas como lo es el control de sistemas no lineales. La cual servirá para obtener la posición, ángulo y detalles a partir de la imagen tomada de la cámara

conectada al "DSP", este contiene los algoritmos para calcular un algoritmo wavelet para, posteriormente realizar una etapa de control para el sistema, que este último queda fuera del alcance de esta tesis.

1.5. Estado del arte del procesamiento digital de señales

Las wavelets son funciones matemáticas que acortan los datos en diferentes componentes de frecuencia, y por eso estudiar cada componente con una resolución correspondiente a su escala. Estas tienen diferentes ventajas con respecto a los tradicionales métodos de Fourier en el análisis físico de situaciones donde las señales tienen discontinuidades y picos marcados. Las wavelets fueron desarrolladas independientemente en los campos de matemáticas, física cuántica, ingeniería eléctrica y geología sísmica. Intercambiando entre estos campos durante los últimos diez años se dirigió a nuevas aplicaciones wavelet como lo son compresión de imágenes, turbulencia, visión humana, radar y predicción de terremotos [14]. Amara Graps describe en [14] la historia de las wavelets comenzando con Fourier, comparando las transformadas wavelet con las transformadas de Fourier, propiedades de estado y otros aspectos espaciales de las wavelets y finalizando con algunas aplicaciones importantes como compresión de imágenes, tonos musicales y de-noising de datos.

Las wavelets son funciones de ceros significativos. Tienen un enorme rango de aplicaciones prácticas. Como compresión de datos mediante wavelets y el análisis de wavelets en el aspecto astronómico. Una aplicación de las transformadas wavelets es el modelo de visión multi escala para la detección de rasgos en galaxias de la via láctea, en la detección de rasgos positivos hacia la región formada por Sagitario B y catalogando la detección de rasgos negativos (nubes negras) de otra región [32].

Se propone una nueva clase de red wavelet para identificación de sistemas no lineales. En las nuevas redes, como modelo de estructura para sistemas de grandes dimensiones se escoge para ser una super imposición de un número de funciones con pocas variables. Expandiendo cada función usando wavelets de descomposición truncadas, las redes no lineales multi variables pueden ser convertidas en lineales en los parámetros de regresión, los cuales pueden ser resueltos usando el tipo de método del último cuadrado. Una aproximación de la selección eficiente del término del modelo del algoritmo del último cuadrado ortogonal hacia delante (OLS) y la taza de la reducción del error (ERR) es aplicado para resolver los problemas en la linealidad de los parámetros. La principal ventaja en la nuevas redes wavelet es que esta explora los aspectos atractivos de las descomposiciones multi escala wavelet y la capacidad de las redes neuronales tradicionales. Adoptando el análisis de expansión de la varianza (ANOVA), ahora las redes wavelet pueden manejar los problemas de identificación no lineal para grandes dimensiones [4].

Se propone una plantilla wavelet deformable (DWT) para la descripción de la forma del objeto. La DWT ofrece no solamente información global a pequeñas escalas, pero también rasgos locales a altas escalas diferenciales. Por lo que, esta provee una herramienta natural para la representación de multiresolución y puede ser usada conve-

nientemente en un procedimiento jerárquico. En la representación, se direccionan tres pasos de procesamiento en la DWT basada en el reconocimiento automático de blancos. Estos son: 1) pre procesamiento de imagen y extracción de la forma del blanco, 2) normalización de los rasgos de la forma, y 3) descomposición wavelet [18].

Una aproximación a la segmentación de imágenes basada en la detección de los bordes de una imagen critica por la formulación del problema como un tema de clasificación. La meta principal de esta investigación es para identificar mejor los cambios abruptos de la imagen sin incrementar la presencia de ruido en la imagen resultante. La metodología que se sugiere esta basada en la transformada wavelet de dos dimensiones aplicada a la imagen original subsecuentemente involucrada. Este proceso es considerado como un procedimiento de clasificación empleando técnicas de entrenamiento supervisadas y, mas específicamente, análisis discriminante multi-variable paso por paso [21].

Una aproximación eficiente wavelet basada en multi-resolución al problema de visión estéreo. Se define una función iterativamente minimizada. La minimización es desarrollada en la representación de imágenes en el espacio de la wavelet. Utilizando la teoría de representación de operadores en espacios que abarcan funciones escalares y por lo tanto toman ventaja de una aproximación simplificada de diferenciación [16].

Un nuevo y eficiente método de extracción de rasgos en la transformada rápida wavelet que especialmente trata con la valoración del proceso de parámetros o estados en una aplicación dada usando los rasgos extraídos de los coeficientes wavelet del proceso de señales medido. Puesto que estos parámetros se valoran usando todos los coeficientes wavelet muchas veces tienden a ser tediosos o recaen en la imprecisión de los resultados, una rutina de pre procesamiento rasgos robustos correlacionados al proceso de parámetros de interés es altamente deseable. Este método divide la matriz de los coeficientes wavelet calculados en clusters de iguales vectores fila. La filas que representan un importante rancio de frecuencias (para interpretación de señales) tienen un gran número de clusters que las filas que representan menor rango de frecuencias. Los rasgos de la señal procesada son eventualmente calculados por por las normas Euclidianas de los clusters [26].

Líneas y bordes proveen importante información para reconocimiento de objetos y categorización. Además, un modelo brillante se basa en la interpretación simbólica de la representación cortical multi escala de las líneas/bordes. La representación multi escala del esquema para la extracción de líneas/bordes para células simples y complejas puede ser usada para reconstrucción visual, pero también para rendimiento no foto realista [31].

Una wavelet ortogonal se aproxima al pre blanqueado del filtro para detector los objetos Gaussianos con ruido de Markov. La implementación de filtros de bancos de la transformada wavelet actúa jerárquicamente como la operación de los detectores a escalas de objetos discretos. Si el objeto es detectado y su escala acontece que coincide con uno de los calculados por la transformada wavelet y si el ruido de fondo es realmente de Markov, entonces su realización óptima es realizada por el retenimiento de la apropiada sub banda de la imagen. En realidad, la Gaussiana puede ser mas bien una

aproximación tosca de la imagen, y el ruido de fondo puede derivar de Markov. En este caso veremos a la descomposición wavelet como el medio para calcular el conjunto de rasgos ortogonales para la entrada de un clasificador. Usando un clasificador de supervisión lineal aplicado a los vectores de rasgos comprendidos en muestras que tomadas de las muestras de la enésima octava. el mapa resultante de la prueba estadística de valores indica la presencia y localización de objetos. El objeto por si mismo es reconstruir por la prueba estadística para enfatizar las subbandas wavelet, seguido por el cálculo de la transformada inversa wavelet [34].

Un nuevo método para extraer los contornos de una imagen, basado en wavelets, consiste en tres pasos. Primero, por descomposición de una wavelet ortogonal la imagen se descompone en un conjunto de aproximaciones wavelet y detalles wavelet. Después, la extracción del contorno se logra aplicando la técnica de reconstrucción wavelet. En el tercer paso, se implementa la extracción de los contornos de multi resolución con respecto a reconstrucción por iteraciones wavelet. Una nueva combinación de contornos de multi resolución resulta en claros contornos de la imagen. Este procedimiento de reconstrucción esta acompañado por una estimación de distribución de energía de la cual el tiempo de descomposición wavelet puede ser controlados adaptablemente. Nuestro método no tiene operación redundante comparada con el esquema de la transformada wavelet. Por lo tanto, el tiempo de cálculo y los requerimientos de memoria son menos que en la transformada wavelet [9].

Un método de detección de bordes de multi resolución basado en la representación wavelet en el cual los bordes son detectados del máximo local de los detalles de la imagen en la transformada wavelet ortogonal. La idea es calcular un filtro óptimo de detección de filtros para bordes rampa. Esta muestra que la aproximación del borde de cada escala en su descomposición wavelet ortogonal puede ser modelada por un contorno de rampa. Como el detector de bordes multi escala es importante no solamente para las aplicaciones usuales tal como reconocimiento de patrones y segmentación de imágenes, sino también para objetos basados en codificación de imágenes donde la reconstrucción de la imagen original de los bordes a diferentes escalas se hace posible por el análisis y síntesis wavelet [20].

Se considera el problema de la detección de bordes direccional y de multi escala. La transformada wavelet de M bandas ortogonal y de fase lineal se usa para descomponer la imagen en canales $M \times M$. Entonces estos canales son combinados de tal forma que cada combinación, a la cual nos referimos como filtro de descomposición, resultando en cero cruces en las correspondientes localizaciones de bordes a diferentes direcciones y resoluciones e inherentemente ejecuta la regularización contra el ruido. Aplicando el detector de cero cruces a las salidas de los filtros de descomposición, se obtienen los mapas de bordes de la resolución deseada y detección. Además, con la aplicación del operador de energía de Teager en la etapa de análisis, es posible obtener la reducción de cero cruces innecesarios. Finalmente los mapas de bordes de la imagen son obtenidos a través de simples combinaciones de los mapas de bordes direccionales [2].

El límite de un la imagen de un objeto esta distorsionada por el ruido u otros

elementos del sistema que se mezclan con la imagen durante la transmisión. Por lo que el límite de la imagen que va a ser detectada y exactamente dividida por un método óptimo de detección de bordes. Después de plantear los limites del objeto aplicando la morfología adaptable en el retenimiento de la imagen de entrada. El borde óptimo es detectado usando una red celular neural wavelet (CNN). el método propuesto se compara con el convencional método de Sobel usado en el algoritmo de detección de bordes [36].

Un esquema basado en wavelet en la detección de bordes a multi escala. Multiplicando los coeficientes wavelet en dos escalas adyacentes para magnificar las estructuras significativas y para suprimir ruido, determinamos los bordes como los máximos locales directamente en el producto de la escala después de un retenedor eficiente, en vez primero de formar los mapas del borde en varias escalas y en seguida de sintetizarlas juntas, según lo empleado en muchas técnicas del multi escalas. Se demuestra que la multiplicación de la escala alcanza resultados mejores que cualquiera de las dos escalas, especialmente en el funcionamiento de la localización. Los experimentos en imágenes naturales se comparan con el Laplaciano Gaussiano y Algoritmos Canny de la detección de bordes [22].

La detección de bordes es un problema importante de bajo nivel en visión. La mayoría de los métodos de detección de bordes funcionan en una imagen en una sola resolución y hacen salir un mapa binario de los bordes. Los bordes dentro de una imagen, sin embargo, ocurren generalmente en las varias resoluciones, o escalas, y representan transiciones de diversos grados, o niveles de gradientes. Así, los métodos simples de detección de bordes que hacen salir mapas binarios de bordes no rinden siempre resultados satisfactorios. Se desarrolla un método de detección de bordes a multiresolución que utiliza una descomposición wavelet de multi tasas para generar una serie de imágenes con una resolución progresivamente más baja de los bordes. Los bordes entonces se extraen recurrentemente para formar la serie de mapas de los bordes donde la salida no se restringe para ser binaria y se fija para reflejar el nivel del gradiente en cada punto del borde. La serie de mapas de los bordes se restringe para formar una pirámide del mapa de los bordes que apila. En esta formulación, el mapa del borde (del nivel más bajo) contiene los bordes en todas las escalas mientras que recorta el borde, basado en escala del borde, se realiza en los niveles subsecuentes. Este acercamiento se demuestra para tener métodos de detección previamente definidos de los bordes a multi resolución del excedente de las ventajas. Se presentan los resultados usando imágenes consideradas naturales [33].

Los oceanógrafos y los investigadores de sensación remota han reconocido ampliamente el potencial de usar las imágenes de satélites para estudiar ondas internas oceánicas. Si, como probablemente, los patrones de los tonos de grises de estas imágenes se pueden confirmar para corresponder a los patrones del canal y de la cresta de ondas internas, entonces mucho puede ser aprendido sobre ondas internas de datos basados en los satélites. La utilidad del análisis wavelet como herramienta para la detección ondas internas y longitud de onda es examinada en las versiones continua y discreta

de la transforma wavelet. El fondo teórico de cada procedimiento es brevemente descrito y aplicado usando una wavelet específica para cada caso. Primero, se presenta la construcción de una wavelet base apropiada, basada en un modelo analítico de la onda interna oceanográfica, para detectar y para localizar ondas no lineales de la imagen del océano. La estructura de la base arbitraria de la wavelet derivó del soporte compacto de las wavelets ortonormales de B-tiras se estudian para obtener mas óptimas descomposiciones wavelets discretas. Las comparaciones se hacen para las descomposiciones wavelets basadas en varias familias de wavelet compactamente apoyada. Finalmente, transforma continua wavelet se aplica para estimar las energías y las longitudes de onda dentro de picos de los trenes onda interna detectadas. Los resultados de este estudio demuestran que el análisis wavelet es una herramienta excelente para detectar ondas internas contra ruido de fondo, y estimar, con un buen grado de la precisión, longitudes de onda SAR de perfiles de imágenes del [30].

Se desarrolla un algoritmo multi-hilos de carga-adaptable para calcular la transformada wavelet discreta (DWT) y su aplicación en una plataforma multi-hilos de textura fina. En el cálculo de la DWT, el problema del tamaño se reducen en cada nivel de descomposición y la longitud del cálculo emergente también varía. Este algoritmo paralelo, se escala dinámicamente al tamaño del problema que varía. Durante cualquier iteración, el cociente del número de hilos al número de hilos remotos emanados por un procesador puede ajustarse para ser mayor que 1 controlando los parámetros del algoritmo. Esta aproximación proporciona una oportunidad de interpolar el cálculo y la comunicación sin explícitamente introducir ciclos inactivos en espera de los hilos remotos para acabar. Los resultados experimentales basados en las puesta en práctica del algoritmo en un nodo de 20 emulando la plataforma multi-hilos, EARTH-MANNA, diseñado específicamente para paradigmas multi-hilos de textura fina [24].

Se presenta una novedosa representación 2D de la potencia de las formas de onda del sistema para el análisis automático y detección de acontecimientos transitorios. La representación es integrada por una matriz cuyas filas son formadas por segmentos de tiempo de formas de onda digitales. Por la selección apropiada de la longitud de segmentos del tiempo, los datos 2D exhiben formas onduladas de la imagen. La forma general se desestabiliza inmediatamente siempre que ocurra un acontecimiento transitorio en la calidad de potencia. Proponemos el uso de las transformadas wavelet discreta de dos dimensiones (2D-DWT) para detectar estas perturbaciones. Se ha observado que, después de omitir el espacio de la aproximación de las señales de la transformada wavelet y denoising espacio del detalle de las señales, la transformada inversa 2D-DWT proporciona buenos resultados de la detección y de la localización, incluso para los casos donde los métodos convencionales fallan [8].

La tecnología de identificación biométrica se ha asociado generalmente a usos seguros muy costosos. Una nueva aproximación para el sistema de reconocimiento del iris. Utiliza operadores morfológicos para la detección de bordes del iris. Simplicidad y rapidez el método propuesto de detección de bordes el cual es considerable hacer frente a imágenes binarias. La representación del código binario vía la fase de la wavelet de Daubechies

se consigue de cada imagen del iris y se utiliza un clasificador euclidiano mínimo de la distancia para el proceso de correspondencia [1].

Una nueva aproximación para reconocer el iris del ojo humano es a cero cruces de la transformada wavelet en varios niveles de resolución se calculan sobre círculos concéntricos en el iris, y las señales unidimensionales resultantes (1-D) se comparan con las características modelo usando diversas funciones diferentes [5].

Se Presenta un método para la segmentación automatizada de la vasculatura en imágenes retinales. El método produce segmentaciones clasificando cada píxel de la imagen como un *vessel* o *novessel*, basado en los píxeles del vector de rasgos. Los vectores de rasgos están compuestos por píxeles de intensidad y la transformada wavelet continua de Morlet de dos dimensiones toma las respuestas a múltiples escalas. La wavelet de Morlet es capaz de sintonizar frecuencias específicas, así permitiendo la filtración del ruido y el realce de *vessel* en un solo paso. Se utiliza un clasificador Bayesian con las funciones de probabilidad de densidad de clase condicional descritas como mezclas Gaussianas, produciendo una clasificación rápida, mientras que el poder modelar la decisión compleja emerge y se compara su funcionamiento con el clasificador del error linear de mínimos cuadrados [19].

Se propone esquema eficiente de reconocimiento facial con dos características: 1) representación de las imágenes de la cara por coeficientes wavelet de sub banda de dos dimensiones (2D) y 2) reconocimiento por un método de clasificación modular, personalizada basado en modelos de la memoria asociativa del kernel. Comparado a las proyecciones PCA y las representaciones de la imagen de resolución baja "pulgar-uña", los coeficientes de sub banda wavelet pueden capturar eficientemente características faciales substanciales mientras mantienen complejidad de cómputo baja. Como allí están generalmente las muestras muy limitadas, se construye un modelo de memoria asociativa (AM) para cada persona y propuesta para mejorar el funcionamiento de los modelos AM por métodos kernel. Específicamente, primero se aplica la transformada kernel a cada par posible del entrenamiento de la muestra de las caras y después delinear el espacio de rasgos a altas dimensiones de regreso al espacio de la entrada. El esquema usa una memoria modular auto asociativa para el reconocimiento de la cara es inspirada por la misma motivación que usar autoencoders para el reconocimiento de caracteres ópticos (OCR), para el cual se han probado las ventajas. Por memoria asociativa, todas las caras prototipo de una persona en particular se utilizan para reconstruirse y la reconstrucción del error para una prueba de la imagen de la cara se utiliza para decidir si la cara de prueba es de la persona correspondiente [3].

Un controlador adaptable no lineal wavelet (AWC) que usa redes constructivas wavelet se propone para una clase de sistemas dinámicos no lineales generales. Las redes wavelet se emplean como aproximador universal para dinámicas altamente no lineales y lineales. En virtud de la propiedad ortonormal novedosa y de la propiedad de la multi-resolución de las redes wavelet, la estructura del controlador wavelet adaptable no lineal puede ser ajustada constructivamente en línea, según el funcionamiento que sigue del sistema dinámico. La propiedad ortonormal asegura que agregando una nueva

resolución (nuevas wavelets) no afecta la red existente wavelet que pudo haber sido sintonizada bien. La propiedad del multi resolución asegura la mejora de la precisión de la aproximación cuando se agrega una nueva resolución. Un controlador adaptable robusto wavelet se puede construir primero con una estructura simple. Se introduce un período de adaptación para especificar el período de espera admisible para el error que sigue para converger bajo estructura actual de la wavelet. Si el sistema no puede converger después del período de espera admisible, se considera una nueva resolución wavelet necesaria y agregada directamente. De este modo, el controlador adaptable wavelet se puede construir y sintonizar fácilmente de un grueso a un fino nivel mientras que conserva la estabilidad en un lazo cerrado. También proporciona un grado de libertad adicional para los usuarios, en orden de balancear la complejidad de la y convergencia de la red [17].

1.6. Cuerpo del trabajo de tesis

La estructura de esta tesis es la siguiente:

En el *Capítulo 2* llamado señales y sistemas se da una introducción a los tipos de señales básicas que existen de manera análoga así como también los tipos de sistemas en tiempo discreto. Se discuten los sistemas lineales y de desplazamiento invariante sobre todo porque son más fáciles de analizar y de ejecutar en los cuales se pueden usar las señales antes mencionadas. Además, se mostrara el procesado básico para estas. El énfasis en este capítulo está en las representaciones y la puesta en práctica en señales y sistemas usando MATLAB.

En el *Capítulo 3* denominado Análisis Frecuencial y Transformadas de Fourier se discuten los algoritmos de la transformada de Fourier en tiempo discreto y la transformada rápida de Fourier como herramientas útiles en el análisis y diseño de los sistemas lineales e invariantes en el tiempo. Además de las muchas aplicaciones que estas tienen en el procesamiento digital de imágenes.

En el *Capítulo 4* el cual lleva por nombre Wavelets se muestra una técnica diferente en cuanto a transformadas de tratamiento de imágenes que específicamente se direcciona al problema de la compresión de imágenes, bordes, detección de rasgos y análisis de texturas. Se ven algunas limitaciones en cuanto a la transformada de Fourier y transformadas similares y se definen tres tipos de transformadas que mejoraran el funcionamiento para aplicaciones especificas. Además, se notaran algunas aplicaciones de las transformadas Wavelets.

En el *Capítulo 5* se llevara acabo la aplicación de la teoría presentada en los capítulos anteriores mediante la simulación mediante un software especializado y de ambiente amigable para que los usuarios, el cual usa algoritmos son expresados de forma fácil.

Este software es MATLAB el cual es un sistema interactivo cuyos elementos básicos son un arreglo que no requiere dimensionar. Esto permite resolver varios problemas de cálculo técnico, necesario para resolver los problemas que en este trabajo se presentan.

En el *Capitulo 6* se llevaran acabo las aplicaciones mediante el equipo especializado para el procesamiento de imágenes. Que en nuestro caso es MATLAB 6.5 puesto que provee herramientas y algoritmos para las simulaciones correspondientes de imágenes con wavelets.

Finalmente, en el *Capítulo 7* se llevara acabo la implementación de los conocimientos previamente adquiridos y los que se han llevado a la práctica en las simulaciones dentro del dispositivo denominado procesador digital de señales (DSP) de la serie TMS320C6711.

Capítulo 2

SEÑALES Y SISTEMAS

En este capítulo se dan a conocer las conceptos de señal y sistemas en tiempo discreto. Además estudian un número importante de tipos de señales y sus operaciones. Se discutirán los sistemas lineales e invariantes en el tiempo puesto que son más fáciles de analizar y ejecutar. Así pues, también se darán algunos otros términos básicos como los son las características de estos sistemas y la convolución. Así también se introducirá a lo que es el procesamiento digital de imágenes haciendo uso de MATLAB.

2.1. Conceptos básicos

- Señal: Es una señal física que varía en el tiempo y el espacio. Matemáticamente, se puede describir como una función de una o más variables independientes. Por ejemplo una señal de voz se puede representar con un alto grado de exactitud como una suma de sinusoides de diferentes amplitudes como se muestra en al ecuación (2.1).

$$\sum_{i=1}^{N} A_1(t)\sin[2\pi F_i(t) + \theta_i(i)] \tag{2.1}$$

  donde A_1 es la amplitud, F_i es la frecuencia general y θ es la fase.

- Sistema: Se define como un dispositivo que realiza una operación sobre una señal como por ejemplo un filtro. Si la operación del sistema es lineal el sistema es lineal, si la operación es no lineal el sistema se denomina como no lineal.

- Procesado de señal: Se dice que la señal obtiene un proceso cuando es pasada a través de un sistema, por ejemplo un filtro. Esto implica la separación de la señal deseada de ruido e interferencia. A las operaciones que realiza un sistema generalmente es lo que se conoce como procesado de la señal.

- Algoritmo: Es un método o conjunto de reglas para implementar el sistema mediante un programa que efectué las operaciones matemáticas correspondientes.

- Frecuencia: Esta directamente relacionada con el tiempo. Relacionada en un movimiento periódico llamado oscilación armónica [28].
- Muestreo: Se concreta en la toma de valores de una señal continua en sucesivos instantes de tiempo. El muestreo de señales consiste en la construcción de secuencias a partir de señales continuas. Un problema que salta a la vista es la pérdida de información.
- Cuantificación: En la conversión analógica/digital de una señal para su tratamiento con computadora además del proceso de muestreo hay que tener en cuenta el de cuantificación, efecto éste debido a la limitación del número de cifras de los registros en los que se almacenan sus valores [27]. Es el proceso de convertir una señal continua en discreta [28].

2.2. Ventajas de procesamiento digital de señales contra procesamiento analógico de señales

Como ya se ha mencionado existen diversas razones por lo que el procesamiento digital de señales es preferible con respecto al procesamiento análogo de señales en las que se pueden mencionar las siguientes:

Primero, un sistema digital programable nos da la opción de rediseño de las operaciones de procesado con solo cambiar el programa. Mientras que el procesamiento análogo implica el rediseño del hardware, seguido de una comprobación y verificación del funcionamiento del mismo.

Los sistemas digitales permiten una mayor precisión en sus componentes. Mientras que en los sistemas análogos las tolerancias de los componentes hacen mas difícil la controlar la precisión.

Las señales que se obtienen de los sistemas digitales son fáciles de almacenar sin deterioro o pérdida en la fidelidad de la señal, por lo que estas señales se vuelven transportables para un procesado no necesariamente en tiempo real en un laboratorio remoto.

Los métodos de procesamiento digital de señales nos permiten aplicar algoritmos mas complejos mientras que en el procesado análogo es mas difícil aplicar funciones matemáticas en sus señales [28].

Una desventaja importante del procesado analógico de señales es su alcance limitado para realizar usos complicados del procesamiento de señales. Esto se traduce a no flexibilidad en el procesamiento y la complejidad en el diseño de sistemas. Todos éstos conducen generalmente a productos costosos. Por otro lado, usando una aproximación al procesado digital de señales, es posible convertir una computadora personal barata en un procesador digital poderoso.

Los sistemas que usan la aproximación de procesado digital de señales pueden ser desarrolladas usando el software que funciona en una computadora de fines generales.

Por lo tanto el procesado digital de señales es relativamente conveniente de desarrollar y probar, y el software es portátil.

Las operaciones de procesado digital de señales se basan solamente en las adiciones y las multiaplicaciones, conduciendo a una capacidad de procesamiento extremadamente estable - por ejemplo estabilidad independiente de temperatura.

La operación de procesado digital de señales se puede modificar fácilmente en tiempo real, a menudo por un cambio simple de programación, o recargando registros.

Procesado digital de señales tiene un costo más bajo debido a la tecnología VLSI, que reduce costos de memorias, compuertas, microprocesadores y así sucesivamente [35].

2.3. Clasificación de señales

Los métodos que se usan en procesamiento digital de señales se aplican a señales con características específicas. Por lo que en esta sección se darán a conocer la clasificación de estas señales para aplicaciones concretas.

2.3.1. Señales multicanal y multidimensionales

Una *señal multicanal* se puede definir como un vector de señales el cual contiene dentro de si misma un conjunto de dos o más señales de la forma:

$$S(t) = \begin{bmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \end{bmatrix} \tag{2.2}$$

Por ejemplo, la señal de la Figura 2.1 nos muestra las tres componentes de una señal vectorial que representa la aceleración de una superficie terrestre.

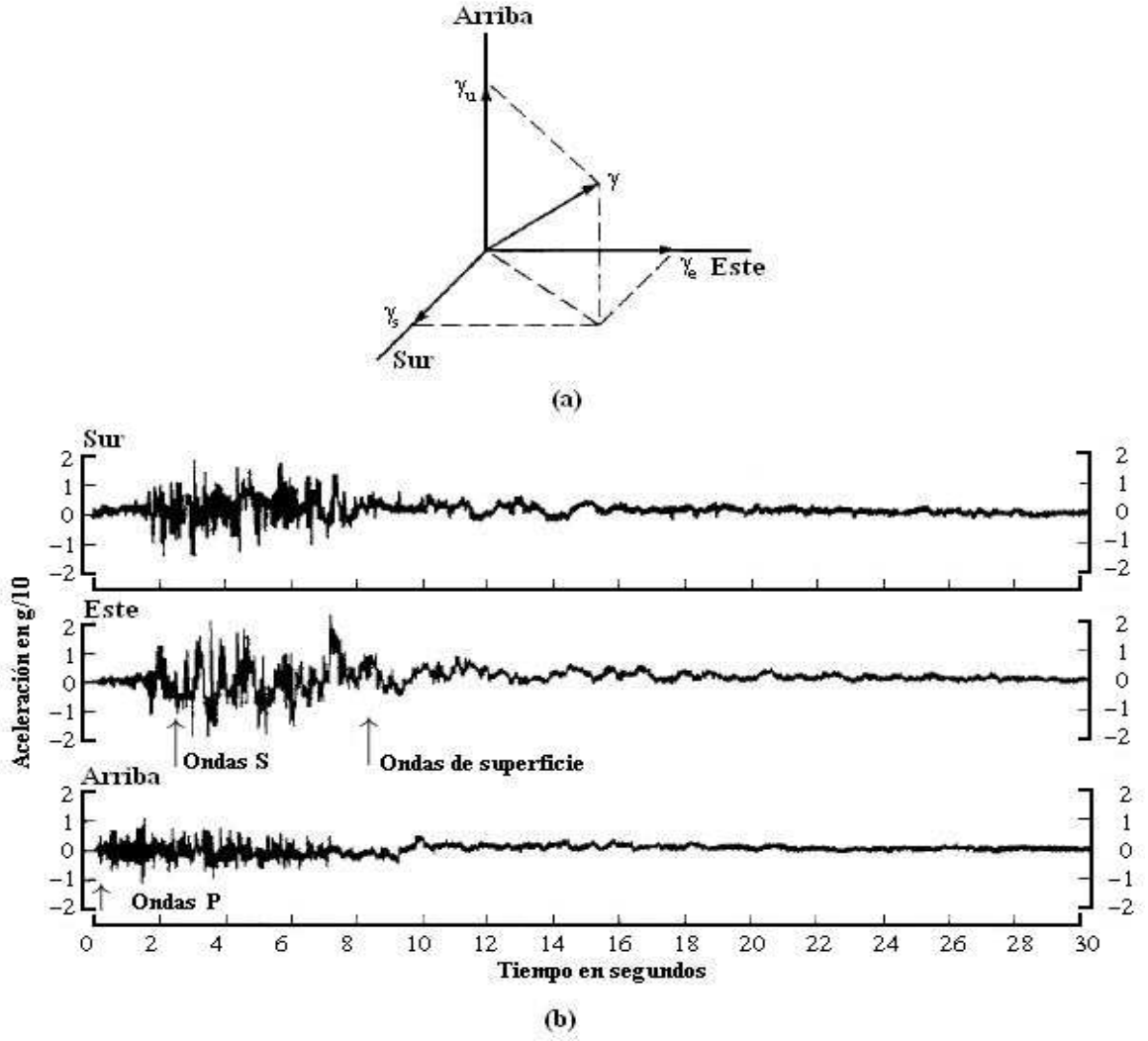

Figura 2.1: Las tres componentes de la aceleración de la tierra a pocos kilómetros del epicentro de un terremoto.

Por otra parte si, una función posee una sola variable independiente se le conoce como *unidimensional*. Por el contrario, se le denomina *multidimensional* si posee M variable independientes. La imagen de la Figura 2.2 dan un ejemplo de una señal bidimensional dado la intensidad o brillo $I(x, y)$. Por lo que podemos decir que la señal de televisión es una señal tridimensional compuesta de la forma:

$$\mathbf{I}(x, y, t) = \begin{bmatrix} I_r(x, y, t) \\ I_g(x, y, t) \\ I_b(x, y, t) \end{bmatrix} \tag{2.3}$$

Figura 2.2: Ejemplo de una señal de dos dimensiones

2.3.2. Señales en tiempo continuo y señales en tiempo discreto

Las señales se pueden clasificar dependiendo de las características de la variable (independiente) tiempo y los valores que esta puede tomar.

Las señales en tiempo continuo o señales análogas son aquellas que están definidas para todos los valores en el tiempo y pueden tomar cualquier valor en el intervalo continuo (a, b) donde a puede ser $-\infty$ y b puede ser ∞. Matemáticamente, estas funciones se describen como señales continuas de variable continua. Por ejemplo, la onda de voz de la Figura 2.3 es una señal análoga.

Las señales en tiempo discreto están definidas para ciertos valores en el tiempo. Estos valores no necesariamente deben ser equidistantes, pero en la práctica se toman valores equidistantes conforme al interés que se aplique. Las señales en tiempo discreto se definen por una secuencia de números reales o complejos. Por ejemplo la señal

$$x(n) = \begin{cases} 0.8^n, & \text{si } n \geq 0; \\ 0, & \text{en otro caso.} \end{cases} \tag{2.4}$$

La cual se muestra en la Figura 2.4.

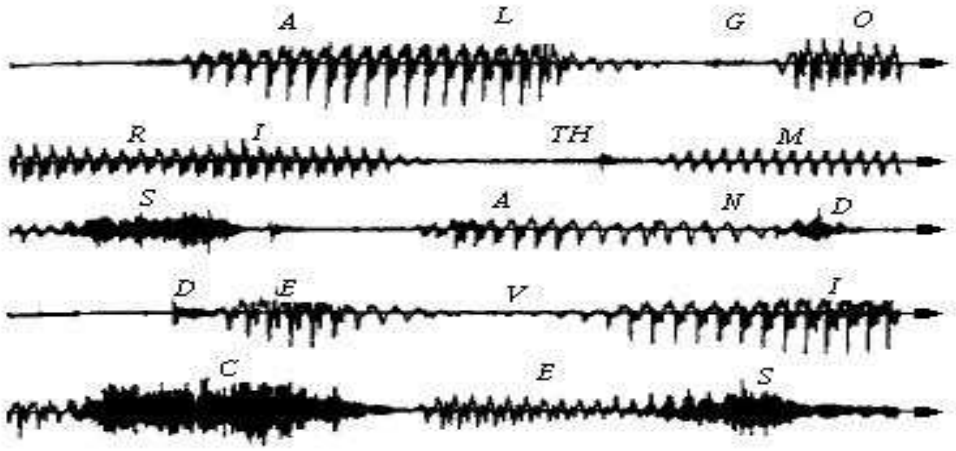

Figura 2.3: Ejemplo de una señal de voz.

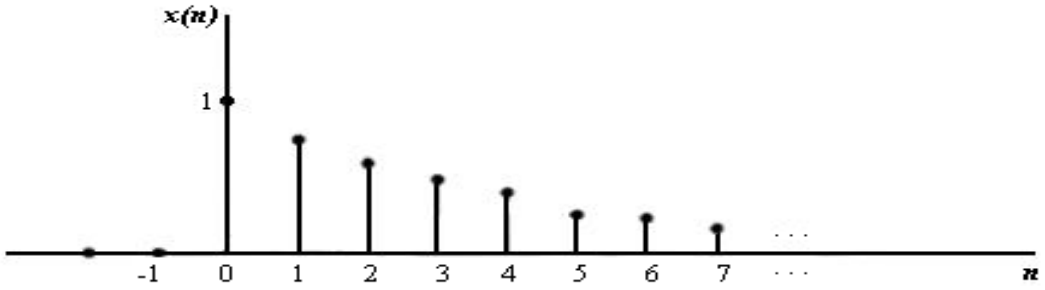

Figura 2.4: Representación gráfica de una señal discreta $x(n) = 0.8^n$ para $n > 0$ y $x(n) = 0$ para $n < 0$.

Existe en la práctica dos formas de originarse las señales:

1. Eligiendo los valores de una señal análoga en determinados instantes de tiempo
2. Acumulando una variable a lo largo de un determinado tiempo

Las señales de tiempo discreto podemos clasificarlas en:

1. Señales de energía
2. Señales de potencia
3. Señales periódicas
4. Señales aperiódicas

5. Señales simétricas (pares)

6. Señales antisimétricas (impares)

2.3.3. Señales deterministas y señales aleatorias

Cualquier señal que pueda ser descrita por una forma matemática explícita, un conjunto de datos o una regla bien definida se le conoce como *determinista*. Además, esto resalta el hecho de que los valores de las señales tanto presentes como pasados como futuros, reconocen exactamente.

Por lo contrario, cuando las señales no pueden ser determinadas por un grado de precisión mediante formas matemáticas o son muy complicadas para una aplicación práctica nos referimos a que son señales aleatorias. Por ejemplo la salida de un generador de ruido o la señal de voz de la Figura 2.3 [28].

2.4. Señales en tiempo discreto y sus operaciones

En el estudio de las señales y sistemas en tiempo discreto existen varias señales básicas que juegan un rol importante en el procesamiento digital de señales. Las cuales se definen a continuación [28]:

1. *impulso unitario*

   El impulso unitario se denomina $\delta(n)$ y se define como:

$$\delta(n-n_o) = \begin{cases} 1, & n=0; \\ 0, & n \neq 0. \end{cases} = \{..., 0, 0, \underset{\uparrow}{1}, 0, 0, ...\} \tag{2.5}$$

   En otras palabras, el impulso unitario es una señal que vale cero siempre excepto en $n = 0$ donde vale uno. La representación del impulso unitario se muestra en la Figura 2.5. En MATLAB la función $zeros(1, N)$ genera un vector fila de N ceros, que se pueden utilizar para poner en ejecución a $\delta(n)$ sobre un intervalo finito. Sin embargo la relación lógica $n == 0$ es una manera elegante de poner a $\delta(n)$ en ejecución. Por ejemplo al poner en práctica la ecuación (2.5) en la cual el impulso unitario se encuentra fuera de cero.

   Sobre el intervalo $n_1 \leq n \leq n_2$, se utiliza la siguiente función de MATLAB:

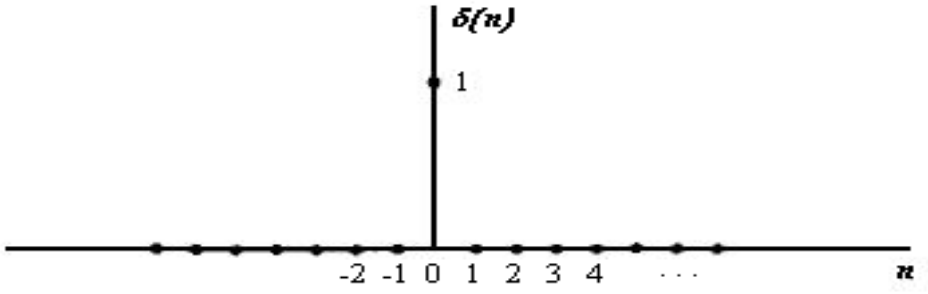

Figura 2.5: Representación gráfica de la señal impulso unitario.

MATLAB 2.1: Función para un impulso unitario finito.

```
function [x,n] impseq(n0,n1,n2)
%genera a x(n)= delta (n-n0); n1<= n <= n2
%---------------------------------------
%[x,n] impseq(n0,n1,n2)
%
n= [n1:n2]; x= [(n-n0) == 0];
```

2. *Escalón unidad*

El impulso unitario se denomina $u(n)$ y se define como:

$$u(n-n_o) = \begin{cases} 1, & n \geq n_0; \\ 0, & n < 0. \end{cases} = \{..., 0, 0, \underset{\uparrow}{1}, 1, 1, ...\} \tag{2.6}$$

El valor de la secuencia escalón unidad en el índice (temporal) n es igual a la suma acumulada hasta el índice n de todos los valores anteriores de la secuencia impulso. Es decir, se interpreta como la suma de impulsos atrasados. La cual se muestra en la Figura 2.6 [23]. En MATLAB la función $ones(1, N)$ genera un vector fila de N unos. Puede ser usado para generar $u(n)$ sobre un intervalo finito. Una aproximación elegante es de nuevo utilizar la operación de relación lógica $n >= 0$ para poner en practica la ecuación (2.6).
Sobre el intervalo $n_1 \leq n \leq n_2$, utilizaremos la siguiente función de MATLAB:

MATLAB 2.2: Función para un escalón unitario finito.

```
function [x,n]= stepseq(n0,n1,n2)
%genera a x(n)= u(n-n0); n1<= n <= n2
%---------------------------------------
%[x,n] stepseq(n0,n1,n2)
%
n= [n1:n2]; x= [(n-n0) >= 0];
```

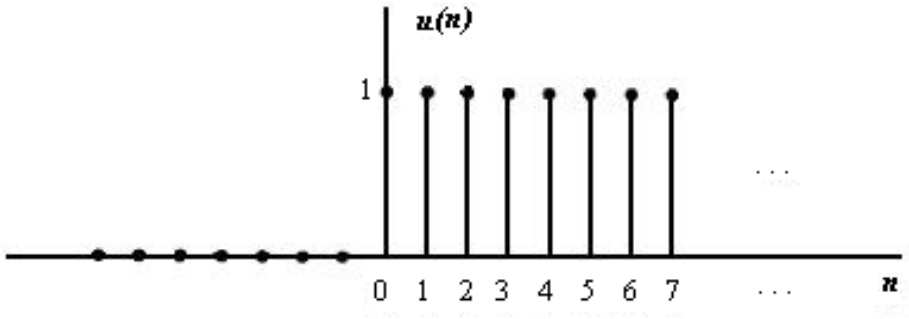

Figura 2.6: Representación gráfica de la señal de escalón unitario.

3. *Rampa unidad*

Se denota como $u_r(n)$ y se define como [28]:

$$u_r(n) = \begin{cases} n, & \text{para } n \geq 0; \\ 0, & \text{para } n < 0. \end{cases} \tag{2.7}$$

La cual se muestra en la Figura 2.7 .

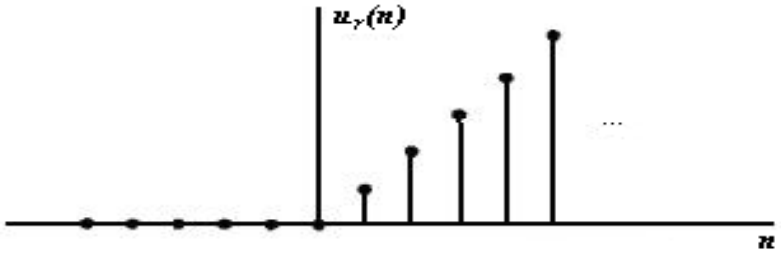

Figura 2.7: Representación gráfica de la señal de rampa unitaria.

4. *Secuencia exponencial de valores reales*

$$x(n) = a^n, \forall n; a \in \mathbb{R} \tag{2.8}$$

En MATLAB un arreglo de operadores "·^"se requiere para poner en practica una secuencia exponencial real. Por ejemplo, para generar, $0 \leq n \leq 10$ se necesita el código siguiente de MATLAB.

```
>> n = [0:10];   x = (0.9) .^;
```

5. *Exponencial del valor complejo*

$$x(n) = e^{(\alpha + j\omega_0)n}, \forall n \tag{2.9}$$

Donde se llama s a una atenuación y ω_0 es la frecuencia en radianes. La función exp de MATLAB se utiliza para generar secuencias exponenciales. Por ejemplo para generar $x(n) = exp[2 + j3(n)]$, $0 \leq n \leq 10$ necesitaremos el código siguiente de MATLAB.

```
>> n = [0:10];  x = exp((2+3j)*n)
```

6. *Secuencia sinusoidal*

$$x(n) = \cos(\omega_0 n + \theta), \forall n \tag{2.10}$$

Dónde θ es la fase en radianes. La función en MATLAB cos (o sin) se usa para generar secuencias sinusoidales. Por ejemplo, para generar $x(n) = 3cos(0.1\pi n + \pi/3) + 2sin(0.5pn)$, $0 \leq n \leq 10$ se necesita el código siguiente de MATLAB.

```
>> n = [0:10]; x = 3*cos(0.1* pi* n +pi/3) + 2*sin(0.5*pi*n)
```

7. *Secuencias aleatorias*

Muchas secuencias prácticas no se pueden describir por expresiones matemáticas como los casos de arriba. Estas secuencias se llaman secuencias al azar (o estocásticas) y son caracterizadas por parámetros de las funciones asociadas a la densidad de probabilidad o de sus momentos estadísticos. En MATLAB dos tipos (de pseudo) secuencias al azar están disponibles. El $rand(1, N)$ genera una secuencia al azar de la longitud N cuyos elementos se distribuyan uniformemente entre $[0, 1]$. El $randn(1, N)$ genera una secuencia al azar gaussiana de la longitud N con el medio 0 y variación 1. Otras secuencias al azar pueden ser generadas usando transformaciones de las funciones antes mencionadas.

8. *Secuencia periódica*

Una secuencia $x(n)$ es periódica si $x(n) = x(n + N)$ $\forall n$. El número entero más pequeño N que satisfaga la relación anterior se le llama período fundamental. Utilizaremos $\tilde{x}(n)$para denotar una secuencia periódica. Para generar P períodos de $\tilde{x}(n)$ a partir del período $\{x(n),\ 0 \leq n \leq N - 1\}$, se puede copiar $x(n)$ cada P tiempos:

```
>> xtilde = [x, x, ... , x];
```

Pero una aproximación elegante es utilizar las capacidades poderosas de indexación de MATLAB. Primero generamos una matriz que contiene P filas de valores de $x(n)$. Entonces podemos concatenar P filas en un vector fila largo usando la construcción (:), sin embargo, los trabajos de esta construcción solamente funcionan en las columnas. Por lo tanto se tiene que utilizar la transportación matricial del operador ($'$) para proporcionar el mismo efecto en filas.

```
>> xtilde = x' * ones(1, P);      % P columns of x;  X is a row vector
>> xtilde = xtilde (:);           % long column vector
>> xtilde = xtilde';
```

Observe que las ultimas dos líneas pueden combinarse en una para condensar la codificación [35].

2.4.1. Operaciones en secuencias

Aquí describiremos brevemente las operaciones de las secuencias básicas y sus equivalentes en MATLAB.

1. *Adición de señales:* Un ejemplo de una simple suma dada por.

$$\{x_1(n)\} + \{x_1(n)\} = \{x_1(n) + x_1(n)\}$$

Es puesta en practica por MATLAB mediante el operador aritmético +. Sin embargo las longitudes de $x_1(n)$ y $x_2(n)$ deben ser igual. Si las secuencias son longitudes desiguales, o si las posiciones de la muestra son diferentes para las secuencias de longitud iguales, entonces no podemos utilizar directamente al operador +. Se tiene primero que aumentar $x_1(n)$ y $x_2(n)$ de modo que tengan el mismo vector n (y por lo tanto la misma longitud) de la posición. Esto requiere la atención cuidadosa a las operaciones de la indexación de MATLAB. En particular, la operación lógica de la intersección &, las operaciones relacionadas como <= y == y la función *find* se requieren para hacer $x_1(n)$ y $x_2(n)$ de longitud igual. La función siguiente, llamada *sigadd* demuestra estas operaciones.

MATLAB 2.3: Función para sumar secuencias de longitudes iguales.

```
function [y, n] = sigadd (x1, n1, x2, n2)
% implementa y(n)= x1(n) + x2(n)
%-------------------------------------------
% [y, n] = sigadd (x1, n1, x2, n2)
% y = Secuencia de suma sobre n, la cual incluye n1 y n2
% x1= Primera secuencia sobre n1
% x2= Segunda secuencia sobre n2 (n2 puede ser diferente de n1)
%
n=min(min(n1),min(n2)):max(max(n1),max(n2)); % duración de y(n)
y1= zeros (1, length(n)); y2 = y1;           % inicialización
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;   % x1 con duración de y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;   % x2 con duración de y
y = y1+y2;
```

2. *Multiplicación de señales:* Esto es una muestra de una multiplicación (o la multiplicación "punto") dada por.

$$\{x_1(n)\}.\{x_1(n)\} = \{x_1(n)x_1(n)\}$$

Es puesta en practica por MATLAB mediante el arreglo de operadores .*. Otra vez las mismas restricciones que se solicitan en el operador .* se usan para el operador +. Por lo se ha, desarrollado la función *sigmult*, que es similar a la función *sigadd*.

MATLAB 2.4: Función para multiplicar dos funciones de longitud igual.

```
function [y, n] = sigmult (x1, n1, x2, n2)
% implementa y(n)= x1(n) * x2(n)
%-------------------------------------------
% [y, n] = sigmult (x1, n1, x2, n2)
% y = Secuencia del producto sobre n, la cual incluye n1 y n2
% x1= Primera secuencia sobre n1
% x2= Segunda secuencia sobre n2 (n2 puede ser diferente de n1)
%
n=min(min(n1),min(n2)):max(max(n1),max(n2)); % duración de y(n)
y1=zeros(1,length(n)); y2=y1;                % inicialización
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;   % x1 con duración de y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;   % x2 con duración de y
y = y1.*y2;
```

3. *Escalamiento:* En esta operación cada muestra es multiplicada por un escalar α.

$$\alpha\{x(n)\} = \{\alpha x(n)\}$$

Una operación aritmética * se utiliza para poner en práctica la operación de escalamiento en MATLAB.

4. *Desplazamiento:* En esta operación cada $x(n)$ es desplazado por la cantidad k para obtener una secuencia $y(n)$ desplazada.

$$y(n) = \{x(n-k)\}$$

Si dejamos $m = (n-k)$, entonces $n = m + k$ y la operación antedicha se da por

$$y(n+k) = \{x(m)\}$$

Por lo tanto esta operación no tiene ningún efecto en el vector x, pero el vector x es cambiado por la adición de k en cada elemento esto se demuestra en la función $sigshift$

MATLAB 2.5: Función para desplazar el vector x.

```
function [y, n] = sigshift (x, m, n0)
% implementa y(n) = x(n - n0)
%---------------------------------------
% [y, n] = sigshift (x, m, n0)
%
n = m + n0;  y = x;
```

5. *Imagen espejo:* En esta operación que cada muestra de $x(n)$ se refleja alrededor $x = 0$ para obtener $y(n)$ de la secuencia reflejada.

$$y(n) = \{x(-n)\}$$

En MATLAB esta operación es puesta en practica por la función $fliplr(x)$ para los valores de la muestra y por la función $fliplr(n)$ para la posición de la muestra según lo demostrado en la función $sigfold$.

MATLAB 2.6: Función para reflejar la función $x(n)$.

```
function [y, n] = sigfold (x, n)
% implementa y(n) = x(-n)
%---------------------------------------
% [y, n] = sigfold (x, n)
%
y = fliplr(x);    n = - fliplr(n);
```

6. *Adición simple:* Esta operación diferencia de la operación de la adición de señales. Agrega todos los valores de la muestra de $x(n)$ entre n_1 y el n_2.

$$\sum_{n=n_1}^{n_2} x_n = x(n_1) + ... + x(n_2)$$

Es puesta en práctica por la función `sum (x (n1:n2))`.

7. *Productos entre muestras:* Esta operación también diferencia de la operación de la multiplicación de señales. Multiplica todos los valores de las muestras de $x(n)$ entre n_1 y el n_2.

$$\prod_{n=n_1}^{n_2} x_n = x(n_1) \times ... \times x(n_2)$$

Es puesta en practica por la función `prod(x(n1:n2))`.

8. *Energía de la señal:* La energía de la secuencia $x(n)$ se da por.

$$\varepsilon_x = \sum_{-\infty}^{\infty} x_n \cdot x(n) = \sum_{-\infty}^{\infty} |x(n)|^2$$

Donde el súper índice $*$ denota la operación de la conjugación compleja. La energía de una secuencia finita de duración $x(n)$ puede ser calculada en MATLAB usando.

```
>> Ex = sum (x  .*  conju(x));    % una aproximación
>> Ex = sum (abs(x) .^2);         % otra aproximación
```

9. *Señal de potencia:* El promedio de la potencia de una secuencia periódica con periodo fundamental N esta dada por [35]:

$$p_s = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2$$

2.5. Conversión analógica/digital y digital/analógica

En este mundo complejo toda clase de señales y en varias formas nos rodean. Algunas de las señales son naturales, pero muchas mas señales son artificiales. Algunas señales son necesariamente (el habla), algunas son agradables (música), mientras que muchas son indeseadas o innecesarias en una situación dada. En compañía de la ingeniería, las señales son portadoras de información útil e indeseada. Por lo tanto, realzar y extraer la información útil de una mezcla de información que está en conflicto es la forma más simple de procesamiento de señales. Más generalmente, el procesamiento de señales es una operación diseñada para extraer, realzar, almacenar, y transmitir la información

útil. Por lo tanto, el procesamiento de señales tiende a ser aplicación dependiente [35]. Para procesar señales análogas por medios digitales es necesario convertirlas en formato digital, esto es, transformarlas en secuencias de números finitos. Este procedimiento se denomina *conversión análoga digital (A/D)* y los dispositivos correspondientes son *conversiones A/D (ADC's)*. Conceptualmente la conversión análoga/digital se puede ver como un proceso de tres pasos, como se muestran en la Figura 2.8.

1. *Muestreo*: Esta se refiere a la conversión de la señal en tiempo continuo a una señal en tiempo discreto mediante las muestras obtenidas de la señal continua en instantes de tiempo discreto. Así, $x_a(t)$ es la entrada al muestreador, la salida es $x_a(nT) \equiv x(n)$, donde T es el *intervalo de muestreo*.

2. *Cuantificación*: Es la conversión de una señal en tiempo discreto con valores continuos a una señal en tiempo discreto con valores discretos (señal digital). El valor de cada muestra de la señal se representa mediante un valor seleccionado de un conjunto de valores posibles. La diferencia entre la muestra sin cuantificar $x(n)$ y la salida cuantificada $x_q(n)$ se denomina error de cuantificación.

3. *Codificación*: En el proceso de codificación, cada valor discreto $x_q(n)$ se representa mediante secuencia binaria de b bits.

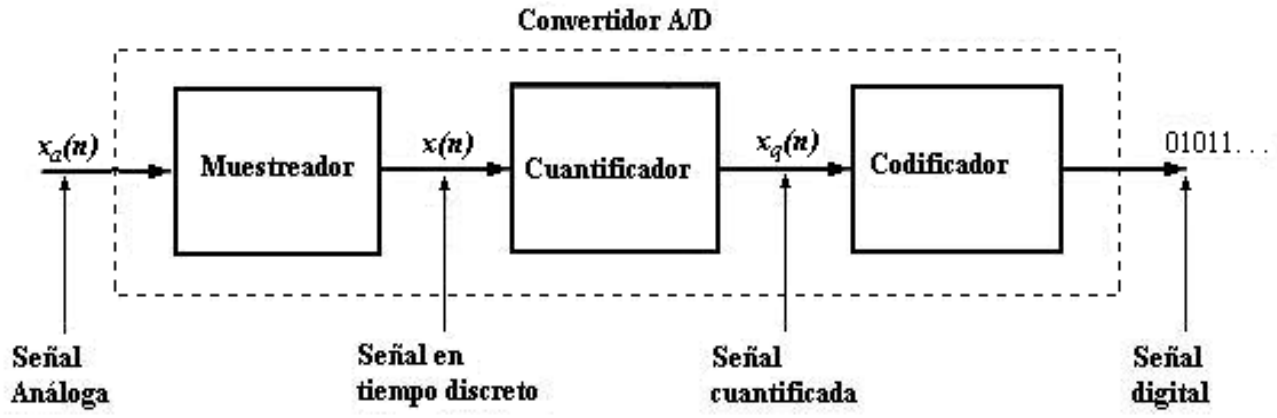

Figura 2.8: Componentes básicos del convertidor análogo/digital (ADC).

Mientras que el proceso contrario se denomina *conversión digital/analógica A/D* en el cual, el dispositivo para este proceso es conocido como convertidor digital analógico (DAC's), y trabaja mediante la interpolación de muestras.

Desde el punto de vista de la programación, el convertidor más simple es el llamado retenedor de orden cero que se muestra en la Figura 2.9 y que simplemente mantiene el valor constante de una muestra hasta que recibe la siguiente. Aunque, se pueden obtener mejoras usando la interpolación lineal para interconectar las muestras de con

segmentos de línea recta. Utilizando técnicas de orden superior se pueden obtener una mejor interpolación [28].

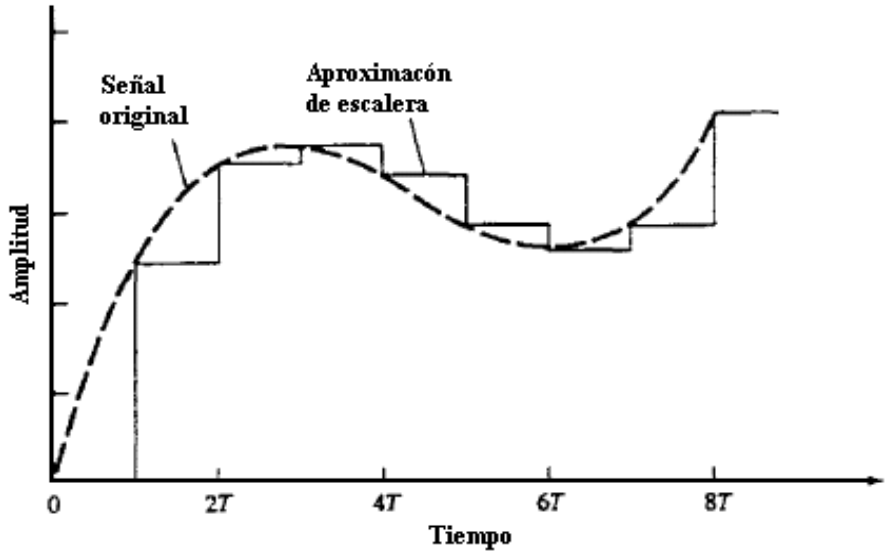

Figura 2.9: Convertidor digital/análogo (Retenedor de orden cero).

2.6. Representación digital de señales multidimensionales (imágenes)

Como se ha visto en los temas anteriores las imágenes son señales de tipo multidimensional. Por lo que a partir de este tema se empezara a tratar temas relacionados con estas señales ya que son la parte primordial de este trabajo.

El termino *imagen monocromática* o solamente *imagen* se refieren a una función bidimensional de luz $f(x, y)$ donde x y y representan coordenadas espaciales y el valor f es cualquier punto (x, y) del brillo o nivel de gris de la imagen en ese punto. La Figura 2.2 muestra el convenio de ejes que se usan para representar una señal bidimensional (imagen). De vez en cuando se vuelve útil representar una imagen en perspectiva de un tercer eje que represente el brillo, esta aparecería como una superficie de picos estrechos con numerosos cambios en los niveles de brillo otras mas suaves en donde los niveles de brillo varíen muy poco o incluso no varíen. De esta forma, asignando valores proporcionalmente más grandes a zonas más brillantes se lograría que la representación de la altura sea correspondiente al brillo de la imagen [10].

El proceso de conversión es llamado *digitalización* y se muestra en la Figura 2.8. Cada píxel de localización, se muestrea y cuantifica el brillo de la imagen [6]. Una *imagen digital* $f(x, y)$ que ha sido discretizada tanto en la coordenadas espaciales como

en el brillo. Esta puede considerarse como una matriz cuyos índices de fila y columna identifican un punto de la imagen y el valor correspondiente de la matriz indica el nivel de gris de ese punto. Estos elementos de la distribución digital se denominan *elementos de la imagen* o también llamados *píxeles* o *pels* [10].

2.7. Terminología de procesamiento digital de imágenes

Puesto que las imágenes se presentan en varias formas, unas visibles y otras no, unas abstractas y otras de manera física, unas son adecuadas para su análisis y otras no. debido a esto es importante mencionar algunos de los términos más comunes que se emplean en procesamiento digital de imágenes [6].

- *Imagen:* Es una representación, o imitación de un objeto o cosa. Una imagen contiene información descriptiva acerca del objeto que representa, muestra su información de manera que permite al observador visualizar su subjetividad. Una pintura es un tipo de imagen. Webster la define como "la representación hecha mediante la pintura, dibujo, o fotografía, algo lucido, gráfico, una descripción exacta de un objeto o cosa de manera que sugiere una imagen mental o una idea exacta de la misma cosa".

- *Digital:* Esta palabra se relaciona al cálculo por métodos numéricos o unidades discretas.

- *Imagen digital:* Se define como la representación numérica de un objeto, los píxeles son unidades discretas y la escala de grises cuantificada suple la componente numérica.

- *Procesamiento:* Es el acto de someter algo a proceso. Un proceso son series de acciones u operaciones para obtener el resultado deseado.

- *Procesamiento digital de imágenes:* Es una representación numérica de un objeto a una serie de operaciones en orden de mantener el resultado deseado. En el caso de imágenes este cambia su forma para hacerlas mas deseables o atractivas. Comienza con una imagen produce una versión modificada de esa imagen

- *Gráficas computacionales:* Se refiere al procesado y despliegue de imágenes de cosas que existen conceptualmente o de descripciones matemáticas. El aspecto principal es siempre la generación de una imagen a partir de un modelo que describe al objeto.

- *Vision computacional:* Se refiere al desarrollo de sistemas que pueden interpretar el contenido de escenas naturales.

- *Asimilación digital:* Este término se refiere en abarcar cualquier manipulación de la imagen relacionada con los datos por computadora. Esto se refiere también a las gráficas de computadoras, su visiona si como el procesamiento digital y su análisis.

- *Digitalización:* Es el proceso de convertir una imagen de su forma original a su forma digital.

- *Display:* Es la generación de una imagen visible de una imagen digital.

- *Escaneo:* Esto es la dirección selectiva de localizaciones especificas en el dominio de la imagen.

- *Píxel:* Llamado también elemento de la imagen, se refiere a cada una de las subregiones de dirección en el proceso de escaneo.

2.8. Etapas fundamentales de procesamiento digital de imágenes

El tratamiento digital de señales se comprende de varias etapas fundamentales para realizar el procesamiento digital de una imagen. Por lo que en esta sección se enfocara a las técnicas para el procesamiento de imágenes y la siguiente sección se enfocaran hacia el hardware que se necesita para ello. Por lo que en la Figura 2.10 se muestra un diagrama a bloque de estas sencillas etapas.

- *Adquisición de imágenes:* Es decir la adquisición de una imagen digital. Para ello se necesita un sensor de imágenes y la posibilidad de digitalizar la señal producida por el sensor.

- *Procesamiento:* La función básica del procesamiento es la de mejora la imagen de forma que se aumenten la posibilidades de éxito en los procesos posteriores. Tratando típicamente de las técnicas para mejorar el contraste, eliminar el ruido, etc.

- *Segmentación:* Consiste en partir una imagen de entrada en sus partes constituyentes u objetos. Esto se refiere al reconocimiento de caracteres, extrayendo los caracteres individuales y palabras de fondo.

- *Representación:* A la salida de la segmentación se obtienen datos de píxel en bruto, por lo que es necesario convertir los datos a una forma adecuada para el procesamiento de imágenes por una computadora. Lo primero es observar si los datos corresponden a un contorno o a una región completa. Sin embargo, para algunas aplicaciones ambas representaciones coexisten,

Figura 2.10: Etapas fundamentales de procesamiento digital de imágenes.

- *Descripción:* También llamada selección de rasgos, consiste en extraer rasgos con alguna información cuantitativa de interés o que sean fundamentales para diferenciar una clase de objetos de otra.
- *Reconocimiento:* Es el proceso que asigna una etiqueta a un objeto basándose en la información proporcionada.
- *Interpretación:* Implica asignar significado a un conjunto de objetos reconocidos

Aunque no se ha dicho nada sobre la *base del conocimiento* con la interacción con las demás etapas que se muestran en la Figura 2.10. Esto es el conocimiento sobre un dominio del problema esta codificado como un sistema de procesamiento de imágenes como una base de datos de conocimiento. Este conocimiento puede ser tan simple como detallar las regiones de una imagen donde se tiene un interés y tan compleja como manejas una base de datos de imágenes. Además guía las operaciones de cada etapa del procesamiento, controlando la interacción entre ellas. La representación de la

Figura 2.10 indica que la comunicación entre las etapas del procesamiento se basa en el conocimiento previo de cómo debe de ser el resultado [10].

2.9. Elementos de procesamiento digital de imágenes

En el nivel más básico, el procesamiento digital de imágenes requiere una computadora la cual pueda procesar imágenes y dos piezas de equipamiento especial: un digitalizador de imagen y un display para mostrar dicha imagen.

En su forma natural las imágenes no son directamente tratables en el análisis computacional puesto que esta trabajan con datos numéricos (mejor que con los pictóricos) por lo que una imagen debe de ser convertida a datos numéricos antes del ser procesado por una computadora.

La Figura 2.11 nos muestra el sistema completo para el procesamiento digital de imágenes. En el cual el procesamiento digital producido por el digitalizador entra dentro de un almacenamiento temporal en un dispositivo adecuado. En repuesta a las instrucciones del operador, la computadora llama y ejecuta el procesamiento digital de la imagen. Durante la ejecución, la señal de entrada se lee en la computadora línea por línea, generando una imagen de salida, píxel por píxel, y almacenándola en un dispositivo de almacenaje línea por línea. Los pasos del procesamiento están limitados solamente por la imaginación y paciencia del programador. Finalmente cuando la imagen se muestra en pantalla lo hace por medio de una digitalización en reversa [6].

Dentro de estos elementos se denotan cinco elementos fundamentales los cuales se describen a continuación:

2.9.1. Adquisición de imágenes

Para la adquisición de imágenes se debe tener en cuenta dos elementos. El primero de estos es un dispositivo físico sensible a una determinada banda del espectro de energía electromagnética y que produzca una señal eléctrica de salida proporcional al nivel de energía proporcionado. El segundo, es llamado *digitalizador*, el cual se usa para convertir la señal de salida del sistema sensible a forma digital. Entre la amplia categoría de sensores de luz visible e infrarrojos más frecuentemente empleados para este fin se encuentran los microdensotómetros, diseccionadores de imágenes, cámaras vidicom, las matrices de detectores fotosensibles de estado sólido y las matrices de elementos de carga acoplada.

2.9.2. Almacenamiento

Proporcionar la capacidad de almacenamiento suele ser un reto en el diseño de sistemas de procesamiento digital de imágenes. En esta área el almacenamiento digital para aplicaciones de procesamiento digital de señales la se divide en tres categorías principales:

1) Almacenamiento a corto plazo, para ser empleado durante el procesamiento. Entre los dispositivos que se pueden usar se encuentran memorias a corto plazo o memorias temporales las cuales almacenan una o más imágenes a las que se puede acceder con rapidez pero están limitadas por el tamaño físico de la tarjeta y su capacidad de almacenamiento

2) Almacenamiento en línea, para una reutilización relativamente rápida. Para este tipo de almacenamiento se suelen usar discos magnéticos o discos magneto-ópticos con capacidades de hasta 1 Gbyte.

3) Almacenamiento en archivo caracterizado por su acceso poco frecuente. Este tipo de almacenamiento se caracteriza por ser de tipo masivo.

2.9.3. Procesamiento

Esta parte se refiere al procesamiento como los procedimientos que normalmente se expresan en forma de algoritmos, así como la adquisición de imágenes y su representación que pueden ser implementadas en software. En este caso el hardware solo nos sirve para una mayor velocidad en algunas aplicaciones del procesamiento de imágenes. El tratamiento de imágenes se caracteriza por brindar soluciones específicas.

2.9.4. Comunicación

Este elemento implica, comunicaciones locales entre sistemas de procesamiento de imágenes y comunicaciones entre dos puntos, es decir, comunicación de los datos de las imágenes. Para los cuales se emplean técnicas de compresión y descompresión de imágenes.

2.9.5. Presentación

Para este elemento los monitores constituyen los principales dispositivos de presentación que se utilizan en los sistemas modernos de procesamiento de imágenes. Los monitores están gobernados por las salidas de una placa de hardware ubicada en la computadora principal. Las señales de salida del módulo de visualización pueden emplearse como entrada de un módulo de grabación de imágenes que produzca una copia impresa de lo que se presenta en la pantalla del monitor. Es decir diapositivas, transparencias o fotografías [10]. En la Figura 2.11 se muestra los elementos funcionales básicos de un sistema de procesamiento digital de imágenes.

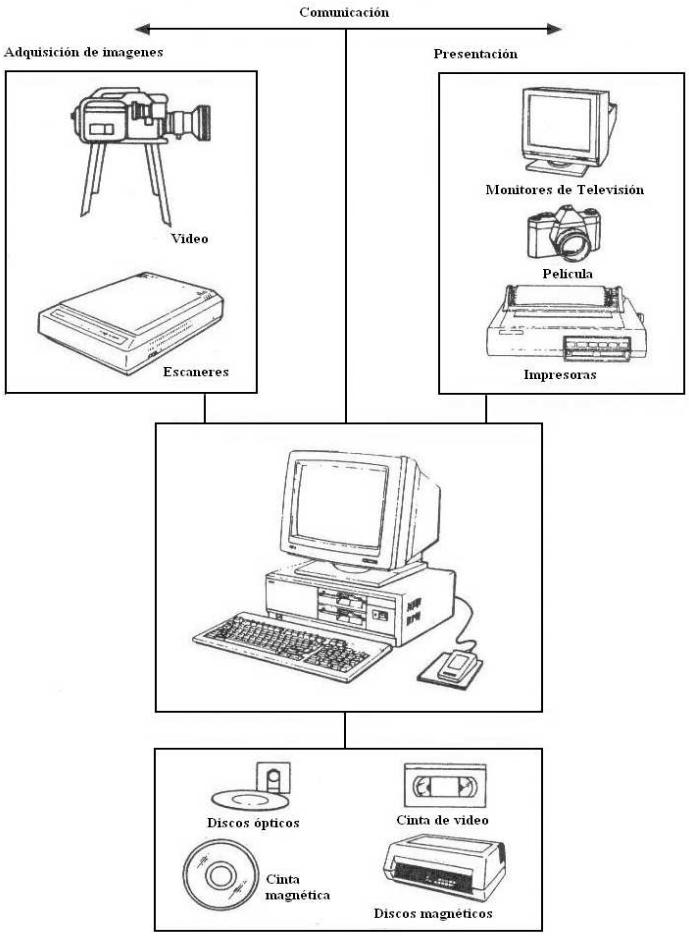

Figura 2.11: Elementos funcionales básicos de un sistema de procesamiento digital de imágenes.

Capítulo 3

ANÁLISIS FRECUENCIAL Y TRANSFORMADAS DE FOURIER

3.1. Introducción

La transformada de Fourier es una de las herramientas útiles en el diseño y análisis de sistemas lineales e invariantes en el tiempo. Estas representaciones implican básicamente su descomposición en componentes sinusoidales (exponenciales complejas). Con esto se dice que la señale esta representada en el *dominio de la frecuencia*.

La mayor parte de la señales de interés práctico se pueden descomponer en la suma de componentes sinusoidales. Para las señales periódicas la descomposición se denomina *serie de Fourier*, para las señales de energía finita la descomposición se denomina *transformada de Fourier* [28]. Del análisis de Fourier se sabe que una función periódica (secuencia) a veces puede ser representada por una combinación lineal de exponenciales complejas armónicamente relacionadas (las cuales son una forma de muestreo). Esto nos da la representación de *series de Fourier discretas (DFS)*. Entonces se extiende la DFS a secuencias de duración finita, las cuales conducen a una nueva transformada llamada *transformada discreta de Fourier (DFT)*. La DFT evita los dos problemas mencionados anteriormente y es una transformada numéricamente computable que es adecuada para implementación en la computadora. Estudiamos sus propiedades y sus usos en detalle en el análisis de sistemas. El cálculo numérico de la DFT para secuencias largas es prohibitivamente una perdida de tiempo. Por consiguiente se desarrolla un cálculo eficiente de la DFT. Estas son llamadas algoritmos de *transformadas rápidas de Fourier (FFT)*. Se estudian estos dos últimos algoritmos en detalle [35].

Comenzaremos nuestro análisis frecuencial de señales con la representación de señales periódicas y aperiódicas en tiempo continuo por medio de las series de Fourier y la transformada de Fourier. Después se dará un tratamiento paralelo con señales periódicas y aperiódicas de señales en tiempo discreto.

3.2. Análisis frecuencial de señales en tiempo continuo

El análisis frecuencial de una señal conlleva la separación de la señal en sus componentes frecuenciales a partir de herramientas fundamentales de análisis de Fourier. Formas de onda tienen diferentes espectros, por ello, este espectro provee una identidad o firma de la señales en el sentido de que ninguna otra señal tiene el mismo espectro. En el tratamiento del análisis frecuencial, se desarrolla las herramientas matemáticas adecuadas para la descomposición de señales en componentes frecuenciales sinusoidales. Además, se desarrolla las herramientas para la síntesis de una señal dada a partir de sus componentes frecuenciales, esto para su representación matemática y gráfica. El terminó *espectro*se emplea al referirse al contenido de la frecuencia de una señal. El proceso de obtención del espectro de una señal dada se le conoce como *análisis espectral o frecuencial* y el proceso de determinar el espectro de una señal práctica a partir de mediciones reales se le conoce como *estimación espectral*. Esta es una distinción muy importante que hay que tomar en cuenta.

3.2.1. Series de Fourier para señales periódicas

La representación matemática básica de las señales periódicas es la serie de Fourier, que es una suma ponderada de sinusoides relacionadas armónicamente. Se recorda que una combinación lineal de exponenciales complejas, armónicamente relacionadas da la forma

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t} \tag{3.1}$$

es una señal periódica de periodo fundamental $T_p = 1/F_0$. De esta manera, podemos considerarlas señales exponenciales

$$\{e^{j2\pi k F_0 t}, \quad k = 0, \pm 1, \pm 2, \ldots\}$$

como los bloques básicos a partir de los cuales pueden construir señales periódicas de diferentes tipos mediante la elección adecuada de frecuencias y de los coeficientes $\{c_k\}$. F_0 determina el periodo fundamental de $x(t)$ y los coeficientes $\{c_k\}$ especifican la forma de onda. Suponiendo que se tiene una señal periódica $x(t)$, la cual podemos representar mediante la serie de (3.1), denominada *serie de Fourier*, cuya frecuencia se elige mediante el inverso del periodo. Para determinar la expresión de los coeficientes $\{c_k\}$. Primero multiplicamos ambos lados de (3.1) por la exponencial compleja

$$e^{-j2\pi l F_0 t}$$

obteniéndose $C_k = \frac{1}{T_p} \int x(t) e^{j2\pi t F_o(t)} dt$.

Un punto importante en la representación de la señal periódica $x(t)$ mediante series de Fourier es, si la serie converge o no a $x(t)$ para todo valor de t, esto es, si la señal $x(t)$ y su serie de Fourier correspondiente

$$\sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t} \tag{3.2}$$

son iguales para todo valor de t. Las denominadas *Condiciones de Dirichlet* garantizan que la serie en (3.2) sea igual a $x(t)$, excepto en aquellos valores de t en los que $x(t)$ es discontinua. Estos valores de t convergen en el valor medio de la discontinuidad. Las condiciones de Dirichlet para señales periódicas son:

1. La señal $x(t)$ tiene un numero finito de discontinuidades en cualquier periodo.
2. La señal $x(t)$ contiene un número finito de máximos y mínimos en cualquier periodo.
3. La señal $x(t)$ es absolutamente integrable en cualquier periodo, esto es,

$$\int_{T_p} |x(t)|dt < \infty \tag{3.3}$$

Todas las señales periódicas de interés práctico satisfacen estas condiciones [28].

3.2.2. Densidad espectral de potencia para señales periódicas

Una señal periódica tiene energía infinita y potencia media finita dada por:

$$P_x = \frac{1}{T_p}\int_{T_p} |x(t)|^2 dt \tag{3.4}$$

Se toma el complejo conjugado de

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{-j2kF_0 t} \tag{3.5}$$

y se sustituye por $x^*(t)$ en en (3.4), obteniendo

$$\begin{aligned} P_x &= \frac{1}{T_p}\int_{T_p} x(t) \sum_{k=-\infty}^{\infty} c_k^* e^{j2\pi k F_0(t)} dt \\ &= \sum_{k=-\infty}^{\infty} c_k^* \left[\frac{1}{T_p}\int_{T_p} x(t) e^{j2\pi k F_0(t)} dt\right] \\ &= \sum_{k=-\infty}^{\infty} |c_k|^2 \end{aligned} \tag{3.6}$$

Por se establece la relación

$$P_x = \frac{1}{T_p}\int_{T_p} |x(t)|^2 = \sum_{k=-\infty}^{\infty} |c_k|^2 \tag{3.7}$$

La cual se denomina *Relación de Parseval* para señales de potencia [28]. Las señales que se obtienen forman un espectro formados por líneas equidistantes que son el inverso de su periodo fundamental T_p como se muestran en la Figura 3.1, la cual muestra los coeficientes de Fourier de un tren de pulsos rectangulares con anchura de pulso τ fija y periodo variable T_p.

De (3.7)se dice que la potencia de la señal se conserva al transformar la señal $x(t)$ al dominio de la frecuencia.

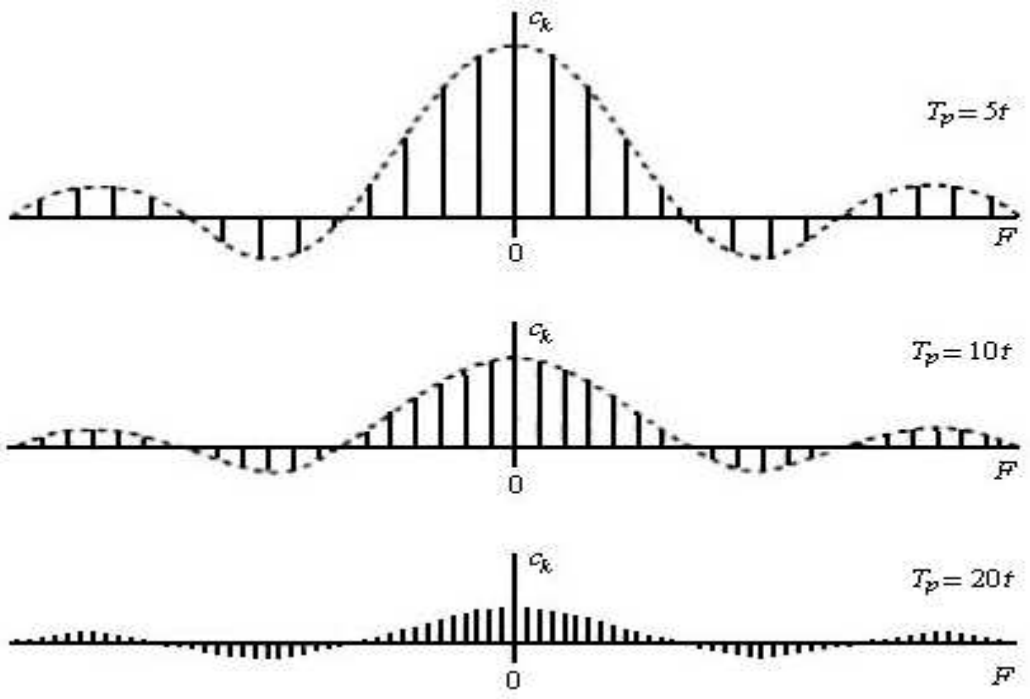

Figura 3.1: Coeficientes de Fourier de un tren de pulsos rectangulares con anchura de pulso τ fija y periodo variable T_p.

3.2.3. Transformada de Fourier continua

Previamente en este capítulo se vio el análisis de Fourier para señales en tiempo continuo y su representación básica mediante la serie de Fourier y la obtención de su espectro. En esta sección se denota la importancia del desarrollo de la transformada de

Fourier en las aplicaciones de procesamiento digital de imágenes así como también la de la obtención de su densidad espectral.

Sea una función continua $f(x)$ de la variable real x. La *Transformada de Fourier* de $f(x)$, indicada por $\mathfrak{F}\{f(x)\}$, se define por la ecuación

$$\mathfrak{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)e^{-2j\pi ux}dx \tag{3.8}$$

donde $j = \sqrt{-1}$[10]. La transformada de Fourier es una transformación lineal integral que, en general, toma una función de n variables reales dentro de otra función que es compleja de n variables reales. La *transformada inversa de Fourier* de $F(u)$ se defina como

$$\mathfrak{F}^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)e^{2j\pi ux}du \tag{3.9}$$

La única diferencia entre las transformaciones de Fourier directa e inversa es el signo del exponente. El teorema integral de Fourier establece que la transformación es reciproca, y

$$\mathfrak{F}\{f(x)\} = F(u) \Rightarrow \mathfrak{F}^{-1}\{F(u)\} = f(x)$$

Las Funciones $f(x)$ y $F(u)$ se llaman *par de transformadas de Fourier*. Para cualquier función $f(x)$, la transformada de Fourier $F(u)$ es única, y viceversa [6]. La transformada de Fourier de una función real es, por lo general, compleja; es decir:

$$F(u) = R(u) + jI(u) \tag{3.10}$$

Donde $R(u)$ y $I(u)$ son las partes real e imaginaria de $F(u)$. A menudo es conveniente expresar (3.8) en forma exponencial, es decir

$$F(u) = |F(u)|e^{j\phi(u)} \tag{3.11}$$

donde

$$|F(u)| = [R^2(u) + I^2(u)]^{1/2} \tag{3.12}$$

y

$$\phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right] \tag{3.13}$$

La función modulo $|F(u)|$ recibe el nombre de *espectro de Fourier* de $f(x)$ y $\phi(u)$ es su *ángulo de fase*. El cuadrado del espectro,

$$\begin{aligned} P(u) &= |F(u)|^2 \\ &= R^2(u) + I^2(u) \end{aligned} \tag{3.14}$$

Se denomina habitualmente *espectro de potencia* de $f(x)$, también llamado *densidad espectral*. La Figura 3.3 muestra diferentes señales con sus espectros de potencia [10].

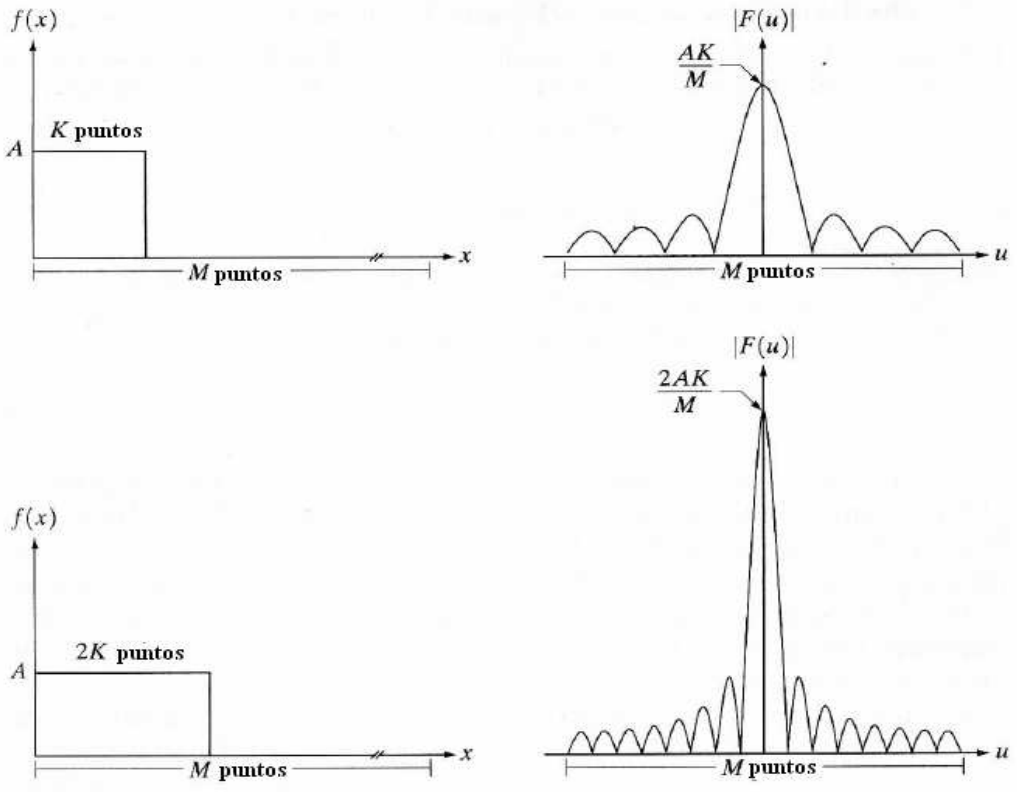

Figura 3.2: Gráficas de una secuencia periódica cuadrada para varios valores de L y N y sus espectros de potencia.

3.3. Análisis frecuencial de señales en tiempo discreto

En la sección anterior se mostró el desarrollo de la serie de Fourier para señales periódicas en tiempo continuo como combinación lineal de exponenciales armónicas complejas armónicamente relacionadas. Pero esta sección se empezará a enfocar al tratamiento digital de imágenes. Y para esto se vera lo que es el análisis de señales en tiempo discreto para así poder ver el conjunto de herramientas que se pueden aplicar para las señales bidimensionales que ocupará el péndulo invertido.

Se ha visto como un sistema lineal e invariante en el tiempo puede ser representado usando su respuesta ante a la muestra de la secuencia unitaria. Esta respuesta, llamada respuesta al impulso unitario $h(n)$, permite calcular la respuesta del sistema a cualquier entrada arbitraria $x(n)$ utilizando la convolución lineal que se muestra a continuación:

$$x(n) \rightarrow h(n) \rightarrow y(n) = h(n) * x(n)$$

Esta representación de la convolución esta basada en el hecho de que cualquier señal puede ser representada por una combinación lineal de muestras unitarias retrasadas y escaladas. Igualmente, también se pueden representar cualquier señal discreta arbitraria como una combinación lineal de las señales básicas introducidas en el capitulo 2. Cada conjunto de señales básicas provee una nueva representación de la señal. Cada representación tiene algunas ventajas y algunas desventajas dependiendo en el tipo de sistema bajo consideración. Sin embargo, cuando el sistema es lineal e invariante en el tiempo, solo la representación sobresale como la más eficaz. Esta basada en el conjunto señales exponenciales complejas $\{e^{ju}\}$ y es llamada Transformada en tiempo discreto de Fourier [35].

3.4. Transformada de Fourier bidimensional

Hace poco, se considero a la transformada de Fourier como una función de una dimensión en el tiempo. En procesamiento digital de imágenes, y en el análisis de sistemas ópticos, comúnmente las entradas y salidas son de dos dimensiones, y en ocasiones son de muchas dimensiones. La aportación en la transformada de Fourier de una dimensión no será una pérdida de tiempo, sin embargo, puesto que la transformada se generaliza para grandes dimensiones.

Por esto se define para funciones de dos dimensiones las transformadas de Fourier directa e inversa, respectivamente como

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dx\ dy \tag{3.15}$$

y

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}du\ dv \tag{3.16}$$

Donde $f(x,y)$ es una imagen y $F(u,v)$ es su espectro. $F(u,v)$ es, en general, una función de valores complejos de dos variables reales de frecuencia u y v [6].

Como en el caso de funciones unidimensionales, el espectro de Fourier, la fase y el espectro de potencia son, respectivamente:

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2} \tag{3.17}$$

$$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \tag{3.18}$$

y

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v) \tag{3.19}$$

Como se observa en la Figura 3.3 de una función bidimensional y su espectro.

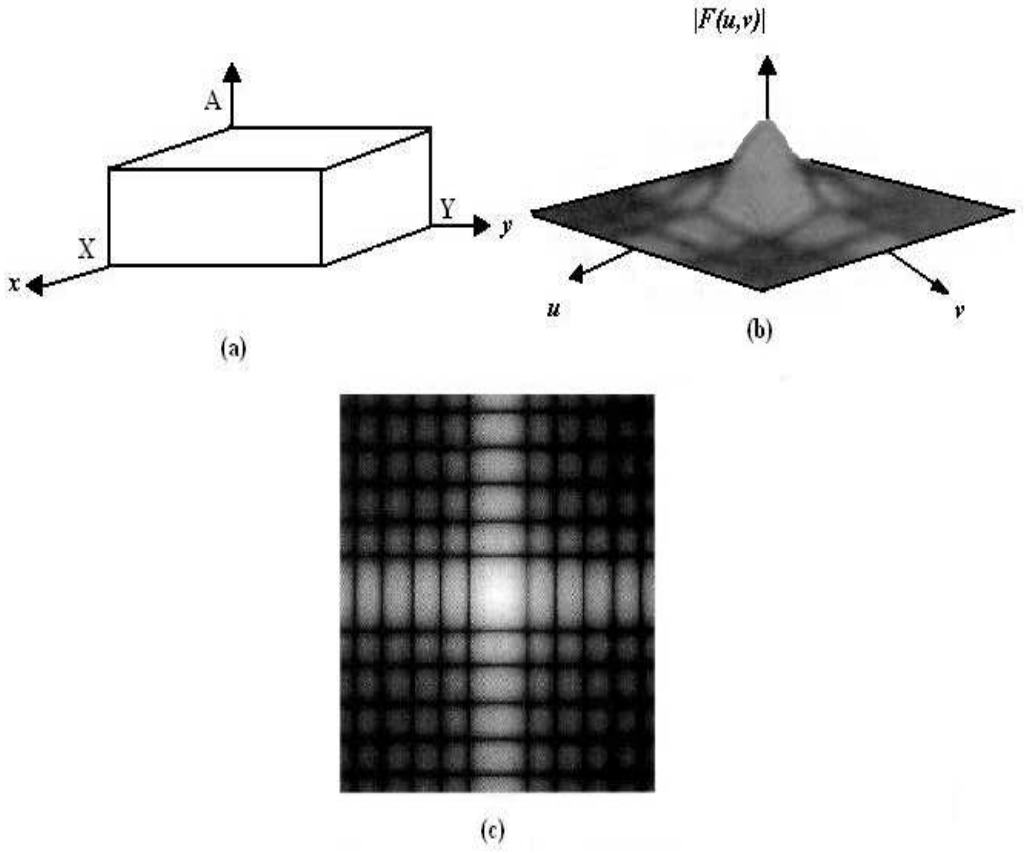

Figura 3.3: (a) Una función bidimensional (2-D), (b) Su espectro de Fourier y (c) Su espectro representado en función de la intensidad.

La cual muestra una representación de esta función en perspectiva multidimensional y el espectro como una función de la intensidad, donde el brillo es proporcional a la amplitud $|F(u, v)|$.

3.5. Transformada de Fourier discreta

Si $f(n)$ es absolutamente sumable, es decir, $\sum_{-\infty}^{\infty} |f(n)| < \infty$, entonces su transformada en tiempo discreto de Fourier esta dada por:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi ux/N} \tag{3.20}$$

La transformada inversa en tiempo discreto de Fourier (IDTFT) de $F(e^{jw})$ esta

dada por:

$$f(x) = \frac{1}{N} \sum_{x=0}^{N-1} F(u) e^{j2\pi ux/N} \tag{3.21}$$

La cual transforma una señal discreta $f(n)$ en una función continua de valores complejos $F(e^{j\omega})$ de variable real ω llamada frecuencia digital, la cual es medida en radianes [35].

En este caso se utilizan dos variables para la cual el par de transformada discreta de Fourier son:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{[-j2\pi(ux/M+vy/N)]} \tag{3.22}$$

para $u = 0, 1, 2 \ldots, M-1$, $v = 0, 1, 2 \ldots, N-1$ y

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{[-j2\pi(ux/M+vy/N)]} \tag{3.23}$$

para $x = 0, 1, 2 \ldots, M-1$, $y = 0, 1, 2 \ldots, N-1$.

Cuando las imágenes se muestrean con la distribución cuadrada $M = N$ y

$$F(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) e^{[-j2\pi(ux+vy)/N]} \tag{3.24}$$

para $u, v = 0, 1, 2, \ldots, N-1$, y

$$f(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) e^{[-j2\pi(ux+vy)/N]} \tag{3.25}$$

Para $x, y = 0, 1, 2, \ldots, N-1$. Puesto que $F(u,v)$ y $f(x,y)$ forman un par de transformadas de Fourier, agrupar esos términos multiplicativos constantes es arbitrario. En la práctica es común que las imágenes se digitalicen en matrices cuadradas, por lo que se tratara frecuentemente con el par de transformadas de Fourier dadas por (3.24) y (3.25). La formulación de (3.22) y (3.23) solo se empleara ocasionalmente, cuando se desee acentuar la importancia de la generalidad del tamaño de la imagen [10].

3.5.1. Propiedades de la DFT

Declaramos la siguientes propiedades sin comprobación:

1. ***Periodicidad:*** La transformada discreta de Fourier $F(e^{j\omega})$ es periódica en ω con periodo 2π.

$$F(e^{j\omega}) = F(e^{j|\omega+2\pi|})$$

Implicación: Necesita solo el periodo de $f(e^{j\omega})$ $(i.e., \omega \in [0, 2\pi],\ o\ [-\pi, \pi], etc.)$ para el análisis y no para el dominio completo $-\infty < \omega < \infty$.

2. ***Simetría:*** Para valores reales $f(n)$, $F(e^{j\omega})$ conjugado simétrico.

$$F(e^{-j\omega}) = F * (e^{j\omega})$$

o.

$$\begin{array}{rcll} Re[F(e^{-j\omega})] & = & Re[F(e^{j\omega})] & \text{simetría par} \\ Im[F(e^{-j\omega})] & = & -Im[F(e^{j\omega})] & \text{simetría impar} \\ |F(e^{-j\omega})| & = & |F(e^{j\omega})| & \text{simetría par} \\ \angle F(e^{-j\omega}) & = & -\angle F(e^{j\omega}) & \text{simetría impar} \end{array}$$

Implicación: Para graficar $F(e^{j\omega})$, ahora necesitamos considerar solamente la mitad del periodo de $X(e^{j\omega})$. Generalmente, en práctica este periodo se escoge para ser $\omega \in [0, \pi]$.
Las dos propiedades anteriores se necesitan para propósitos de la graficación. Ahora se discuten las propiedades útiles restantes, las cuales se dan a continuación sin comprobación. Dada $F(e^{j\omega})$ que es la transformada de Fourier de $f(n)$.

3. **Linealidad:** La transformada en tiempo discreto de Fourier es una transformación lineal; que es.

$$\mathcal{F}[\alpha f_1(n) + \beta f_2(n)] = \alpha\mathcal{F}[f_1(n)] + [f_2(n)] \tag{3.26}$$

Para toda α, β, $f_1(n)$ y $f_2(n)$.

4. **Corrimiento en el tiempo:** Un corrimiento en el dominio del tiempo corresponde al corrimiento de fase.

$$\mathcal{F}[f(n-k)] = F(e^{j\omega})e^{-j\omega k} \tag{3.27}$$

5. **Corrimiento de frecuencia:** La multiplicación por una exponencial compleja corresponde al corrimiento en el dominio de la frecuencia.

$$\mathcal{F}[f(n)e^{j\omega_0 n}] = F(e^{j(\omega-\omega_0)}) \tag{3.28}$$

6. **Conjugación:** La conjugación en el dominio del tiempo corresponde al doblamiento y la conjugación en el dominio de la frecuencia.

$$\mathcal{F}[f^*(n)] = F^*(e^{-j\omega}) \tag{3.29}$$

7. **Doblamiento:** El doblamiento en el dominio del tiempo corresponde al doblamiento en el dominio de la frecuencia.

$$\mathcal{F}[f(-n)] = F(e^{-j\omega}) \tag{3.30}$$

8. **Simetrías en secuencias reales:** Ya hemos estudiado la simetría conjugada de secuencias reales. Estas secuencias reales pueden ser descompuestas en sus partes par e impar

$$f(n) = f_e(n) + f_o(n)$$

Entonces

$$\mathcal{F}[f_e(n)] = Re[F(e^{j\omega})] \tag{3.31}$$

$$\mathcal{F}[f_o(n)] = jIm[F(e^{j\omega})]$$

Implicación: Si la secuencia $f(n)$ es real y par, entonces $F(e^{jw})$ es también real y par. Por lo tanto una gráfica sobre $[0, \pi]$ es necesario para su completa representación.

9. **Convolución:** Esta es una de las más poderosas propiedades que hace conveniente el análisis de sistemas en el dominio de la frecuencia.

$$\mathcal{F}[f_1(n) * f_2(n)] = \mathcal{F}[f_1(n)]\mathcal{F}[f_2(n)] = F_1(e^{j\omega})F_2(e^{j\omega}) \tag{3.32}$$

10. **Multiplicación:** Esta es una dualidad de la propiedad de la convolución.

$$\mathcal{F}[f_1(n) \cdot f_2(n)] = \mathcal{F}[f_1(n)] \circledast \mathcal{F}[f_2(n)] \triangleq \frac{1}{2\pi} \int F_1(e^{j\theta})F_2(e^{j(\omega-\theta)})d\theta \tag{3.33}$$

La operación anterior como operación es llamada *convolución periódica*. Y por lo tanto se denota por $\circledast$.

11. **Energía:**La energía de la señal $f(n)$ se puede escribir como:

$$\begin{aligned} \varepsilon_f &= \sum_{\infty}^{-\infty} |f(n)|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |F(e^{j\omega})|^2 d\omega \\ &= \int_0^{\pi} \frac{|F(e^{j\omega})|^2}{\pi} d\omega \quad \text{para secuencias reales usando simetría par} \end{aligned} \tag{3.34}$$

(3.34) es también conocida como teorema de Parseval. De (3.34) el espectro de densidad de energía de $x(n)$ se define como [35]:

$$\Phi_z(\omega) \triangleq \frac{|F(e^{j\omega})|^2}{\pi} \tag{3.35}$$

La energía de $f(n)$ en la banda de $[w_1, w_2]$ esta dada por:

$$\int_{-\omega_1}^{\omega_2} \Phi_z(\omega) d\omega, \;\; 0 \leq \omega_1 < \omega_2 \leq \pi$$

3.5.2. Propiedades de la DFT bidimensional

En esta sección se mostraran las propiedades de la transformada de Fourier bidimensional que van a ser necesarias para presentaciones posteriores. Aunque el interés principal se centran transformadas discretas bidimensionales, los conceptos subyacentes de algunas de esas propiedades son mas fáciles de comprender cuando se presentan primero en su forma unidimensional.

1. **Separabilidad:** El par de transformadas discretas de Fourier representadas por (3.22) y (3.23) puede representarse en forma separable como:

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1} e^{[-j2\pi ux/N]} \sum_{y=0}^{N-1} f(x,y) e^{[-j2\pi vy/N]} \tag{3.36}$$

y

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1} e^{[-j2\pi ux/N]} \sum_{y=0}^{N-1} F(u,v) e^{[-j2\pi vy/N]} \tag{3.37}$$

La principal ventaja de la propiedad de la separabilidad es que $F(u,v)$ o $f(x,y)$ pueden obtenerse en dos pasos aplicando sucesivamente la transformada de Fourier unidimensional, o su inversa.

2. **Traslación:** Las propiedades de traslación del par de transformadas de Fourier son:

$$f(x,y)e^{[-j2\pi(u_0x+v_0y)/N]} \Leftrightarrow F(u-u_0,v-v_0) \tag{3.38}$$

y

$$f(x-x_0,y-y0) \Leftrightarrow F(u,v) = e^{[-j2\pi(ux_0+vy_0)/N]} \tag{3.39}$$

Donde la doble flecha indica la correspondencia entre una función y su transformada de Fourier y viceversa. De (3.38) se muestra que si se multiplica $f(x,y)$ por el termino exponencial indicado y se toma la transformada del producto, lo que resulta en un desplazamiento del origen del plano de frecuencias al punto (u_0,v_0). Se hace de forma similar para el caso de su transformada inversa.

3. **Periodicidad y simetría conjugada:** La transformada discreta de Fourier y su inversa son funciones periódicas de periodo N; es decir

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N) \tag{3.40}$$

La validez de esta propiedad puede demostrarse mediante la sustitución directa de las variables $(u+N)$ y $(v+N)$ en la ecuación (3.24). Si $f(x,y)$ es real, la transformada de Fourier presenta también simetría conjugada.

$$F(u,v) = F^*(-u,-v) \tag{3.41}$$

o, mas interesante

$$|F(u,v)| = |F(-u,-v)| \tag{3.42}$$

donde $F^*(u,v)$ es el complejo conjugado de $F(u,v)$. La visualización del modulo de la transformada de Fourier con fines interpretativos es a menudo interesante.

4. **Rotación:** Si se introducen las coordenadas polares

$$x = r\cos\theta \quad y = r\sin\theta \quad u = \omega\cos\phi \quad v = \omega\sin\phi \tag{3.43}$$

Entonces $f(x,y)$ y $F(u,v)$ se convierten en $f(r,\theta)$ y en $F(\omega,\phi)$, respectivamente. En otras palabras, haciendo girar $f(x,y)$ un ángulo θ, $F(u,v)$ gira el mismo ángulo. De manera similar, girando $F(u,v)$ se hace girar $f(x,y)$ el mismo ángulo.

5. **Distributividad:**De la definición de transformadas de Fourier continuas o discretas.

$$\mathfrak{F}\{f_1(x,y)+f_2(x,y)\} = \mathfrak{F}\{f_1(x,y)+\mathfrak{F}\{f_2(x,y)\} \tag{3.44}$$

y, en general

$$\mathfrak{F}\{f_1(x,y)\cdot f_2(x,y)\} \neq \mathfrak{F}\{f_1(x,y)\}\cdot\mathfrak{F}\{f_2(x,y)\} \tag{3.45}$$

En otras palabras, la transformada de Fourier y su inversa son distributivas entre si con respecto a la suma pero no a la multiplicación.

6. **Convolución y correlación:** Estas dos se consideran relaciones de la transformada de Fourier que constituyen la unión fundamental entre los dominios espaciales y de frecuencias. Las cuales son de importancia fundamental para la comprensión de las técnicas de procesamiento digital de imágenes.

   - Convolución: El teorema de convolución en dos dimensiones se expresa por la relaciones

   $$f(x,y) * g(x,y) \Leftrightarrow F(u,v)G(u,v) \tag{3.46}$$

   y

   $$f(x,y)g(x,y) \Leftrightarrow F(u,v) * G(u,v) \tag{3.47}$$

   La convolución bidimensional discreta se formúla haciendo que $f(x,y)$ y $g(x,y)$ sean matrices discretas de tamaño $A \times B$ y $C \times D$, respectivamente. De la forma

   $$f(x,y) * g(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)g(x-m, y-n) \tag{3.48}$$

   - Correlación: En el caso bidimensional sean $f(x,y)$ y $g(x,y)$ funciones discretas, su correlación se define como:

   $$f(x,y)^{\circ} g(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m,n)g(x+m, y+n) \tag{3.49}$$

   Una de las principales aplicaciones de la correlación en el procesado de imágenes es en el caso de formatos o ajuste de prototipos, donde el problema consiste en hallar el mayor parecido entre una imagen desconocida y un conjunto de imágenes conocidas.

7. **Muestreo de funciones bidimensionales:** El procedimiento de diseño para estas funciones puede ser formulado matemáticamente haciendo uso de la función impulso bidimensional $\delta(x,y)$ definida como

   $$\int \int_{-\infty}^{\infty} f(x,y)\delta(x-x_0, y-y_0) = f(x_0, y_0) \tag{3.50}$$

   Esta función de muestreo bidimensional consiste en un tren de impulsos separados en Δx en la dirección x y Δy en la dirección y. Donde

   $$\Delta x = x - x_0 \qquad\qquad \Delta y = y - y_0$$

3.6. Transformada corta de Fourier

El análisis de señales ya tiene a su disposición un arsenal impresionante de herramientas. Quizás la más poderosa de estas es el análisis de Fourier, el cual descompone una señal en componentes sinusoidales a diferentes frecuencias. Otra forma de pensar en el análisis de Fourier es en una técnica matemática para cambiar nuestra forma de ver una señal basada en tiempo a una basada en frecuencia. Para muchas señales el análisis de Fourier es extremamente útil porque la frecuencia de la señal contenida es de gran importancia.

En un esfuerzo de corregir las deficiencias del análisis de Fourier, Dennos Gabor (1946) adapto la transformada de Fourier para analizar solo una pequeña sección de la señal en un instante. Esta técnica es llamada ventaneo de la señal. La adaptación de Gabor, llamada transformada en tiempo corto de Fourier (STFT), delinea una señal en una función de dos dimensiones del tiempo y la frecuencia.

LA STFT representa un tipo de compromiso entre las bases del tiempo y la frecuencia de la señal. Provee cierta información acerca de cuando y que frecuencias de la señal ocurren en un evento. Sin embargo solo se puede obtener esta información con una precisión limitada, y esa limitación esta determinada por el tamaño de la ventana. Mientras que el compromiso de la DTFT entre el tiempo y la frecuencia puede ser muy útil, el inconveniente es que una vez que se escoge un determinado tamaño para la ventana, esa ventana es la misma para todas las frecuencias. Muchas señales requieren una aproximación más flexible una donde podamos variar el tamaño de la ventana para determinar mas exactamente el tiempo y/o la frecuencia [37].

3.7. Transformada rápida de Fourier

La DFT introducida hace poco es la única transformada que es discreta tanto como en el dominio de la frecuencia como en el del tiempo, y es definida por sucesiones de duración finita. Aunque es una transformada calculable, la implementación directa es muy ineficiente, especialmente cuando la sucesión de longitud N es muy grande. En 1965 Cooley y Tukey mostraron un procedimiento que substancialmente reduce la cantidad de cálculos involucrados en la DFT [10]. Esto conduce a la explosión de aplicaciones de la DFT, incluyendo el área de procesamiento digital de señales. Por consiguiente esto conduce al desarrollo de otros algoritmos más eficientes. Todos estos algoritmos eficientes son conocidos como algoritmos de la transformada rápida de Fourier [35].

Considere una sucesión de N puntos. Su DFT de N puntos esta dada por

$$F(k) = \sum_{n=0}^{N-1} f(n) W_N^{nk}, \quad 0 \leq k \leq N-1 \tag{3.51}$$

donde $W_N = e^{-j2\pi/N}$. Para obtener una muestra de $F(k)$ se necesitn N multiplicaciones complejas y $(N-1)$ sumas complejas. Por consiguiente para obtener un completo

conjunto de coeficientes de la DFT, necesitamos N^2 multiplicaciones complejas y $N(n-1) \simeq N^2$ sumas complejas. También uno tiene que almacenar N^2 coeficientes complejos de $\{W_N^{nk}\}$ (el cual genera internamente un costo extra). Claramente, el número de cálculos de la DFT para sucesiones de N puntos depende cuadráticamente de N, el cual se denotara por la notación

$$C_N = o(n)^2$$

Para N, o N^2 grandes en la práctica es inaceptable. Generalmente, el tiempo de procesamiento para una suma es menor que para una multiplicación. Así pues desde ahora se concentrara en el número de multiplicaciones complejas, las cuales requieren de 4 multiplicaciones reales y 2 sumas reales.

La descomposición apropiada de la ecuación de la DFT puede hacer que el número de multiplicaciones y sumas sea proporcional a $N \log_2 N$. El procedimiento de descomposición se denomina *Algoritmo de la transformada rápida de Fourier* (FFT, del inglés Fast Fourier Transform). La reducción de N^2 a $N \log_2 N$ operaciones muestra un considerable ahorro en tiempo de cálculo. Evidentemente, la aproximación de la FFT proporciona una apreciable ventaja de cálculo sobre la resolución directa de la transformada de Fourier, especialmente cuando N es demasiado grande [10].

3.8. Comentarios y referencias

El material presentado en este capítulo ha mostrado que el análisis de Fourier por sus muchas características y propiedades es una herramienta importante para el procesamiento de señales e imágenes. El material resumido en este capítulo se obtuvo de los siguientes libros: Proakis y Manolakis [28], Vinay y Proakis [35], Gonzales y Woods [10] y Castleman [6].

Para muchas señales, el análisis de Fourier es extremadamente útil porque las señales contienen frecuencia de alta importancia. Entonces ¿porque necesitamos otras técnicas como el análisis wavelet?. El análisis de Fourier tiene un inconveniente. En transformar al dominio de la frecuencia, la información del tiempo se pierde. Cuando se busca la transformada de Fourier de una señal, es imposible decir cuando un evento en particular esta ocurriendo. Si las propiedades de la señal no varían mucho con el tiempo esto es, si la señal es no estacionaria, este inconveniente no es muy importante. Sin embargo, señales más interesantes contienen características no estacionarias ó transitorias: sentidos, tendencias, cambios abruptos, y comienzos de fines de eventos. Estas características a menudo son las partes más importantes de las señales, y el análisis de Fourier no es suficiente para detectarlos [37]. Por eso el siguiente capítulo tratará el tema de la técnica wavelet que forma el corazón de este trabajo de investigación.

Capítulo 4

WAVELETS U ONDOLETAS

4.1. Introducción

La teoría de wavelet provee un gran bastidor unificado para un número de técnicas las cuales han sido desarrolladas independientemente para varias aplicaciones de procesamientos de señales. Por ejemplo, procesamiento de señales de multiresolución, usadas en visión de computadoras; codificación de sub-banda, desarrollada para la compresión de imágenes y de voz; y la expansión de series de wavelet desarrolladas en aplicaciones matemáticas, han sido recientemente reconocidas como puntos diferentes de una misma teoría.

Por ello, la teoría wavelet cubre un área muy extensa. Que trata los casos de tiempo continuo y también tiempo discreto. Provee de varias técnicas generales que pueden ser aplicadas a muchos problemas en procesamiento de señales, y por lo tanto tiene numerosas aplicaciones potenciales.

En particular, la transformada wavelet es de interés para el análisis de señales no estacionarias, por que provee una alternativa a la clásica transformada de Fourier de tiempo corto (STFT) ó de la transformada de Gabor. Las diferencias básicas son las siguientes. En contraste con la DTFT, las cuales usan un análisis singular de ventanas, la transformada de wavelet usan pequeñas ventanas a bajas frecuencias. Esto es un espíritu de lo que llamamos constante Q o constante relativa del análisis frecuencial del ancho de banda. La transformada wavelet también se refiere al análisis en el tiempo de la frecuencia basado en la distribución Wigner-Ville. Para algunas aplicaciones se desea ver a la trasformada de wavelet como una descomposición de señales en un conjunto de señales básicas. Por ello, las funciones básicas son llamadas *wavelets*. Se obtienen de un prototipo singular de wavelets por dilataciones y contracciones (escalamientos) así como corrimientos. El prototipo wavelet, (llamado wavelet madre) puede ser visto como un filtro pasa bandas, y la propiedad de la constante Q de los otro filtros pasa bandas (wavelets) recae por que son versiones escaladas del prototipo.

Por lo tanto, en la transformada wavelet, la notación de la escala se introduce como una alternativa a la frecuencia induciéndose a lo que se le llama representación de

tiempo-escala. Esto significa que una señal es acotada en un plano de tiempo-escala (el equivalente al plano de tiempo-frecuencia usado en la STFT). Existen varios tipos de transformadas wavelets, y, dependiendo de la aplicación, se prefiere una sobre la otra. Para señales de entrada continua, los parámetros de escala y tiempo pueden ser continuos, conduciendo a la transformada wavelet continua (CWT). Igualmente podría ser discreta conduciendo a la transformada wavelet discreta (DWT). En el último caso utiliza técnicas de procesamiento de señales multi taza y esta relacionado con los sistemas de codificación de sub-bandas utilizadas en compresión de imágenes y voz. Note la analogía con la transformada de Fourier (continua), series de Fourier y transformada de fourier discreta.

La teoría wavelet ha sido desarrollada como un bastidor unificado recientemente, a pesar de eso han tomado lugar ideas similares desde comienzos del siglo. La idea de mirar una señal a varias escalas y analizándola con varias resoluciones han emergido de hecho independientemente diferentes campos de las matemáticas, física, e ingeniería. A mediados de los 80's investigadores de la "escuela francesa"dirigidos por un geofísico, un físico teórico y un matemático (llamados, Morlet, Grossman y Meyer), construyeron fundamentos matemáticos muy fuertes acerca del tema y lo llamaron *Ondeletas* (Wavelets). Que también interactúan con otros campos.

La atención de la comunidad de procesamiento de señales fue atraída cuando Daubechies y Mallat, como suplemento a contribución a la teoría wavelets, establecieron conexiones con los resultados del procesamiento de señales discretas. Desde entonces, un número de contribuciones de teóricas, así como prácticas han sido hechas en varios aspectos de la transformada wavelet, y el tema ha ido creciendo rápidamente. Este capítulo cubre varias de las definiciones básicas y propiedades de la transformada wavelets, muestra las conexiones entre varios campos y muestra aplicaciones en procesamiento de señales. Esto solo para mostrar una vista simple y sintetizada de la teoría wavelet [29].

4.2. Ondas y wavelets

Retomando que la transformada de Fourier usa, como funciones ortonormales básicas, *ondas* sinusoidales, llamadas así por que se parecen a las olas del océano y ondas que se propagan por otros medios. Para una transformada completa, estas funciones se deben extender hasta el infinito en ambos lados. Los vectores básicos de la transformada de Fourier tampoco son cero en el dominio completo, esto es, que no tienen *soporte compacto*.

Tradicionalmente las señales se representan mediante la transformada de Fourier. Una nueva opción es utilizar *ondeletas* (wavelets) en vez de ondas largas. Las wavelets son una alternativa, *no* un reemplazo. Estas nuevas transformadas son más *locales*: encuentran un compromiso entre tiempo y frecuencia.

Por lo que ingenieros y matemáticos han explorado varias transformaciones que tienen funciones básicas de duración limitada. Estas funciones básicas varían de posición así como en frecuencia. Son *ondas* de duración limitada y se refiere a ellas como *wavelets*.

Las transformadas basadas en ellas se llaman *Transformadas Wavelet*. La Figura 4.1 ilustra la diferencia entre ondas y wavelets. Las primeras dos son ondas coseno que difieren en frecuencia pero no en duración, las dos ultimas son wavelets que difieren en frecuencia y posición a lo largo del eje [6].

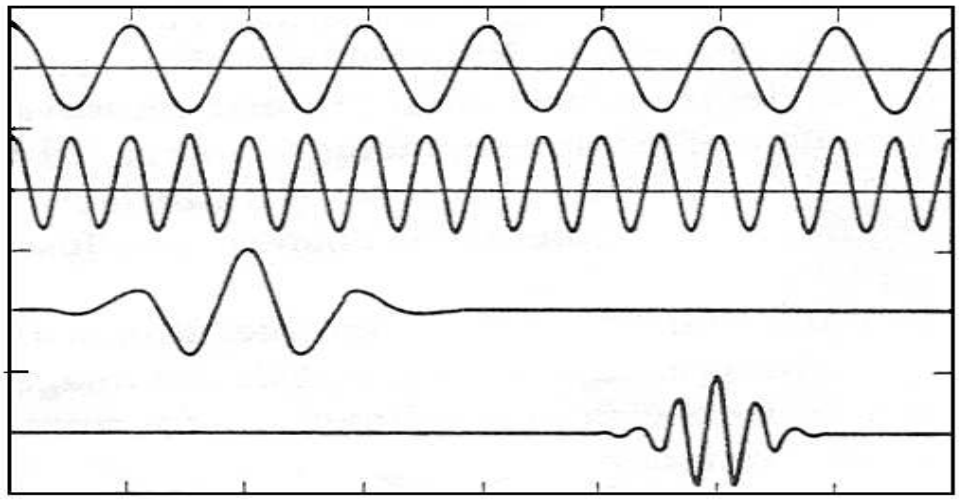

Figura 4.1: Ondas y wavelets

4.2.1. Tipos de Wavelets

Las transformadas wavelets comprenden un amplio conjunto de formas. A lo largo del tiempo se han ido desarrollando diferentes versiones de wavelets, lo que han dado lugar a familias de wavelets. Las diferentes familias de wavelets hacen diferencias entre que tan compactas están las funciones básicas localizadas en el espacio y que tan finas son [14].

Entre las familias de wavelets más comunes se encuentran las siguientes:

- **Wavelet Haar:** El primer participante en la carrera de las wavelet fue un matemático Húngaro llamado Alfred Haar, que introdujo en 1909 las funciones que actualmente se denominan "wavelets de Haar". Estas funciones consisten simplemente en un breve impulso positivo seguido de un breve impulso negativo. Aunque los impulsos breves de las wavelets de Haar son excelentes para la enseñanza de la teoría de las wavelets, no resultan de tanta utilidad en la mayoría de aplicaciones, ya que producen líneas irregulares con picos en lugar de curvas suaves.

- **Wavelet de Morlet:** Morlet, un ingeniero de Elf-Aquitanie, desarrolló su propia forma de analizar las señales sísmicas para crear componentes que estuvieran lo-

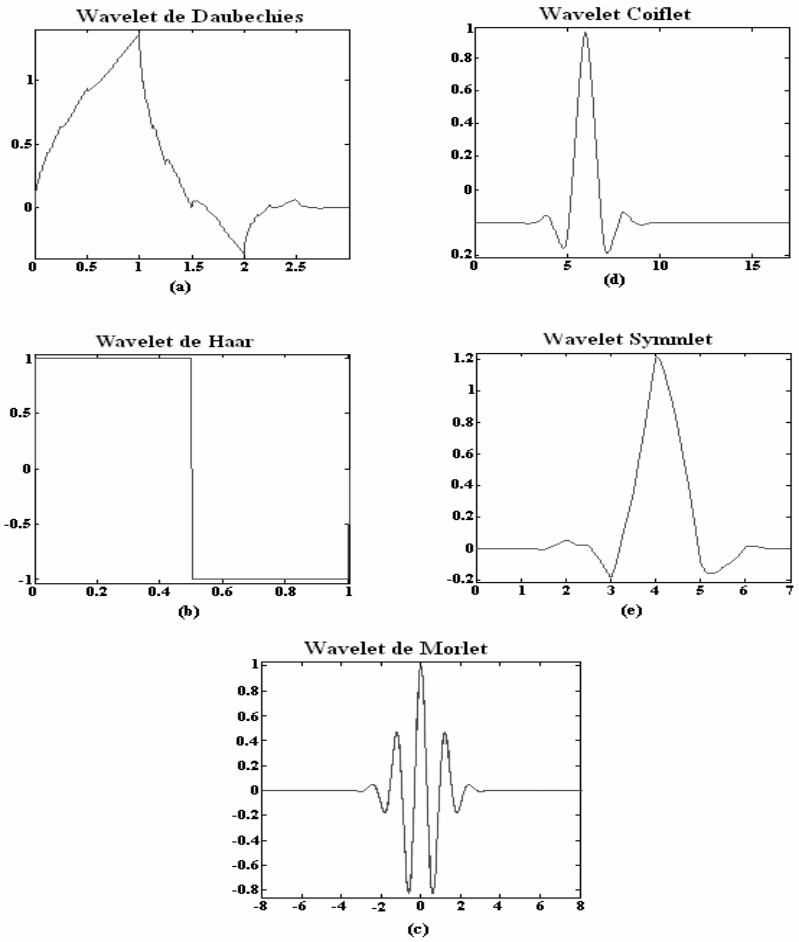

Figura 4.2: Gráficos de varios tipos distintos de wavelets. (a) Wavelet de Daubechies, (b) Wavelet de Haar, (c) Wavelet de Morlet, (d) Wavelet de Coiflet y (e) Wavelet de Symmlet.

calizados en el espacio, a los que denominó "wavelets de forma constante". Posteriormente, se conocerían como wavelets de Morlet. Independientemente de que los componentes se dilaten, compriman o desplacen en el tiempo, mantienen la misma forma. Se pueden construir otras familias de wavelets adoptando una forma diferente, denominada wavelet madre, y dilatándola, comprimiéndola o desplazándola en el tiempo.

- **Wavelet de Daubechies:** La última gran salva de la revolución de las wavelets se disparó en 1987, cuando Ingrid Daubechies, mientras visitaba el Courant Institute de la Universidad de Nueva York y, posteriormente, durante su trabajo en los laboratorios AT&T Bell, descubrió una clase completamente nueva de wavelets, que no sólo eran ortogonales (como las de Meyer) sino que también se podían

implementar mediante sencillas ideas de filtrado digital, de hecho, mediante cortos filtros digitales [38].

- **Otras Wavelets:** Como las de Coiflet y Symmlet son subclases de wavelets que se distinguen por el número de coeficientes y por el número de niveles de iteración. Por ejemplo en la familia de wavelets de Coiflet existen Coiflets con dos momentos de desvanecimiento y otras con tres momentos de desvanecimiento y esta relacionada con el número de coeficientes [14].

Las cuales se muestran en la Figura 4.2

4.3. Análisis en el dominio de la frecuencia

La literatura en procesamiento de señales incluye considerablemente trabajo relativo al análisis de señales en términos de dos dimensiones de espacio *tiempo y frecuencia*. Este procedimiento actualmente precede a la transformada Wavelet, pero ahora ataca dentro del mismo conjunto. Acorde con esto, cada componente de transición de una señal se acota a una posición en el dominio *tiempo-frecuencia* que corresponde a la componente predominante de tiempo y frecuencia.

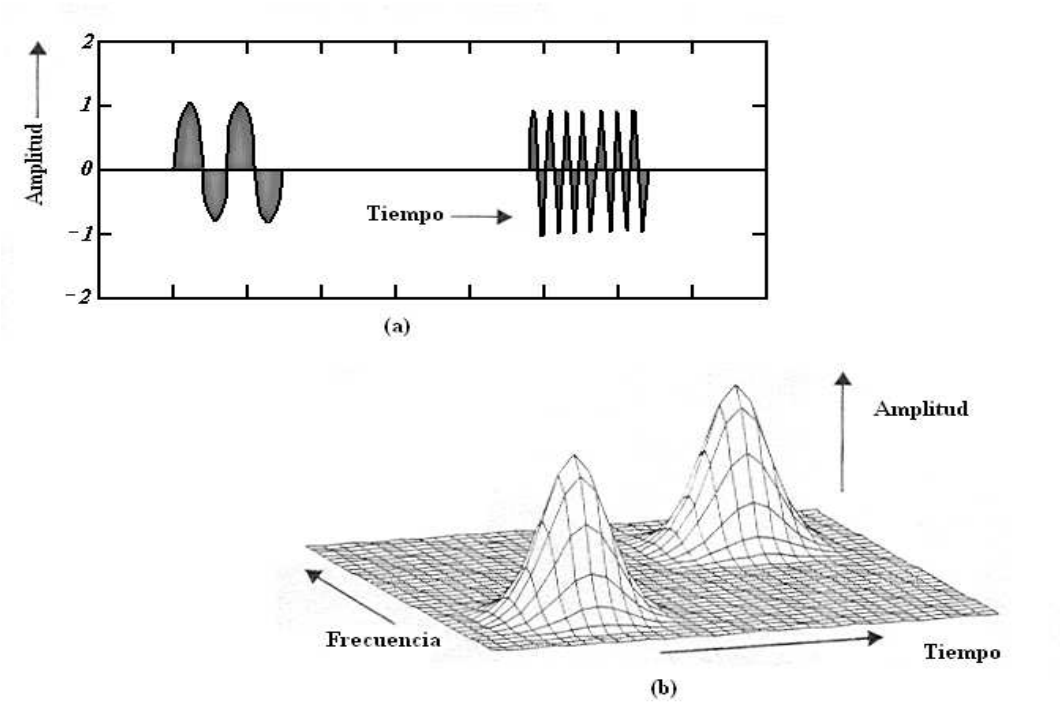

Figura 4.3: Espacio tiempo-frecuencia: (a) Señal, (b) Representación.

En un análisis imaginario, el espacio es de tres dimensiones y puede ser visto como un conjunto de imágenes. Un componente localizado puede aparecer primordialmente en el conjunto que corresponde al componente primordial de la frecuencia. La Figura 4.3 muestra una imagen que contiene dos componentes localizadas que han sido sometidas a dos filtros pasa bandas [6].

4.4. Tipos de transformadas

Dentro del área del procesamiento de señales existen diversas maneras de tratar una señal que en este caso se realiza mediante transformadas entre la cuales se encuentran la transformada de Fourier, la transformada discreta de Fourier y la Transformada rápida de Fourier, las cuales ya han sido analizadas en el capítulo anterior. Pero puesto que la transformada de Fourier y otras transformada similares poseen varias limitaciones en cuanto al procesado de señales, científicos y matemáticos han desarrollado otro tipo de transformadas que prometen mejorar el desarrollo en algunas aplicaciones las cuales reciben el nombre de *Transformadas Wavelets*.

Así como con la transformada de Fourier, existen las tres posibilidades para las transformadas Wavelet: la transformada wavelet continua (CWT), la expansión en series wavelet y la transformada wavelet discreta (DWT). La situación es ligeramente mas complicada, sin embargo, puesto que las funciones básicas wavelet pueden o no ser ortonormales. Un conjunto de funciones básicas de wavelet pueden soportar una transformada si son o no ortonormales [6].

Para señales continuas de entrada, los parámetros de escala y tiempo pueden ser continuos, conduciendo a la transformada wavelet continua (CWT). Igualmente podría ser discreta conduciendo a la transformada wavelet discreta (DWT). En el último caso utiliza técnicas de procesamiento de señales multi taza y esta relacionado con los sistemas de codificación de sub-bandas utilizadas en compresión de imágenes y voz. Note la analogía con la transformada de Fourier (continua), series de Fourier y transformada de fourier discreta [29].

4.5. Expansión en series wavelet

Empezaremos por definir la *Expansión en series wavelet* de una función $f(x) \in L^2(\mathbf{R})$ relativa a la wavelet ψ y la función de escala φ la cual se escribe de la forma:

$$f(x) = \sum_k c_{j_0}(k)\varphi_{j_0,k}(x) + \sum_{j=j_0}^{\infty}\sum_k d_j(k)\psi_{j,k}(x) \tag{4.1}$$

donde j_0 es una escala de comienzo arbitraria y $c_{j_0}(k)$ son normalmente llamados *aproximación* o *coeficientes de escala*; la $d_j(k)$ son referidos como *detalle* o *coeficientes wavelet*.

Esto es porque la primera suma de (4.1) usan funciones de escala que proveen una aproximación a $f(x)$ en la escala j_0 (a menos que $f(x) \in V_{j_0}$ y que sea exacto). para cada escala alta $j \geq j_0$ en la segunda suma, una función de resolución fina, una suma de wavelets, se suma a la aproximación para proveer un incremento en el detalle, si las funciones de expansión forman una base ortonormal o una imagen estrecha, lo cual es siempre el caso, se calculan los coeficientes de expansión como

$$c_{j_0}(k) = \langle f(x), \varphi_{j_0,k}(x) \rangle = \int f(x)\varphi_{j_0,k}(x)dx \tag{4.2}$$

y

$$d_j(k) = \langle f(x), \psi_{j,k}(x) \rangle = \int f(x)\psi_{j,k}(x)dx \tag{4.3}$$

Si la funciones de expansión son parte de una base biortogonal, los terminos ψ y φ deben ser reemplazados por sus funciones duales, $\tilde{\psi}$ y $\tilde{\varphi}$, respectivamente [11].

4.6. Transformada wavelet continua

La transformada wavelet continua (también llamada transformada wavelet integral) fue desarrollada por Grossman y Morlet [6]. La cual transforma una función continua en una función altamente redundante de dos variables, traslación y escala. La transformada resultante es fácil de interpretar y valuable en el análisis tiempo-frecuencia. Aunque nuestro interés es principalmente en señales discretas. La transformada continua wavelet de una función cuadrática integrable, $f(x)$, i.e $\int |f(x)|^2 dx < \infty$ relativa a la wavelet de valores reales $\psi(x)$, es,

$$W_\psi(s,\tau) = \int_{-\infty}^{\infty} f(x)\psi_{s,\tau}(x)dx \tag{4.4}$$

donde

$$\psi_{s,\tau}(x) = \frac{1}{\sqrt{s}}\psi\left(\frac{x-\tau}{s}\right) \tag{4.5}$$

Es un conjunto de funciones básicas wavelet, $\{\psi_{s,\tau}(x)\}$, puede ser generada trasladando y escalándola wavelet básica, $\psi(x)$ y s y τ son llamados parámetros de *escala* y *traslación*, respectivamente. Dada $W_\psi(s,\tau)$ se puede obtener la *transformada continua wavelet inversa*

$$f(x) = \frac{1}{C_\psi}\int_0^\infty \int_{-\infty}^{\infty} W_\psi(s,\tau)\frac{\psi_{s,\tau}(x)}{s^2}d\tau\, ds \tag{4.6}$$

Para la cual

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(u)|^2}{|u|}du \tag{4.7}$$

es el espectro de Fourier y $\Psi(u)$ es la transformada de Fourier de $\psi(x)$, y que además satisface el *criterio de admisibilidad*. La Figura 4.4 muestra la transformada wavelet continua la cual es denominada wavelet de *sombrero mexicano* y el espectro de Fourier de una función de una dimensión [11] [6].

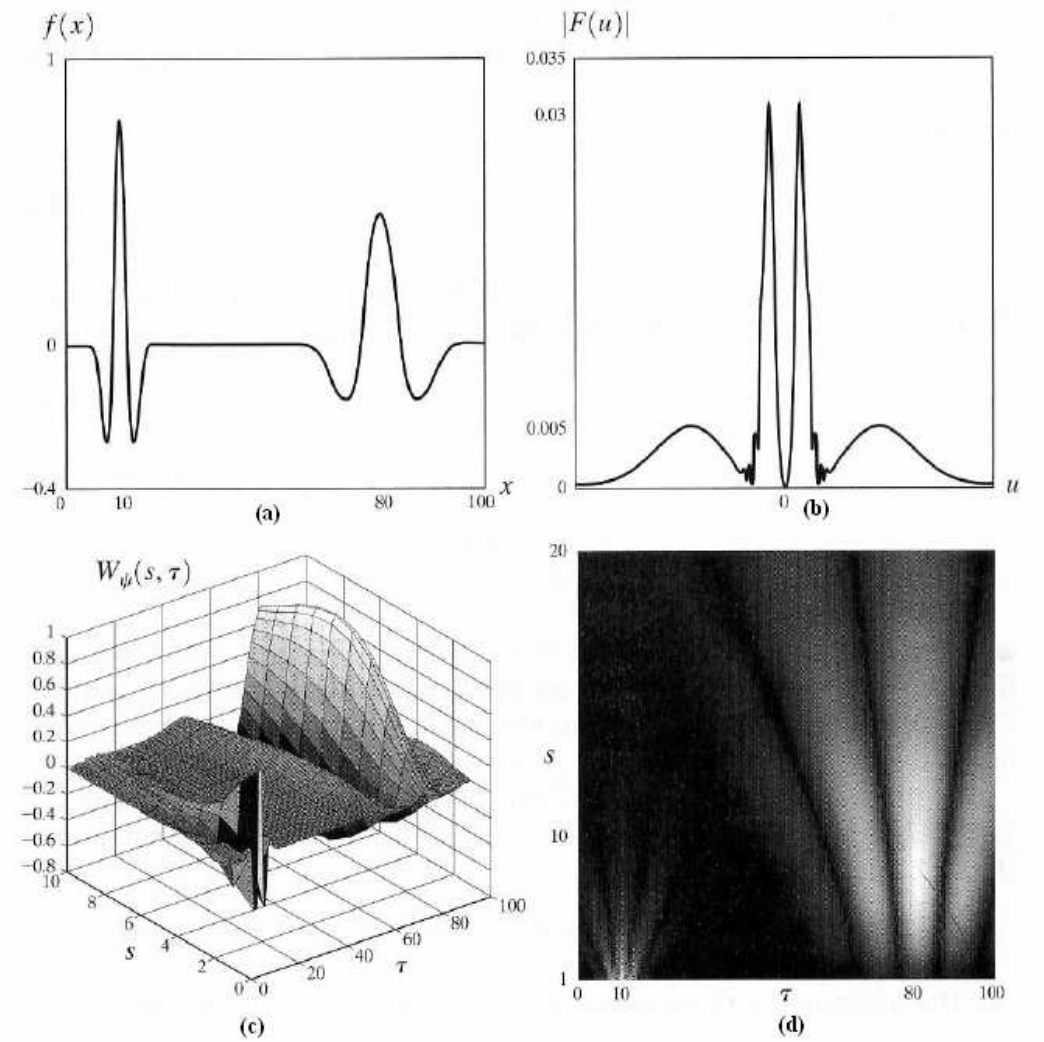

Figura 4.4: Transformada wavelet continua (c y d) Espectro de Fourier (b) Una función de una dimensión (a).

4.6.1. Transformada wavelet continua de dos dimensiones

La transformada wavelet continua $W_\psi(s,\tau)$ de una función de una solo dimensión $f(x)$ es una función de dos variables, una más que la de $f(x)$. La CWT es más que completa, por lo que representa un incremento considerable en la información contenida

y en el volumen requerido para el almacenamiento de datos. Para funciones de más de una variable, esta transformada también incrementa la dimensionalidad por uno. Si $f(x, y)$ es una función de dos dimensiones, su transformada wavelet continua es

$$W_\psi(s, \tau_x, \tau_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\psi_{s,\tau_x,\tau_y}(x, y)dx\, dy \tag{4.8}$$

donde τ_x y τ_y, especifican la traslación en dos dimensiones. La transformada wavelet continua inversa de dos dimensiones es,

$$f(x, y) = \frac{1}{C_\psi}\int_0^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W_\psi(s, \tau_x, \tau_y)\frac{\psi_{s,\tau_x,\tau_y}(x, y)}{s^3}d\tau_x\, d\tau_y\, ds \tag{4.9}$$

donde

$$\psi_{s,\tau_x,\tau_y}(x, y) = \frac{1}{\sqrt{s}}\psi\left(\frac{x - \tau_x}{s}, \frac{y - \tau_y}{s}\right) \tag{4.10}$$

es la wavelet básica de dos dimensiones. La misma generalización se puede extender para funciones de más de dos dimensiones [6].

4.7. Transformada wavelet discreta

La transformada wavelet discreta es muy usada en compresión de imágenes, procesamiento y análisis. Dada un conjunto de funciones básicas ortonormales, se calcula la transformada wavelet discreta, justo como lo haríamos con cualquier otra transformada unitaria, obteniendo una función wavelet básica que sea adecuada. Coma la expansión en series wavelet de la sección anterior delinea una función de una variable continua en una secuencia de coeficientes. Si la función se expande a una secuencia de números, como muestras de una función continua $f(x)$, los coeficientes resultantes son llamados *wavelet discreta*. Por lo que veremos tres técnicas para el desarrollo de la transformada wavelet discreta: (1) Teoría de bancos de filtros, (2) Multiresolución o análisis en tiempo-escala y (3) Codificación de subbanda. Para casos mas simples, las series de expansión definidas anteriormente se vuelve el par de transformadas de la DWT

$$W_\varphi(j_0, k) = \frac{1}{\sqrt{M}}\sum_x f(x)\varphi_{j_0,k}(x) \tag{4.11}$$

$$W_\psi(j, k) = \frac{1}{\sqrt{M}}\sum_x f(x)\psi_{j,k}(x) \tag{4.12}$$

para $j \geq j_0$ y

$$f(x) = \frac{1}{\sqrt{M}}\sum_k W_\varphi(j_0, k)\varphi_{j_0,k}(x) + \frac{1}{\sqrt{M}}\sum_{j=j_0}^{\infty}\sum_k W_\psi(j, k)\psi_{j,k}(x) \tag{4.13}$$

4.7.1. Teoría de bancos de filtros

Investigadores en el área de análisis de voz y procesamiento de señales acústicas han usado por mucho tiempo el concepto de bancos de filtros pasa banda para descomponer la señal en componentes a diferentes frecuencias. Ciertamente, el método es un precursor del *análisis de tiempo-frecuencia*, en la cual los componentes de la señal, en la cual los componentes de la señal se muestran en espacio de dos dimensiones cuyas dimensiones son tiempo de ocurrencia y frecuencia de oscilación. Suponga que se tiene una señal compuesta de dos tonos de quiebre (sinusoides de corta duración) incrustadas en ruido aleatorio. Suponga que se analiza esta señal para detectar el número, frecuencia y posición de los tonos de quiebre. La transformada de Fourier puede reflejar el contenido entero de la señal, pero no siempre es fácil de interpretar. Por ejemplo la información de posición esta codificada en el espectro de fase de forma complicada, mientras que el espectro en amplitud puede mostrar distintos picos debido a cada uno de los componentes transitorios de la señal, por lo que un simple señal frecuencial es insuficiente para resolver los componentes de la señal. Por esto la teoría de bancos de filtros utiliza filtros pasa bandas en paralelo. Las correspondientes salidas son $g_1(x)$, $g_2(x)$, $g_3(x)$. La Figura 4.5 muestra las tres salidas de los filtros pasa bandas. Note que los dos tonos de quiebre emergen de filtros separados. Cada una de las salidas pasa bandas esta formada por la convolución

$$g_i(x) = \int_{-\infty}^{\infty} f(t)h_i(x-t)dt \tag{4.14}$$

Entonces la reflexión en la convolución integral no tiene efecto, y las salidas de los filtros pueden escribirse como

$$g_i(x) = \int_{-\infty}^{\infty} f(t)h_i(t-x)dt = \langle f(t), h_i(t-x)\rangle \tag{4.15}$$

Por lo tanto podemos ver a $\{g_i(x)\}$ como el conjunto de transformadas wavelet (de dos dimensiones), donde $\{h_i(x)\}$ es el conjunto de wavelets.

4.7.2. Multiresolución o análisis en tiempo-escala

Mucho del desarrollo en el análisis wavelet recae en el campo que generalmente es llamado *análisis de multiresolución*. Este desarrollo fue hecho por limitaciones de la transformada de Fourier mencionadas anteriormente. La teoría de filtros de bancos ofrece una aproximación de representación de señales compuesta de componentes oscilatorios, como notas musicales o tonos de quiebre. Sin embargo, en análisis de imágenes los componentes de interés localizados, a menudo no son enteramente oscilatorios y solo incluyen un ciclo o una parte de un ciclo, como líneas, bordes o puntos. Los objetos de una imagen se ven a diferentes tamaños de escala. Las transformadas wavelet se han desarrollado a través de líneas de multiresolución. En el análisis tiempo-frecuencia, una señal se representa en el espacio de dos dimensiones, pero aquí el eje vertical es la escala

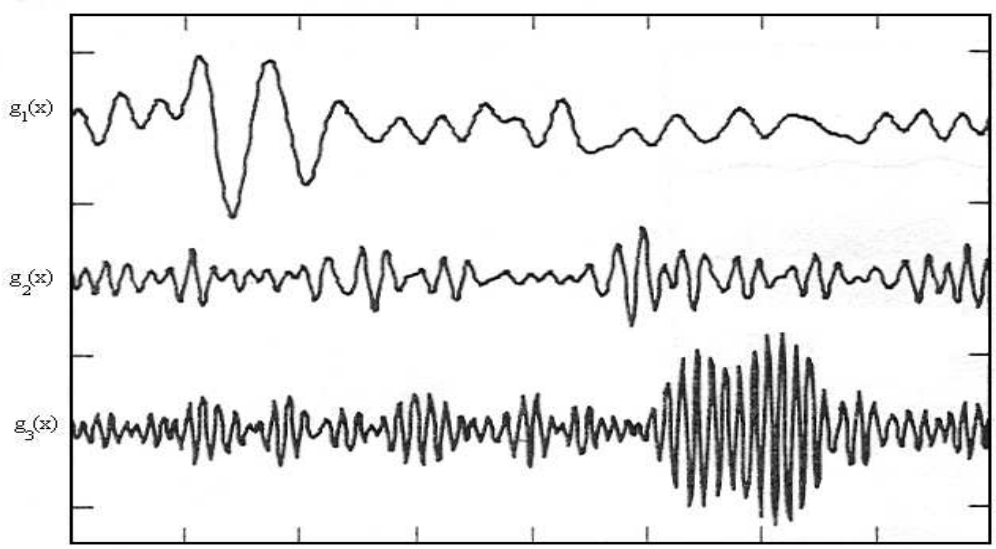

Figura 4.5: Salidas del filtro pasa bandas

antes que frecuencia. La escala se logra dilatando y contrayendo la wavelet básica para formar un conjunto de funciones básicas. La wavelet básica, $\psi(x)$, es escalada como $\psi(x/a)$ (la cual se ensancha si $a > 1$ y contraída si $a < 1$) para formar el conjunto de funciones básicas. A gran escala a, las funciones básicas dilatadas se usan para grandes rasgos, mientras que un a pequeña se pretende para pequeños detalles. Algunos métodos para el análisis de multiresolución son: (1) Algoritmos piramidales y (2) Codificación de pirámide Laplaciana.

4.7.3. Codificación de subbanda

Mas allá del fondo de la transformada wavelet discreta, ahora se describe la técnica de tiempo-frecuencia llamada *codificación de subbanda*. Originalmente diseñada para codificación compacta para señales digitalizadas de audio, la codificación de subbanda pretende descomponer la señal (o imagen) dentro de los componentes de la banda estrecha (filtrado del pasa bandas) y la representa, sin redundancia, en la cual una forma posible de reconstruir la señal original sin errores. Dada la señal de banda limitada $f(t)$, que es

$$\mathfrak{F}\{f(t)\} = F(s) = 0 \quad \text{para} \quad |s| \geq s_{\max} \tag{4.16}$$

ss muestrea la señal con un espaciado de muestra uniforma Δt para formar

$$f(i\Delta t) \quad i = 0, 1, \ldots, N-1 \quad s_{\max} \leq S_N = \frac{1}{2\Delta t} \tag{4.17}$$

donde s_N es la frecuencial de Nyquist. Empieza el análisis dividiendo en partes los ejes de frecuencia en subintervalos desarticulados [6].

4.8. Aplicaciones

Quizás el mayor potencial de las wavelets ha sido reclamado en la compresión digital. Puesto que las transformadas wavelet discretas son esencialmente sistemas de codificación de subbanda y puesto que los codificadores de subbanda han sido exitosos en la compresión de voz e imágenes, es claro que las wavelets encontraran inmediata aplicación en sistemas de compresión [29]. Entre las aplicaciones más comunes se encuentran las siguientes:

- *Realce de imágenes:* La DWT descompone una imagen en componentes de diferentes tamaños, posición y orientación. Como el filtrado digital en el dominio de la frecuencia de Fourier, uno puede alterar la amplitud de los coeficientes en el dominio de la transformada wavelet para obtener la transformada inversa. Esto puede selectivamente acentuar los componentes interesantes a expensas de los indeseables

- *Fusión de imágenes:* La fusión de imágenes combinan dos o más imágenes del mismo objeto en una sola imagen que es más fácil de interpretar que cualquiera de las originales. Esta técnica encuentra aplicaciones en interpretación de imágenes multiespectrales, así como imágenes médicas, donde las imágenes de la misma parte del cuerpo son obtenidas por diferentes modalidades de asimilación [6].

- *Visión humana:* En los pasados 1980 David Marr empezó a trabajar en el laboratorio de inteligencia artificial del MIT en visión artificial para robots. Su teoría fue que el procesamiento de imágenes en el sistema visual humano tiene una complicada estructura jerárquica que involucra varias leyes de procesamiento. A cada nivel de procesamiento, el sistema de la retina provee una representación visual, que escala progresivamente de manera geométrica. Sus argumentos se enfocan en la detección de cambios de intensidad. Su teoría fue que los cambios de intensidad ocurren a diferentes escalas en la imagen, por tanto su detección óptima requiere el uso de operadores a diferentes escalas, este operador fue una wavelet que hoy en día es llamada "wavelet de Marr"

- *Compresión de huellas digitales del FBI:* Entre 1924 y hoy el FBI recolecto más de 30 millones de huellas digitales. En 1993 desarrolló estándares para la digitalización y compresión, mediante la digitalización a una resolución de 500 píxeles por pulgada con 256 niveles de gris por píxel. Hasta ahora, la principal aplicación excepcional de las wavelets ha sido la compresión de imágenes digitales. Son el eje central del nuevo estándar de imágenes digitales JPEG-2000 y del método WSQ (del inglés Wavelet Scalar Quantization, cuantización escalar wavelet) que utiliza el FBI para comprimir su base de datos de huellas dactilares.Como se ve en la Figura 4.6.

Figura 4.6: Huella digital de la mano izquierda digitalizada por el FBI.

- *Notas musicales:* Victor Wickerhouser sugirió que un conjunto de wavelets puede ser útil en la síntesis de voz. Su idea era que un generador de paquete de wavelets simples puede reemplazar un gran número de osciladores. A través de la experimentación, un músico puede determinar las combinaciones de los paquetes de ondas que producen sonidos interesantes. Una muestra de notas producidas por un instrumento, puede ser descompuesta en sus paquetes de coeficientes wavelet. Para reproducir las notas se cargan sus coeficientes en un generador de paquetes wavelet para ver el resultado [14]. Como se ve en la Figura 4.7.

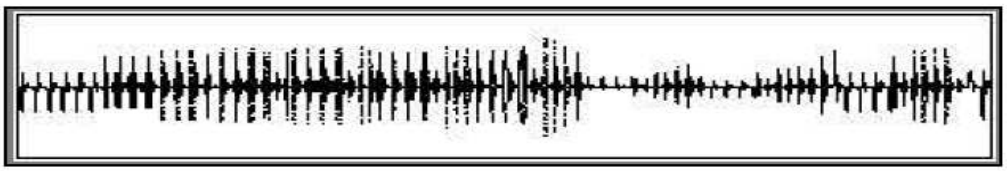

Figura 4.7: Wavelets para musica: Representacion gráfica del tono de quiebre de Wickerhouser.

- *Compresión de imágenes:* La transformada wavelet discreta descompone una imagen en un conjunto de pequeñas imágenes sucesivamente ortonormales. Más aun, mientras el histograma de nivel de grises de la imagen original puede ser de cualquier forma, aquellos reflejos de las transformadas wavelet son comúnmente unimodales y simétricas cerca de cero. Esto simplifica el análisis estadístico de las propiedades de la señal. Como se ve en la Figura 4.8.

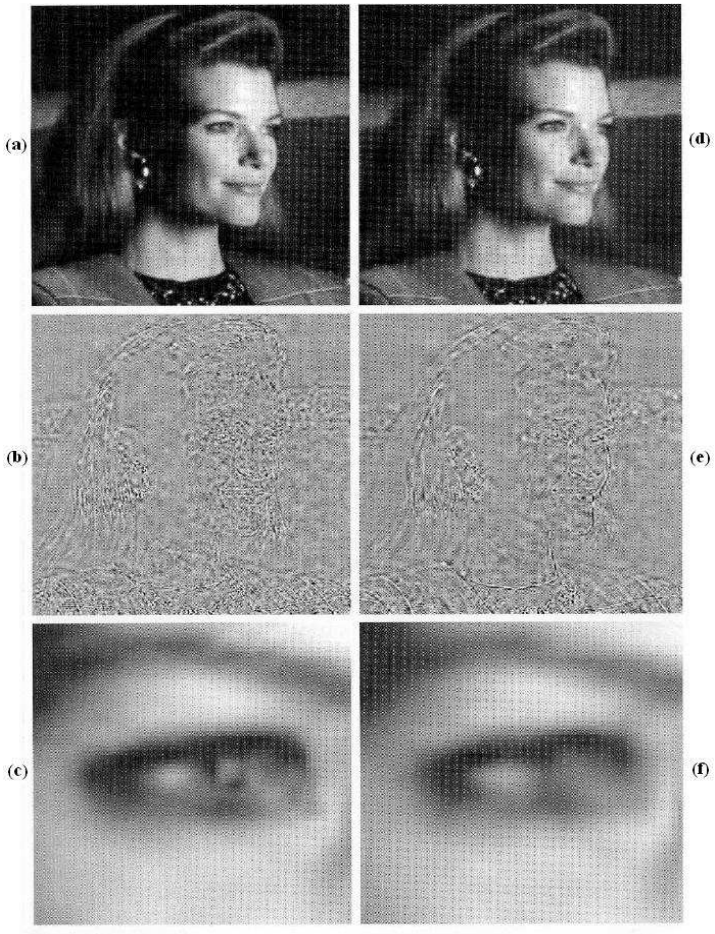

Figura 4.8: (a), (c) y (e) Resultados de la codificación wavelet con una taza de compresión de 108 a 1; (b), (d) y (f) Resultados similares de una compresión de 167 a 1.

4.9. Comentarios y referencias

Puesto que el análisis wavelet corrige las deficiencias del análisis de Fourier, y que por su gran multiresolución provee un análisis más fino. Esto es de gran ayuda para el procesamiento de imágenes como se verá en el siguiente capítulo donde implementarán estos algoritmos a un sistema carro péndulo invertido.

Aunque existen diversas transformadas dentro del análisis, para el uso de señales digitales se emplea la transformada wavelet discreta y la transformada wavelet rápida las cuales son capaces de utilizar señales analógicas para representarlas en su forma digital.

Esto nos lo muestran los siguientes autores con cada uno de sus libros, los cuales sirven de base para la estructura de este capítulo: Graps, A., *An introduction to wavelets.*,

Gonzales, R. C., Woods R. E., *Digital image processing.*, Castleman, K. R. y *Digital image processing.*

Para su implementación en un sistema real es conveniente realizar simulaciones para comprobar que efectivamente el sistema funciona eficazmente, esto ahorra gastos de tiempo y esfuerzo así como también en cuestiones económicas, como se ha visto en otras investigaciones. Como se vera en el próximo capítulo.

Capítulo 5

RESULTADOS DE SIMULACIONES

5.1. Introducción

Una característica importante, fundamental y significativa en el diseño de sistemas de procesamiento de imágenes es el nivel de prueba y experimentación que normalmente es requerida antes de llegar a una solución. Esta característica implica que la habilidad para formular aproximaciones y prototipos rápidos candidatos a soluciones generales juega un gran papel en reducir el costo y tiempo requerido para llegar a un sistema de implementación viable [12].

El procesamiento digital de señales es un área caracterizada por la necesidad de un extenso campo experimental para establecer la viabilidad de proponer soluciones a un problema dado. En este capítulo se ven los resultados de la teoría que a lo largo de este trabajo se ha expuesto mediante un software el cual esta diseñado para proveer soluciones a los problemas de procesamiento de imágenes, este software es MATLAB desarrollado por la compañía "Mathworks.inc"[37].

5.2. Acerca de MATLAB

MATLAB® es un lenguaje de alto rendimiento para cálculos técnico. Este integra cálculo, visualización y programación en un ambiente fácil de usar donde los problemas y soluciones son expresados en una notación matemática familiar. Los usos típicos incluyen:

- Matemáticas y cálculo simbólico.
- Desarrollo de algoritmos.
- Adquisición de datos.
- Modelado, simulación y creación de prototipos.

- Análisis de datos, exploración y visualización.
- Gráficas científicas y de ingeniería.
- Aplicación de desarrollo, incluyendo construcción de interfaces gráficas para usuarios.

MATLAB es un sistema interactivo cuyos elementos básicos son un arreglo que no requiere dimensionar. Esto permite resolver varios problemas de calculo técnico, especialmente aquellos con formulaciones de matrices y vectores, se podría tomar una fracción de tiempo para escribir un programa en lenguaje escalar no interactivo como *C* o Fortran. El nombre de MATLAB permanece del laboratorio matriz. MATLAB fue originalmente escrito para proveer fácil acceso al software de matrices desarrollado por los proyectos de LINPACK y EISPACK. Hoy, MATLAB incorpora las librerías LAPACK y BLAS, incrustando el estado del arte en software para cálculo de matrices. MATLAB ha sido envuelto por un periodo de varios años con la ayuda de varios usuarios. En ambientes universitarios es la herramienta de enseñanza estándar en cursos avanzados de matemáticas, ingeniería y ciencia. En la industria, MATLAB es la herramienta que se escoge para investigaciones de alta productividad, desarrollo y análisis. Lo más importante para los usuarios de MATLAB son los *toolboxes* o cajas de herramientas que permiten aprender y aplicar tecnología especializada. Estas son colecciones comprensivas de funciones de MATLAB (M-files) que extienden el ambiente de MATLAB para resolver particulares clases de problemas. Estas áreas incluyen procesamiento de señales, sistemas de control, redes neuronales, lógica difusa, wavelets, simulación y muchas otras.

5.3. Wavelets en MATLAB

Las herramientas para wavelets son una colección de funciones construidas en el ambiente de calculo matemático de MATLAB ®. Esta provee herramientas para el análisis y síntesis de señales e imágenes, y herramientas para aplicaciones estadísticas, usando wavelets y paquetes de wavelets con el ambiente de trabajo de MATLAB. Las herramientas proveen dos clases de herramientas:

- Funciones en la línea de comandos
- Herramientas gráficas interactivas

La primer categoría de herramientas están hechas de funciones que puedes llamar directamente desde la línea de comandos o desde tus propias aplicaciones. Muchas de esta funciones son *M-files*, las cuales son series de declaraciones que implementan análisis de wavelet especializadas. Se ve el código de estas funciones usando la siguiente declaración:

```
type function_name
```

La segunda categoría de herramientas es una colección de herramientas de interfase gráfica que ofrece un acceso a una funcionalidad extensa. Para acceder a estas herramientas se escribe

```
wavemenu
```

desde la línea de comandos.

En este capítulo se verá la aplicación de las transformadas Wavelet de la imagen de un péndulo invertido tomada por una cámara conectada a un procesador digital de señales (DSP) la cual se muestra en la Figura 5.1.

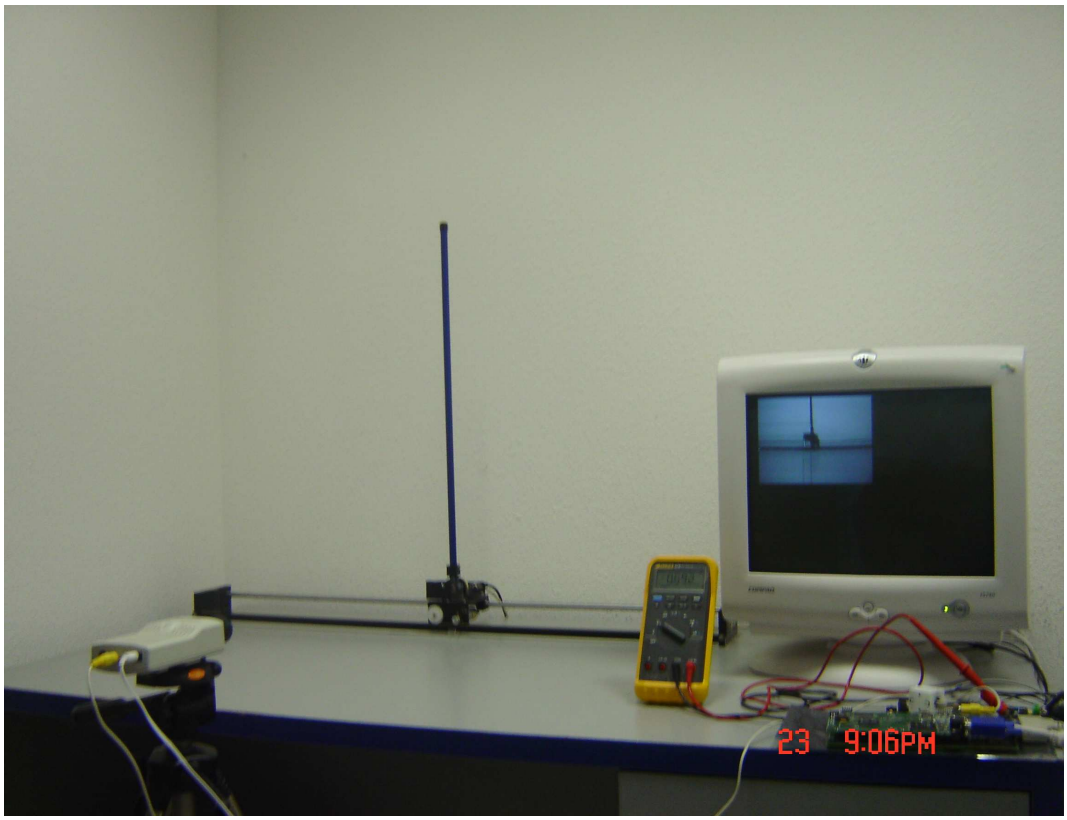

Figura 5.1: Imagen del péndulo invertido controlada por el DSP.

5.3.1. Línea de comandos

Cuando las imágenes digitales son vistas o procesadas a múltiples resoluciones, la herramienta matemática para su tratamiento idóneo es la *Transformada wavelet discreta* (DWT). Además de que llega a ser eficiente, un alto campo de trabajo para la representación y almacenamiento de imágenes de *multiresolución*, la DWT provee una poderosa comprensión dentro de las características frecuenciales y espaciales de la imagen. Mientras que por otro lado, la transformada de Fourier, revela solamente los atributos de frecuencia de la imagen [12].

En este capítulo, veremos el cálculo de la transformada wavelet discreta mediante el software ya mencionado, lo cual se lleva acabo de dos formas. Como ya se dijo anteriormente, se pueden usar la línea de comandos mediante algoritmos del programa ya establecidos y definidos así como mediante funciones creadas por el usuario llamadas "M-files", las cuales no se encuentran dentro de las herramientas de wavelets en MATLAB. La otra manera es mediante al ambiente gráfico de las herramientas de wavelet la cual se puede acceder mediante el comando `wavemenu` esto se vera en la siguiente sección.

De la Figura 5.1 solo se utilizara la parte izquierda superior para su análisis, por lo tanto esta queda como se muestra en la Figura 5.2.

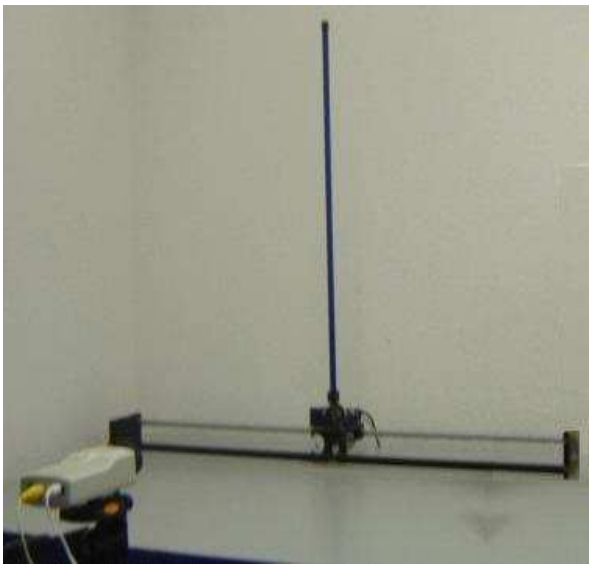

Figura 5.2: Imagen a tratar del péndulo invertido tomada por el DSP.

Dentro del ambiente de MATLAB sólo soporta los siguientes formatos de imagen:

Nombre del formato	Descripción	Extensión reconozida
TIFF	Tagged Image File Format	.tif .tiff
JPEG	Join Photographic Experts Group	.jpg .jpeg
GIF	Grafics Interchange Format	.gif
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

Tabla 5.1: Formatos de imagen que soporta MATLAB.

En este trabajo el tipo de imagen a utilizar es del tipo JPEG. Por lo tanto el código necesario para introducir la imagen dentro de un arreglo en MATLAB es el siguiente:

```
>> I=imread('DSP1.jpg'); % Carga el archivo de imagen y lo guarda en la variable I.
>> whos % Muestra el tamaño y la clase de las variables guardadas.
>> save I % Guarda la variable I.
>> ls % Muestra tus archivos en el directorio.
>> % Ahora existe un archive llamado "I.mat" en tu directorio que
>> % contiene la variable I.
>> clear % Limpia la memoria de MATLAB.
>> load I % Carga la variable I que guardamos anteriormente.
>> whos % Checa que ya esta guardada.
>> imshow(I) % Muestra la imagen.
>> I=im2double(I); % Convierte a variable a un arreglo doble.
>> whos % Checa que la variable a sido convertida a doble.
```

Mediante este código se guarda la imagen dentro de MATLAB pero lo hace en un arreglo de tipo *unit-8* y para propósitos de simulación en cualquiera de la formas vistas anteriormente de be ser de tipo *double* por ello se usa la función,

```
>> I=im2double(I);
```

Mediante este arreglo se gráfica el arreglo el cual se ve en la Figura 5.3.

Todo esto es necesario para poder poner en práctica el uso de wavelets. Que para este propósito se requiere obtener los bordes de la imagen ya antes mencionada. Por lo que se usa la transformada rápida wavelet para calcular la transformada discreta wavelet. Todo esto mediante programas hechos en MATLAB (M-files) e instrucciones dentro de la línea de comandos. Aunque existen diversas instrucciones, aquí se presentan funciones diferentes a las comunes, puesto que poseen características diseñadas para obtener los resultados que requerimos.

Se empieza con la función `wavefast` la cual es una variante de la función `wavedec2`, la cual calcula la descomposición wavelet multinivel de dos dimensiones. Cuyo código es el siguiente:

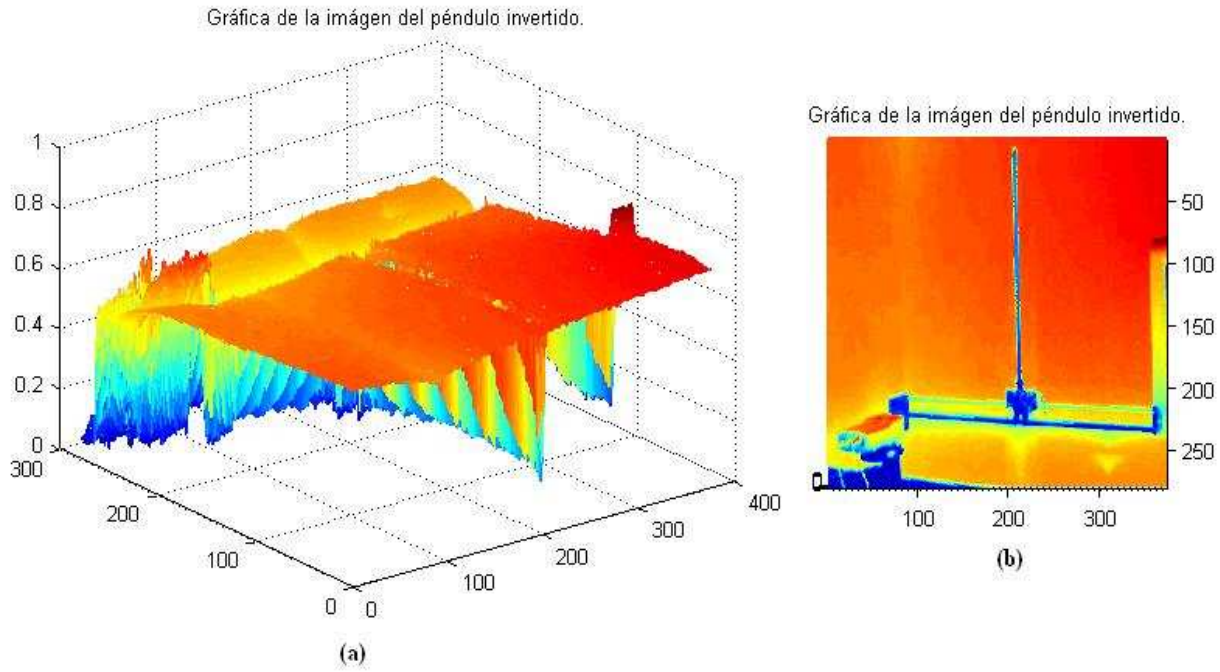

Figura 5.3: (a) Gráfica de la imagen en 3-D y (b) Gráfica de la imagen en 2-D.

MATLAB 5.2: Esta función ejecuta la transformada rápida wavelet multinivel de dos dimensiones

```
function [c,s]=wavefast(x,n,varargin)
% WAVEFAST ejecuta la transformada rápida wavelet  multinivel de dos
% dimensiones.
% [C,L]=WAVEFAST(X, N, LP, HP) Ejecuta la FWT de 2 niveles de
% una imagen (o matriz) X con respecto a la descomposición de filtros
% LP y HP
%
% [C, L]=WAVEFAST(X, N, WNAME) ejecuta la misma operación pero retoma
% los filtros LP y HP para wavelets WNAME usando WAVEFILTER
%
% El parámetro N de escala debe ser menor o igual a log2 de la máxima
% dimensión de la imagen. Los filtros LP y HP deben de ser par. Para
% reducir la distorsión de los bordes, X es simétricamente extendida.
% Que es, si X=[c1 c2 c3 ... cn] (en 1D), entonces su extensión
% simétrica debe ser [... c3 c2 c1 c1 c2 c3... cn cn cn-1 cn-2...].
%
% SALIDAS:
% La matriz c es un vector de coeficientes de descomposición:
%
% c=[a(n) h(n) v(n) d(n) h(n-1) ... v(1) d(1)]
%
% Donde a, h, v, y d son vectores columnas que contienen matrices de
% coeficientes de aproximación horizontal, vertical y diagonal,
% respectivamente. C tiene 3n + 1 secciones donde n es el número de
% descomposiciones wavelet.
```

```
%
% La matriz S es una (n+2) x 2 matriz de contabilidad:
%
% S=[sa(n,:); sd(n,:); sd(-n,:);...; sd(1,:); sx]
%
% Donde sa y sd son aproximaciones y detalles de tamaño de la entrada
% de datos
%
% Compruebe los argumentos de entrada por sentido común
error(nargchk(3,4,nargin));

if nargin == 3
    if ischar(varargin{1})
        [lp, hp]=wavefilter(varargin{1},'d');
    else
        error('Nombre de la Wavelet faltante.');
    end
else
    lp=varargin{1};     hp=varargin{2};
end

fl=length(lp);          sx=size(x);

    if (ndims(x)~= 2) | (min(sx) < 2) | ~isreal(x) | ~isnumeric(x)
        error ('X debe ser real, una matriz numerica.');
    end
    if (ndims(lp) ~=2) | ~isreal(lp) | ~isnumeric(lp) | ndims(hp) ~= 2
        | ~isreal(hp) | ~isnumeric(hp) | fl~=length(hp) | rem(fl,2)~=0
        error('LP y HP deben ser vectores filtros numéricos pares y de
              longitud real igual');
    end
    if ~isreal(n) | ~isnumeric(n) | (n < 1) | (n > log2(max(sx)))
        error(['N debe ser un escalar real entre 1 y
              log2(max(size((X)))).']);
    end
% Inicializa el comienzo de las estructuras de datos de salida y la
% aproximación inicial
c=[];      s=sx;       app=double(x);
% Para cada descomposición ...
for i=1:n
    % Extiende la aproximación simétricamente
    [app,keep]= symextend(app,fl);
    % Convoluciona las filas con HP y muestre de bajada. Entonces
    % convoluciona las columnas con HP y LP para conseguir los
    % coeficientes diagonal y vertical
    rows=symconv(app, hp, 'row', fl, keep);
    coefs=symconv(rows, hp, 'col', fl, keep);
    c=[coefs(:)' c];    s=[size(coefs); s];
    coefs=symconv(rows, lp, 'col', fl, keep);
    c=[coefs(:)' c];
    % Convoluciona las filas con LP y el muestreo de bajada. Entonces
    % convoluciona las columnas con HP y LP para conseguir los
    % siguientes coeficientes diagonal y vertical
    rows=symconv(app, lp, 'row', fl, keep);
    coefs=symconv(rows, hp, 'col', fl, keep);
    c=[coefs(:)' c];
    app=symconv(rows, lp, 'col', fl, keep);
end
```

```
% Anexo final a las estructuras.
c=[app(:)' c];      s=[size(app); s];
% ------------------------------------------------------------------
function [y, keep]=symextend(x, fl)
% Calcula el numero de coeficientes a mantener después de la
% convolución y el muestreo de bajada. Entonces extiende a x en dos
% dimensiones.
keep=floor((fl+size(x)-1)/2); y=padarray(x,[(fl-1) (fl-1)],
'symmetric', 'both');
% ------------------------------------------------------------------
function y=symconv(x, h, type, fl, keep)
% Convoluciona las filas o columnas de x con h, muestreo de bajada
% y extrae la sección central puesto que esta simétricamente extendida
if strcmp(type, 'row')
    y=conv2(x,h);
    y=y(:, 1:2:end);
    y=y(:, fl / 2 + 1 : fl / 2 + keep(2));
else
    y=conv2(x,h');
    y=y(1:2:end,:);
    y=y(fl / 2 + 1 : fl / 2 + keep(1), :);
end
```

La siguiente rutina de prueba usa funciones `tic` y `toc` para comparar el tiempo de ejecución de la función `wavedec2` y nuestra función `wavefast`.

MATLAB 5.3: Esta función compara wavedec2 y wavefast.

```
function [ratio, maxdiff]=fwtcompare(f,n,wname)
% FWTCOMPARE Compara wavedec2 y wavefast
% [RATIO, MAXDIFF]=FWTCOMPARE(F,N,WNAME) compara la operación de la
% herramienta wavedec2 y el uso de la función wavefast
%
% ENTRADAS:
% F:        Imagen a ser transformada.
% N:        Numero de escalas a calcular.
% WNAME:    Wavelet a usar.
%
% SALIDAS
% RATIO:    Ejecución de la taza de tiempo (función hecha/de las
% herraminetas).
% MAXDIFF:  Máxima diferencia de coeficientes.
%
% Obtiene la transformada y calcula el tiempo de wavedec2.
tic; [c1, s1]=wavedec2(f, n, wname); reftime=toc;
% Obtiene la transformada y calcula el tiempo de wavefast.
tic; [c2, s2]=wavefast(f, n, wname); t2=toc;
% Compara los resultados
ratio=t2/(reftime+eps); maxdiff=abs(max(c1-c2));
```

Por lo que para la imagen de la Figura 5.2 y una wavelet de quinta escala con respecto a una wavelet de Daubechies de 4to orden, la función denota que

```
>> load f
>> [ratio, maxdiff]=fwtcompare(F, 5, 'db4')
ratio =
    0.0212
maxdiff =
  1.0214e-014
```

La función `wavework` es el fundamento de las rutinas desarrolladas las cuales están basadas en las formas familiares de cortar, copiar y pegar de las aplicaciones modernas de procesamiento.

MATLAB 5.4: Esta función se usa para editar las estructuras de descomposición de las wavelets

```
function [varargout]=wavework(opcode, type, c, s, n, x)
% WAVEWORK Se usa para editar las estructuras de descomposición de
% las wavelets
% [VARARGOUT]=WAVEWORK(OPCODE, TYPE, C, S, N, X) Obtiene los
% coeficientes especificados por TYPE y N para acceder o modificación
% basado en OPCODE
%
% ENTRADAS:
% OPCODE       Operación a ejecutar.
% ---------------------------------------------------------------------
% 'copy'        [varargout]= Y = Matriz de coeficientes requerida
%               (via TYPE y N)
% 'cut'         [varargout]= [NC, Y] = Nuevo vector de descomposición
%               (con la matriz de coeficientes requerida puesta a cero)
%               Y la Matriz de coeficientes requerida
% 'paste'       [varargout]= [NC] = Nuevo vector de descomposición con
%               la matriz de coeficientes reemplazada por X.
%
% TYPE          Categoria de coeficientes
% ---------------------------------------------------------------------
% 'a'           Coeficientes aproximados.
% 'h'           Detalles horizontales.
% 'v'           Detalles verticales.
% 'd'           Detalles diagonales.
%
% [C,S]         Es la estructura de descomposición de las herramientas
% wavelet.
% N es el nivel de descomposición (Ignorado se TYPE='a').
% X es la matriz de coeficientes de dos dimensiones a pegar.
%
```

```
error(nargchk(4, 6, nargin));

if (ndims(c) ~=2) | (size(c, 1) ~=1)
    error('C debe ser un vector fila.');
end

if (ndims(s) ~=2) | ~isreal(s) | ~isnumeric(s) | (size(s, 2) ~=2)
    error('S debe ser real, un arreglo numérico de dos columnas.');
end

elements=prod(s, 2);        % elementos de la matriz de coeficientes.

if (length(c) < elements(end)) | ~(elements(1) + 3 *
sum(elements(2:end - 1)) >= elements(end))
    error('[C S] debe ser una estructura de descomposición wavelet
           estandar');
end

if strcmp(lower(opcode(1:3)), 'pas') & nargin < 6
    error('No existen argumentos de entrada.');
end
if nargin < 5
    n=1;        % El nivel por defecto es 1.
end
nmax=size(s, 1) - 2;      % Niveles máximos en [c, s].

aflag=(lower(type(1)) == 'a'); if ~aflag & (n > nmax)
    error('N excede las descomposiciones en [C, S]');
end
switch lower(type(1))      % Hace punteros dentro de C.
    case 'a'
        nindex=1;
        start=1;    stop=elements(1);   ntst=nmax;
    case {'h', 'v', 'd'}
        switch type
            case 'h', offset=0;     % Compensa los detalles.
            case 'v', offset=1;
            case 'd', offset=2;
        end
        nindex=size(s, 1)-n;    % Índice para detallar información.
        start=elements(1)+3*sum(elements(2:nmax-n+1))+offset*elements
              (nindex)+1;
        stop=start + elements(nindex)-1;
        ntst=n;
    otherwise
        error('TYPE debe empezar con "a", "h", "v" o "d".');
end

switch lower(opcode)        % Requiere acción.
    case {'copy', 'cut'}
        y=repmat(0, s(nindex, :));
        y(:)=c(start:stop);     nc=c;
        if strcmp(lower(opcode(1:3)), 'cut')
            nc(start:stop)=0; varargout={nc, y};
        else
            varargout={y};
        end
```

```
    case 'paste'
        if prod(size(x)) ~= elements(end-ntst)
            error('X no esta dimensionado para el pegado que se
                   requiere.');
        else
            nc=c;     nc(start:stop)=x(:);          varargout={nc};
        end
    otherwise
        error('OPCODE irreconocible.');
end
```

Las siguientes tres funciones `wavecut`, `wavecopy` y `wavepaste` usan la función anterior para manipular C (que es la matriz de coeficientes de descomposición) usando una sintaxis más intuitiva.

wavecut

MATLAB 5.5: Esta función pone a cero los coeficientes en una estructura de descomposición wavelet.

```
function [nc, y]=wavecut(type, c, s, n)
% WAVECUT pone a cero los coeficientes en una estructura de descompo-
% sición wavelet.
% [NC, Y]=WAVECUT(TYPE, C, S, N) Regresa una nuevo vector de  desompo-
% sición el cual detalla o aproxima los coeficientes (basados en TYPE
% y N.) han sido puestos a cero. los coeficientes que fueron puestos
% a cero son regresados en Y.
%
% ENTRADAS:
% TYPE          Categorías de coeficientes.
% ---------------------------------------------------------------------
% 'a'           Coeficientes aproximados.
% 'h'           Detalles horizontales.
% 'v'           Detalles verticales.
% 'd'           Detalles diagonales.
% [C,S]         Es la estructura de datos.
% N especifica el nivel de descomposición (Ignorado se TYPE='a').
%
error(nargchk(3, 4, nargin)); if nargin == 4
    [nc, y]=wavework('cut', type, c, s, n);
else
    [nc, y]=wavework('cut', type, c, s);
end
```

wavecopy

MATLAB 5.6: Esta función retoma los coeficientes de la estructura de descomposición de la wavelet.

```
function y=wavecopy(type, c, s, n)
% WAVECOPY retoma los coeficientes de la estructura de descomposición
% de la wavelet.
% Y=WAVECOPY(TYPE, C, S, N) Regresa el arreglo de coeficientes
% (basados en TYPE y N.)
%
% ENTRADAS:
% TYPE          Categorías de coeficientes.
% ---------------------------------------------------------------
% 'a'           Coeficientes aproximados.
% 'h'           Detalles horizontales.
% 'v'           Detalles verticales.
% 'd'           Detalles diagonales.
% [C,S]         Es la estructura de datos.
% N especifica el nivel de descomposición (Ignorado se TYPE='a').
%
error(nargchk(3, 4, nargin)); if nargin == 4
    y=wavework('copy', type, c, s, n);
else
    y=wavework('copy', type, c, s);
end
```

wavepaste

MATLAB 5.7: Esta función pone los coeficientes en una estructura de descomposición wavelet.

```
function nc=wavepaste(type, c, s, n, x)
% WAVEPASTE pone los coeficientes en una estructura de descomposición
% wavelet.
% NC=WAVEPASTE(TYPE, C, S, N, X) Regresa una nueva estructura de
% descomposición después de pegar X dentro de su fundamento en TYPE y N
%
% ENTRADAS:
% TYPE          Categorías de coeficientes.
% ---------------------------------------------------------------
% 'a'           Coeficientes aproximados.
% 'h'           Detalles horizontales.
% 'v'           Detalles verticales.
% 'd'           Detalles diagonales.
% [C,S]         Es la estructura de datos.
% N especifica el nivel de descomposición (Ignorado se TYPE='a').
% X es el aproximado  de dos dimensiones o detalles de la matriz de
% coeficientes cuyas dimensiones son apropiadas para descomposición de
% nivel N
%
error(nargchk(5, 5, nargin)); nc=wavework('paste',type,c,s,n,x);
```

La función `wave2gray` crea una subimagen y escala los coeficientes para revelar mejores diferencias e insertar los bordes para delinear la aproximación y varios detalles de las matrices vertical, horizontal y diagonal.

MATLAB 5.8: Esta función muestra los coeficientes de descomposición Wavelet.

```
function w=wave2gray(c, s, scale, border)
% WAVE2GRAY Muestra los coeficientes de descomposición wavelet.
% W=WAVE2GRAY=(C, S, SCALE, BORDER) Muestra y regresa los coeficientes
% wavelet de la imagen.
%
% EJEMPLOS:
% wave2gray(c, s);                    Despliega w/default.
% foo=wave2gray(c, s);                Despliega y regresa.
% foo=wave2gray(c, s, 4);             Amplifica los detalles.
% foo=wave2gray(c, s, -4);            Amplifica los valores absolutos.
% foo=wave2gray(c, s, 1, 'append');   Mantiene el valor de los bordes.
%
% ENTRADAS/SALIDAS:
% [C, S] es el vector de descomposición wavelet y la matriz de
% contabilidad.
%
% ESCALA        Define los coeficientes de escala.
% ------------------------------------------------------------------
% 0 o 1         Máximo rango (Por defecto.)
% 2, 3 ...      Amplifica los defaults por el factor de escala.
% -1, -2, ...   Amplifica los valores absolutos por abs(scale).
%
% BORDES        Limita entre las descomposiciones wavelet.
% ------------------------------------------------------------------
% 'absorb'      Los bordes reemplazan a la imagen (Por defecto).
% 'append'      Los bordes se incrementan con la anchura de la imagen.
%
% Imagen W:     ------- ------ -------------- -------------------
%               |       |      |              |
%               | a(n)  | h(n) |              |
%               |       |      |              |
%               ------- ------     h(n-1)     |
%               |       |      |              |
%               | v(n)  | d(n) |              |       h(n-2)
%               |       |      |              |
%               ------- ------ --------------
%               |              |              |
%               |   v(n-1)     |    d(n-1)    |
%               |              |              |
%               ------- ------ -------------- -------------------
%               |                             |
%               |          v(n-2)             |       d(n-2)
%               |                             |
%
% Aquí n denota el escalón de  la descomposición escala, y h, v, d son
% aproximaciones de detalles  de los coeficientes diagonal, horizontal
% y vertical respectivamente.
% Comprueba los argumentos de entrada por lógica.
```

```
error(nargchk(2, 4, nargin));

if (ndims(c) ~=2) | (size(c, 1) ~=1)
    error('C debe ser un vector fila');
end

if (ndims(s) ~=2) | ~isreal(s) | ~isnumeric(s) | (size(s, 2) ~=2)
    error('S debe ser real un arreglo numérico de dos columnas');
end
elements=prod(s, 2); if (length(c) < elements(end)) |
~(elements(1) + 3 * sum(elements(2:end-1)) >= elements(end))
    error('[C S] debe ser una estructura de descomposición wavelet
           estandar.');
end

if (nargin > 2) & (~isreal(scale) | ~isnumeric(scale))
    error('SCALE debe ser un valor real, escalar numérico');
end

if (nargin > 3) & (~ischar(border))
    error('BORDER debe ser un caracter de cadena.');
end

if nargin == 2
    scale=1;            % Escala por defecto.
end if nargin < 4
    border='absorb';    % Borde por defecto.
end
% Coeficientes de escala y determina el relleno.
absflag= scale < 0; scale=abs(scale);
    if scale == 0
    scale=1;
end

[cd, w]=wavecut('a',c,s);   w=mat2gray(w);
cdx=max(abs(cd(:)))/scale; if absflag
    cd=mat2gray(abs(cd), [0, cdx]);     fill=0;
else
    cd=mat2gray(cd, [-cdx, cdx]);       fill=0.5;
end
% Construye una imagen gris en una descomposición al mismo tiempo.
for i=size(s, 1) - 2:-1:1
    ws=size(w);

    h=wavecopy('h',cd,s,i);
    pad=ws-size(h);     frontporch=round(pad/2);
    h=padarray(h,frontporch,fill,'pre');
    h=padarray(h,pad - frontporch,fill,'post');

    v=wavecopy('v',cd,s,i);
    pad=ws-size(v);     frontporch=round(pad/2);
    v=padarray(v,frontporch,fill,'pre');
    v=padarray(v,pad - frontporch,fill,'post');

    d=wavecopy('d',cd,s,i);
    pad=ws-size(d);     frontporch=round(pad/2);
    d=padarray(d,frontporch,fill,'pre');
    d=padarray(d,pad - frontporch,fill,'post');
```

```
        % Agrega un pixel blanco de borde.
        switch lower(border)
            case 'append'
                w=padarray(w, [1 1], 1, 'post');
                h=padarray(h, [1 0], 1, 'post');
                v=padarray(v, [0 1], 1, 'post');
            case 'absorb'
                w(:, end)=1;          w(end, :)=1;
                h(end, :)=1;         v(:, end)=1;



            otherwise
                error('Parámetro BORDER irreconocible.');
        end
        w=[w h; v d];            % Coeficientes concatenados.
    end

    if nargout == 0
        imshow(w);               % Despliega el resultado.
    end
```

La siguiente rutina de comandos calcula la DWT de dos dimensiones de la imagen de la Figura 5.2 con respecto a la wavelet de Daubechies y despliega los coeficientes resultantes

```
>> load f
>> [c, s]=wavefast(F, 2, 'db4');
>> wave2gray(c, s);
>> figure; wave2gray(c, s, 8);
>> figure; wave2gray(c, s, -8);
```

Los resultados se muestran en la Figura 5.4.

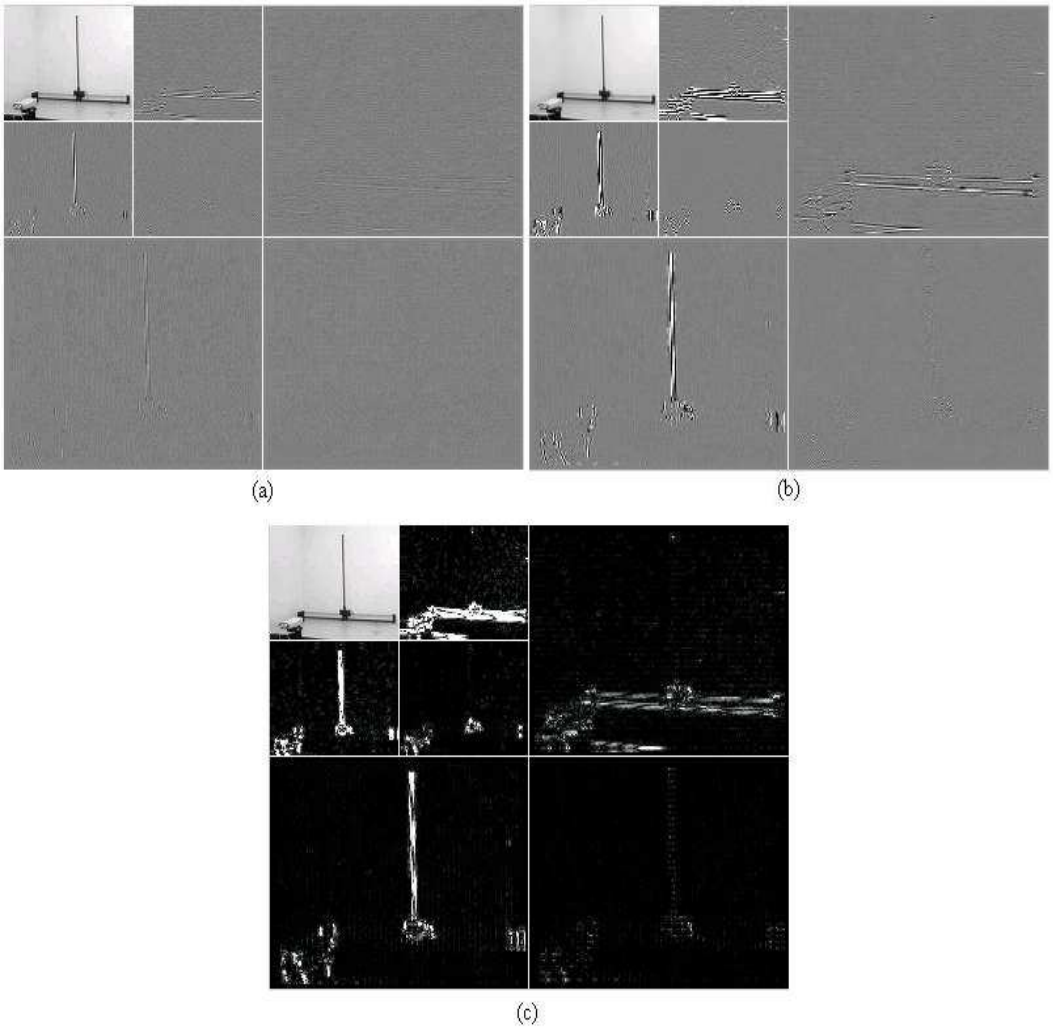

Figura 5.4: Gráficas de la transformada discreta wavelet de la imagen en la Figura 5.2. (a) Escala automática, (b) Escala adicional por 8 y (c) Valores absolutos escalados por 8.

La siguiente rutina `waveback`, es la función final que se necesita este paquete de funciones para procesamiento digital de imágenes en conjunción con las herramientas incluidas en MATLAB. Esta función tiene similitud con la función `waverec2`.

MATLAB 5.9: Esta función ejecuta la FWT inversa multinivel de dos dimensiones.

```
function [varargout]=waveback(c, s, varargin)
% WAVEBACK ejecuta la FWT inversa multi nivel de dos dimensiones.
% [VARARGOUT]=WAVEBACK(C, S, VARARGIN) calcula la reconstrucción
% wavelet 2-D de N niveles parcial o completa de la estructura de
% descomposición [C, S].
%
% SINTAXIS:
% Y=WAVEBACK(C, S, 'WNAME');    Matriz Y de salida de la IFWT
% Y=WAVEBACK(C, S, LR, HR);     Usando filtros de reconstrucción pasa
%                               bajas y pasa altas (LR y HR) o los
%                               filtros obtenidos por WAVEFILTER con
%                               'WNAME'.
%
% [NC, NS]=WAVEBACK(C, S, 'WNAME', N);    Saca la nueva wavelet.
% [NC, NS]=WAVEBACK(C, S, LR, HR, N);     Estructura de descomposición
%                                         [NC, NS] después de la
%                                         reconstrucción de N pasos.
%
% Compruebe los argumentos de entrada y salida por lógica.
%
error(nargchk(3,5,nargin)); error(nargchk(1,2,nargout));

if (ndims(c) ~=2) | (size(c, 1) ~=1)
    error('C debe ser un vector fila');
end

if (ndims(s) ~=2) | ~isreal(s) | ~isnumeric(s) | (size(s, 2) ~=2)
    error('S debe ser real, un arreglo numérico de dos columnas.');
end

elements=prod(s, 2); if (length(c) < elements(end)) | ~(elements(1)
+ 3 * sum(elements(2:end - 1)) >= elements(end))
    error('[C S] debe ser una estructura de descomposición wavelet
           estándar ');
end

% Niveles máximos en [C, S].
nmax=size(s, 1)-2;
% Obtiene el tercer parámetro de entrada e inicia la comprobación
% de las banderas.
wname=varargin{1};      filterchk=0;        nchk=0;

switch nargin
    case 3
        if ischar(wname)
            [lp, hp]=wavefilter(wname, 'r');    n=nmax;
        else
            error('Filtro indefinido');
        end
        if nargout ~= 1
            error('Numero incorrecto de argumentos de salida.');
        end
```

```
    case 4
        if ischar(wname)
            [lp, hp]=wavefilter(wname, 'r');
            n=varargin{2};      nchk=1;
        else
            lp=varargin{1};     hp=varargin{2};
            filterchk=1;        n=nmax;
            if nargout ~=1
                error('Número incorrecto de argumentos de salida.');
            end
        end
    case 5
        lp=varargin{1};     hp=varargin{2};     filterchk=1;
        n=varargin{3};      nchk=1;
    otherwise
        error('Numero improcedente de argumentos de entrada.');
end

fl=length(lp); if filterchk
    if (ndims(lp)~=2) | ~isreal(lp) | ~isnumeric(lp) | (ndims(hp)~=2) |
        ~isreal(hp) | ~isnumeric(hp) | (fl~=lentgh(hp)) | rem(fl, 2)~=0
        error('LP y HP deben ser pares y vectores de filtros numéricos
               de igual longitud.');
    end
end

if nchk & (~isnumeric(n) | ~isreal(n))      % Comprueba la escala de N.
    error('N debe ser un numero real.');
end

if (n > nmax) | (n < 1)
    error('Numero invalido (N)  de reconstrucciones requeridas.');
end

if (n ~=nmax) & (nargout ~=2)
    error('No hay suficientes argumentos de salida.');
end

nc=c;       ns=s;       nnmax=nmax;     % Descomposición inicial.
for i=1:n
    % Calcula la nueva aproximación.
    a=symconvup(wavecopy('a',nc,ns),lp,lp,fl,ns(3,:))+symconvup
      (wavecopy('h',nc,ns,nnmax),hp,lp,fl,ns(3,:))+symconvup
      (wavecopy('v',nc,ns,nnmax),lp,hp,fl,ns(3,:))+symconvup
      (wavecopy('d',nc,ns,nnmax),hp,hp,fl,ns(3,:));
    % Actualizar descomposición.
    nc=nc(4*prod(ns(1, :)) + 1:end);        nc=[a(:)' nc];
    ns=ns(3:end, :);                        ns=[ns(1, :); ns];
    nnmax=size(ns, 1) - 2;
end

% Para reconstrucciones completas, se reformatea la salida como 2-D.
if nargout == 1
    a=nc;       nc=repmat(0, ns(1, :));     nc(:)=a;
end

varargout{1}=nc; if nargout == 2
    varargout{2}=ns;
end
```

```
% --------------------------------------------------------------------
function z=symconvup(x, f1, f2, fln, keep)
% Muestrea de subida las filas y convoluciona las columnas con f1;
% muestrea de subida las columnas y convoluciona las filas con f2;
% entonces extrae el centro asumiendo extensión simétrica.

y=zeros([2 1] .* size(x));      y(1:2:end, :)=x; y=conv2(y, f1');
z=zeros([1 2] .* size(y));      z(:, 1:2:end)=y; z=conv2(z, f2);
z=z(fln - 1:fln + keep(1) - 2, fln - 1:fln + keep(2) - 2);
```

La siguiente rutina de prueba usa funciones para comparar el tiempo de ejecución de la función de MATLAB `waverec2` y nuestra función `wavefast` usando una simple modificación a la rutina de `fwtcompare` .

MATLAB 5.10: Esta función compara waverec2 y waveback.

```
function [ratio, maxdiff]=ifwtcompare(f,n,wname)
% FWTCOMPARE Compara waverec2 y waveback.
% [RATIO, MAXDIFF]=IFWTCOMPARE(F,N,WNAME) compara la operación de la
% herramienta waverec2 y el uso de la función waveback
%
% ENTRADAS:
% F:        Imagen a ser transformada e inversamente transformada.
% N:        Numero de escalas a calcular.
% WNAME:    Wavelet a usar.
%
% SALIDAS
% RATIO:    Ejecución de la taza de tiempo (función hecha/de las
% herramientas).
% MAXDIFF:  Máxima diferencia de la imagen generada.
%
% Obtiene la transformada y obtiene la salida y calcula el tiempo de
% waveback.
[c1, s1]=wavedec2(f, n, wname); tic; g1=waverec2(c1, s1, wname);
reftime=toc;
% Obtiene la transformada y obtiene la salida y calcula el tiempo de
% waveback.
[c2, s2]=wavefast(f, n, wname); tic; g2=waveback(c2, s2, wname);
t2=toc;
% Compara los resultados
ratio=t2/(reftime+eps); maxdiff=abs(max(max(g1-g2)));
```

Para probar esta función se aplica una transformada de cinco escalas con respecto a una wavelet de Daubechies de 4to orden por lo que entonces se tiene,

```
>> load f
>> [ratio, maxdiff]=ifwtcompare(F, 5, 'db4')
ratio =
    0.8140
maxdiff =
  1.1102e-015
```

Con las dos funciones anteriores se demuestra la sensitividad direccional de la transformada wavelet de dos dimensiones y su utilidad en la detección de bordes.

```
>> load f
>> imshow (F)
>> [c, s]=wavefast(F, 1, 'sym4');
>> figure; wave2gray(c, s, -6);
>> [nc, y]=wavecut('a', c, s);
>> figure; wave2gray(nc, s, -6);
>> edges=abs(waveback(nc, s, 'sym4'));
>> figure; imshow(mat2gray(edges));
```

Las gráficas resultantes se muestran en la Figura 5.5.

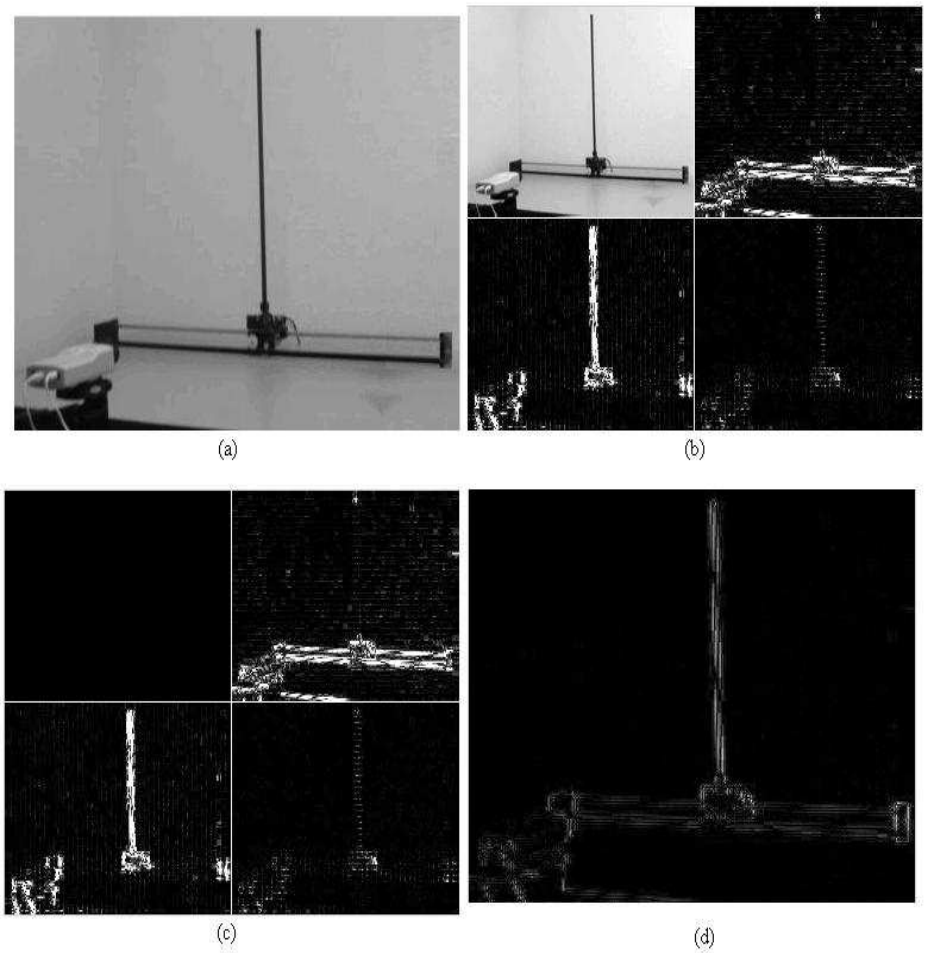

Figura 5.5: Wavelets en detección de bordes. (a) Una vista simple de la Figura 5.2, (b) Su transformada wavelet, (c) La transformada modificada poniendo a cero todos los coeficientes de aproximación y (d) Imagen de bordes resultante de calcular los valores absolutos de la transformada inversa.

Las wavelets como su contraparte en Fourier, son instrumentos efectivos alisando o difuminando imágenes. Para estilizar el proceso de alisado, utilizamos la siguiente función denominada `wavezero`

MATLAB 5.11: Esta función pone a cero los detalles de los coeficientes de la transformada wavelet

```
function [nc, g8]=wavezero(c, s, l, wname)
% WAVEZERO pone a cero los detalles de los coeficientes de la
% transformada wavelet
% [NC, G8]=WAVEZERO(C, S, L, WNAME) pone a cero los  nivel de
% coeficientes de detalle L en la estructura de descomposición wavelet
% [C, S] y calcula la transformada inversa resultante con respecto a
% las wavelets WNAME.
%
[nc, foo]=wavecut('h',c,s,l); [nc, foo]=wavecut('v',nc,s,l); [nc,
foo]=wavecut('d',nc,s,l); i=waveback(nc,s,wname);
g8=im2uint8(mat2gray(i)); figure; imshow(g8);
```

Usando la función `wavezero`, se pueden generar una serie de figuras alisadas en aumento, mediante la siguiente secuencia de comandos,

```
>> load f
>> [c, s]=wavefast(F, 4, 'sym4');
>> wave2gray(c,s,20);
>> [c, g8]=wavezero(c,s,1,'sym4');
>> [c, g8]=wavezero(c,s,2,'sym4');
>> [c, g8]=wavezero(c,s,3,'sym4');
>> [c, g8]=wavezero(c,s,4,'sym4');
```

Las gráficas resultantes se muestran en la Figura 5.6.

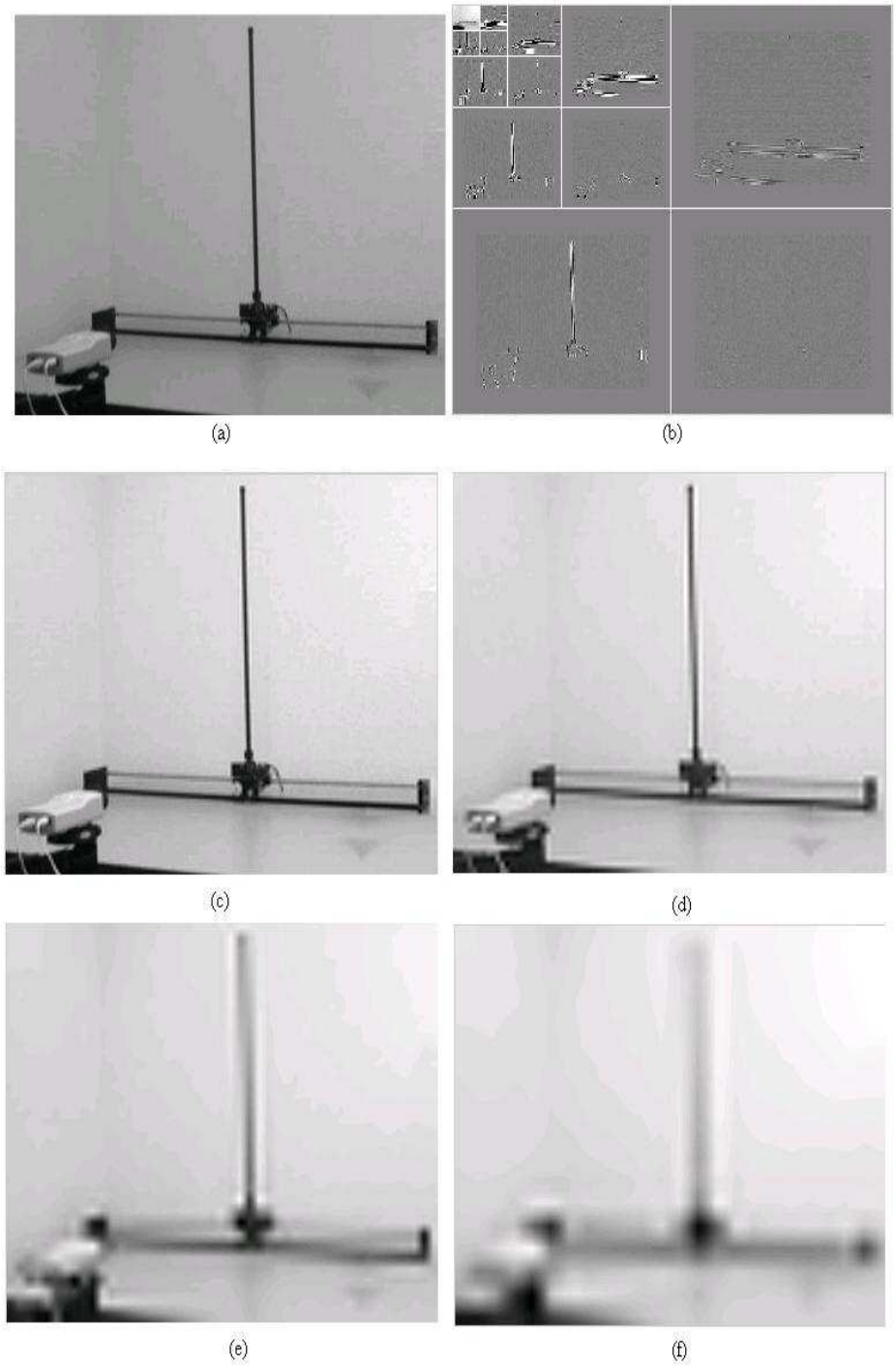

Figura 5.6: Wavelet basado en alisado de imagen. (a) Imagen original, (b) Su transformada wavelet, (c) Transformada inversa después de poner a cero los primeros niveles de los coeficientes de detalle; (d), (e) y (f) Resultados similares después de poner a cero los segundos, terceros y hasta cuartos niveles de los coeficientes de detalle.

5.3.2. Ambiente gráfico

Existe otra forma de realizar las simulaciones para obtener los bordes, mediante la aplicación de las transformadas wavelets. Esta otra opción es mediante la aplicación del modo gráfico de MATLAB para wavelets. Esto se hace mediante el comando `wavemenu` el cual se debe de escribir sobre la línea de comandos. El modo gráfico se ve como se muestra en la Figura 5.6.

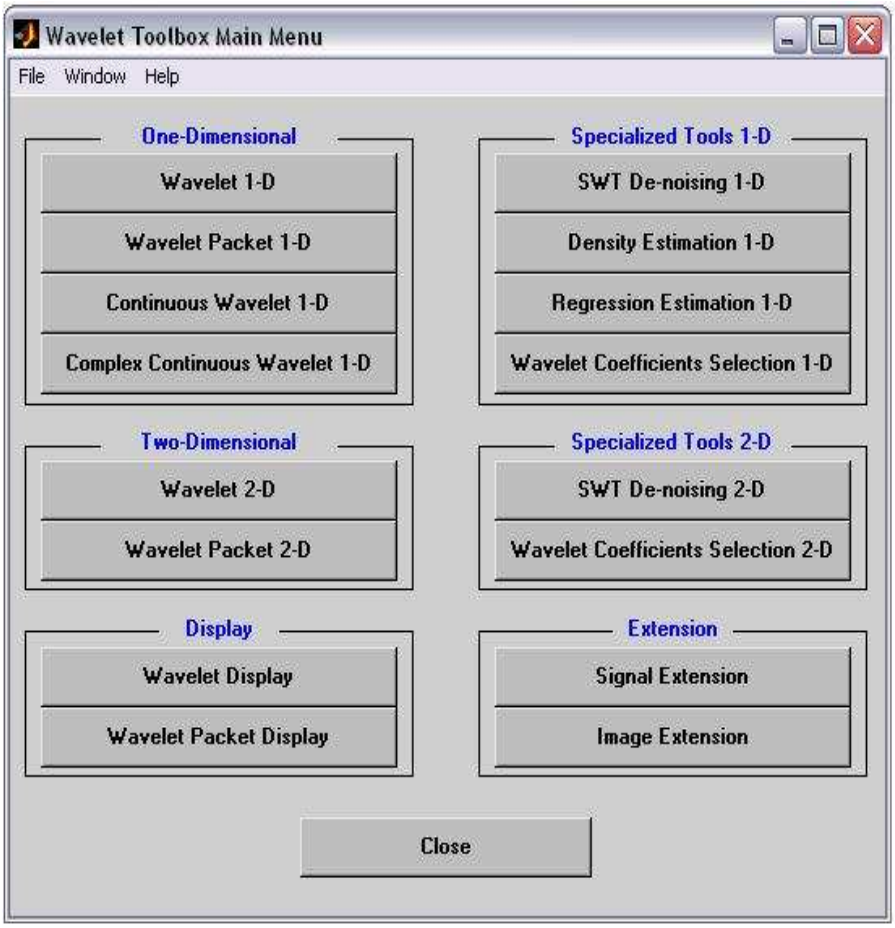

Figura 5.7: Wavemenu.

Previamente hay que procesar la imagen en la línea de comandos para posteriormente poder utilizar el ambiente gráfico de MATLAB. Esto se hace mediante el código que se muestra a continuación:

```
>> I=imread('DSP1.jpg');
% Carga el archivo de imagen y lo guarda en la variable I.
>> save I
% Guarda la variable I.
>> clear
% Limpia la memoria de MATLAB.
>> load I
% Carga la variable I guardada anteriormente.
>> imshow(I)
% Muestra la imagen.
>> [I,map] = rgb2ind(u,256);
% Obtiene el arreglo de la imagen indexada y el arreglo de colores de la imagen
>> I=rgb2gray(I);
% Convierte el arreglo indexado a un arreglo de la imagen de formato    intensidad
>> imshow(I)
% Muestra la imagen.
```

Este código es necesario puesto que se necesita tener un arreglo de la imagen en formato de incendiada y un arreglo que contenga los colores de la imagen en caso de ser una imagen a color. Los resultados de este código son necesarios guardarlos para obtener el archivo de tipo `m`. Posteriormente se utiliza el comando `wavemenu` y se selecciona el botón de *Wavelet 2-D*, en el cual utilizando los menús, se selecciona el menú "File"después *load*, e *image*. Con esto se carga el archivo *m* previamente guardado.

Mediante esta aplicación se pueden aplicar diversas opciones que se le aplican a las wavelets como los son la descomposición de paquetes wavelet, filtrado de coeficientes, reconstrucción, de-noising, histogramas, compresión, residuos de imagen, selección de coeficientes y estadísticas de diversos tipos. Además de otras técnicas relacionadas con wavelets en una sola dimensión.

Por cuestiones prácticas sólo mostraremos el análisis de este método con las wavelets que se utilizaron en los resultados anteriores que es la wavelet de Daubechies de 4 coeficientes y de nivel 2 y la wavelet de Symmlet de 4 coeficientes al nivel 1 y 4; para demostrar que los resultados obtenidos con ambos métodos son los mismos.

Por lo que los resultados de este método se muestran en la Figura 5.8.

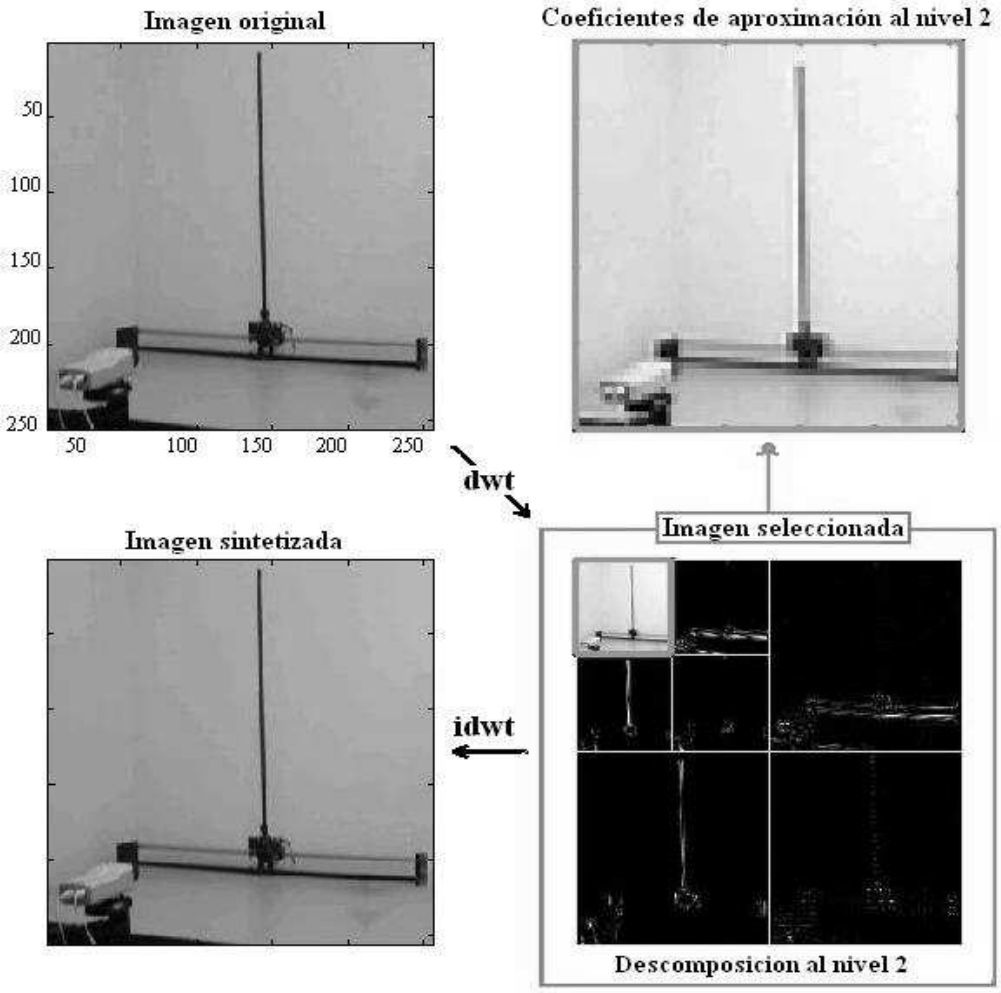

Figura 5.8: Resultados de simulaciones mediante el método gráfico aplicando la Wavelet de Daubechies a nivel 2.

La siguiente demostración es aplicando a la misma imagen la wavelet de Symmlet de cuatro coeficientes nivel 1. La cual se muestra en la Figura 5.9.

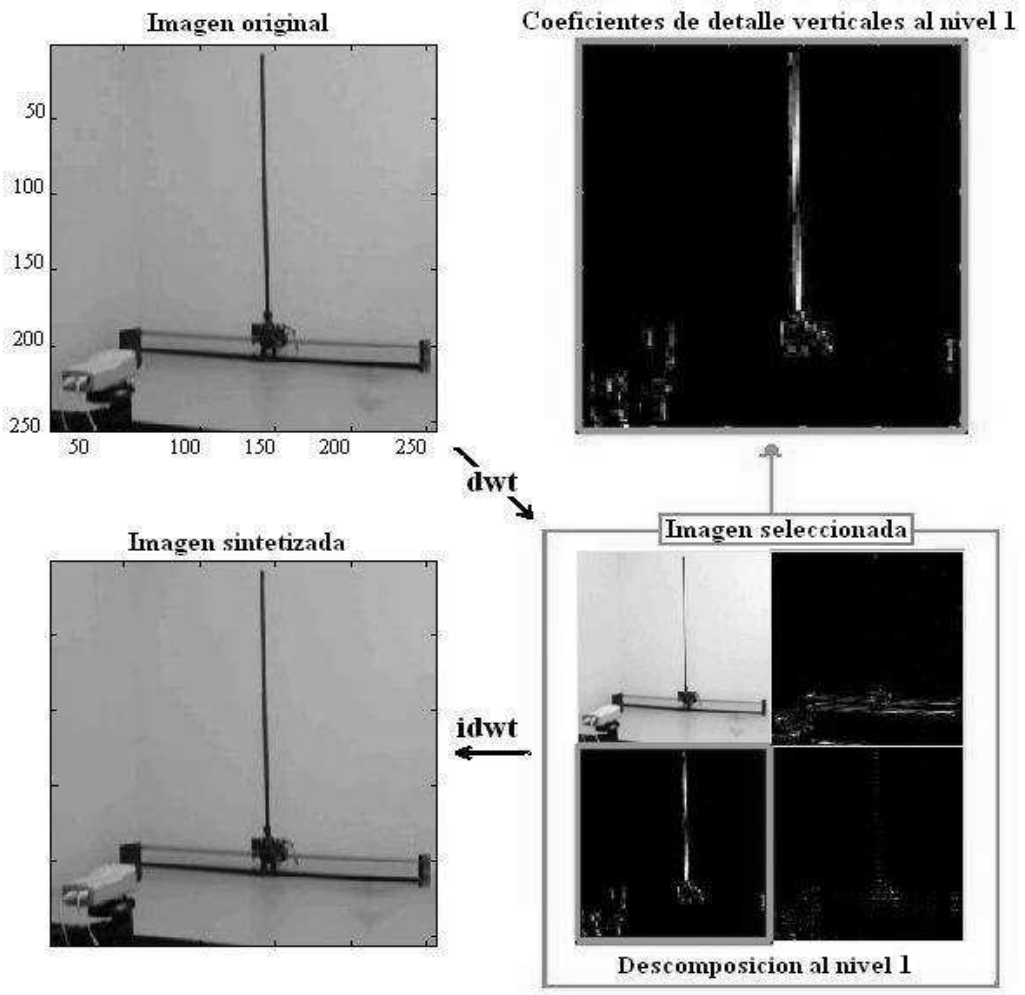

Figura 5.9: Resultados de simulaciones mediante el método gráfico aplicando la Wavelet de Symmlet a nivel 1.

La siguiente demostración es aplicando a la misma imagen la wavelet de Symmlet de cuatro coeficientes nivel 4. La cual se muestra en la Figura 5.10

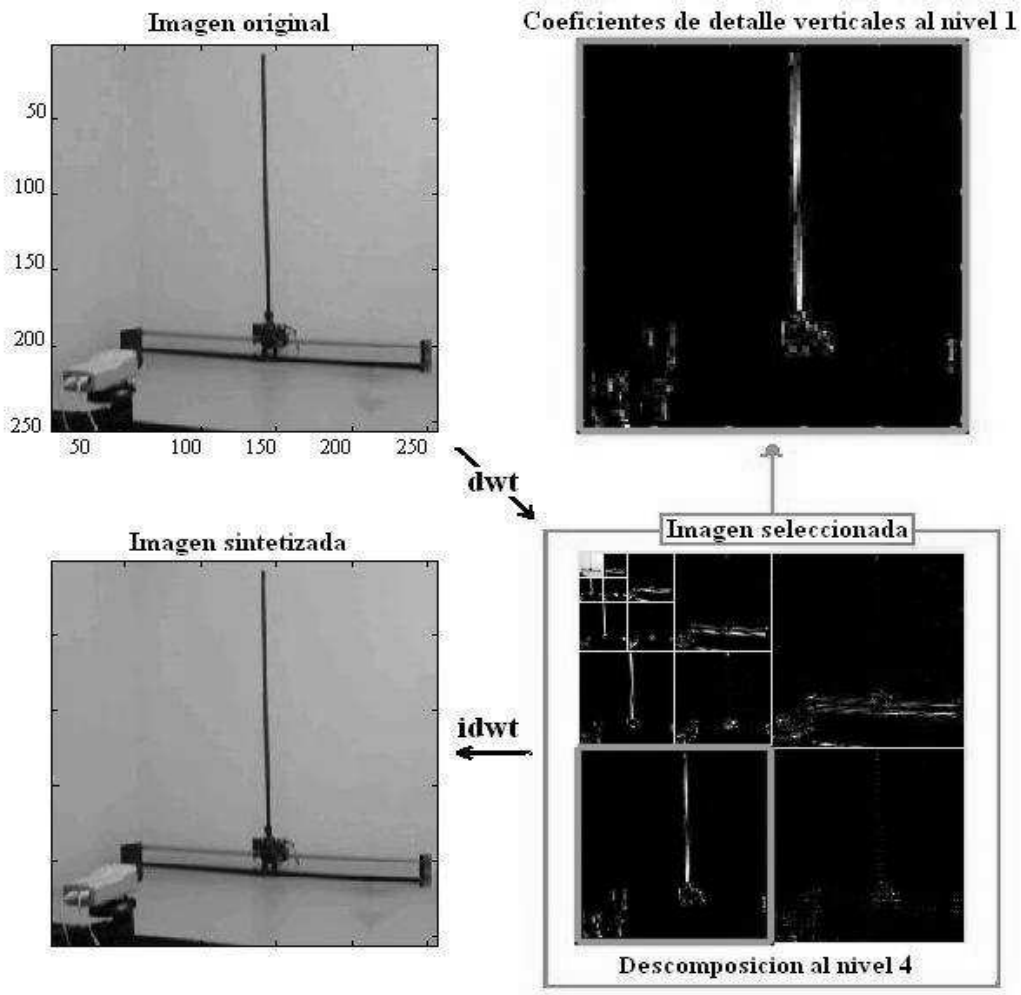

Figura 5.10: Resultados de simulaciones mediante el método gráfico aplicando la wavelet de Symmlet a nivel 4.

Como se ve en estas imágenes los resultados obtenidos mediante el método gráfico son iguales a los obtenidos mediante la línea de comandos.

5.4. Comentarios

La simulación es una parte muy importante en el desarrollo de un proyecto de procesamiento digital de señales e imágenes, puesto que permite observar el comportamiento de los algoritmos numéricos antes de ser implantados en tiempo real. Esto con la finalidad de ahorrar tiempo (hombre y maquina) y costos.

Pero aunque en la simulación se puede observar todo esto, la parte mas importante en un trabajo de investigación es la aplicación de los conocimientos y experiencias dentro del equipo de laboratorio y llevar acabo las pruebas que corroboren las simulaciones, esto es lo que se muestra en el siguiente capítulo.

Capítulo 6

RESULTADOS DE LABORATORIO

6.1. Introducción

Las wavelets han sido desarrolladas para analizar los componentes de frecuencia de una señal de acuerdo con la escala. Estas proveen más información que la transformada de Fourier para señales que tienen discontinuidades o picos [39].

Una de las aplicaciones más comunes, es la detección de bordes, extracción del ruido, descomposición y reconstrucción. Para mostrar estas aplicaciones usaremos las librerías IMGLIB e IDK de un procesador digital de señales (DSP) TMS320C6000, que provee Texas Instruments. El código se implementa en el TMS320C6000 [39].

6.2. Transformada wavelet discreta

La transformada wavelet discreta (DWT) fue desarrollada para aplicar la transformada wavelet al mundo digital. Los bancos de filtros se usan para aproximar el comportamiento de la transformada wavelet continua. La señal es descompuesta con filtros pasa bajas y pasa altas. Los coeficientes de estos filtros son calculados usando análisis matemáticos [39]. Como se muestra en la Figura 6.1.

donde:

- LPd: Filtro de descomposición pasa bajas.
- HPd: Filtro de descomposición pasa altas.
- LPr: Filtro de reconstrucción pasa bajas.
- HPr: Filtro de reconstrucción pasa altas.

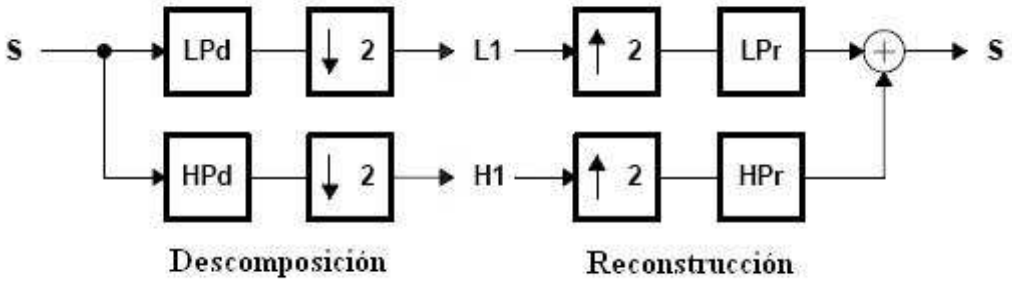

Figura 6.1: Transformada Wavelet Discreta.

6.3. Wavelets y reconstrucción perfecta de bancos de filtros

Los bancos de filtros descomponen la señal en componentes de baja y alta frecuencia. Los componentes de baja frecuencia usualmente contienen la mayor parte de la frecuencia de la señal. Esto es llamado aproximación. Los componentes de alta frecuencia contienen los *detalles* de la señal. La descomposición wavelet puede ser implementado usando filtros de bancos de dos canales. La idea principal es que la reconstrucción perfecta de filtros de bancos implementa expansiones de series de señales de tiempo discreto. Las Figuras 6.2 y 6.3 muestran los diagramas de descomposición y reconstrucción wavelet respectivamente.

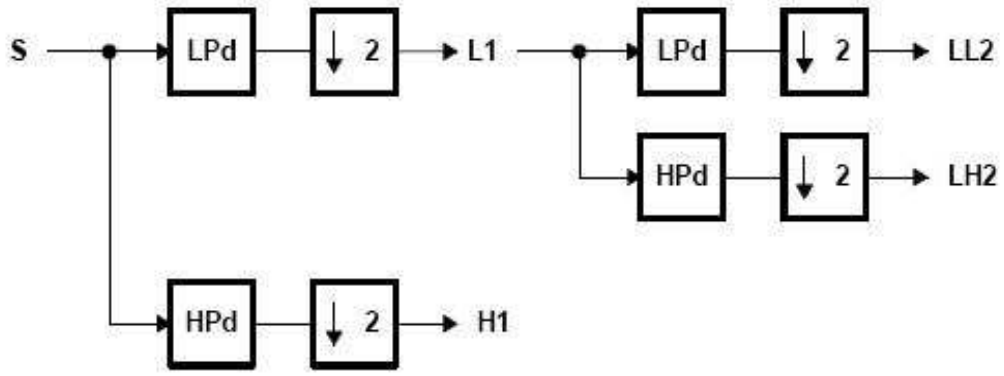

Figura 6.2: Descomposición Wavelet de dos niveles.

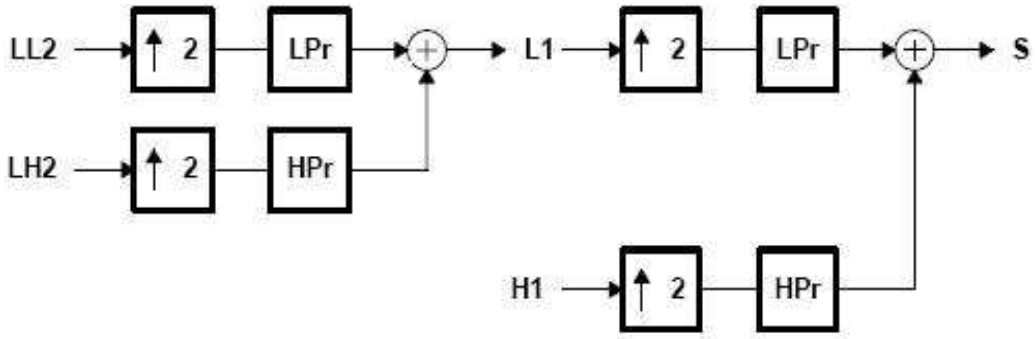

Figura 6.3: Reconstrucción Wavelet de dos niveles.

La entrada y la reconstrucción son iguales; esto es llamado reconstrucción perfecta. Las dos estructuras mas populares de descomposición son la de *pirámide* y la de *paquete wavelet*. La primera descompone solo una aproximación (componentes de baja frecuencia) mientras que la segunda descompone la aproximación y los detalles (componentes de baja y alta frecuencia) [39]. Para delimitar el tema en este trabajo se usara solo la descomposición de pirámide el cual se muestra en la Figura 6.4.

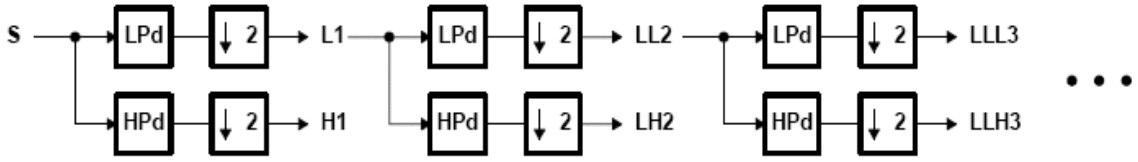

Figura 6.4: Descomposición de pirámide.

6.4. Wavelets y procesamiento de imágenes

Las wavelets han encontrado una gran variedad de aplicaciones en el campo de procesamiento de imágenes. El estándar JPEG 2000 usa wavelets para la compresión de imágenes. Otras aplicaciones de procesamiento de imágenes como reducción de ruido, detección de bordes y análisis de huellas digitales deben ser estudiadas en la literatura.

6.4.1. Descomposición de imágenes

En la descomposición wavelet de una imagen, la descomposición se realiza fila por fila y después columna por columna. Por ejemplo, una imagen de $M \times N$, hay que filtrar cada fila y después muestrear hacia abajo para obtener dos imágenes de $N \times (M/2)$. Después cada columna y se submuestrea la salida del filtro para obtener cuatro imágenes de $(N/2) \times (M/2)$.

De las cuatro subimágenes obtenidas como se ven en la Figura 6.5, la primera obtenida filtrando las columnas y las filas con un pasa bajas referida como una imagen LL. La otra se obtiene filtrando las filas con un pasa altas y las columnas con un pasa altas se refieren como imágenes LH. La otra que se obtiene filtrando las filas con un pasa bajas y las columnas con un pasa bajas se refieren como imágenes HL. La subimagen obtenida filtrando las columnas y las filas con un pasa altas referida como una imagen HH. Cada una de las imágenes obtenidas de esta manera pueden ser filtradas y sub muestreadas para obtener cuatro subimágenes mas. Como se muestra en la Figura 6.5.

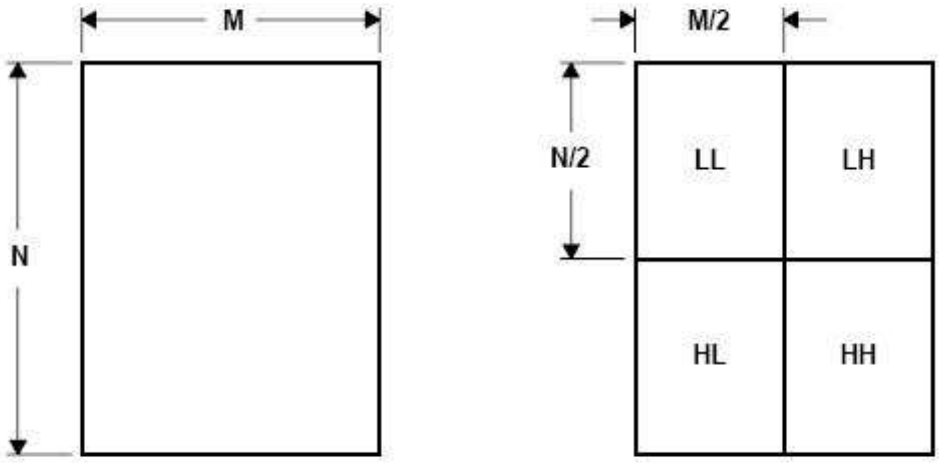

Figura 6.5: Imagen original de la descomposición wavelet 2-D.

Tres de las más populares formas de descomponer una imagen son: pirámide, sapcl y paquete wavelet, como se muestra en la Figura 6.6.

- En la estructura de pirámide, solo la subimagen LL se descompone después de cada descomposición en cuatro subimágenes más.
- En la estructura de descomposición de paquete wavelet, cada subimagen (LL, LH, HL, HH,) se descomponen después de cada descomposición.
- En la estructura de spacl, después del primer nivel de descomposición, cada subimagen es descompuesta en pequeñas subimágenes, y solo la subimagen LL es descompuesta [39].

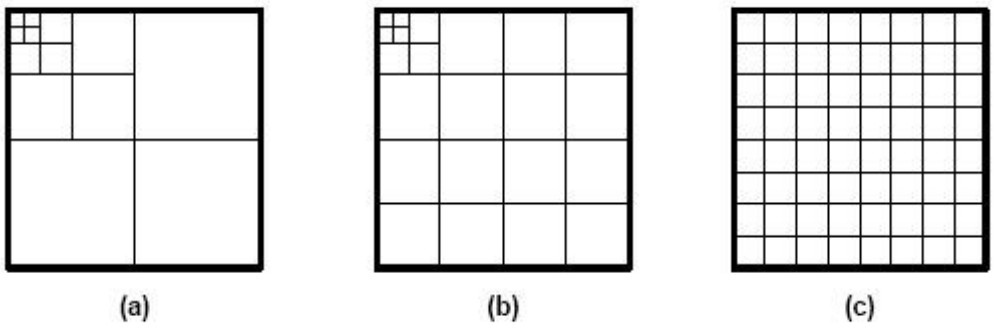

Figura 6.6: Tres estructuras de descomposición de una imagen: (a) Pirámide, (b) Sapcl y (c) Paquete wavelet.

6.5. Aplicaciones de wavelet de dos dimensiones

Los códigos siguientes, presentan aplicaciones del procesamiento digital de imágenes mediante wavelets, primero de una imagen de 256x256 y de imágenes tomadas desde una cámara desde el procesador digital de señales en tiempo real. Las imágenes tomadas serán de un péndulo invertido el cual se muestra en la Figura 6.7, el cual se mostro en el Capitulo 5 en la Figura 5.2.

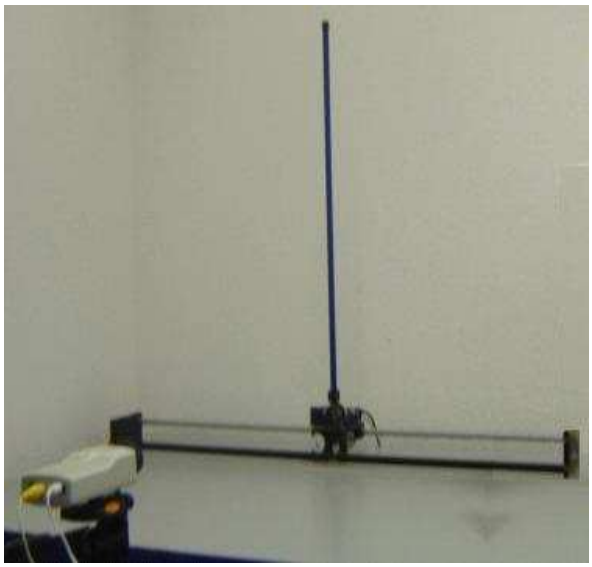

Figura 6.7: Imagen a tratar del péndulo invertido tomada por una cámara conectada a el DSP

6.5.1. Ejemplo de detección de bordes

En esta aplicación, la detección de bordes 2-D que se hace como se muestra en la Figura 6.8(a) y el resultado de la descomposición de un nivel muestra que los bordes se detectan como se muestra en la Figura 6.8(b).

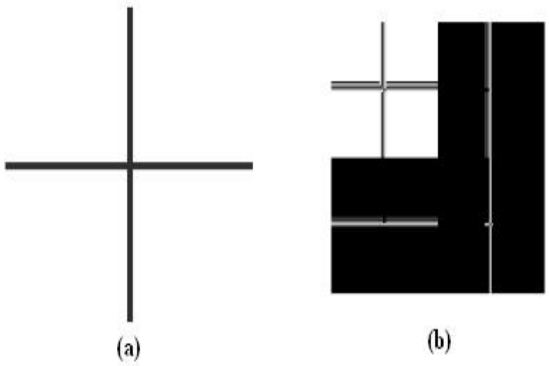

Figura 6.8: (a) Imagen usada en la aplicación de detección de bordes y (b) Detección de bordes 2-D de un nivel.

6.6. Aplicación sobre una sola imagen

Para la aplicación de la imagen de la Figura 6.7 se ordenan los datos de la matriz obtenida de 256x256 como se muestra en parte del siguiente código llamado `imagesample1`.

```
#pragma DATA_SECTION( goldhill, "DSP1")
short goldhill[128][128] = {

{163, 164, 164, 165, 165, 164, 164, 163, 163, 163, 163, 163, 163, 163, 163, 163,
 164, 164, 164, 163, 163, 162, 162, 162, 162, 162, 162, 161, 161, 160, 160, 160,
 160, 160, 160, 159, 159, 158, 158, 158, 159, 159, 159, 158, 158, 157, 157, 157,
 157, 157, 158, 157, 156, 155, 154, 153, 151, 151, 151, 152, 152, 153, 153, 153,
 156, 156, 156, 157, 157, 158, 158, 158, 158, 158, 158, 159, 159, 160, 160, 160,
 159, 159, 160, 161, 161, 162, 163, 163, 161, 161, 161, 162, 162, 163, 163, 163,
 165, 165, 165, 165, 165, 165, 165, 165, 164, 164, 165, 166, 166, 167, 168, 168,
 167, 167, 167, 167, 167, 167, 167, 167, 168, 168, 168, 168, 168, 168, 168, 168,
 169, 169, 170, 169, 170, 170, 169, 169, 176, 169, 161, 164, 169, 173, 171, 169,
 171, 170, 170, 171, 171, 172, 172, 172, 171, 172, 172, 173, 173, 172, 172, 171,
 173, 173, 173, 173, 173, 173, 173, 173, 174, 174, 174, 174, 174, 174, 174, 174,
 174, 174, 174, 173, 173, 172, 171, 171, 178, 177, 175, 174, 173, 174, 174, 175,
 176, 176, 176, 176, 176, 176, 176, 176, 177, 178, 178, 178, 178, 177, 176, 175,
 178, 178, 178, 178, 178, 178, 178, 178, 179, 179, 179, 179, 179, 179, 179, 179,
 179, 180, 180, 181, 181, 180, 180, 179, 181, 181, 181, 181, 181, 181, 181, 181,
```

```
 180, 181, 183, 184, 184, 183, 181, 180, 182, 182, 182, 182, 182, 182, 182, 182},

{163, 164, 164, 165, 165, 164, 164, 163, 163, 163, 163, 163, 163, 163, 163, 163,
 164, 164, 164, 163, 163, 162, 162, 162, 162, 162, 162, 161, 161, 160, 160, 160,
 160, 160, 160, 159, 159, 159, 158, 158, 159, 159, 159, 158, 158, 157, 157, 157,
 157, 157, 158, 157, 156, 155, 154, 153, 151, 151, 152, 152, 152, 153, 153, 153,
 156, 156, 156, 157, 157, 158, 158, 158, 158, 158, 158, 159, 159, 160, 160, 160,
 159, 159, 160, 161, 161, 162, 163, 163, 161, 161, 161, 162, 162, 163, 163, 163,
 165, 165, 165, 165, 165, 165, 165, 165, 164, 164, 165, 166, 166, 167, 168, 168,
 167, 167, 167, 167, 167, 167, 167, 167, 168, 168, 168, 168, 168, 168, 168, 168,
 169, 169, 170, 169, 170, 170, 169, 169, 167, 166, 166, 171, 173, 173, 172, 171,
 171, 170, 170, 171, 171, 172, 172, 172, 171, 172, 172, 173, 173, 172, 172, 171,
 173, 173, 173, 173, 173, 173, 173, 173, 174, 174, 174, 174, 174, 174, 174, 174,
 174, 174, 174, 174, 173, 173, 173, 173, 178, 177, 175, 174, 173, 174, 175, 175,
 176, 176, 176, 176, 176, 176, 176, 176, 177, 178, 178, 178, 178, 177, 176, 175,
 178, 178, 178, 178, 178, 178, 178, 178, 179, 179, 179, 179, 179, 179, 179, 179,
 179, 180, 180, 181, 181, 180, 180, 179, 181, 181, 181, 181, 181, 181, 181, 181,
 181, 182, 182, 183, 183, 182, 182, 181, 182, 182, 182, 182, 182, 182, 182, 182},
  .                                   .                                       .
  .                                   .                                       .
  .                                   .                                       .
{163,  16,  15,  13,  15,  16,  17,  18,  17,  14,  35,  56,  64,  31,   12, 15,
  16,  16,  15,  15,  15,  14,  14,  14,  14,  15,  19,  19,  13,  84,  122, 78,
  26,  16,  10,  12,  15,  13,  10,  8,    9,  10,  10,  11,  12,  13,  14,  13,
  12,  10,  12,  13,  14,  11,  13,  16,  13,  15,  17,  20,  20,  20,  19,  18,
  17,  18,  18,  19,  20,  21,  21,  22,  16,  17,  18,  20,  21,  23,  24,  26,
  21,  21,  22,  21,  19,  17,  17,  17,  15,  16,  17,  18,  20,  22,  24,  27,
  27,  28,  31,  34,  39,  44,  48,  50,  64,  67,  72,  77,  81,  86,  91,  95,
 100, 101, 106, 111, 116, 120, 123, 125, 132, 133, 136, 138, 142, 145, 147, 147,
 145, 144, 145, 146, 147, 147, 147, 146, 144, 144, 144, 143, 143, 142, 142, 142,
 141, 141, 142, 143, 143, 144, 145, 145, 146, 147, 147, 148, 148, 148, 149, 148,
 149, 149, 149, 150, 150, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151,
 150, 151, 151, 152, 152, 151, 151, 150, 151, 151, 151, 151, 151, 151, 151, 151,
 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151,
 149, 149, 149, 150, 150, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 150,
 151, 152, 152, 152, 152, 152, 152, 152, 151, 151, 151, 151, 151, 151, 151, 151,
 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151},

{163,  17,  15,  15,  15,  16,  17,  18,  12,  15,  20,  43,  83,  36,  17,  19,
  16,  16,  15,  15,  15,  13,  14,  14,  20,  22,  15,  20,  14,  39, 103, 126,
  38,  24,  13,  13,  16,  14,   8,   4,  10,  10,  11,  12,  13,  13,  14,  14,
  15,  8,    9,  16,  20,  15,  14,  16,  15,  16,  18,  19,  18,  15,  12,   8,
  17,  17,  17,  16,  15,  15,  14,  14,  17,  17,  16,  16,  16,  15,  15,  15,
  13,  16,  18,  18,  17,  17,  18,  19,  21,  19,  17,  17,  19,  21,  21,  21,
  20,  19,  18,  18,  18,  19,  20,  20,  22,  25,  29,  32,  34,  38,  42,  45,
```

```
  53,  57,  62,  70,  78,  84,  90,  92, 100, 101, 104, 109, 114, 119, 123, 124,
 133, 133, 135, 137, 139, 140, 141, 141, 144, 144, 144, 143, 143, 142, 142, 142,
 141, 141, 142, 143, 143, 144, 145, 145, 146, 147, 147, 148, 148, 148, 149, 148,
 149, 149, 149, 150, 150, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151,
 150, 151, 151, 152, 152, 151, 151, 150, 151, 151, 151, 151, 151, 151, 151, 151,
 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151,
 149, 149, 149, 150, 150, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 150,
 151, 152, 152, 152, 152, 152, 152, 152, 151, 151, 151, 151, 151, 151, 151, 151,
 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151, 151},
};
```

La cual se introduce en el programa código llamado `2D_wavelet.c` el cual se muestra a continuación y que utiliza como se ha estado manejando una wavelet de Daubechies de cuatro coeficientes y de nivel 2.

```
#include <stdio.h>
#include <stdlib.h>
#include<imagelib.h>
#include<wavelet.h>
#include "imagesample1.h"

#define WIDTH  256 #define HEIGHT 2568
#pragma DATA_SECTION(temp_wksp, ".wksp_array");
short temp_wksp[WIDTH];


void main( ) {
    int     i, j;
    short     **image;
    int     Width;

    Width = 256;
    // Allocate pointers to rows
    image = (short **)malloc((size_t)(Width*sizeof(short*)));
    if(!image) printf("allocation failure 1 in matrix()");

    for( i = 0; i < HEIGHT; i++ )
        image[i] = &goldhill[i][0];

    // Perfect Reconstruction of Pryamid Decomposition
    //=========================================================
    IMG_wave_decom_two_dim( image, temp_wksp, Width, Width, db4, 2 );
    IMG_wave_recon_two_dim( image, temp_wksp, Width, Width, db4, 2 );
    //---------------------------------------------------------
```

```
        // Edge Detection
    //=====================================================
    for( i = 0; i < HEIGHT; i++ )
        for( j = 0; j < WIDTH; j++ )
            image[i][j] = 255;
    for( i = (HEIGHT>>1)-1; i <=(HEIGHT>>1)+1; i++ )
        for( j = 0; j < WIDTH; j++ )
            image[i][j] = 50;
    for( j = (WIDTH>>1)-1; j <=(WIDTH>>1)+1; j++ )
        for( i = 0; i < HEIGHT; i++ )
            image[i][j] = 50;
     IMG_wave_decom_two_dim( image, temp_wksp, WIDTH, HEIGHT, db4, 2 );
    //-----------------------------------------------------
}
```

El cual se soporta mediante dos librerías llamadas `imagelib.h` y `wavelet.h`. El primero `imagelib.h` se sustenta los comandos utilizados en el programa anterior mientras que `wavelet.h`, emplea los coeficientes de los filtros de descomposición a usar. El resultado obtenido se muestra en la Figura 6.9

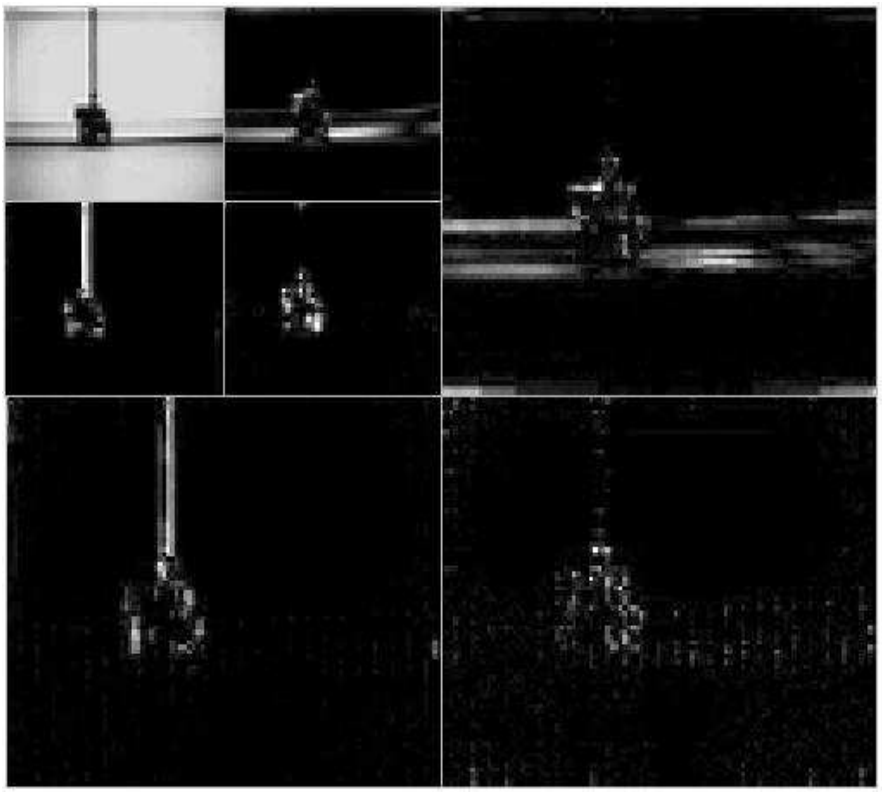

Figura 6.9: Imagen obtenida del procesador digital de señales para una sola imagen.

6.7. Aplicación sobre una imagen en tiempo real

Para el caso llevar acabo la transformada wavelet en tiempo real se necesita llevar acabo el código del siguiente programa denominado `2D_wave_trans`. El cual se muestra a continuación:

```
#include <stdio.h>
#include <stdlib.h>
#include <c6x.h>
/*--------------------------------------------------------------------------*/
/* Header files that use ImageLIB components */
/*--------------------------------------------------------------------------*/
#include "filters.h"
#include "csl.h"
#include "cache.h"
#include "dat.h"
#include "wavelet.h"
/*--------------------------------------------------------------------------*/
/* Las imágenes normales en la tarjeta de captura IDK  son de 640 por 480. /*
/* El conjunto de datos usados para la prueba es de 256 por 256. Estos /*
/* están definidos en IMG_ROWS e IMG_COLS, y TMG_ROWS y TMG_COLS son un /*
/* conjunto de 256 por 256 para el test_data usados. /*
/*--------------------------------------------------------------------------*/
#define IMG_COLS 640
#define IMG_ROWS 480
#define TMG_ROWS 256
#define TMG_COLS 256
/*--------------------------------------------------------------------------*/
/* Se crea una sección externa llamada ext_sect para la memoria externa /*
/* Se declaran tres arreglos en la memoria externa para el tamaño actuales /*
/* que se esperan en el escenario del IDK. Estos son los siguientes: /*
/* */
/* IMAGE DATA */
/* input_ch_data: Arreglo de 640 por 480 caracteres con 8 bit píxeles
*/
/* output_ch_data: Arreglo de 640 by 480 caracteres con 8 bit píxeles */
/* */
/* SCRATCH_PAD: scratch pad externo para almacenar temporalmente el resultado */
/* */
/* Scratch pad externo dos veces el tamaño de la imagen y 12 líneas por /*
/* contexto */
/* */
/* Un arreglo intermedio es un arreglo corto: */
/* Así que necesitamos espacio para almacenar arriba de 2 arreglos cortos /*
/* del tamaño de la imagen y 6 líneas de contexto */
```

```
/* Por lo tanto la memoria externa tiene IMG_COLS * (IMG_ROWS + 6) * 4 */
/* */
/* Consumo de memoria externa: */
/* */
/* input_ch_image: arreglo de 640 por 480 char --> 307200 --> 30 K bytes */
/* output_ch_image: arreglo 640 por 480 char --> 307200 --> 30 K bytes */
/* ext_mem: 2 arreglos de 646 por 480 shorts --> 1.24 M bytes */
/* */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Alinea arreglos de imágenes externas y scratch pad en los limites dword /*
/* y declara secciones. También declara los arreglos con el tamaño correcto /*
/*---------------------------------------------------------------------------*/
#pragma DATA_ALIGN(input_ch_data, 8);
#pragma DATA_ALIGN(output_ch_data, 8);
#pragma DATA_ALIGN(ext_scratch_pad,8);
#pragma DATA_SECTION(input_ch_data,".image:ext_sect");
#pragma DATA_SECTION(output_ch_data,".image:ext_sect");
#pragma DATA_SECTION(ext_scratch_pad,".image:ext_sect");
unsigned char input_ch_data[ IMG_COLS * IMG_ROWS];
unsigned char output_ch_data[IMG_COLS * IMG_ROWS];
char ext_scratch_pad[IMG_COLS * (IMG_ROWS + 6) * 2 * 2];
/*---------------------------------------------------------------------------*/
/* Crea una seccion en la memoria interna llamada chip_image que contendrá /*
/* varias líneas de la imagen externa DMA'ed dentro del trabajo interno /*
/* de los buffers. El tamaño del buffer interno esta destinado para el peor /*
/* caso de uso de todos los algoritmos combinados. Esto pasa en el algoritmo /*
/* de wavelet vertical donde se requieren 8 líneas de entrada para producir /*
/* dos líneas de salida. Por lo que el requerimiento de de memoria interna es /*
/* de 42 líneas de la imagen de entrada /*
/* Create section in internal memory called chip_image that will contain */
/* various lines of the external image DMA'ed into internal working */
/* buffers. The size of the internal buffer is allocated for the worst */
/* case usage of all algorithms combined. This happens in the vertical */
/* wavelet algorithm where 8 lines of input are required for producing 2 */
/* lines of output. Thus the internal memory requirement is 42 lines of */
/* the input image. */
/* */
/* 42 * 640 = 26880 bytes = 26.25 k Bytes */
/*---------------------------------------------------------------------------*/
#pragma DATA_ALIGN(int_scratch_pad, 8);
#pragma DATA_SECTION(int_scratch_pad, ".chip_image:int_sect");
char int_scratch_pad[ IMG_COLS * 21 *2];
/*---------------------------------------------------------------------------*/
/* Empieza el código principal: */
```

```
/*-------------------------------------------------------------------------*/
int main()
{
/*-------------------------------------------------------------------------*/
/* Las estructuras IMAGE para imágenes de entrada par e impar son /*
/* in_image_ev y in_image_od, y son las entradas del código wavelet.*/
/* Las estructuras IMAGE par alas imagines de salida son out_image. /*
/* Los detalles de salida SCRATCH_PAD y de entrada scratch pad. /*
/*-------------------------------------------------------------------------*/
IMAGE in_image_ev, in_image_od;
IMAGE out_image;
SCRATCH_PAD scratch_pad;
/*-------------------------------------------------------------------------*/
/* Los parámetros wavelet incluyen 8 filtros adaptables. Actualmente /*
/* se ejecuta solo una escala de descomposición. /*
/*-------------------------------------------------------------------------*/
WAVE_PARAMS wave_params; int err;
/*-------------------------------------------------------------------------*/
/* El conjunto de parámetros particulares para el campo par. /*
/* a) dirección de arranque b) columnas c) filas. En este caso la mitad /*
/* de las filas se asumen que son del campo par y la otra mitad se asume /*
/* que son del campo impar. /*
/*-------------------------------------------------------------------------*/
in_image_ev.img_data = input_ch_data;
in_image_ev.img_cols = TMG_COLS;
in_image_ev.img_rows = TMG_ROWS >> 1;
/*-------------------------------------------------------------------------*/
/* El conjunto de parámetros particulares para el campo impar. /*
/* a) dirección de arranque b) columnas c) filas. En este caso puesto que /*
/* puesto que una imagen continua, para simular los campos, el campo impar /*
/* es el conjunto que apunta a la segunda linea. /*
/*-------------------------------------------------------------------------*/
in_image_od.img_data = input_ch_data + TMG_COLS;
in_image_od.img_cols = TMG_COLS;
in_image_od.img_rows = TMG_ROWS >> 1;
/*-------------------------------------------------------------------------*/
/* El conjunto de parámetros para la salida de la imagen /*
/* a) dirección de arranque b) columnas c) filas. Las filas de la imagen /*
/* de salida será la suma de las filas de salida de las filas de entrada /*
/* del campo par e impar. /*
/*-------------------------------------------------------------------------*/
out_image.img_data = output_ch_data;
out_image.img_cols = TMG_COLS;
out_image.img_rows = TMG_ROWS;
/*-------------------------------------------------------------------------*/
```

```
/* El conjunto de parámetros particulares para el scratch pad externo y el /*
/* scratch pad interno a) scratch pad externo b) tamaño del scratch pad /*
/* externo c) scratch pad interno d) tamaño del scratch pad interno /*
/*-------------------------------------------------------------------------*/
scratch_pad.ext_data = ext_scratch_pad;
scratch_pad.ext_size = sizeof(ext_scratch_pad);
scratch_pad.int_data = int_scratch_pad;
scratch_pad.int_size = sizeof(int_scratch_pad);
/*-------------------------------------------------------------------------*/
/* El conjunto de parámetros particulares del codigo wavelet /*
/* a) Direccionamiento del filtro pasa bajas b) Direccionamiento del /*
/* filtro pasa altas c) Numero de escalas de descomposición /*
/* Actualmente solo se soporta una sola escala. /*
/*-------------------------------------------------------------------------*/
wave_params.qmf_ext = qmf_ext;
wave_params.mqmf_ext = mqmf_ext;
wave_params.scale = 1;
/*-------------------------------------------------------------------------*/
/* Inicializa CSL y coloca en modo L2 para ser la mitad del cache/half /*
/* SRAM. Permite el caching de esta región. Ejecuta una limpieza del cache /*
/* para remover cualquier etiqueta que hayan sido previamente ocultada.
/*-------------------------------------------------------------------------*/
CSL_Init();
CACHE_SetL2Mode(CACHE_32KCACHE);
CACHE_EnableCaching(CACHE_CE00);
CACHE_Clean(CACHE_L2, 0x80020000, 0xF2000);
/*-------------------------------------------------------------------------*/
/* Abre un canal para DMA a ser ejecutada, y conseguir el manejo a ser /*
/* pasado a un algoritmo. El algoritmo wavelet es llamado wavelet_codec /*
/* */
/* wavelet_codec(&in_image_ev, &in_image_od, &out_image, */
/* &scratch_pad, &wave_params); */
/* */
/* in_image_ev: apunta a la estructura para el campo par */
/* in_image_od: apunta a una estructura para el campo par */
/* out_image: apunta a una estructura para la imagen de salida */
/* scratch_pad: apunta a una estructura para scratch pad */
/* wave_params: apunta a una estructura para el código wavelet */
/* img_type: FLDS para los campos par e impar y PROG para progresivos */
/* Si img_type es PROG entonces in_image_od se ignora y */
/* se asuma que la la imagen es contigua empezando desde la */
/* dirección en in_image_ev. Si img_type es FLDS, entonces la mitad */
/* de las filas se asumen  en el campo par y la otra mitad en el /*
/* campo impar */
/*-------------------------------------------------------------------------*/
```

```
DAT_Open( 0, DAT_PRI_LOW, 0 );
wavelet_codec( &in_image_ev, &in_image_od, &out_image,
               &scratch_pad, &wave_params, FLDS);
DAT_Close( 0, DAT_PRI_LOW, 0);
return(1);
}
```

Los resultados de este código se muestran en la Figura 6.10.

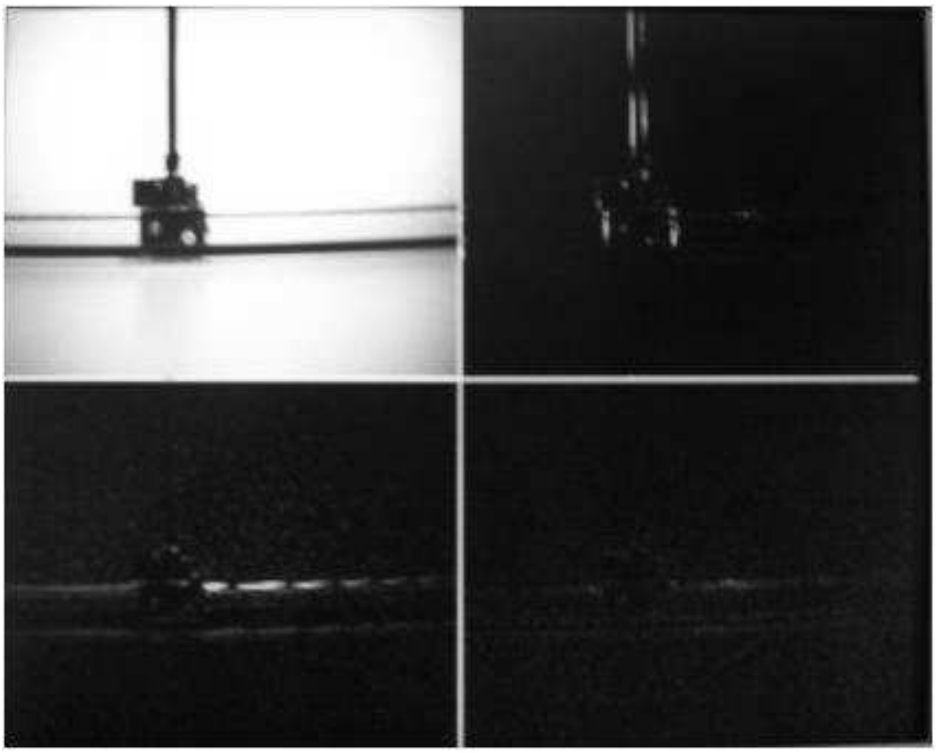

Figura 6.10: Imagen obtenida del procesador digital de señales para una imagen en tiempo real.

6.8. Comentarios

En este capítulo se observo que las simulaciones que se llevaron acabo en el capítulo anterior representaron una gran ayuda al llevarlas a la práctica puesto que permitió el desarrollo de mejores programas que al ser ejecutados bajo un procesador digital de señales y una tarjeta equipo de desarrollo de imágenes (IDK) conectadas a una cámara se obtuviera una mas detallada detección de rasgos, y obtención de patrones.

Capítulo 7

CONCLUSIONES

Mediante este trabajo de investigación se dio una breve introducción a los sistemas de procesamiento digital de señales e imágenes, en cuanto a su historia, características, procesos que se llevan acabo dentro del hardware especificado y conceptos básicos, los cuales se usan a lo largo de este documento.

Dando una breve pero concisa explicación del análisis de Fourier, para denotar sus funcionamiento vemos que el análisis de Fourier por sus muchas características y propiedades es una herramienta importante para el procesamiento de señales e imágenes. Para muchas señales, el análisis de Fourier es extremadamente útil porque las señales contienen frecuencia de alta importancia. Pero el análisis de Fourier tiene un inconveniente. Al transformar al dominio de la frecuencia, la información del tiempo se pierde. Cuando se busca la transformada de Fourier de una señal, es imposible decir cuando un evento en particular esta ocurriendo. Si las propiedades de la señal no varían mucho con el tiempo esto es, si la señal es no estacionaria, este inconveniente no es muy importante. Sin embargo, señales más interesantes contienen características no estacionarias ó transitorias: sentidos, tendencias, cambios abruptos, y comienzos de fines de eventos. Estas características a menudo son las partes más importantes de las señales, y el análisis de Fourier no es suficiente para detectarlos.

Para corregir todos estos inconvenientes en el análisis de Fourier se utiliza la técnica wavelet, la cual puede determinar cuando un evento en particular esta ocurriendo. Puesto que el análisis wavelet corrige las deficiencias del análisis de Fourier, y que por su gran multiresolución provee un análisis más fino. Siendo de gran ayuda para el procesamiento de imágenes.

Aunque existen diversas transformadas dentro del análisis, para el uso de señales digitales se emplea la transformada wavelet discreta y la transformada wavelet rápida las cuales son capaces de utilizar señales analógicas para representarlas en su forma digital.

La técnica de wavelets o también conocida como técnica de ondoletas, en la que la descomposición wavelet de una imagen, se realiza fila por fila y después columna por columna. Con esto se detectan los valores cambiantes dentro de la matriz de la

imagen y con ello la detección de bordes mediante la ubicación de valores que cambian bruscamente dentro de una escala de grises aplicada a la matriz de ésta. Con ello se logra un reconocimiento de patrones mediante la detección de bordes. El resultado obtenido se muestra en pantalla mediante sus coeficientes vertical, horizontal y diagonal.

Se hace la adquisición de imágenes y su procesamiento mediante procesador digital de señales (DSP) de la serie TMS320C6711 junto con una tarjeta de procesamiento digital de imágenes IDK (Imaging Development Kit). Las imágenes tomadas son de un sistema péndulo invertido que se encuentra bajo control. Todo esto se hace con la finalidad de obtener datos que ayuden en la implementación de controles que hacen uso de una cámara de visión como sensor (este tema esta fuera del alcance de este trabajo).

Por lo que de este trabajo de investigación se concluye que el procesamiento digital de imágenes mediante la técnica wavelet representa una gran ventaja con respecto al procesamiento con análisis de Fourier, además que las wavelets tienen una gran aplicación en diversas áreas: procesamiento de señales, procesamiento digital de imágenes, control, entre otras. Por ello, la técnica wavelet cubre un área muy extensa. Que trata los casos de tiempo continuo y también tiempo discreto. Provee de varias técnicas generales que pueden ser aplicadas a muchos problemas en procesamiento de señales, y por lo tanto tiene numerosas aplicaciones potenciales.

Apéndice A

ESPECIFICACIONES TÉCNICAS DEL DSP TMS320C6000

Las familias de dispositivos TMS320C62x / TMS320C67x, ejecutan un máximo de 8 instrucciones, de 32 bits, por ciclo. El CPU contiene 32 registros de propósito general, de 32 bits y 8 unidades funcionales. Estos dispositivos tienen un conjunto completo de herramientas de desarrollo y optimización, que incluyen un compilador C eficiente, un optimizador de ensamble para simplificar la planificación y programación del lenguaje ensamblador, y un depurador con interfase gráfica, basada en Windows, para visualizar las características de ejecución en el código fuente.

Además, contiene una tarjeta de emulación de hardware compatible con la interfase del emulador TI XDS510. Estas herramientas cumplen con los estándares 1149.11990, revisión de acceso a puerto y arquitectura de verificación de límites, de la IEEE.

Las características de los dispositivos C62x / C67x, incluyen:

1. Un CPU avanzado VLIW (very long instruction word) con 8 unidades funcionales, que incluyen 2 multiplicadores y 6 ALU's (unidades lógico aritméticas).

   a) Ejecuta un máximo de 8 instrucciones por ciclo, diez veces más que los DSP's típicos.

   b) Permite rápido tiempo de desarrollo en diseños con código RISC altamente efectivos.

2. Empaquetado de instrucción.

   a) Obtiene el tamaño del código equivalente a las 8 instrucciones ejecutadas en serie o en paralelo.

   b) Reduce el tamaño del código, el consumo de energía y el *fetch* del programa.

3. Ejecución condicional de todas las instrucciones.

   a) Reduce los saltos costosos.
   b) Incrementa el paralelismo para mantener un alto desempeño.

4. Ejecuta el código programado, en unidades funcionales independientes.
5. Proporciona soporte eficiente de memoria para una variedad de aplicaciones de 8, 16 y 32 bits de datos.
6. Maneja operaciones aritméticas de 40 bits, adicionando precisión extra a los coders y otras aplicaciones computacionalmente intensivas.
7. Proporciona soporte de normalización y saturación, en operaciones aritméticas claves.
8. Soporta operaciones comunes, halladas en aplicaciones de control y manipulación de datos, como: manipulación de campos, extracción de instrucción, activación, desactivación y conteo de bits.

Además, el C67x tiene las siguientes características:

- Un máximo de 1336 MIPS (millones de instrucciones por segundo), a 167 MHz.
- Un máximo de 1 G FLOPS (operaciones en punto flotante por segundo), a 167 MHz, para operaciones de precisión simple.
- Un máximo de 250 M FLOPS a 167 MHz, para operaciones de doble precisión.
- Un máximo de 688 M FLOPS a 167 MHz, para operaciones de multiplicación y acumulación.
- Soporte de hardware para operaciones de punto flotante, de simple y doble precisión (con formato IEEE).
- Multiplicación entera de 32 x 32 bits, con resultado de 32 o 64 bits.

Los dispositivos C62x / C67x tienen la siguiente variedad de opciones de memoria y periféricos:

- Amplia memoria RAM para ejecución rápida de algoritmos.
- Soporta interfases para memoria externa de 32 bits (SDRAM, SBSRAM, SRAM y otras memorias asíncronas), para aumentar el rango de memoria externa y maximizar el desempeño del sistema.
- Acceso a la memoria y periféricos de los dispositivos C62x / C67x a través del puerto *host*.

- Controlador del multicanal DMA.
- Puerto(s) serie multicanal.
- Timer(s) de 32 bits.

A.1. Arquitectura de los dispositivos TMS320C 62X/C67X

Los dispositivos 'C6211, 'C6711, 'C6701, 'C6201 y 'C6202 operan a 150, 150, 167, 200 y 250 MHz respectivamente. Todos estos DSP's ejecutan un máximo de 8 instrucciones por ciclo. El DSP 'C6211 es de punto fijo mientras que el 'C6711 es de punto flotante.

Los procesadores 'C62x/C67x consisten de tres partes: el CPU, los periféricos y la memoria. Ocho unidades funcionales operan en paralelo (seis ALU's y dos multiplicadores), con dos conjuntos similares de cuatro unidades funcionales básicas. Las unidades se comunican usando un camino cruzado entre dos clasificaciones de registros, cada una de los cuales contiene 16 registros de 32 bits. La Figura A.1 muestra el diagrama de bloque de los dispositivos TMS320C62x/C67x.

A.1.1. Unidad de procesamiento central (CPU)

El CPU contiene:

- Unidad *fetch* de programa.
- Unidad de despacho de instrucción.
- Unidad de decodificación de instrucción.
- 32 registros de 32 bits.
- Dos caminos de datos (*path*), cada uno con cuatro unidades funcionales.
- Registros de control.
- Lógica de control.
- Lógica de interrupción, emulación y prueba.

El CPU tiene dos caminos de datos (A y B), cada camino tiene cuatro unidades funcionales (.L, .S, .M y .D) y un archivo de registros que contiene 16 registros de 32 bits (*register file*). Las unidades funcionales ejecutan operaciones de lógica, corrimiento, multiplicación y direccionamiento de datos. Todas las instrucciones aceptan operaciones de carga y almacenamiento sobre los registros. Las dos unidades de direccionamiento de datos (.D1 y .D2) son exclusivamente responsables de toda la transferencia de datos entre los archivos de registros y la memoria.

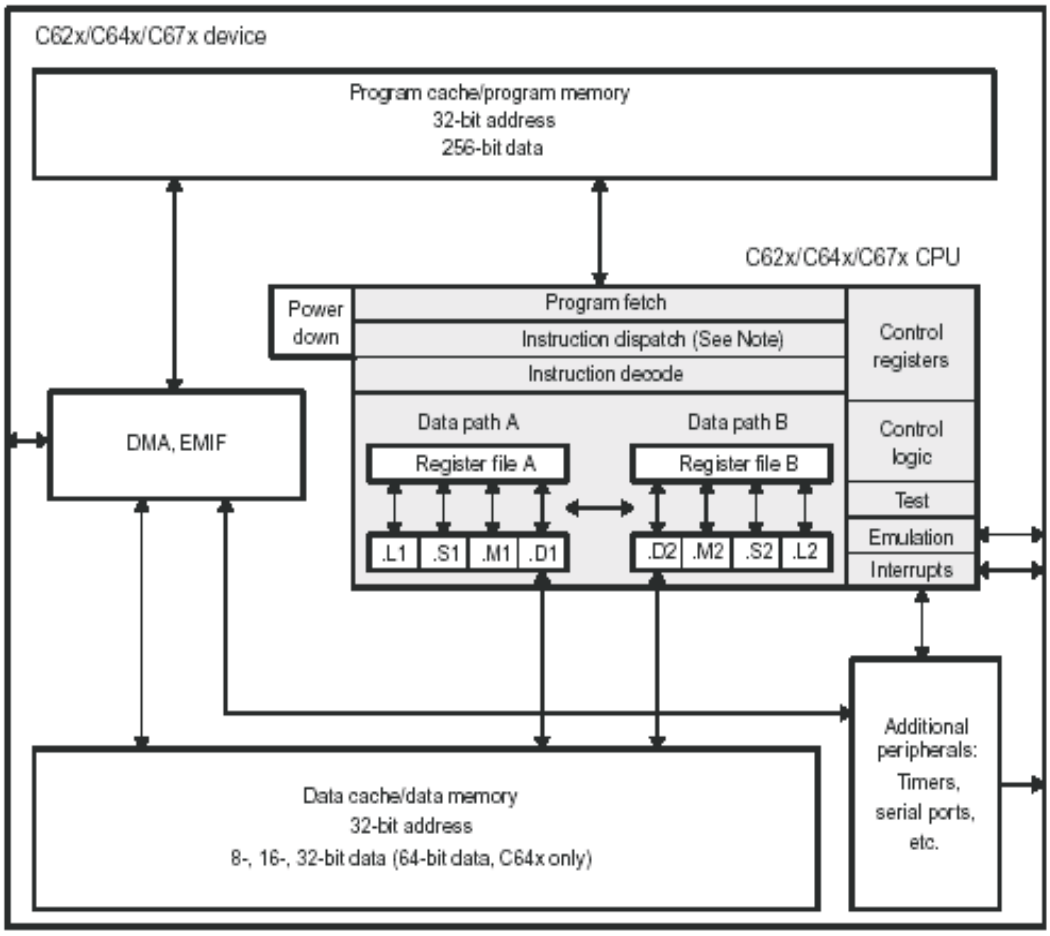

Figura A.1: Diagrama de bloques de TMS320C62x/C67x.

A.1.2. Caminos de datos del CPU

Los caminos de datos del CPU consisten de: dos archivos de registros de propósito general (A y B), ocho unidades funcionales (.L1, .L2, .S1, .S2, .M1, .M2, .D1 y .D2), dos caminos de lectura de memoria (LD1 y LD2), dos caminos de almacenamiento de memoria (ST1 y ST2), dos caminos cruzados entre los archivos de registros (1X y 2X) y dos caminos de direccionamiento de datos (DA1 y DA2). La Figura A.2 muestra el camino de los datos del CPU 'C67x.

A.1.3. Archivos de registros de propósito general (*register files*)

Hay dos archivos de registros de propósito general (A y B) en los caminos de datos del 'C62x/'C67x. Cada uno de esos archivos contiene 16 registros de 32 bits (A0-A15) para el archivo A y (B0-B15) para el archivo B. Los registros de propósito general

pueden ser usados para manejar datos o punteros de direccionamiento de estos. Los registros A1, A2, B0, B1 y B2 pueden ser utilizados como registros de condición. Los registros 4-A7 y B4-B7 pueden ser usados para el direccionamiento circular.

Los archivos de registros de propósito general soportan datos de 32 y 40 bits de punto fijo. Los datos de 32 bits, pueden estar contenidos en cualquier registro de propósito general. Los datos de 40 bits están contenidos en dos registros; los 32 bits menos significativos del dato (LSB) son colocados en un registro par y los restantes ocho bits más significativos (MSB) son colocados en los ocho bits menos significativos del registro próximo superior (que es siempre un registro impar). El 'C67x también usa ese par de registros para colocar valores de punto flotante de doble precisión de 64 bits.

A.1.4. Unidades funcionales

Las ocho unidades funcionales en los caminos de datos del 'C62x/C67x pueden ser divididas en dos grupos de cuatro; cada unidad funcional, en un camino de datos, es casi idéntica a la unidad correspondiente, en el otro camino de datos (p. ej. .L1 es muy similar a .L2). Las unidades funcionales son descritas en el Cuadro A.1.

La mayoría de los caminos en el CPU, soportan operaciones de 32 bits, y algunas soportan operaciones largas (40 bits). Cada unidad funcional tiene su propio puerto de escritura de 32 bits, en un archivo de registros de propósito general. Todas las unidades terminan en 1 (por ejemplo, .L1) cuando se refiere al archivo de registros A y en 2 cuando se refiere al archivo de registros B. Cada unidad funcional tiene dos puertos de lectura de 32 bits, para cada operando *src1* y *src2*. Cuatro unidades (.L1, .L2, .S1 y .S2) tienen un puerto extra de 8 bits para ejecutar operaciones de 40 bits. Debido a que cada unidad tiene su propio puerto de escritura de 32 bits, las ocho unidades pueden ser usadas en paralelo en cada ciclo.

A.1.5. Archivos de registros de control del TMS320 C62x/C67x

Una unidad (.S2) puede leer de y escribir hacia los registros de control. Mostrados en la Figura A.2. El Cuadro A.2, menciona y describe, los registros de control contenidos en el archivo de registros de control. A cada registro se accede con la instrucción MVC.

El 'C67x posee tres registros de configuración adicionales, para soportar operaciones de punto flotante (ver cuadro A.3). Los registros especifican los modos de redondeo de punto flotante para las unidades .M y .L. También contienen campos de bit para advertir si *src1* y *src2* son NaN (no es un número) o números desnormalizados. Además si resulta *overflow* o *underflow*, es inexacto, infinito o inválido. Hay campos que advierten si una división por cero fue efectuada o si una comparación fue ejecutada con un NaN.

Unidad Funcional	Operaciones en punto fijo	Operaciones en punto flotante
Unidad .L(.L1, .L2)	Operaciones aritméticas y comparación de 32 y 40 bits. Cuenta de 0's o 1's. Mas a la izquierda, para 32 bits. Normalización de 32 y 40 bits. Operaciones lógicas de 32 bits.	Operaciones aritméticas. Operaciones de conversión: DP→SP, INT→DP, INT→SP.
Unidad .S(.S1, .S2)	Operaciones aritméticas de 32 bits. Corrimientos de 32/40 bits y operaciones campos de bits en 32 bits. Operaciones lógicas de 32 bits. Saltos, generación de constantes. Transferencia de registros de/hacia registros (solamente .S2)	Comparación recíproca. Operaciones de raíz cuadrada. Operaciones de valor absoluto. Operaciones de Conversión SP a DP
Unidad .M(.M1, .M2)	Operaciones de multiplicación de 16x16 bits.	Operaciones de multiplicación de 32x32 bits. Operaciones de multiplicación de punto flotante.
Unidad .D(.D1, .D2)	Sumas, restas y cálculos de direccionamiento circular de 32 bits. Carga y almacenamiento con *offset* constante de 5 bits. Carga y almacenamiento con *offset* constante de 15 bits. (solo .D2)	Lectura de palabras dobles con *offset* constante de 5 bits.

Tabla A.1: Unidades funcionales y las operaciones realizadas

Abreviatura	Nombre	Descripción
AMR	Registro de modo de direccionamiento.	Especifica si utiliza direccionamiento lineal o circular para cada uno de los ocho registros; también contiene el tamaño para el direccionamiento circular.
CSR	Registro de control de estado.	Contiene el bit de interrupción global, los bits de control del cache y otros bits de control de estado diversos.
IFR	Registro de bandera de interrupción.	Despliega el estado de las interrupciones.
ISR	Registros para activar interrupción.	Permite activar interrupciones manualmente.
ICR	Registro para interrupción.	Permite limpiar interrupciones pendientes manualmente.
IER	Registro para retorno de interrupción	Permite habilitar / deshabilitar interrupciones individuales.
NRP	Puntero de retorno de interrupción no mascarable	Contiene la dirección de retorno de una interrupción no mascarable.
PCE1	Contador del programa, fase E1	Contiene la dirección del paquete *fetch* (contiene el paquete de ejecución del pipeline) en la etapa E1.

Tabla A.2: Registros de control

Abreviatura	Nombre	Descripción
FADCR	Registro de configuración del sumador del punto flotante	Específica el modo *underflow*, modo de redondeo, NaN y otras excepciones para la unidad .L
FAUCR	Registro de configuración auxiliar de punto flotante	Especifica modos de *underflow*, modos de redondeo y otras excepciones para la unidad .S
FMCR	Registro de configuración del multiplicador de punto flotante	Especifica modos de *underflow*, modos de redondeo y otras excepciones para la unidad .M

Tabla A.3: Registros de control extendido para el TMS320C67x

A.1.6. Caminos entre archivos de registros (*register files cross paths*)

Cada unidad funcional lee directamente de y escribe directamente hacia el archivo de registros, dentro de su propio camino de datos. Esto es, las unidades .L1, .S1, .D1 y .M1 escriben en el archivo de registros A y las unidades .L2, .S2, .D2 y .M2 escriben en el archivo de registros B. Los archivos de registros son conectados a las unidades funcionales del archivo de registros opuesto, a través de los caminos cruzados 1X y 2X. Esos caminos cruzados permiten a las unidades funcionales, de un camino de datos, acceder a operandos de 32 bits, del lado opuesto. El camino cruzado 1X permite a las unidades funcionales del camino de datos A, leer su operando fuente del archivo de registros B. El camino cruzado 2X permite a las unidades funcionales del camino de datos B, leer su operando fuente del archivo de registros A.

A.1.7. Caminos de memoria, cargas y almacenamiento

Hay dos caminos de 32 bits, para leer los datos de memoria en los registros de almacenamiento: LD1 para el archivo de registros A y LD2 para el archivo de registros B. El 'C67x también tiene un segundo camino de carga de 32 bits para ambos archivos de registros A y B. Este segundo camino permite a la instrucción LDDW leer simultáneamente dos registros de 32 bits en los lados A y B. Existen además dos caminos de 32 bits, ST1 y ST2, para almacenar valores de los registros a la memoria, para cada archivo de registros. Los caminos de lectura largos .L y .S son compartidos con los caminos de almacenamiento.

A.1.8. Caminos de direccionamiento de datos

Los caminos de direccionamiento de datos (DA1 y DA2) mostrados en la Figura (A.2) colocados fuera de las unidades .D, permiten generar direcciones de datos de un archivo de registros. Con eso se sostienen cargas y almacenamientos en memoria, desde el otro archivo de registros. Sin embargo, las cargas y almacenamientos ejecutados en paralelo, deben cargar a y del mismo archivo de registros. Aunque también existe la alternativa de que ambos usen un camino cruzado al registro opuesto.

A.1.9. Mapeo entre instrucciones y unidades funcionales

El cuadro (A.4) muestra el mapeo entre las instrucciones y las unidades funcionales para las instrucciones de punto fijo del TMS320C62x/C67x. El cuadro (A.5) muestra el mapeo entre las instrucciones y las unidades funcionales para las instrucciones de punto flotante del TMS320C67x.

Unidad .L	Unidad .M	Unidad .S		Unidad .D	
ABS	MPY	ADD	SET	ADD	STB(15-bitt offset) Solo .S2
ADD	MPYU	ADDK	SHL	ADDAB	STH(15-bitt offset) Solo .S2
ADDU	MPYUS	ADD2	SHR	ADDAH	STW(15-bitt offset) Solo .S3
AND	MPYSU	AND	SHRU	ADDAW	SUB
CMPEQ	MPYH	B disp	SHRL	LDB	SUBAB
CMPGT	MPYHU	B IRP	SUB	LDBU	SUBAH
CMPGTU	MPYHUS	B NRP	SUBU	LDH	SUBAW
CMPLT	MPYHSU	B reg	SUB2	LDHU	ZERO
CMPLTU	MPYHL	CLR	XOR	LDW	
LMBD	MPYHLU	EXT	ZERO	LDB	
MV	MPYHULS	EXTU		LDBU	
NEG	MPYHSLU	MV		LDH	
NORM	MPYLH	MVC		LDHU	
NOT	MPYLHU	MVK		LDW	
OR	MPYLUHS	MVKH		MV	
SADD	MPYLSHU	MVLKH		STB	
SAT	SMPY	NEG		STH	
SSUB	SMPYHL	NOT		STW	
SUB	SMPYLH	OR			
SUBU	SMPYH				
SUBC					
XOR					
ZERO					

Tabla A.4: Mapeo de instrucciones de punto fijo y unidades funcionales

Unidad .L	Unidad .M	Unidad .S	Unidad .D
ADDDP	MPYDP	ABSDP	ADDAD
ADDSP	MPYI	ABSSP	LDDW
DPINT	MPYID	CMPEQDP	
DPSP	MPYSP	CMPEQSP	
INTDP		CMPGTDP	
INTDPU	CMPGTSP		
INTSP		CMLTDP	
INTSPU	CMPLTSP		
SPINT		RCPDP	
SPTRUNC		RCPSP	
SUBDP		RSQRDP	
SUBSP		RSQRSP	
SPDP			

Tabla A.5: Mapeo de instrucciones de punto flotante y unidades funcionales

A.2. Modos de direccionamiento

Los modos de direccionamiento son lineales por default para los 'C62x y 'C67x aunque también existe el modo de direccionamiento circular. El modo de direccionamiento se especifica con el registro modo de direccionamiento (AMR).

Con todos los registros se puede ejecutar el direccionamiento lineal. Solo en ocho de ellos se puede ejecutar el direccionamiento circular: del A4 a A7 8que son usados por la unidad .D1) y del B4 a B7 (que son usados por la unidad D2). Ninguna otra unidad puede ejecutar direccionamiento circular.

Las instrucciones LDB/LDH/LDW, STB/STH/STW, ADDAB/ADDAH/ ADDAW, y SUBAB/SUBAH/SUBAW se apoyan en el registro AMR, para determinar que tipo de calculo del direccionamiento es ejecutado por esos registros.

Los CPUs 'C62x/'C67x tienen arquitectura de carga/almacenamiento, lo que significa que la única manera de acceder datos en memorias es con la instrucción de carga o almacenamiento. El cuadro (A.6) muestra la sintaxis de un direccionamiento indirecto, para su localización en memoria.

A.3. Interrupciones

Los CPUs 'C62x/'C67x tienen 14 interrupciones. Estas son *reset*, la interrupción no mascarable (NMI) e interrupciones de la 4 a la 15. Estas interrupciones corresponden a las señales `RESET`, NMI e INT4-INT15 respectivamente, sobre los limites del CPU. En

Tipo de direccionamiento	Ninguna Modificación del registro de dirección	Preincremento o predecremento del registro de dirección	Postincremento o postdecremento del registro de dirección
Registro Indirecto	*R	*++R	*R++
		*- - R	*R - -
Registro Relativo	*+R[ucst5]	*++R[ucst5]	*R++[ucst5]
	*-R[ucst5]	*- -R[ucst5]	*R - -[ucst5]
Base + Index	*+R[offsetR]	*++R[offsetR]	*R++[offsetR]
	*-R[offsetR]	*- -R[offsetR]	*R- -[offsetR]

Tabla A.6: Generación de direccionamiento indirecto para load/store

los mismos dispositivos 'C62x/'C67x, estas señales pueden estar ligadas directamente a los pines del dispositivo, conectando periféricos al chip, o pueden ser desactivadas permanentemente, cuando están ligadas e inactivas en el chip. Generalmente, `RESET` y NMI son conectadas directamente a los pines del dispositivo. Las características del servicio de interrupción incluyen:

- El pin IACK del CPU es usado para confirmar la recepción de una petición de interrupción.
- Los pines INUM0 INUM3 indican el vector de interrupción que esta siendo utilizado.
- Los vectores de interrupción son reubicables.
- Los vectores de interrupción consisten de un paquete *fetch*. Con los paquetes se proporciona un rápido servicio.

A.4. Periféricos

Los periféricos disponibles en los dispositivos TMS320C6000 se muestran en el cuadro (A.7). Los periféricos que son accesibles al usuario se configuran con un conjunto de registros de control delimitados en memoria. El controlador del bus de periféricos realiza el arbitraje para el acceso a los periféricos. La lógica de configuración de Boot esta conectada por señales externas y la lógica de baja energía es accesible directamente por el CPU. La Figura A.3 muestra un diagrama de bloques, con los periféricos de los dispositivos 'C6211/'C6711.

1. **Controlador DMA.** El controlador DMA transfiere datos entre rangos de direccionamiento en el mapa de memoria, sin intervención del CPU. El controlador DMA tiene cuatro canales programables y cinco canales auxiliares.

Periféricos	C6201	C6202	C6211	C6701	C6711
Controlador de acceso directo a memoria (DMA)	Y	Y	N	Y	N
Controlador de acceso directo a memoria mejorado (EDMA)	N	N	Y	N	Y
Interfase al Puerto Host (HPI)	Y	N	Y	Y	Y
Bus de Expansión	N	Y	N	N	N
Interfase de Memoria Externa (EMIF)	Y	Y	Y	Y	Y
Configuración del Boot	Y	Y	Y	Y	Y
Puertos Seriales de Multicanal (McBSPs)	2	3	2	2	2
Selector de Interrupción	Y	Y	Y	Y	Y
Timers de 32 - bits	2	2	2	2	2
Lógica de Energía Baja	Y	Y	Y	Y	Y

Tabla A.7: Periféricos de los dispositivos TMS320C6000

2. **Controlador EDMA.** El controlador EDMA mejora las mismas funciones del controlador DMA. El EDMA tiene dieciséis canales programables, así como un espacio de RAM para soportar múltiples configuraciones de futuras transferencias.

3. **HPI.** El HPI es un puerto paralelo por medio del cual, un procesador host puede acceder directamente al espacio en memoria del CPU. El dispositivo host tiene facilidad de acceso debido a que es el maestro de la interfase. El host y el CPU pueden intercambiar información a través de la memoria interna o externa. En suma, el host tiene acceso directo a los periféricos de memoria mapeada.

4. **Bus de expansión.** El bus de expansión es un reemplazo de para el HPI, así como una expansión del EMIF. La expansión proporciona dos áreas distintas de funcionalidad, (puerto host y puertos de Entrada/Salida) que pueden coexistir en un sistema. El puerto host del bus de expansión puede operar en modo asíncrono de forma (esclavo), similar al HPI o en modo síncrono (maestro/esclavo). Estos modos permiten la interfase de dispositivos para una variedad de protocolos del bus host. FIFOs síncronos y los dispositivos periféricos de Entrada/Salida asíncronos pueden conectarse al bus de expansión.

5. **EMIF.** El EMIF soporta una interfaz de baja adherencia (*glueless*) para varios dispositivos externos incluye:

   a) SDRAM de ráfaga síncrona (SBSRAM).

   b) DRAM síncrona (SDRAM).

   c) Dispositivos asíncronos, incluyendo SRAM,ROM y FIFOs.

   d) Dispositivo externo de memoria compartida.

6. **Configuración del Boot.** Los dispositivos TMS320C62x/C67x proporcionan una variedad de configuraciones del boot, que determinan las acciones de inicialización que ejecuta el DSP, después del reset del dispositivo. Estas incluyen: cargas de código de un espacio externo de ROM sobre el EMIF y cargas de código a través del HPI/bus de un host externo.

7. **McBSP.** El puerto serial multicanal con buffer (McBSP) esta basado en las interfases estándar del puerto serie, encontrada en los dispositivos con plataformas TMS320C2000 y 'C5000. Resumiendo, el puerto puede almacenar muestras seriales en un buffer de memoria automáticamente, con la ayuda del controlador DMA/EDMA. Este también tiene capacidad de multicanal, compatible con los estándares de conexión a redes T1, E1, SCSA y MVIP. Proporciona:

   a) Comunicación full-Duplex.

   b) Registros de datos de doble buffer para flujo continuo de datos.

   c) Tramado independiente y temporización para dispositivos y transmisión.

   d) Interfase directa a codecs estándar, chips de interfase analógica (AICs) y otros dispositivos A/D y D/A conectados en serie.

8. Tiene las siguientes capacidades:

   a) Interface directa a:

      1) Tramas T1/E1.
      2) Dispositivos conforme a ST - BUS^{TM}.
      3) Dispositivos conforme a IOM - 2.
      4) Dispositivos conforme a AC97.
      5) Dispositivos conforme a IIS.
      6) Dispositivos SPI^{TM}.

   b) Transmisión y recepción multicanal de 128 canales.

   c) Un selector del ancho del tamaño del dato, que incluye 8, 12, 16, 20, 24 y 32 bits.

   d) Ley - μ y Ley - A de compasión.

   e) Transferencia inicial de 8 bits con LSB(bit menos significativo)o MSB(bit más significativo).

   f) Polaridad programable para ambas tramas de sincronización y relojes de datos.

   g) Reloj interno altamente programable y generación de trama.

9. **TIMER.** Los dispositivos 'C6000 tienen dos timer de propósito general que son usados para:

   a) Eventos del timer.

   b) Eventos del contador.

   c) Generador de pulsos.

   d) Interrupción del CPU.

   e) Enviar eventos de sincronización a el contador DMA/EDMA.

10. **Selector de Interrupción.** El conjunto de periféricos del 'C6000'producen 14 a 16 fuentes de interrupción. El CPU tiene 12 interrupciones disponibles. El selector de interrupción permite elegir entre las 12 interrupciones, la que necesita su sistema. El selector de interrupción, también permite cambiar la polaridad de entrada para la interrupción externa.

11. **Lógica de bajo consumo de energía.** La lógica de bajo consumo de energía permite reducir el reloj para disminuir el consumo de energía. La mayoría de la potencia de operación de la lógica CMOS, se disipa durante la conmutación del circuito de un estado lógico a otro. Para prevenir algo o toda la lógica del chip de conmutación, se pueden realizar ahorros significativos de energía, sin perder datos ni contexto operacional.

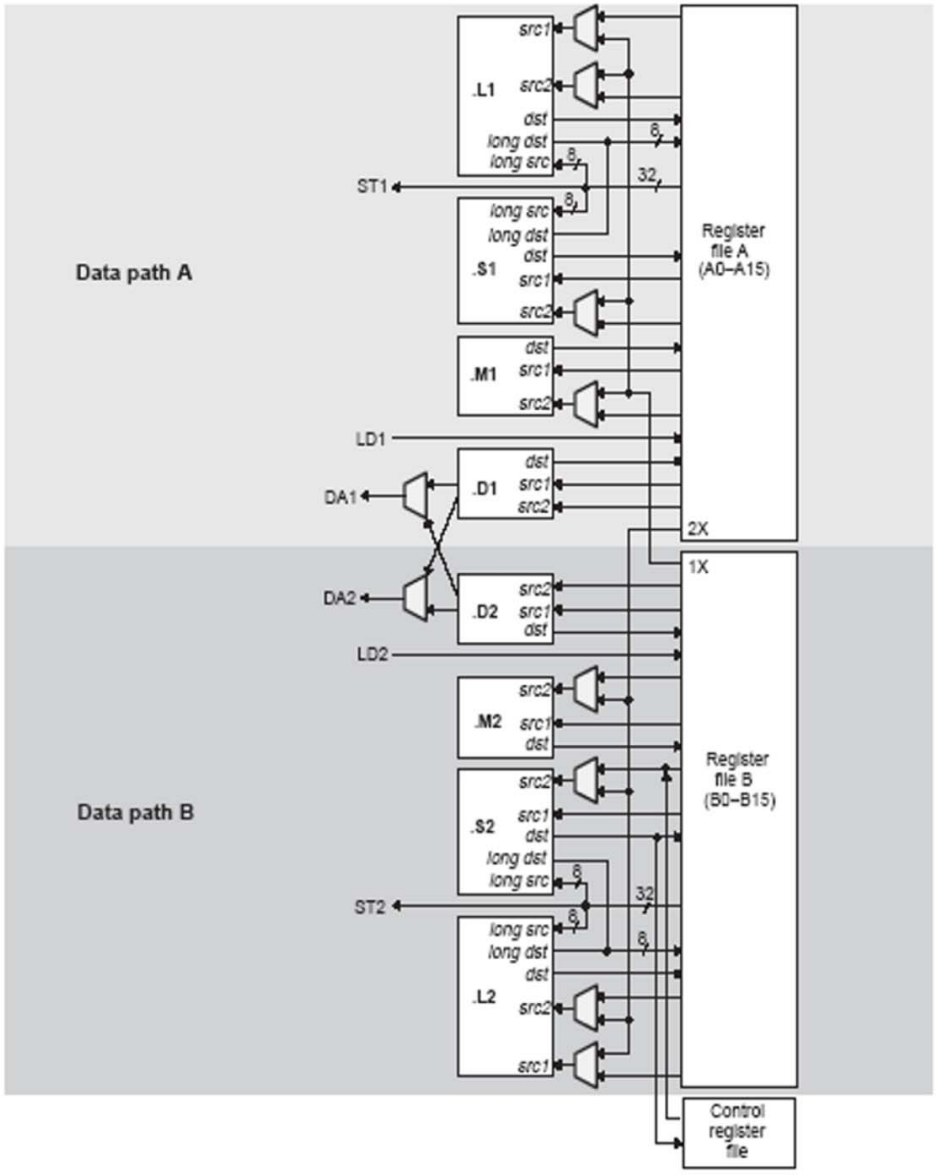

Figura A.2: Camino de datos del TMS320C67x.

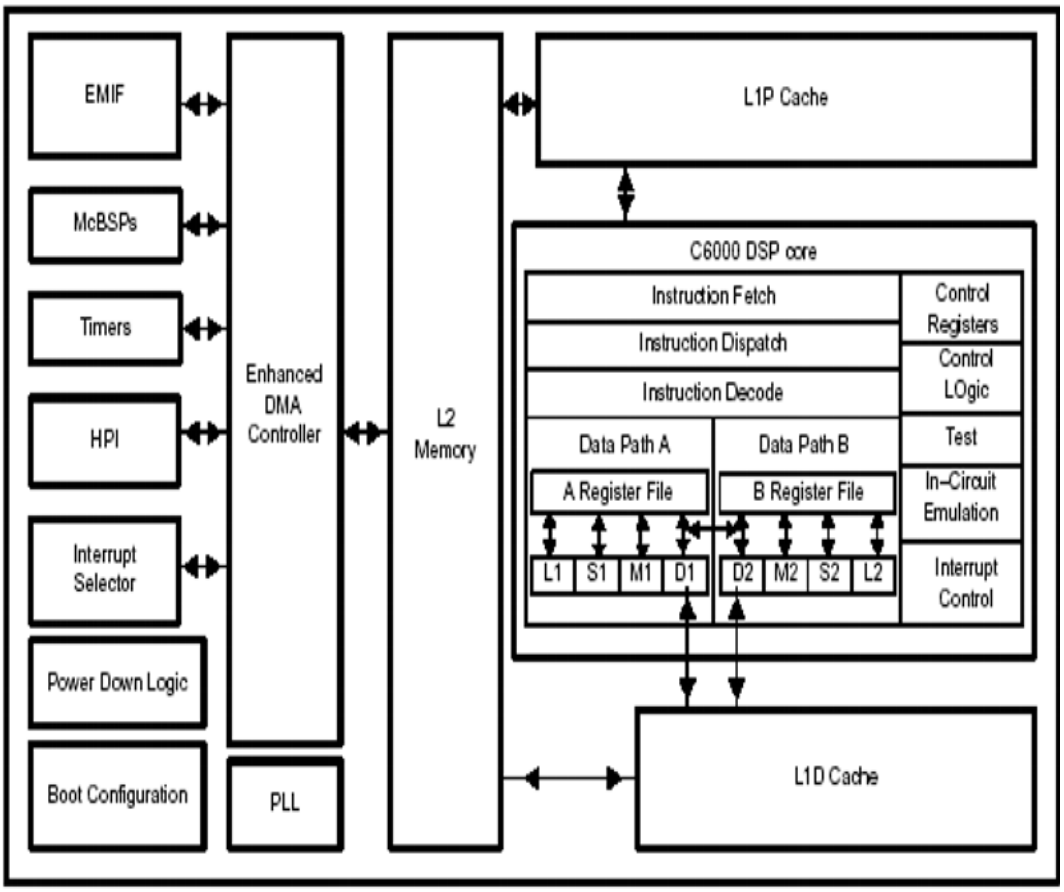

Figura A.3: Diagrama de bloques de los dispositivos TMS320C6211C6711.

Bibliografía

[1] Ahmad P. and Babak N. A., *A Novel Iris Recognition System Using Morphological Edge Detector and Wavelet Phase Features.* Control and Intelligent Processing Center of Excellence, 2000.

[2] Aydin T., Yemez Y., Anarim E. and Sankur B., *Multidirectional and multiscale edge detection via M-band wavelet transform.* IEEE Transactions on Image Processing, Vol. 5, No 9, 1996.

[3] Bai-Ling Z., Haihong Z., and Shuzhi S. G., *Face Recognition by Applying Wavelet Subband Representation and Kernel Associative Memory.* IEEE Transactions On Neural Networks, Vol. 15, No. 1, 2004.

[4] Billings S.A. and Hua-Liang Wei., *A new class of wavelet networks for nonlinear system identification.* IEEE Transactions on Neural Networks, Vol. 16, No. 4, 2005.

[5] Boles W. W. and Boashash B., *A Human Identification Technique Using Images of the Iris and Wavelet Transform.* IEEE Transactions On Signal Processing, Vol. 46, No. 4, 1998.

[6] Castleman K. R., *Digital image processing.* Pearson Education, 2002.

[7] Chuvieco E., *Fundamentos de la Teledetección Espacial.* Colección Monografías y Tratados GER, Ediciones Rialp, S.A., Madrid, 1990.

[8] Dogan G. E. and Omer N. G., *Power Quality Event Detection using Joint 2D-Wavelet Subspaces.* IEEE Transactions On Instrumentation And Measurement., 2002.

[9] Feng L., Tang Y. Y. and Yang L.H., *A wavelet approach to extracting contours of document images.* Fifth International Conference on Document Analysis and Recognition, 2002.

[10] Gonzales R. C. y Woods R. E., *Tratamiento digital de imágenes.*, Addison-Wesley / Diaz de Santos, 1998.

[11] Gonzales R. C. and Woods R. E., *Digital image processing.* Prentice Hall, 2002.

[12] Gonzales R. C. and Woods R. E., *Digital image processing using MATLAB.* Prentice Hall, 2003.

[13] Gopinath, R.A., Odegard, J.E., Burrus, C.S., *Optimal wavelet representation of signals and the wavelet sampling theorem.*, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 41, No. 4, 1994.

[14] Graps A., *An introduction to wavelets.* Computational Science and Engineering, IEEE, Vol. 3 , No. 3, 2003.

[15] Graps A., *An introduction to wavelets.* Computational Science and Engineering, IEEE, Vol. 2 , No. 2, 1995.

[16] Guy C. and Yehoshua Y. Z., *Wavelet-Based Multiresolution Stereo Vision.* 16th International Conference on Pattern Recognition (ICPR'02) - Vol. 3, 2002.

[17] Jian-xin X. and Ying T., *Nonlinear adaptive wavelet control using constructive wavelet networks.* American Control Conference, 2001.

[18] Jin Li. and C. C. Jay Kuo., *Automatic traget shape recognition via deformable wavelet templates.* Signal and image instituteand Department of electrical engineering systems. University of Southern Californa, 2002.

[19] Joao V. B. Soares, Jorge J. G. Leandro, Roberto M. Cesar-Jr., Herbert F. Jelinek, and Michael J. Cree., *Retinal Vessel Segmentation Using the 2-D Morlet Wavelet and Supervised Classification.* IEEE for possible publication, 2005.

[20] Karrakchou M. and Li W. *Optimal ramp edge detection by orthogonal wavelet transform.* Symposium on Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International, 2002.

[21] Karras D.A., Karkanis, S.A. and Mertzios, B.G., *A classification approach to image structure segmentation based on the wavelet transform.* EUROMICRO 97. New Frontiers of Information Technology. Short Contributions. Proceedings of the 23rd Euromicro Conference, 1997.

[22] Lei Zhang. and Bao P., *A wavelet-based edge detection method by scale multiplication.* 16th International Conference on Pattern Recognition, 2002.

[23] Oppenheim A. V., Schafer R. W. and Buck J. R., *Tratamiento de señales en tiempo discreto*, Prentice Hall, 2000.

[24] Parimala T., Ashfaq A. K., Gerd H. and Guang R. G., *A fine-grain load-adaptive algorithm of the 2D discrete wavelet transform for multithreaded architectures.* Journal of Parallel and Distributing Company, 2003.

[25] Parker J. R., *Algorithms for image processing and computer vision.* Wiley Computer publishing, 2002.

[26] Pittner, S. and Kamarthi, S. V., *Feature Extraction From Wavelet Coefficients for Pattern Recognition Tasks.* IEEE Transactions on pattern analysis and machine intelligence, Vol. 21, No. 1, 1999.

[27] Platero C., *Introducción al procesamiento digital de señales.* Universidad Politécnica de Madrid, 1999.

[28] Proakis J. G. and Manolakis D. G., *Tratamiento digital de señales.* Prentice hall, Madrid 1998.

[29] Rioul O. and Vetterli M., *Wavelets and signal processing.* IEEE SP Magazine, 1991.

[30] Rodenas J.A. and Garello R., *Internal wave detection and wavelength estimation from ERS-1 SAR images via wavelet analysis.* Seventh International Conference on Electronic Engineering in Oceanography, 1997. Technology Transfer from Research to Industry., 2002.

[31] Rodrigues J. and Du Buf, J. M. H., *Improved line/edge detection and visual reconstruction.* University of Algarve - Faro, Portugal, 2002.

[32] Shashikiran, G., *Wavelets in the astronomical context.* Physical Research Laboratory, 2003.

[33] Siddique J.I. and Barner, K.E., *Wavelet-based multiresolution edge detection utilizing gray level edge maps.* 1998 International Conference on Image Processing, 2002.

[34] Strickland R.N. and He Il Hahn., *Wavelet transform methods for object detection and recovery* IEEE Transactions on Image Processing, Vol. 6 , No. 5, 1997.

[35] Vinay K. I. and Proakis J. G., *Digital signal processor using MATLAB v. 4.0.* PWS Publishing Company, 1997.

[36] Young-Hyun B., Oh-Sung B. and Sung-Rung M., *Image edge detection using adaptive morphology Meyer wavelet-CNN.* International Joint Conference on Neural Networks, 2003.

[37] MATLAB Mathworks, version 6.5.0.180913a R13, Junio 2002.

[38] http://www7.nationalacademies.org/spanishbeyonddiscovery/mat_008276-04.html

[39] *TMS320C6000 Imaging Developer's Kit (IDK) User's Guide*, Texas Instruments Incorporated, 2001.

[40] CPS *Code composer studio version 2.0.1*, Texas Instruments Incorporated, 2001.