Técnicas de Compresión de Imágenes Aplicables a un Ambiente de Oficina Distribuido

Autor: Lic. Claudia Cecilia Russo

Director : Ing. Armando De Giusti

INDICE

Objetivo

Se realizó un análisis de las tecnologías actuales de compresión de datos, especialmente compresión de imágenes aplicables en transmisión multimedial en ambientes de oficinas distribuído.

En particular se comparan las técnicas JPEG estándar, JPEG modificado, transformada BTW y transformada de wavelet, desde el punto de vista de la eficiencia de compresión en relación con el grado de pérdida producido.

Se investigó e implementó un algoritmo de particionamiento adaptivo aplicable en JPEG y la mejora obtenible a partir de él, comparando con resultados publicados en la bibliografía actual.

Estructura de la Tesis

En el Capítulo I: "*Compresión de datos. Conceptos y técnicas básicas*" se introduce a los conceptos básicos de compresión de datos presentando algunos algoritmos clásicos. Se hace una breve descripción de los diferentes modos de compresión, con pérdida o sin pérdida, se pesenta una clasificación de algoritmos.

En el Capítulo II "*JPEG (Joint Photografics Expert's Group*" se describe el método estándar propuesto por la ISO.

En el Capítulo III "*MPEG (Motion Picture Expert Group*" se describe el método MPEG utilizado para la compresión de cuadros de video., además de las normas MPEG-1, MPEG-2.

En el Capítulo IV: "*Transformada de wavelet*" se presenta Fourier como antecesor de las wavelet. Se muestran las nociones básicas para el manejo y comprensión de las wavelet como método de compresión y trataremos de cuantificar la pérdida producida en el compresor.

En el Capítulo VI: "*Transformada de Burrows Wheeler (BWT)*" si bien es un método aplicable a texto es un método muy interesante por sus resultados de compresión y por su manera de manejar la información.

En el Capítulo VII: "*JPEG con Particionamiento Adaptivo*" se presenta una variación del método estándar JPEG, donde se obtienen mejoras en los radios de compresión obtenidos en las pruebas realizadas. Se realiza una comparación con el JPEG estándar.

En el Capítulo correspondiente a "*Análisis Comparativo. Experiencias*" se presentan las comparaciones de las aplicaciones realizadas con la transformada de

wavelet y el método JPEG, al igual que con el método JPEG adaptivo presentado en el capítulo anterior.

En el Capítulo de "*Conclusiones finales*" se presenta un análisis más exhaustivo de los radios alcanzados por los métodos presentados y se realizan conclusiones puntuales de los mismos.

Introducción

Los sistemas de computadoras personales, han llegado a ser portables, además de ser más pequeñas y más potentes en velocidad de procesamiento y capacidad de memoria. Las interfaces gráficas conducen en forma experta al usuario a través de las entradas y salidas de programas de aplicación con la simplicidad de apuntar y cliclear.

Como consecuencia, las computadoras han llegado a ser consideradas como ayudantes de la productividad en lugar de meras herramientas de oficina.

La descentralización de las organizaciones en la última década condujo, seguramente al advenimiento de la computadora personales. El crecimiento de las computadoras personales de escritorio y de las redes de área local enlazando grupos de trabajo en localidades geográficamente dispersas, alimentó la demanda de mayor conectividad para lugares internos, así como en las redes externas. Redes de valor añadido ofrecían servicios como mensajería electrónica e intercambio electrónico de datos (texto, imagen, etc.), que posibilitan que organizaciones dispares se comunicasen electrónicamente entre sí.

La computadora, el teléfono y la televisión han convergido hacia una tecnología digital de acceso sencillo. Junto con esta convergencia ha llegado la capacidad de crear, manipular, almacenar y transferir grandes cantidades de medios digitales por las redes. Las redes han incrementado enormemente su ancho de banda y velocidad para soportar pesados volúmenes de tráfico, que combinados con las nuevas tecnologías como, fibra óptica, Ethernet rápida, retransmisión de tramas y modo de transferencia asíncrono (ATM), están preparando el camino hacia un sistema de redes de comunicaciones de datos de alta velocidad, fiable y robusto. Esta red central virtual posibilitará la tecnología para la participación interactiva y autodirigida en el uso de medios electrónicos para adquirir información, productos y servicios.

La documentación en soporte de papel es, en la mayor parte de las empresas el único medio legal para avalar operaciones. Sin embargo sería deseable poder almacenarlas en un medio magnético para luego, en forma eficaz poder acceder a ella.

El principal inconveniente de un documento en papel (texto o imagen) es que requiere ser procesado manualmente. Una vez impreso en papel es necesario enviarlo físicamente a la sección que corresponda. El tiempo de traslado del documento es dependiente de la distancia y de los medios disponibles entre origen y destino. Cuando el receptor lo recibe deberá procesarlo.

Es deseable pensar en tener toda la información posible digitalizada y que cualquier área de la empresa tenga acceso en un tiempo razonable a la dicha información, hoy en día es un tema de estudio dentro de las empresas el rápido acceso de la información. Esto evitará introducir errores por la transmisión por interpretación errónea, por omisión e incluso por extravío de todo el documento.

Una piedra angular de la era de la información es la posibilidad de suministrar información sin considerar cuándo y dónde se hace la demanda. El acceso independiente del tiempo y del lugar a las necesidades de información, ha conducido ha sistemas de redes y hacia formas simplificadas de localizar y recuperar la información.

Aunque los medios utilizados para el almacenamiento digital de información son cada vez más amplios, razones de economía y de velocidad de acceso parecen recomendar la utilización de técnicas que permitan la reducción del espacio preciso para describir información.

También es importante considerar el volumen de datos a utilizar en el caso de la transmisión de datos. En estas condiciones, un mayor volumen se traduce en un tiempo de transmisión mayor, lo que, a su vez, redunda en un coste superior, una ocupación mayor del canal y, lo que es peor aún, una creciente probabilidad de error en la transmisión.

La tendencia hacia la miniaturización y los requerimientos cada vez más rigurosos (capacidad de memoria, rápido acceso, etc.), nos hace pensar en técnicas basadas en la manipulación de la representación de los datos pero preservando el significado de los mismos. Estamos hablando de Compresión de datos (técnica semántica).

Se introducirán los conceptos de compresión ilustrándose los métodos más habituales y estableciéndose un paralelismo entre cada método y sus aplicaciones.

En lo que respecta a la compresión de imágenes, se analizarán los métodos basados en la el contexto de los mismos. La compresión de imágenes juega un rol crucial en aplicaciones muy diversas e importantes como teleconferencias, censado remoto (uso de maquinaria satelital para aplicaciones climáticas y recursos terrestres), documentos e imágenes médicas, etc. En resumen, un número cada vez mayor de aplicaciones dependen de una manipulación, almacenamiento y transmisión eficiente de imágenes binarias, en escala de grises y color.

La compresión de una imagen, según se utilice un algoritmo u otro, puede o no preservar el cien por cien de la información contenida. En cualquier caso, de lo que se trata es de eliminar cualquier redundancia presente en la imagen, con el menor nivel de degradación posible.

Ya el simple hecho de digitalizar una imagen supone incrementar de forma considerable el ancho de banda de la misma. Si bien el tratamiento digital ofrece ventajas más que discutibles, su contrapartida es, precisamente, este gran incremento en el ancho de banda requerido. Así, una imagen típica de televisión se transmite con un ancho de banda de unos 5 Mhz., mientras que, digitalizando la misma en los, como máximo 625x830 pixeles que la componen, a 8 bits cada uno, requeriría alrededor de 40 Mhz.. Es obvio que el proceso de digitalización introduce una gran redundancia que es preciso eliminar.

La problemática principal de la compresión se centra en que no existe un método universal válido para cualquier imagen genérica. El método a aplicar en cada caso es fuertemente dependiente de la imagen concreta, y el grado máximo de compresión alcanzable depende directamente de la misma.

Compresión. Conceptos y técnicas básicas

Introducción y Terminología

Compresión lógica

Cuando una base de datos es diseñada, uno de los pasos es obtener la mayor reducción de datos posibles. Esta reducción resulta de la eliminación de campos de información redundantes, representando los campos restantes con la menor cantidad de indicadores lógicos posibles.

Muchos métodos de compresión lógica de datos pueden ser considerados durante el proceso de análisis de la base de datos. Cada método puede resultar en distintos grados de reducción de almacenamiento. Por consiguiente, cuando la base de datos o una porción de la base de datos comprimida lógicamente es transmitida entre distintas locaciones, el tiempo de transmisión de datos es reducido ya que los datos fueron reducidos.

Compresión física

La compresión física puede ser vista como un proceso de reducción de la cantidad de datos antes de entrar en un canal de transmisión y la expansión de los mismos en su formato original al llegar a su destino final. Aunque ambas compresiones pueden resultar en reducciones de tiempo de transmisión, existen diferencias de aplicación entre las dos técnicas.

Terminología

La compresión de un bloque de datos original es llamada proceso de codificación, y el código resultante es llamado bloque de datos comprimido. Inversamente a este proceso, el bloque de datos comprimido es descomprimido para reproducir el bloque de datos original. Por lo tanto para nosotros es exactamente lo mismo hablar del bloque de datos original o del bloque de datos descomprimido.

El grado de reducción obtenido como resultado del proceso de compresión es conocido como "*Radio de Compresión*". Este grado mide la cantidad de datos comprimidos en comparación con la cantidad de datos originales, tal que:

Radio de Compresión = Long. del bloque / Long. bloq. comp.

Es obvio que mientras mayor sea el radio de compresión más efectiva es la técnica empleada. La técnica empleada no será efectiva si el radio de compresión es menor o igual a 1.

Cuando los datos son comprimidos, el radio de compresión varia en proporción a la susceptibilidad de los datos a el/los algoritmo/s usado/s. Así, se debe concentrar la atención sobre un radio de compresión promedio y no sobre un radio alcanzado con un archivo en particular. En general, buenos algoritmos pueden alcanzar un radio de compresión promedio de 1,5 mientras que excelentes algoritmos basados sobre técnicas de procesamiento sofisticados alcanzarán un promedio de radio de compresión mayor a 2.0 (para archivos de texto) [Nelson, 1991].

Técnicas básicas de Compresión de datos

Las técnicas de compresión de datos tienen el objetivo de aplicar una transformación m'=f(m) que, aprovechando la redundancia del dato m, minimice el mensaje m'.

En general, cuanta más información se tenga del objeto a comprimir, más eficiente será el método de compresión, dado que la codificación puede estar basada en caracteres simples, bloques de dos, tres o más caracteres, estructuras completas de un texto (palabras o párrafos), bits, bloques de bits, etc. [Held, 1994].

Por ejemplo, si el algoritmo tiene en cuenta la sintaxis de un lenguaje para comprimir un programa, entonces se obtendrán mejores logros codificando en función de las palabras claves. En imágenes se puede tener en cuenta las zonas con igual color, textura, etc.

Esto sugiere que en algunos casos sea necesario un estudio previo del objeto a comprimir para lograr mejor compresión aplicando el método que más se ajusta.

Desde el punto de vista de la transmisión de información se puede ver al conjunto de datos como una secuencia de símbolos S_1, S_2, ..., S_n. Estos símbolos forman un conjunto, posiblemente infinito.

Los distintos métodos se pueden agrupar en tres clases, según el concepto en el que se basen:

- ✓ Cantidad de símbolos del conjunto.
- ✓ Frecuencia de aparición de los símbolos en el texto.

- ✓ Contexto en el que aparecen los símbolos.

Cantidad de símbolos del conjunto.

En algunas aplicaciones los datos pueden reducirse a un conjunto finito expresado en ASCII.

Por ejemplo en un proyecto de automatización de una empresa que comercializa libros, los títulos de los mismos pueden ser representados por un conjunto finito de símbolos.

Si un título standard tiene N caracteres, expresado en ASCII requerirá 7*N bits. De las 2^{7*N} combinaciones posibles sólo serán utilizadas como mucho 2^{25} (más de treinta y tres millones de ejemplares). Si a cada libro se le asigna un número de secuencia, la cantidad de bits necesarios por ejemplar se reduce de 7*N a veinticinco o menos. Nótese que N (cantidad de caracteres por título) es del orden de 10 a 100 [Mallat].

Frecuencia de aparición de los símbolos en el texto.

En la mayoría de los textos algunos símbolos o letras aparecen más a menudo que otros. Esta observación sugiere un esquema de codificación en el cual a los símbolos más comunes les sean asignados códigos cortos, y a los símbolos menos frecuentes, códigos largos.

Contexto en el que aparecen los símbolos:

El método de Huffman asume que la probabilidad de que aparezca un símbolo es independiente de su predecesor inmediato. Un esquema más complicado podría ser determinar la probabilidad condicional de cada símbolo dado su predecesor. Para el alfabeto, esto implica tener 26 tablas, una para la frecuencia de distribución de cada letra seguida de una A, otra para la B, etc.

Si hay una fuerte relación entre los símbolos y sus sucesores, el uso de este método logrará buenos resultados, aún si la distribución de los símbolos es plana (sin tener en cuenta su predecesor).

La desventaja del método de la probabilidad condicional es la gran cantidad de tablas requeridas. Si hay k símbolos, las tablas tendrán k^2 entradas.

Existe otro método que puede ser usado para codificar string de bits que contienen mayor cantidad de ceros que de unos. Esta situación se da en transmisión de matrices ralas e imágenes.

Cada código representa la cantidad de ceros que hay entre dos unos consecutivos. Para manejar sucesiones largas de ceros, el símbolo que consiste en todos unos significa que la verdadera distancia es 2^{k-1} más el valor del o los siguientes símbolos.

Por ejemplo, el string de bits

0001000001000000100000000000001000000100010000000110100000 1

está formado por secuencias de longitud 3, 5, 6, 14, 6, 3, 7, 0, 1 y 5. Esto podría ser codificado usando símbolos de tres bits como:

011 101 110 111 111 000 110 011 111 000 000 001 101

obteniendo una ganancia del 34%.

Este método sirve también para ser aplicado en cualquier secuencia de símbolos repetidos en forma consecutiva, secuencias de blancos, fines de línea y ceros a izquierda.

Algoritmo de LZW

La ventaja fundamental de este método es que realiza el estudio del texto a medida que lo comprime, es decir que no necesita realizar un estudio previo [Sayood, 1996].

En líneas generales, el algoritmo va formando una tabla con los grupos de caracteres que recibe y les asigna un código. La compresión se logra cuando reemplaza las próximas apariciones de los string de caracteres por códigos simples asignados.

Los códigos generados por el algoritmo pueden ser de cualquier longitud, pero siempre tendrán más bits (alrededor de 12 o 16) que un carácter simple, la longitud del código generado depende de la cantidad de símbolos de la tabla. Los primeros 256 códigos (cuando se usan caracteres de ocho bits) son asignados por defecto al conjunto de caracteres estándar.

Algoritmo

STRING = carácter de entrada.
Mientras haya más caracteres de entrada:
CARACTER = carácter de entrada.
Si STRING+CARACTER está en la tabla:
STRING = STRING+CARACTER.
Si no: SALIDA = código de STRING
incorporar STRING+CARACTER a la tabla
STRING = CARACTER.
Fin.

Otra de las ventajas del método es que no se necesita pasar la tabla de string al receptor, ya que el algoritmo de descompresión la vuelve a construir a medida que procesa el texto. Esto es posible porque el algoritmo de compresión siempre devuelve las componentes de un string antes de sacar su código.

Este proceso agrega un nuevo string a la tabla cada vez que recibe un nuevo código. En síntesis la tarea se reduce a traducir cada código recibido en un string y enviarlo a la salida.

Desafortunadamente hay un caso de excepción que causa algunos problemas en el algoritmo de descompresión.

Si hay un string que consiste en un par (STRING, CARACTER) ya definido en la tabla, y la entrada es una secuencia de STRING, CARACTER, STRING, CARACTER, STRING, la salida del algoritmo de descompresión es un código que aún no ha podido definir.

Para manejar esta situación se agrega al algoritmo una rutina que da una solución al problema planteado. Por ser este el único caso que el proceso encuentra un código indefinido, lo resuelve agregando a CODIGO_VIEJO el valor de CARACTER.

Las desventajas principales del método son dos:

- ✓ La tabla de string adquiere gran tamaño rápidamente: este aspecto se puede mejorar reemplazando partes de un string con códigos ya existentes. Por ejemplo, guardar el código 260 (de la fig.3) como /257.
- ✓ La búsqueda de un string en la tabla requiere un costo importante de tiempo: si la tabla se arma con algún criterio especial, se obtiene una ganancia considerable de tiempo utilizando algoritmos de búsqueda para recuperar los string.

Algoritmo de Huffman

El algoritmo recodifica los símbolos de la fuente de acuerdo a la estructura probabilística de la misma, asignando menos unidades de información a los

símbolos más probables y más a los menos probables. Para realizar esto, sea analiza la estructura probabilística de la fuente generando el histograma de la misma; a partir de este se genera un Código de Huffman; y por último sé recodifica la fuente [Held, 1994].

El algoritmo es el siguiente. Inicialmente se calcula las probabilidades de ocurrencia de los símbolos de la fuente (Crear_histograma). A continuación se procede a generar el código a partir de las probabilidades calculadas (Armar_Arbol). Para la generación del código se utiliza un arreglo de nodos hojas que contienen los símbolos y sus ocurrencias o probabilidades.

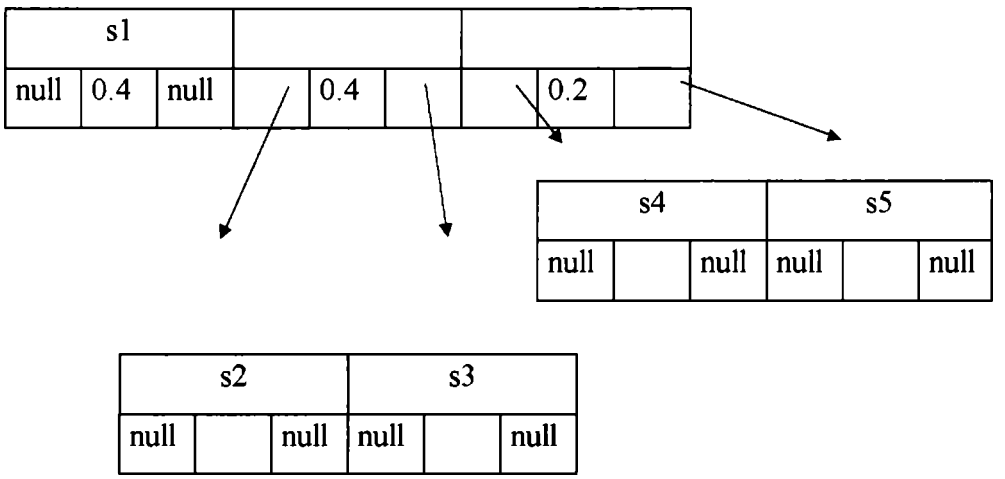
s1			s2			s3			s4			s5		
null	0.4	null	null	0.2	null	null	0.2	null	null	0.1	null	null	0.1	null

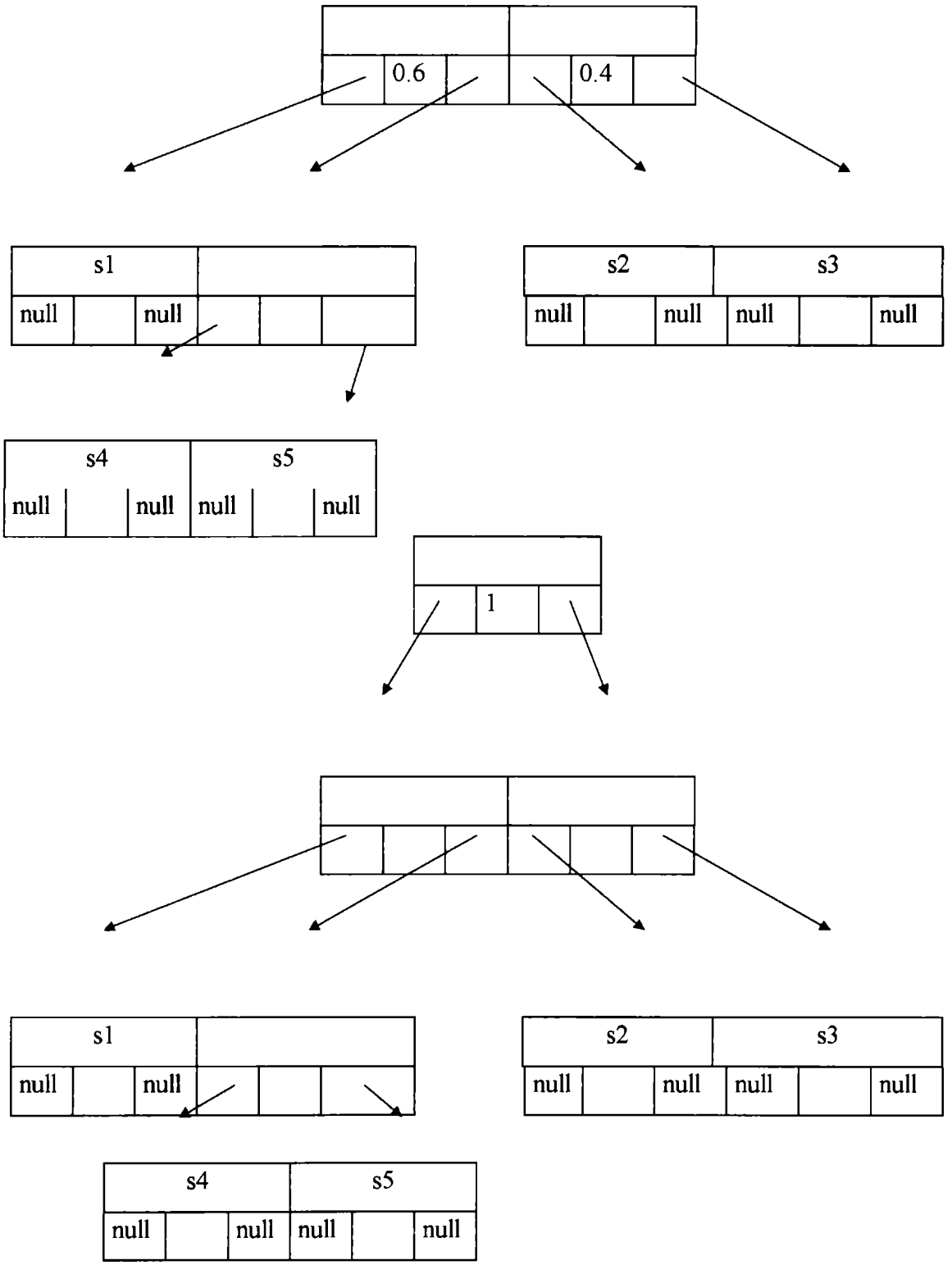
Se recorre el arreglo para encontrar los nodos de mínimas ocurrencias. Una vez identificados estos nodos, al mínimo se lo elimina del arreglo y al nuevo mínimo se le suma la cantidad de ocurrencias del eliminado y se le insertan un nuevo hijo izquierdo y un nuevo hijo derecho. El hijo izquierdo contiene el carácter, el hijo izquierdo y el hijo derecho del nuevo mínimo; el hijo derecho contiene el carácter, el hijo izquierdo y el hijo derecho del mínimo eliminado.

s1			s2			s3					
null	0.4	null	null	0.2	null	null	0.2	null		0.2	

s4			s5		
null		null	null		null

Este proceso se repite hasta que solo quede un nodo en el arreglo.





Una vez generado el árbol de codificación, lo pasamos a un arreglo para tener acceso directo a la palabra de código correspondiente a un símbolo (Armar_codigo_y_header). Se recorre el árbol en pre-order guardando el camino

corriente (agregando a la palabra de código un '0' para el hijo izquierdo y un '1' para el hijo derecho). Cada vez que se alcanza una hoja, se asigna a la celda del arreglo correspondiente al carácter de la hoja el camino corriente.

arreglo[s1] = "00"
arreglo[s2] = "10"
arreglo[s3] = "11"
arreglo[s4] = "010"
arreglo[s5] = "011"

Además de generar el arreglo de acceso directo se aprovechó esta función para generar la parte del header del archivo comprimido correspondiente al código de la siguiente manera. Cada vez que se saltaba de un nodo a otro se sacaba un 0, cada vez que se alcanzaba una hoja se sacaba un 1 y el carácter de la hoja. Como veremos más adelante, sólo con esto se puede reconstruir el árbol de decodificación. La secuencia de salida para nuestro ejemplo sería la siguiente: 001's1'01's4'1's5'01's2'1's3'.

Estructura del archivo comprimido

Tamaño del Archivo Original
Código Comprimido (Árbol de Decodificación)
Datos recodificados

El tamaño del archivo original es necesario ya que no siempre se utilizan todos los bits del último byte del archivo de salida y puede ser que estos bits "*sobrantes*" conformen una palabra de código válida.

Refinamiento del Proceso de Compresión:

```
Crear_histograma(Archivo_de_entrada, &Histograma);
Armar_arbol(Histograma, &Arbol);
Armar_codigo_y_header(Arbol, &Header, &Codigo);
```

Grabar_header(Archivo_de_salida, Header);

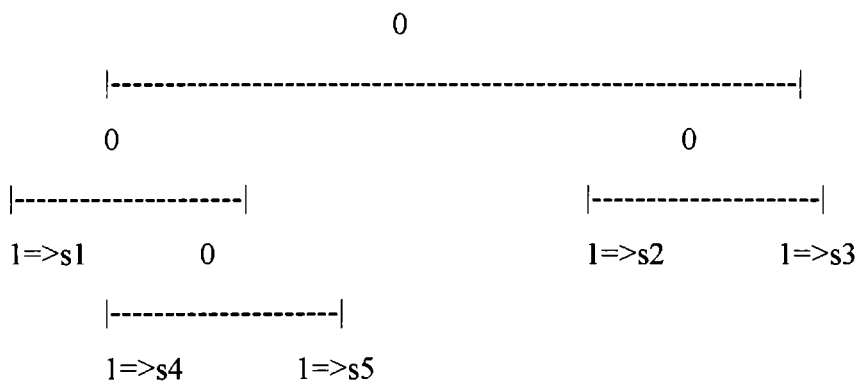
Codificar(Archivo_de_entrada, Codigo,Archivo_de_salida);

A continuación se describe el Proceso de Descompresión. Esto implica la lectura del Header (Tamaño del archivo Original y Árbol de Decodificación Comprimido). La reconstrucción del Árbol de Decodificación y la decodificación propiamente dicha.

El proceso de reconstrucción del árbol a partir del header es el siguiente:

1. Leer bit ,
2. Si el valor del bit es '0', crear un nodo y realizar el paso 1, primero para el hijo izquierdo y luego para el hijo derecho.
3. Si el valor del bit es '1', leer el símbolo (8 bits), crear un nodo hoja y asignarle el símbolo leído

Veamos como quedaría el árbol reconstruido para la salida de nuestro ejemplo: 001's1'01's4'1's5'01's2'1's3' .



El proceso de decodificación se realiza de la siguiente manera:

1. nos posicionamos en la raíz del árbol de decodificación
2. leemos un bit
3. si el bit es '0', nos movemos al hijo izquierdo, si es '1' al hijo derecho

4. si el nodo en el que estamos es una hoja, sacamos el carácter y recomenzamos en el punto 1.
5. si el nodo no es una hoja, saltamos al punto 2.

Notar que a este proceso le falta el criterio de terminación que consiste en contar la cantidad de caracteres que se sacaron, hasta llegar al tamaño del archivo original. Esto fue ignorado deliberadamente para facilitar la comprensión.

Refinamiento del Proceso de Descompresión

```
Rearmar_arbol(Archivo_de_entrada,&Arbol);
Decodificar(Archivo_de_entrada, Arbol, Archivo_de_salida);
```

Run - Length

La codificación Run-Lengh es un método de compresión de datos que reduce físicamente cualquier tipo de secuencia repetida de caracteres, una vez que la secuencia alcance un nivel predefinido de ocurrencia [Nelson, 1991]. Para la situación en que el carácter repetido sea nulo, esta técnica sería parecida a la de supresión de nulos.

De una manera similar al método de supresión de nulos, el empleo de codificación Run-Lengh requiere normalmente el uso de un carácter especial, indicador de compresión, seguido por el carácter que se repite, seguido por la cantidad de repeticiones del mismo.

Cuando se emplean códigos como ASCII o EBCDIC, una buena elección del carácter especial es aquella que no es encontrada, frecuentemente, en los datos a comprimir.

Proceso de codificación

El proceso de compresión convierte un conjunto de caracteres comprimidos en tres caracteres que son: el carácter especial, el carácter repetido y la cantidad de repeticiones.

Nótese que, empleando la supresión de nulos, se requieren dos caracteres y que en esta técnica se requieren tres caracteres. Mientras que en algunas corridas de nulos esto no es de mayor relevancia, cuando hay un exceso de nulos en los datos resulta de suma importancia la consideración de técnicas de compresión combinadas para mejorar los resultados.

La cantidad máxima de caracteres repetidos que se puede indicar con el contador de repeticiones es 255, o si suponemos que el hecho de que esté el carácter especial indica que hay al menos 4 caracteres repetidos, el contador en 1 indicaría 5 caracteres repetidos y así, el contador en 255 indicaría 259 repeticiones.

La eficiencia de esta técnica depende del número de ocurrencias de caracteres repetidos en los datos ha ser comprimidos. Este método provee en general un buen radio de compresión, en archivos con las características enunciadas.

Compresión con pérdida vs. compresión sin pérdida

Las técnicas de compresión proporcionan pérdida o no de la integridad de los datos. Los esquemas de compresión con pérdida interpretan el archivo de datos, hacen suposiciones algorítmicas y eliminan realmente datos del archivo. Esto técnicamente produce una pérdida de datos, sin embargo, el espectador percibe poca pérdida de calidad. Esta técnica es aconsejable para imágenes en las que perder un cierto nivel de sus características es aceptable para el espectador. Los esquemas con pérdida son necesarios para comprimir suficientemente archivos de vídeo con el fin de

almacenarlos para reproducirlos en redes por ejemplo. Las relaciones de compresión alcanzadas de 200:1 hoy en día son con pérdida de información.

Los esquemas de compresión sin pérdida conservan todos los datos del archivo de datos. La compresión sin pérdida se recomienda para aplicaciones como hojas electrónicas o imágenes médicas en las que la pérdida de cualquier dato puede convertir el archivo en inutilizable o impreciso. Además, las relaciones de compresión para los esquemas sin pérdida son mucho más pequeñas que las de los esquemas con pérdida, habitualmente de 2 o 3 a 1 como máximo.

Además hay técnicas que caen bajo la categoría de Motion Compression. Solamente se aplican en secuencias de imagen relacionadas como las encontradas en dibujos en movimiento y producción de video, secuencias de imágenes médicas y aplicaciones de teleconferencia. Con ella se remueve la redundancia entre frames de imagen secuenciales.

Las metodologías de compresión disponibles de texto, audio, gráficos y vídeo constan de estándares internacionales o esquemas patentados. La industria de computadoras está desarrollando constantemente algoritmos de compresión que son más eficientes, tienen menos pérdida y son más rápidos en comprimir y descomprimir. También están realizando esfuerzos para proporcionar metodologías sofisticadas de compresión, descompresión y reproducción que pueden ser fácilmente transportadas a diversas plataformas de computadoras personales.

Opciones de Hardware y Software de Compresión

La tecnología de Compresión está actualmente disponible como una opción hardware o software. Incrustar el algoritmo de compresión en el hardware mejora enormemente la velocidad del proceso de compresión, pero con un gasto muy grande. El hardware de compresión debe estar disponible a un precio razonable para que sea competitivo con los productos de software actuales. Las tarjetas de compresión de

computadoras con MPEG incorporado son nuevas en el mercado. La compresión JPEG se puede conseguir usando hardware o software.

El uso de la compresión en datos multimediales

En multimedia es un tema de investigación el manejo que se realiza de la información. La velocidad y el almacenamiento de procesos constituyen un factor que actuará como limitación para la aplicación de recursos multimediales. Por este motivo se impone la necesidad de un sistema para la compresión de datos (texto, imágenes, planillas, etc.) de manera que funciona dentro de los límites de velocidad y almacenamiento razonable.

Los sistemas de compresión de datos basan su funcionamiento en el hecho de que un fotograma de vídeo por ejemplo contiene información redundante. Veamos como funciona en términos generales un sistema de compresión de datos. En lugar de codificar cada pixel de la pantalla, los algoritmos de compresión de datos almacenan las características del primer fotograma en una serie. La instrucción puede ser "*Todos los pixeles de este fotograma debe ser de color negro*". Si el segundo fotograma tiene un circulo blanco en el centro la instrucción podría ser "*Todos los pixel de este fotograma deben ser negros excepto la región central que tiene un circulo blanco de radio 50 pixel*". Podemos mejorar aun más la tasa de compresión considerando solamente las diferencias entre los fotogramas. Si un tercer fotograma añade un punto rojo en el centro del circulo blanco, la instrucción seria: "*Este fotograma añade un punto rojo en el centro*". La instrucción para un cuarto fotograma podría ser "*Añadir una línea verde en la parte superior*" [Shaddock, 1992].

En realidad, la compresión de datos es mucho más complicada ya que en el mundo real no se encuentran zonas definidas por un único color. La idea es que el fotograma archivado sea una versión considerablemente reducida del fotograma mostrado. La información redundante ha sido eliminada.

No todos los sistemas de compresión preveen preservar fielmente la imagen original. La compresión de datos con pérdida además de eliminar información redundante elimina información correspondiente a la imagen original. Debemos procurar que la información eliminada no sea detectable para el ojo humano. El JPEG (Joint Photografics Expert's Group), es un estándar de compresión con pérdida de imágenes inmóviles, es utilizado por fabricantes de hardware para vídeo digital. MPEG (Motion Picture Expert Group), en el estándar de compresión fotograma a fotograma. Busca las diferencias entre imágenes sucesivas y solo registra los cambios producidos, como no graba información redundante entre fotogramas ofrece una compresión mas alta que la dada por el JPEG. Pero siempre hay un atributo que pagar. Los sistemas MPEG hacen que el acceso a un fotograma individual sea costoso ya que solo se graba el primer fotograma en su integridad [Mallat].

Preservación de la Información

El primer punto a tener en cuenta es si se desea preservar o no el cien por cien de la información o, lo que es lo mismo, si se admite una degradación parcial de la imagen durante el proceso de compresión.

Así, en un primer momento se pueden dividir los algoritmos de compresión según su fidelidad. Para aquellos que no preserven la totalidad de la información presente en la imagen, es preciso establecer un criterio de fidelidad. En general se usan **criterios de fidelidad objetiva,** que suelen definirse como una función matemática entre pixel antes y después de ser comprimido y vuelto a descomprimir. De esta forma puede definirse el error absoluto por pixel, el error cuadrático y el error cuadrático medio o rms como:

$$\sigma^2 = \frac{1}{N_n} \sum_{1}^{N} (x^n - y^n)^2$$

donde N es el tamaño en pixel, $\{x_n\}$ es la salida de la fuente y $\{y_n\}$ es la secuencia reconstruida .

Puede también ser definida la relación señal-ruido (SNR) de una imagen a partir del error cometido al codificar la misma así:

$$SNR = \frac{\sigma_x^2}{\sigma_d^2}$$

Análogamente pueden definirse **criterios de fidelidad subjetivos** que aunque menos rigurosos suelen ofrecer resultados más aceptables cuando el propósito fundamental es visualizar la imagen. Dadas las peculiaridades del sistema visual humano, la calidad subjetiva de dos imágenes puede ser muy distinta aunque presenten una relación señal-ruido similar. Esto puede permitir utilizar técnicas de gran compresión que, aunque objetivamente degraden mucho la imagen, tengan un impacto subjetivo escaso.

Esquema general de Compresión: Calificación *de* Algoritmos

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

- ✓ con pérdida.
- ✓ sin pérdida.

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

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

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

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

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

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

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

Sistema Completo de un Compresor de Imágenes

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

Compresión: Dada una imagen original I, se aplica una transformación T que consiste en mapear los pixels desde un dominio, por ejemplo espacial, hacia otro, por ejemplo frecuencial, en busca de redundancia. Luego, en base a un criterio se determina cuáles y cuántos coeficientes deben seleccionarse. Posteriormente, este conjunto pasa

por un módulo de cuantificación. En éste proceso se llevan los coeficientes, en base a una tabla específica, a un espacio con menor precisión, requiriendo así menos bits para su representación. Este conjunto de coeficientes cuantificados pasa a un proceso de Codificación; generalmente se usa un código entrópico que se encarga de asignar de manera eficiente pocos bits a los coeficientes más frecuentes y muchos bits a los menos frecuentes. Finalmente obtenemos un conjunto de datos que representan la imagen comprimida.

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

Sistema Visual Humano

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

Por otro lado en ojo humano es menos sensitivo a las frecuencias espaciales altas, bordes de una imagen, que a las frecuencias bajas, texturas de una imagen; por ejemplo si en un monitor desplegamos un patrón continuo de pixels blancos y negros alternados uno y otro, el ojo humano tiende a detectar este patrón como un gris uniforme y continuo en lugar del "*mosaico*" que tiene a la vista. Esta deficiencia es explotada codificando con pocos bits los coeficientes que representan frecuencias altas y con muchos bits los de frecuencias bajas [Shaddock, 1992].

JPEG (Joint Photografics Expert's Group)

JPEG (Joint Photographic Experts Groups) es uno de los standard más ampliamente conocidos para compresión de imágenes con pérdida. El mismo fue el resultado de la colaboración de la International Standards Organizations (ISO) y lo que fue CCITT (hoy ITU_T), una parte de la Naciones Unidas. Fue diseñado para comprimir imágenes, full color y en escala de grises, de escenas del mundo real y naturales. Se obtienen buenos resultados al aplicarlo sobre fotografías, trabajos de arte y material similar, aunque no ocurre así con letras, dibujos simples o dibujos de líneas. El JPEG maneja sólo imágenes quietas, pero hay un estándar relacionado llamado MPEG para imágenes en movimiento [Lindley].

JPEG fue diseñado para explotar las limitaciones del sistema visual humano y de esta forma es aplicable a imágenes que serán observadas por las personas.

El JPEG tiene pérdida, es decir, la imagen descomprimida no es exactamente la misma que con la que se empezó. (Hay algoritmos para la compresión de imágenes sin pérdida, pero el JPEG logra una compresión mucho mayor que lo que es posible con los métodos sin pérdida).

El JPEG está diseñado para explotar las limitaciones conocidas del ojo humano, notablemente el hecho de que pequeñas variaciones de color se perciben con menos precisión que pequeñas variaciones en brillo o luminosidad. De esta manera, el JPEG está destinado a comprimir imágenes que serán miradas por seres humanos. Si usted planea analizar estas imágenes con una máquina, los pequeños errores introducidos por el JPEG pueden ser un problema, aunque sean invisibles al ojo.

El Standard JPEG define tres sistemas de codificación diferentes:

- ✓ Un sistema de codificación baseline con pérdida, que utiliza como base la Transformada del Coseno Discreta (DCT)
- ✓ Un sistema de codificación extendido para aplicaciones con requerimientos de más precisión, de reconstrucción progresiva, etc.
- ✓ Un sistema de codificación independiente sin pérdida para compresión reversible.

Para ser compatible con JPEG un producto debería incluir soporte para el sistema baseline. El standard propone una sintaxis que debería cumplir cualquier archivo o secuencia de bits para llamarse JPEG, dejando amplias libertades en las etapas de cuantificación y codificación, de manera que los desarrolladores puedan efectuar mejoras y optimizaciones. No se especifica ningún formato de imagen, resolución espacial o particularidades sobre el color.

Una propiedad útil del JPEG es que el grado de pérdida puede variarse ajustando los parámetros de compresión. Esto significa que el autor de la imagen puede "*negociar*" el tamaño del archivo contra la calidad de la imagen de salida. Se pueden obtener archivos "*extremadamente*" pequeños si no le importa una calidad pobre; esto es útil para aplicaciones como la indexación de archivos de imágenes. Inversamente, si no le agrada la calidad de la salida en el setting de compresión por defecto, se puede aumentar la calidad hasta quedar satisfecho, y aceptar una compresión menor [Sayood, 1996].

¿Porqué usar JPEG?

Hacer que los archivos de imágenes sean más chicos es una ganancia cuando queremos transmitir archivos a través de redes y para lograr bibliotecas de imágenes. El poder comprimir un archivo color de 2 Mbyte a, digamos, 100 Kbytes hace una gran diferencia en espacio de disco y en tiempo de transmisión! Y el JPEG puede proporcionar fácilmente una compresión de 20:1 de datos color. Si comparamos GIF y JPEG, la relación de tamaño es normalmente de 4:1.

Si un software de visualización no soporta JPEG directamente, tendrá que convertir el JPEG a otro formato para ver la imagen. Incluso con un visualizador capaz de ver JPEG, lleva más tiempo decodificar y ver una imagen JPEG que ver una imagen de un formato más simple como ser GIF. De esta manera, usar JPEG es esencialmente un trueque entre tiempo y espacio: se resigna algo de tiempo para poder almacenar o transmitir una imagen de forma más barata. Pero vale la pena notar que cuando hay una transmisión por red o telefónica, los ahorros de tiempo de transferir un archivo más chico pueden ser mayores que el tiempo necesario para descomprimir el archivo.

La segunda ventaja fundamental del JPEG es que almacena información color: 24 bits/pixel (16 millones de colores). GIF, el otro formato de imágenes ampliamente usado en la red, sólo puede almacenar 8 bits/pixel (256 colores o menos). GIF se ajusta razonablemente a pantallas baratas de computadora - la mayoría de las computadoras comunes no pueden mostrar más de 256 colores distintos a la vez. Pero el hardware full color se está abaratando constantemente, y las imágenes JPEG se ven *mucho* mejor que las GIF en este tipo de hardware. Dentro de un par de años el GIF probablemente parecerá tan obsoleto como parece hoy el formato MacPaint en blanco y negro. Más aun, el JPEG es mucho más útil que el GIF para intercambiar imágenes entre personas que tienen hardware muy variante, porque evita que se prejuzgue cuántos colores usar. Por lo tanto el JPEG es considerablemente más apropiado que el GIF para usar como formato estándar por ejemplo en la WWW.

A mucha gente le da miedo el término *"compresión con pérdida"*. Pero cuando hay que representar escenas del mundo real, "*ninguna*" imagen digital puede retener toda la información que llega al ojo. en comparación con la escena del mundo real, el JPEG pierde mucha menos información que el GIF. La verdadera desventaja de la compresión con pérdida es que si una imagen se comprime y descomprime repetidas veces, se pierde un poco de calidad cada vez. Esta es una objeción seria para algunas aplicaciones, pero no tiene la menor importancia para muchas otras.

Descripción del algoritmo JPEG baseline

El sistema recomendado por JPEG es una modificación del esquema original propuesto por Chen y Pratt y se basa en la técnica de codificación de utilizando DCT. En esta sección se describe el algoritmo JPEG baseline, el cual es ilustrado en cada una de sus partes con el bloque de 8x8 de la imagen LENA mostrado en la tabla 1.

121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	110
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

TABLA 1: Bloque de 8x8 de LENA

¿Cómo funciona el JPEG?

La compresión JPEG es un proceso en varias etapas. Igual que otras formas de compresión con pérdida, toma los datos de la imagen y realiza un proceso regularizador que hace que contenga más patrones repetitivos que los que originalmente contenía. Como resultado, la compresión sin pérdida -la codificación de Huffman, en el caso del JPEG- puede funcionar muy efectivamente.

Las imágenes color (no son tratadas en este trabajo) primero se traducen a un espacio de color, como ser YUV o CIELAB, que almacena la información de brillo, o luminancia, separada de la información de color, o crominancia. Esto saca ventaja del

hecho de que el ojo humano es mucho más sensible a variaciones pequeñas de luminancia que a variaciones pequeñas de crominancia, en particular en el extremo azul del espectro. También explota el hecho de que la mayoría de las imágenes contienen áreas bastante grandes en donde los valores de los pixels adyacentes son muy similares en los canales de crominancia.

El siguiente paso, opcional, es hacer un submuestreo de los canales de crominancia. Esta es una de las dos partes del proceso en la que se pierden datos, y típicamente se hace sólo con settings de JPEG de alta compresión y baja calidad. Básicamente, submuestreo significa que se descarta sistemáticamente información de color por filas o columnas de pixels a una razón dada; si se descarta la información de color cada dos filas y cada dos columnas de pixels, se reducen los datos de crominancia en un 75%. Cuando se descompone la imagen, los valores de los pixels faltantes se interpolan a partir de los que están presentes. En la mayoría de las imágenes el efecto no se nota, ya que el canal de luminancia contiene la mayor parte de la información visualmente importante. Puede llegar a observarse una leve pérdida de saturación, pero muy poco más cambia en forma visible.

Luego se registran los datos usando una función matemática llamada Transformada de Coseno Discreta, o DCT. Trabajando con bloques de 8x8 pixels, el DCT analiza las frecuencias espaciales de la imagen tanto horizontal como verticalmente. La parte con pérdida del proceso es la cuantización, que es la forma en que la ingeniería se refiere a la división que reduce la cantidad de bits necesarios para expresar cada valor de frecuencia. Esos valores son luego comprimidos sin pérdida, primero con RLE y luego con la codificación de Huffman. Esta es la parte del JPEG que hace que sean posibles altas razones de compresión.

El especialista en JPEG define dos módulos diferentes "*back end*" para la salida final de los datos comprimidos: se permite la codificación Huffman o la codificación aritmética. La elección no afecta la calidad de la imagen, pero la codificación aritmética normalmente produce un archivo comprimido más pequeño. En imágenes típicas la codificación aritmética produce un archivo un 5 a un 10 % menor que la codificación

Huffman (todos los números de tamaños de archivos citados corresponden a la codificación Huffman).

Desafortunadamente, la variante particular de codificación aritmética especificada por el estándar JPEG está sujeta a patentes. De esta manera "*no se puede usar legalmente la codificación aritmética del JPEG*" a menos que obtenga las licencias de estas compañías. (La excepción "*uso experimental*" de la Ley de Patentes permite que la gente pruebe un método patentado en el contexto de la investigación científica, pero cualquier uso comercial o personal de rutina es una infracción).

La descompresión simplemente invierte este proceso. Los datos se expanden usando la descodificación de Huffman, los valores resultantes se multiplican, se aplica una DCT inversa, y los valores de luminancia y crominancia se traducen de vuelta al RGB.

Veamos con un ejemplo proceso anteriormente mencionado para una imagen en escala de grises:

Transformación

Como ya dijimos la transformada utilizada por el esquema JPEG es la transformada DCT. Los valores de los pixels de la imagen de entrada son decrementados en $2^{(P-1)}$, donde P es el número de bits utilizados para representar cada pixel. En nuestro caso se utilizan imágenes de 8 bits cuyos pixels toman valores entre 0 y 255 de manera que al substraerles 128 sus valores finales de entrada variarán entre -128 y 127.

El algoritmo comienza con el particionamiento de la imagen en bloques de 8x8 pixels. Si alguna de las dimensiones de la imagen no es múltiplo de ocho el codificador

replica la última columna y/o fila hasta que el tamaño final sea múltiplo de 8. Esas filas y columnas adicionales son removidas durante el proceso de decodificación. Luego cada bloque es transformado independientemente usando DCT, dando como resultado otra matriz de 8x8 pero donde sus coeficientes varían en el rango de - 1023 . . 1023 .

Si se toma el bloque de 8x8 de pixels de la tabla 1, se substrae 128 de sus valores y se aplica la transformada DCT a la matriz resultante, se obtienen los coeficientes DCT de la tabla 2. En esta última puede notarse que los coeficientes de más baja frecuencia cercanos al vértice superior izquierdo toman valores absolutos más grandes que los coeficientes de más alta frecuencia. Esto es lo que generalmente ocurre, ya que existen excepciones como en los casos en que se tiene gran cantidad de variaciones en el bloque (ej. bordes)

39.88	6.56	-2.24	1.22	-0.3	-1.08	0.79	1.13
-102.43	4.56	2.26	1.12	0.35	-0.63	-1.05	-0.48
37.77	1.31	1.77	0.25	-1.5	-2.21	-0.10	0.23
-5.67	2.24	-1.32	-0.81	1.41	0.22	-0.13	0.17
-3.37	-0.74	-175	0.77	1.99	-0.26	-1.30	0.76
5.98	-0.13	-0.45	-0.77	1.99	-0.26	1.46	0.00
3.97	5.52	2.39	-0.55	-0.05	-0.84	-0.52	-0.13
-3.43	0.51	-1.07	0.87	0.96	0.09	0.33	0.01

TABLA 2: Coeficientes DCT del bloque de pixels de LENA

Cuantificación

JPEG utiliza cuantificación midtread uniforme para cuantificar los diversos coeficientes de un bloque transformado.

Las longitudes de los intervalos de cuantificación están organizados en una tabla de cuantificación y pueden ser vistos como parte fija de esta etapa del proceso. Un ejemplo de tabla de cuantificación recomendada por JPEG se muestra en la tabla 3. Cada valor cuantificado es representado por un label. El label correspondiente al valor del coeficiente transformado θ_{ij} es :

$$l_{ij} = \left[\frac{\theta_{ij}}{Q_{ij}} + 0.5 \right]$$

donde θ_{ij} es el elemento ij de la tabla de cuantificación, y [x] es la parte entera de x. Como ejemplo consideremos el coeficiente θ_{00} de la tabla 2 cuyo valor es 39.88. De la tabla 3 tenemos que Q_{00} es 16 y por lo tanto

$$l_{00} = \left[\frac{39.88}{16} + 0.5 \right] = [2.9925] = 2$$

El valor reconstruido es obtenido del valor cuantificado multiplicado por la correspondiente entrada en la tabla de cuantificación. Para el caso anterior el valor reconstruido del coeficiente θ_{00} es

$$l_{00} x Q_{00} = 2x16 = 32$$

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

TABLA 3: Matriz de Cuantificación default de JPEG

El error de cuantificación en este caso es 39.88 - 32=7.88. Siguiendo los mismos pasos para los demás coeficientes se obtienen los labels de la tabla 4.

2	1	0	0	0	0	0	0
-9	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

TABLA 4: Calculo de Labels

Por la tabla 3 podemos ver que las longitudes de los intervalos de cuantificación se incrementan a medida que se baja desde el coeficiente DC (posición 0,0) hacia los coeficientes de alto orden. Debido a que el error de cuantificación es una función creciente de las longitudes de los intervalos, los errores más grandes de cuantificación ocurrirán en los coeficientes de más alta frecuencia. La decisión de cual debe ser la

longitud de los intervalos depende de como serán percibidos los errores en esos coeficientes por el sistema visual humano.

Coeficientes en posiciones diferentes tienen importancia perceptual diferente. Los errores de cuantificación en el coeficiente DC y en los AC (coeficientes de posiciones distintas de 0,0) de baja frecuencia son mas fácilmente detectables que los errores en coeficientes de más alta frecuencia. Por lo tanto podríamos utilizar longitudes de intervalos más grandes para los coeficientes menos importantes

Distintos grados de pérdida se pueden lograr multiplicando la matriz de la tabla 3 por un escalar. La flexibilidad dada para esta etapa permite construir matrices de cuantificación adaptivas acorde a las características de la imagen que está siendo comprimida. Por ejemplo en zonas donde se producen variaciones bruscas (como el pelo o los bordes del sombrero en la imagen LENA) podrían aparecer coeficientes de alta frecuencia con un peso importante a la hora de reconstruir la imagen y que podrían no ser tenidos en cuenta si se utilizara una matriz no adaptiva.

Categoría	Palabra de Código	Longitud Total
0	010	3
1	011	4
2	100	5
3	00	5
4	101	7
5	110	8
6	1110	10
7	11110	12
8	111110	14
9	1111110	16
A	11111110	18
B	111111110	20

TABLA 5: Código Default de JPEG

Codificación

En este paso los coeficientes DC y AC resultantes de la cuantificación son codificados utilizando códigos de longitud variable, pero ambos en forma diferente.

Los coeficientes DC (coeficiente de la posición 0,0) son codificados en forma diferencial con respecto al mismo elemento del bloque previo. La razón de esto es que el elemento de esta posición es un múltiplo del valor promedio de los pixels del bloque y, que en general estos promedios no difieren substancialmente entre bloques vecinos. Es de esperarse que esto ocurra en la mayor cantidad de situaciones.

Por lo tanto para codificar las diferencias se asigna un código Huffman al conjunto de categorías en las que pueden caer dichos valores. Una diferencia en

particular se codifica grabando primero la palabra de código correspondiente a su categoría y agregando luego bits extras al final de esa palabra de código para especificar su valor en una cantidad indicada por el número de categoría. Las categorías y sus correspondientes rangos de valores son mostrados en la tabla 6.

Si generalizamos para una categoría de diferencia (por ejemplo K) se utilizan K bits para codificar, ya sea K bits menos significativos de la diferencia positiva o bien, K bits menos significativos de la diferencia negativa menos 1.

La tabla 5 muestra un código default provisto por JPEG donde la primer columna contiene categorías de diferencia, la segunda la palabra de código para esa categoría y la tercera la cantidad total de bits necesaria para codificar una diferencia.

Dado que la categoría 0 contiene solamente un elemento no se necesitan bits extras para especificar el valor. La categoría 1 contiene dos elementos de manera que solo se necesita agregar 1 bit al final del código Huffman para la categoría 1 para especificar alguno de los dos elementos. De igual forma se necesitan 2 bits para especificar un elemento en la categoría 2, 3 bits para la categoría 3 y n bits para la categoría n.

Nro. de Categoría	Rangos de valores
0	0
1	-1 1
2	-3 -2 2 3
3	-7 - 4 4 7
4	-15 ... -8 8 15
5	-31 ... -16 16 31
6	-63 ... -32 32 63
7	-127 -64 64 ... 127
8	-255 ... -128 128 ... 255
9	-511 ... -256 256 ... 511
10	-1023 -512 512 ... 1023
11	-2047 -1024 1024 ... 2047
12	-4095 ... -2048 2047 2048
13	-8191 ... -4096 4096 ... 8191
14	-16383 ... -8192 8192 ... 16383
15	-32767 16384 16384 ... 32767
16	32768

TABLA 6: Categorías de coeficientes y sus respectivos rangos

El código binario para los coeficientes AC es generado en una forma diferente. La matriz transformada y cuantificada es reordenada siguiendo un trayecto de zigzag (ver Figura 1) para obtener una secuencia unidimensional de los coeficientes cuantificados, de manera que la secuencia resultante se encuentre dispuesta de acuerdo al crecimiento de las frecuencias. El mayor beneficio que resulta del nuevo reordenamiento es que se puede tomar ventaja de una mayor longitud en las corridas de ceros.

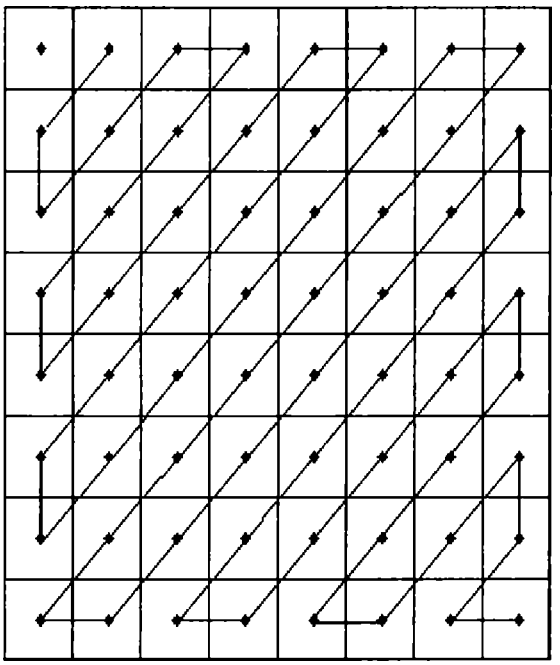


Figura 1. Secuencia de zigzag para un bloque de 8x8 de coeficientes

Concretamente los coeficientes AC son codificados usando un código de longitud variable como el de la tabla 7 donde se definen: C como la categoría en la que cayó un coeficiente y Z como el número de coeficientes con valor cero que ocurrieron en la secuencia después del último coeficiente distinto de cero. De acuerdo con la bibliografia para bloques de 8x8 y pixels de 8 bits C puede tomar valores que varían de 0 a 10. Para los valores de Z, JPEG establece en su tabla de codificación default un valor máximo de 15 ceros consecutivos. De esta manera cuando en la secuencia de zigzag es encontrado un coeficiente cuyo valor cae en una cierta categoría C precedido por Z ceros se graba el código correspondiente a la entrada Z/C en la tabla. Seguidamente se graba el valor del coeficiente en C bits de la misma forma en la que se graban las diferencias entre coeficientes DC.

Z/C	Palabra de Código	Longitud total
0/0	**1010(=EOB)**	**4**
0/1	00	3
0/2	01	4
0/3	100	6
0/4	1011	8
0/5	11010	10
.....	.....	.....
F/0 (ZRL)	**111111110111**	**12**
F/1	1111111111110101	17
F/2	1111111111110110	18
F/3	1111111111110111	19
F/4	1111111111111000	20
F/5	1111111111111001	21
.....	.....	.....

TABLA 7 Algunas palabras de código de la tabla de codificación default provista por JPEG

Existen dos palabras de código especiales llamadas EOB, utilizada para indicar que el resto de los coeficientes en la secuencia de zigzag son ceros, y ZRL para el caso en que el número de ceros consecutivos en la secuencia excede 15.

Como se ve, al igual que en la cuantificación, también se proveen tablas de codificación default aunque el usuario puede construir las propias de manera de que puedan ser adaptadas a las características de la imagen que esta siendo comprimida.

MPEG (Motion Picture Expert Group)

Introducción

Las redes locales, salvo en algunas instalaciones de 100 Mbps, no están en las mejores condiciones para manejar el tráfico de vídeo o multimedia, especialmente cuando hay varias estaciones de trabajo operando simultáneamente.

Por otra parte, las comunicaciones de banda ancha incluso de doble vía, se están volviendo más comunes en aplicaciones como videoconferencia, educación, entrenamiento, etc., donde las señales ya son digitales aunque otras deben convertirse debido a su origen (TV, cable, etc.) pudieran ser analógicas, lo que es usual para su difusión masiva.

Las señales digitales correspondientes pueden ser comprimidas en origen, viajar de esta manera y ser descomprimidas en destino.

El Grupo de Expertos de MPEG ha venido desarrollando normas de compresión para vídeo, bajo la forma de TV de broadcast, TV por cable, videoconferencia, vídeo a demanda, etc. así como la combinación con audio incluso estereofónico.

El uso de MPEG en las LANs y PCs se ha visto demorado frente a otras formas de compresión que pueden descomprimir más económicamente por software. Sin embargo, la necesidad de mayores resoluciones, así como también una previsible convergencia con los televisores, facilitará su mayor difusión.

Oficialmente hay dos tipos de MPEG: el MPEG-1 y MPEG-2. Antes de comenzar a consolidarse el MPEG-2, se estuvo usando (por ejemplo para avisos digitales) una extensión no oficial del primero llamada MPEG-1.5 o MPEG+ para velocidades de hasta 6 Mbps.

Previo a un mayor análisis del MPEG y las normas mencionadas, es conveniente analizar el proceso del manejo del video, comenzando por las señales disponibles o las que se generan con los diferentes sistemas.

El MPEG usa muchas de las técnicas del JPEG, pero agrega una compresión entre marcos para aprovechar las similitudes que normalmente existen entre los marcos sucesivos. Debido a esto, el MPEG típicamente comprime una secuencia de vídeo, en un factor aproximadamente tres veces mayor que los métodos "*M-JPEG*" (JPEG a marcos individuales de una secuencia de vídeo) para una calidad similar. Las desventajas del MPEG son (1) requiere mucha más calculo computacional para generar la secuencia comprimida (ya que detectar similitudes visuales es difícil para una computadora), y (2) es difícil editar una secuencia MPEG en una base de frame-by-frame (marco a marco) (ya que cada marco está íntimamente ligado a los que están a su alrededor). Este último problema ha hecho los métodos "*M-JPEG*" bastante populares para productos de edición de vídeo [Clarke, 1995] [Glassner, 1996, Vol 1].

Base del Sistema MPEG

La norma MPEG describe dos clases de cuadros: codificados individualmente o por comparación entre ellos. En el primer caso se habla de codificación espacial; en el segundo caso se habla de codificación temporal. Los cuadros a su vez se dicen intracodificados en el primer caso e intercodificados en el segundo.

Los cuadros intracodificados se llaman también cuadros I y se comprimen exclusivamente por la redundancia espacial dentro del propio cuadro que se considera, es decir no se lo compara con los otros. Como estos cuadros guardan información explícita de los pixels que lo componen, se los puede usar como referencia para eliminar errores que se puedan ir acumulando con los otros sistemas de codificación.

Los cuadros intercodificados consideran la redundancia temporal además de la espacial, basándose en la comparación con cuadros de referencia, lo que permite un mayor índice de compresión. Hay dos tipos de cuadros intercodificados.

Los cuadros P o cuadros Predecibles hacia delante se basan en el contenido de un cuadro de referencia anterior, efectuando lo que se llama una estimación del movimiento. Su extensión comprimida puede ser menor que un cuadro I.

Por su parte, los cuadros B son Bidireccionales, porque se basan en la interpolación entre un cuadro de referencia anterior y otro posterior. Su extensión comprimida puede ser menor aún que la de un cuadro P. Sin embargo, ya plantea una

posible limitación a un sistema en línea, puesto que la información de un cuadro se genera en función de otro posterior. Los cuadros de referencia para un cuadro B, sólo pueden ser P o I y no otro B.

La estimación del movimiento ayuda especialmente en el caso de movimientos rápidos que dan lugar a imágenes borrosas en cada cuadro. La estimación del movimiento se hace en base a la velocidad y dirección al comparar bloques en cuadros sucesivos. De esta manera se generan vectores de movimiento dados por sus componentes cartesianas (a lo largo y a lo ancho). Hoy en día sin embargo el uso de captores mejorados en cámara de vídeo con obturbador electrónico, permite obtener imágenes nítidas. Pero aún así la estimación se hace necesaria cuando hay que hacer conversión de normas NTSC y PAL, por la diferencia de cuadros por segundo.

El manejo de los cuadros expuesto hasta aquí hace que el MPEG pueda ofrecer una compresión mayor que el JPEG que fue pensado para imágenes fijas.

Efectivamente, la ventaja del MPEG es que a partir de cuadros que establece como referencia, compara cuadros sucesivos, generando información solos de las diferencias. Puede llegar a predecir la ubicación de bloques de una imagen, cuando las comparaciones se hacen no sólo hacia atrás sino hacia delante. Esto impone condiciones especiales de exigencia a los sistemas en tiempo real, puede ser salvada con codecs de alta velocidad.

La Norma MPEG-1

La versión MPEG-1, como ya dijimos, toma sólo un cuadro de información muestreada, con lo que la compresión necesaria se reduce a 4 veces respecto del MPEG-

2. De esta manera basta un factor de entre 20 a 30 veces para generar información entre 1 y 1.5 Mbps con un límite de 1,86 Mbps.

MPEG-1 no produce cuadros con campos entrelazados. Simplemente repite una vez cada cuadro, evitando así las fluctuaciones propias del régimen de repetición de 30 fps (cuadros por segundo).

En una PC, MPEG-1 puede decodificarse por software a baja velocidad (lo que inevitablemente produce saltos en las imágenes) o por hardware, pero posee más costo.

En realidad, MPEG-1 acepta una especificación mínima de 15 cuadros por segundo, de modo que de esta manera y con la señal de salida convenientemente comprimida, pueda ocupar un canal de sólo 150 Kbps. En estas condiciones puede ser manejada por una LAN típica sin mayores complicaciones y hasta en comunicaciones WAN vía ISDN.

La Norma MPEG-2

Originalmente, MPEG-2 se pensó para videoconferencia vía satelital. El sistema de compresión se mejoró y se incluyó la capacidad de presentar campos entrelazados, teniendo a la vista el sistema de satélite de Broadcast Directo (DBS). De hecho MPEG comenzó a difundirse gracias a dicho tipo de servicios especialmente el de DirecTV. Sin embargo no todas las transmisiones son de MPEG-2 pues la norma no se había terminado cuando comenzaron estos tipos de servicios.

MPEG-2 puede ofrecer un factor de compresión mayor que MPEG-1 puesto que, al trabajar con entrelazado, puede explotar la correlación entre los campos que constituyen un mismo cuadro generado menos información redundante.

Compresión y Descompresión MPEG

El sistema MPEG es básicamente asimétrico en cuanto a la compresión y descompresión. En las instalaciones que comprimen fuera de línea, el equipamiento puede ser tal que insuma más de 50 veces el tiempo de descompresión en el equipo del usuario. Y en todo caso, en los sistemas que deben operar en tiempo real, impone exigencias especiales bajo la forma de instalaciones de gran poder de computación.

COMPRESION

1. Para la estimación del movimiento en compresión, el MPEG divide los cuadros en macrobloques. Cada macrobloque tiene 16x16 pixels que son tratados por la luminancia, mientras que tomados de a dos, es decir 8x8, son tomados por ambas señales de crominancia.

Los macrobloques también pueden ser I, P, B, bajo las siguientes condiciones:

a) En un cuadro I solo puede haber macrobloques I.

b) En un cuadro P solo puede haber macrobloques I y P.

c) En un cuadro B puede haber macrobloques de cualquier tipo.

Los macrobloques B o P resultan en diferencias del bloque que se analiza y el bloque que mejor se ajuste en un cuadro de referencia. En estos casos no se describe cada pixel sino que se supone que todos los pixels de un bloque se mueven juntos. Lo que se describe entonces es la diferencia entre dos bloques como conjuntos. Este criterio tiene sentido pues se puede demostrar que en una escena en movimiento hay menos

correlación entre pixels que ocupan el mismo lugar en cuadros sucesivos que entre pixels con ubicaciones algo desplazadas (en sentido horizontal y/o vertical) entre cuadros sucesivos. Esto se debe al movimiento de un objeto completo respecto del fondo y el proceso correspondiente ya mencionado de la estimación del movimiento genera un vector de movimiento que identifica el bloque del cuadro de referencia que mejor se adapta al bloque del cuadro que se analiza, estableciendo las diferencias correspondientes.

Pero habrá otros bloques que se repetirán prácticamente iguales en la misma posición de cuadros sucesivos. En presencia de los bloques recién comentados que implican movimiento, ahora se presenta información del fondo de la imagen, con escasa o nula variación, que es precisamente lo que rescatan los macrobloques del tipo P o B, y que aprovechara el sistema para aumentar la compresión.

Por otra parte, los macrobloques B van mas allá, porque pueden interpolar bloques basándose en una referencia anterior y otra posterior.

2. Cada macrobloque resultara entonces en una sucesión de valores que, o lo describen directamente (macrobloques I), o indirectamente basándose en diferencias con macrobloques (P y B) del cuadro de referencia anterior y posterior, para cuadros B.

Los valores temporales resultantes, son tratados por la Transformada de Coseno Discreto (DCT), con la que se pasa del dominio del tiempo al dominio de las frecuencias. Los valores obtenidos corresponden a amplitudes a diferentes frecuencias. Las imágenes con poco detalle tienen muchos ceros en las frecuencias elevadas. El conjunto de todos los valores constituyen lo que se llama el espectro de la señal temporal original. La envolvente de este espectro tiene sus mayores componentes a bajas frecuencias, reduciéndose las amplitudes hacia las frecuencias mas elevadas.

El proceso mismo del DCT tiene como efecto concentrar la mayor parte de la energía (dada por las amplitudes señaladas) en las frecuencias bajas. No hay en realidad compresión en este paso.

3. Los valores resultantes del paso anterior pasan el Cuantizador. Aquí primero los valores se dividen por una constante y se redondean los resultados. Debido precisamente al efecto de DCT ya comentado y a la división efectuada, la cuantificación elimina algunos valores de alta frecuencia o frecuencia menor, valores de baja energía que casi igualan a cero. De esta manera, pueden anularse (pasan a ser cero) algunas componentes de alta frecuencia, así como algunas amplitudes originalmente reducidas de baja frecuencia. Esto es equivalente a decir que los 8 bits de un pixel se reducen todos a cero.

4. Si los datos obtenidos se ven simplemente como una sucesión de bits (independientemente de los valores de los diferentes pixels), el flujo resultante esta compuesto por unos y ceros.

La señal resultante se trata por medio de la Codificación por la Longitud o run-length encoding, que consiste en contar y registrar la cantidad de bits sucesivos del mismo tipo. El resultado obtenido se trata ahora por la codificación Huffman como una forma de codificación de longitud variable. Una codificación de este tipo de dice entrópica porque asigna la menor cantidad de bits (bajo la forma de palabra o conjunto de bits) a la agrupación que ocurre mas frecuentemente y viceversa.

DESCOMPRESION

Se puede ver como el proceso inverso. Comienza entonces por Huffman y extensión de cada grupo de bits iguales. Al tiempo que los vectores de movimiento, pasan a la cuantización inversa que recupera y aplica el factor de escala correspondiente. Luego viene el DCT inverso que permite regresar al dominio temporal. Los valores resultantes son directamente los correspondientes para pixels de bloques I, pero

obviamente solo las diferencias en los casos de los bloques P y B. Los valores relativos se tratan con la información de compensación del movimiento que se obtiene de los vectores mencionados antes. De esta manera se obtienen los valores específicos correspondientes a los bloques P y B. Finalmente viene la conversión de colores.

Transformada de Wavelet

La transformada de Fourier ha sido usada como una herramienta confiable en el análisis de señales durante muchos años. Inventada a principios del siglo XIX (1800) por Jean-Basptiste-Joseph Fourier, la transformada de Fourier se ha tomado como en la piedra angular del análisis de señales moderno.

La transformada de Fourier ha probado ser increíblemente versátil en aplicaciones que van desde el reconocimiento de patrones al procesamiento de imágenes. Sin embargo, sufre de ciertas limitaciones. Recientemente una nueva clase de transformación, la transformación wavelet, ha demostrado ser tan poderosa y versátil como la transformación de Fourier, pero sin algunas de sus limitaciones [Ueda, 1995].

Antes de presentar como funcionan las transformadas wavelet, examinaremos a su predecesor.

Como funciona la transformada de Fourier

Una función en el dominio del tiempo es traducida por la transformación de Fourier en una función en el dominio de la frecuencias, en donde es analizada por su contenido de frecuencia. Esta traducción ocurre porque la transformación de Fourier, expande la función original en términos de las funciones bases ortonormales de las ondas del seno y del coseno a cada frecuencia.

La transformación de Fourier, trabaja bajo la asunción que la función original en el dominio del tiempo es de naturaleza periódica. Como resultado la transformada de Fourier, tiene dificultades con funciones como componentes transitorios, o sea, componentes localizados en el tiempo, esto se pone especialmente de manifiesto, cuando una señal tiene transiciones agudas. Otro problema es que la transformación de Fourier

de una señal, no lleva ninguna información, perteneciente a la translación de la señal en el tiempo. Las aplicaciones que utilizan la transformada de Fourier as menudo solucionan el primer problema distribuyendo los datos de entrada en ventanas, de manera tal que los valores muestreados converjan a cero en los puntos finales. Intentos para resolver el segundo problema tales como el desarrollo de la transformación de Fourier de tiempo corto, han tenido solo éxito parcial.

En años recientes se han descubiertos nuevas familias de funciones bases ortonormales que conducen a transformaciones que sobrepasan los problemas de la transformación de Fourier. Estas funciones bases se llaman *"waveletes"*, y a diferencia de las ondas de la transformación de Fourier no necesitan tener duración infinita. Pueden ser distintas de cero solo durante una pequeña extensión de la función wavelets. Esto permite a la transformación wavelets traducir una función del dominio del tiempo en una representación localizada, no solo en frecuencia (como la transformada de Fourier), sino también en el tiempo. Esta capacidad ha producido nuevos desarrollos en los campos del análisis de señales, procesamiento de imágenes y compresión de datos.

Introducción : Pirámides de imagen

Las pirámides pueden ser usadas para el filtrado de imágenes, construyendo valores a partir de la señal original [Clarke, 1995].

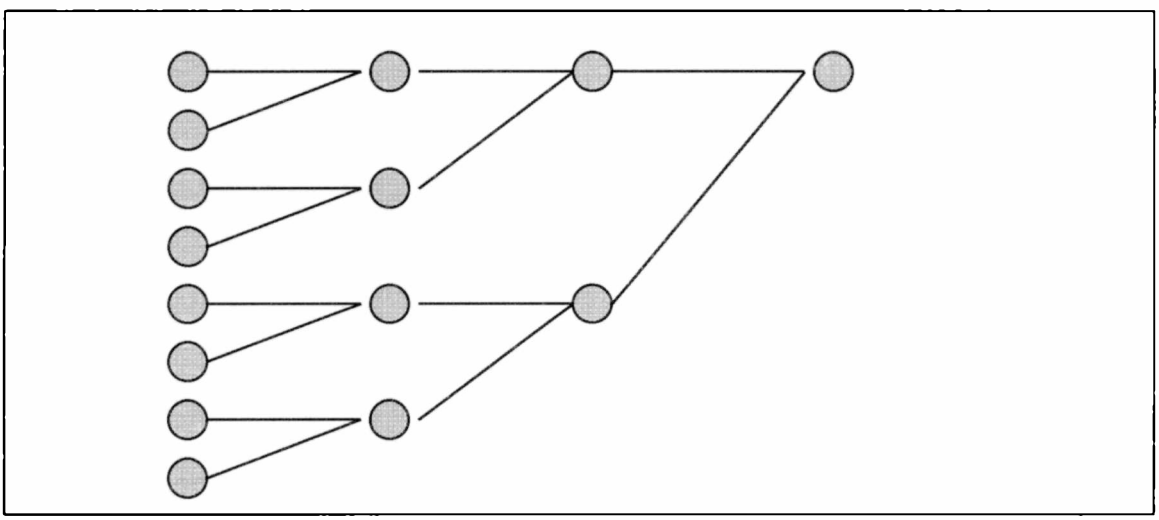


Figura 8

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

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

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

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

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

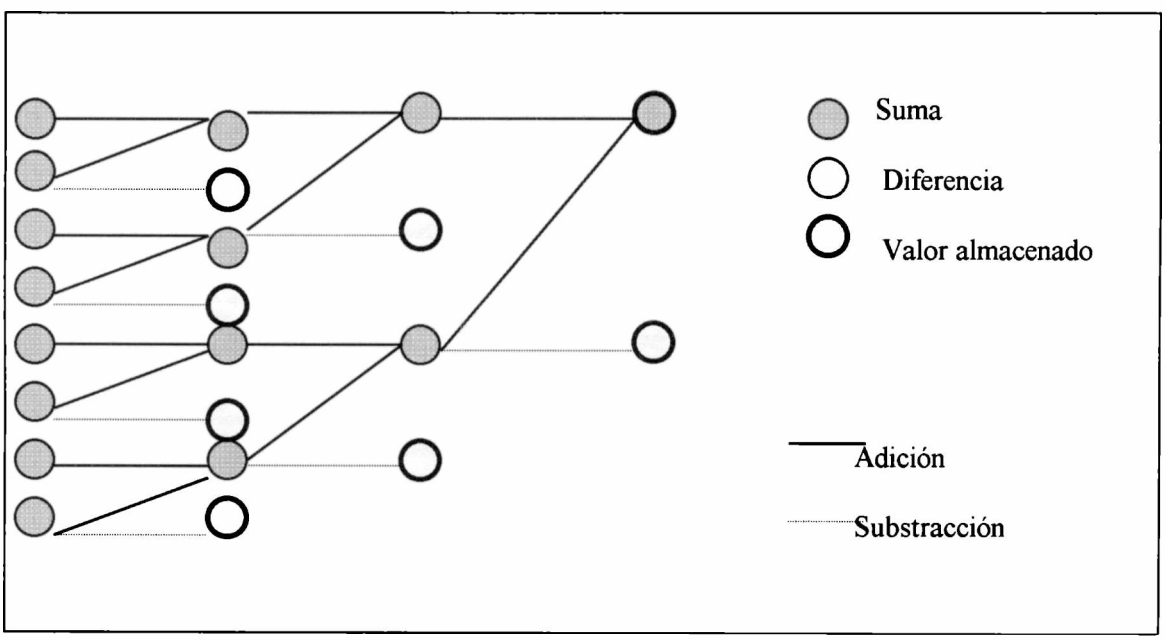


Figura 9

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

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

y la versión detallada submuestreada:

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

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

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

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

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

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

El esquema computacional es el siguiente:

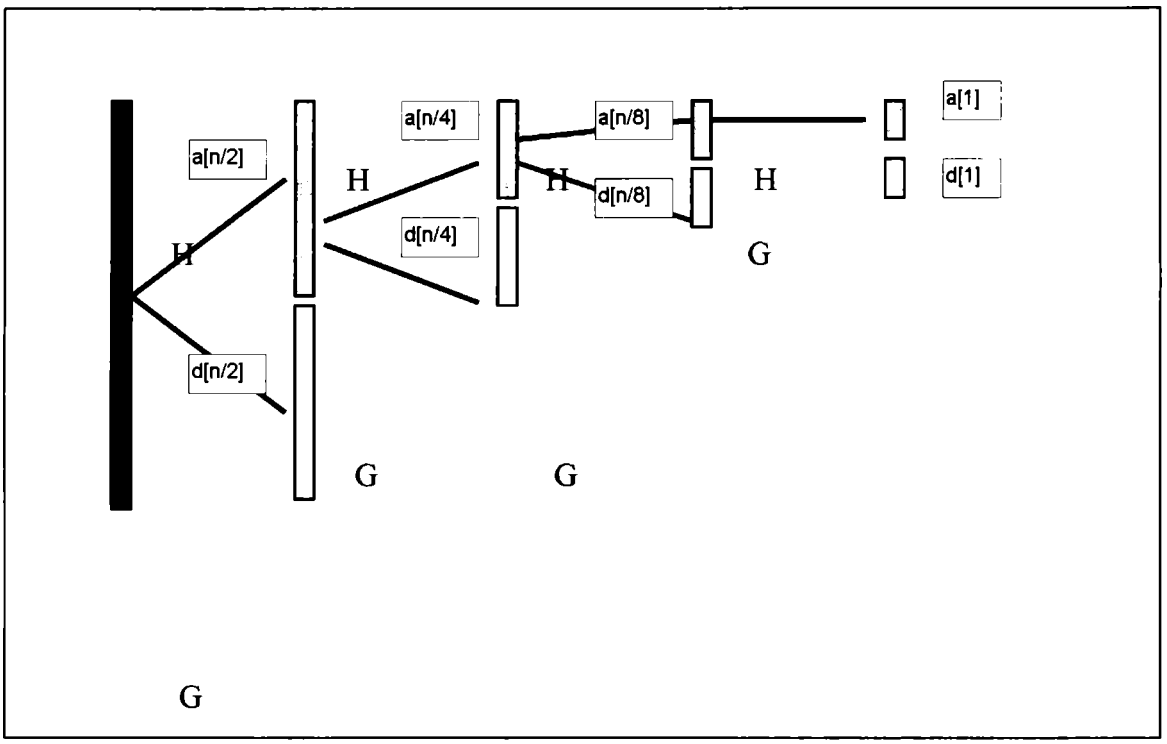


Figura 10

Transformada de Fourier

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

El motivo por el que se estudia las wavelets, es que para algunos análisis de señales, son más convenientes que la transformada de Fourier. La mayoría de las señales que tratamos en una computadora son no estacionarias, esto es, no son estadísticamente la misma en cualquier lugar. En nuestro caso forzamos, mucha información de alta frecuencia en algunos lugares y poca en otros. Para ello se usan técnicas de supermuestreo adaptativas. En las regiones donde la imagen tiene muchos bordes,

texturas y sombras, necesitamos muestrear más veces para determinar las altas frecuencias de la región. Si la imagen tiene pocos cambios de color, podemos muestrear en forma más espaciada.

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

La trasformada de Fourier no permite observar la frecuencia en un intervalo de la 11señal independiente del resto; siempre integra sobre la señal completa retornando luego al intervalo deseado. Si se encuentra una ráfaga de componentes de alta frecuencia en el medio de conjuntos de componentes predominantes de baja frecuencia, la transformada de Fourier no nos dice nada acerca de la ubicación de la ráfaga.

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

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

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

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

Transformada de Fourier Short-Time (por ventanas)

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

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

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

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

Escala y Resolución

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

Lo anteriormente mencionado tiene un significado mayor en procesamiento de imágenes; será útil conocerlo antes de detallar wavelets.

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

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

Para ejemplificar esto, dada una señal discreta de cuatro elementos, s0 =[0,0,2,2]. Se puede crear una nueva versión de la señal promediando pares sucesivos de valores, creando s1=[0,2]. Como el mapa se ha achicado, cada elemento de s1 cubre dos veces el territorio de la señal original, por lo tanto la escala se ha duplicado. Si se repite la operación se obtiene s2=[1] ; la escala de s2 es cuatro veces la escala de s0 y la resolución ha cambiado. Supongamos que volvemos a la señal original s0, y si se filtran

los componentes de baja frecuencia, creando sf=[0.25,0.75,1.25,1.75]. La escala de sf es la misma que s0, pero la resolución de la señal ha cambiado.

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

La Ecuación de Dilación y la Transformada de Haar

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

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

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

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

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

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

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

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

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

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

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

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

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

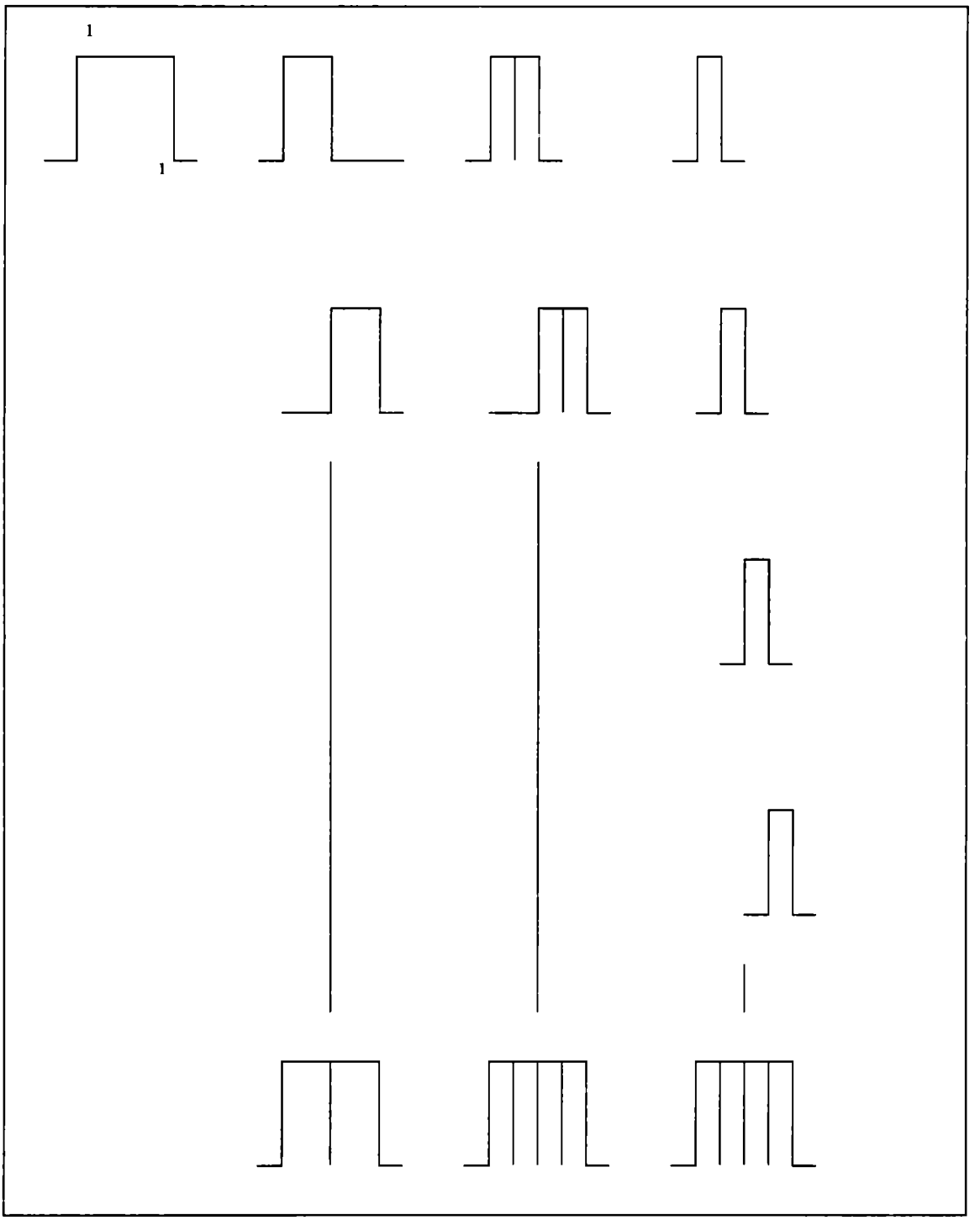


Figura 11 - Ecuación de dilación.

Descomposición y Reconstrucción

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

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

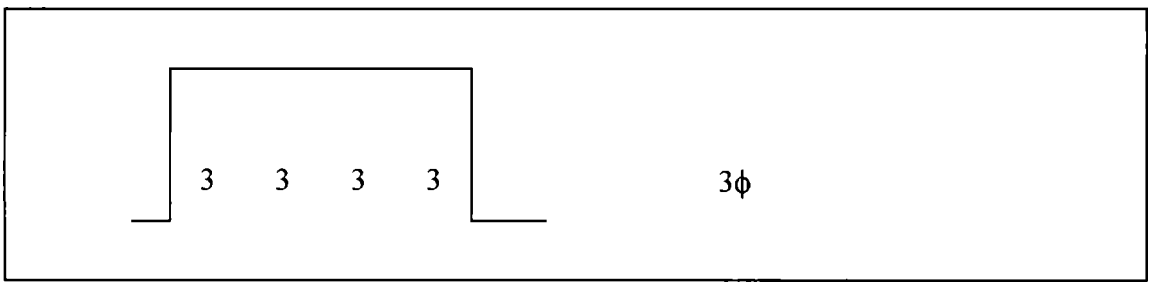


Figura 12 - Función constante a trozos en una unidad de intervalo.

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

Se puede ver cómo contribuyen con el total:

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

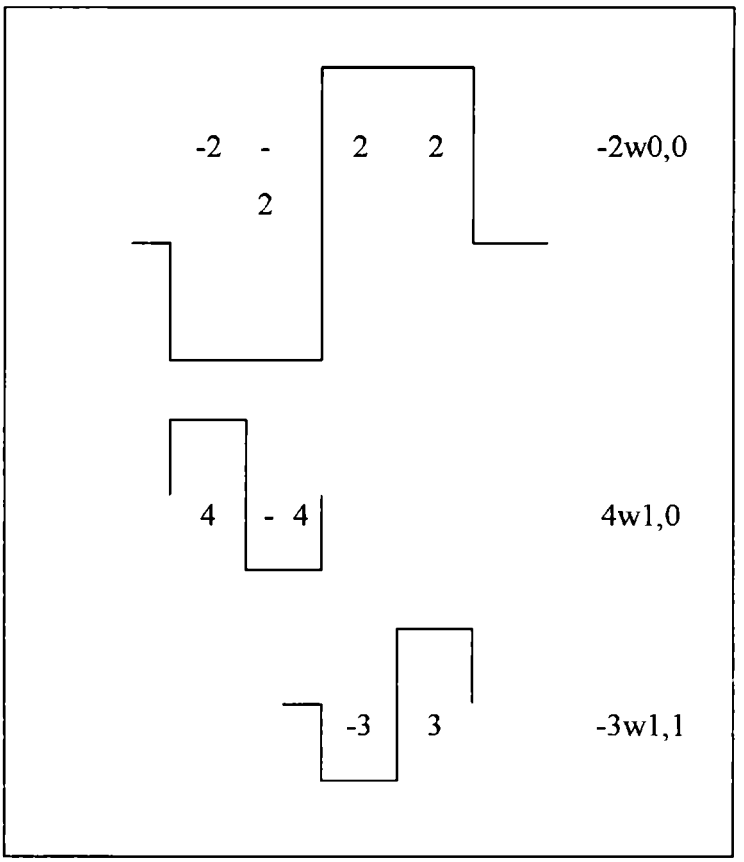


Figura 13 - Las primeras dos generaciones de wavelets.

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

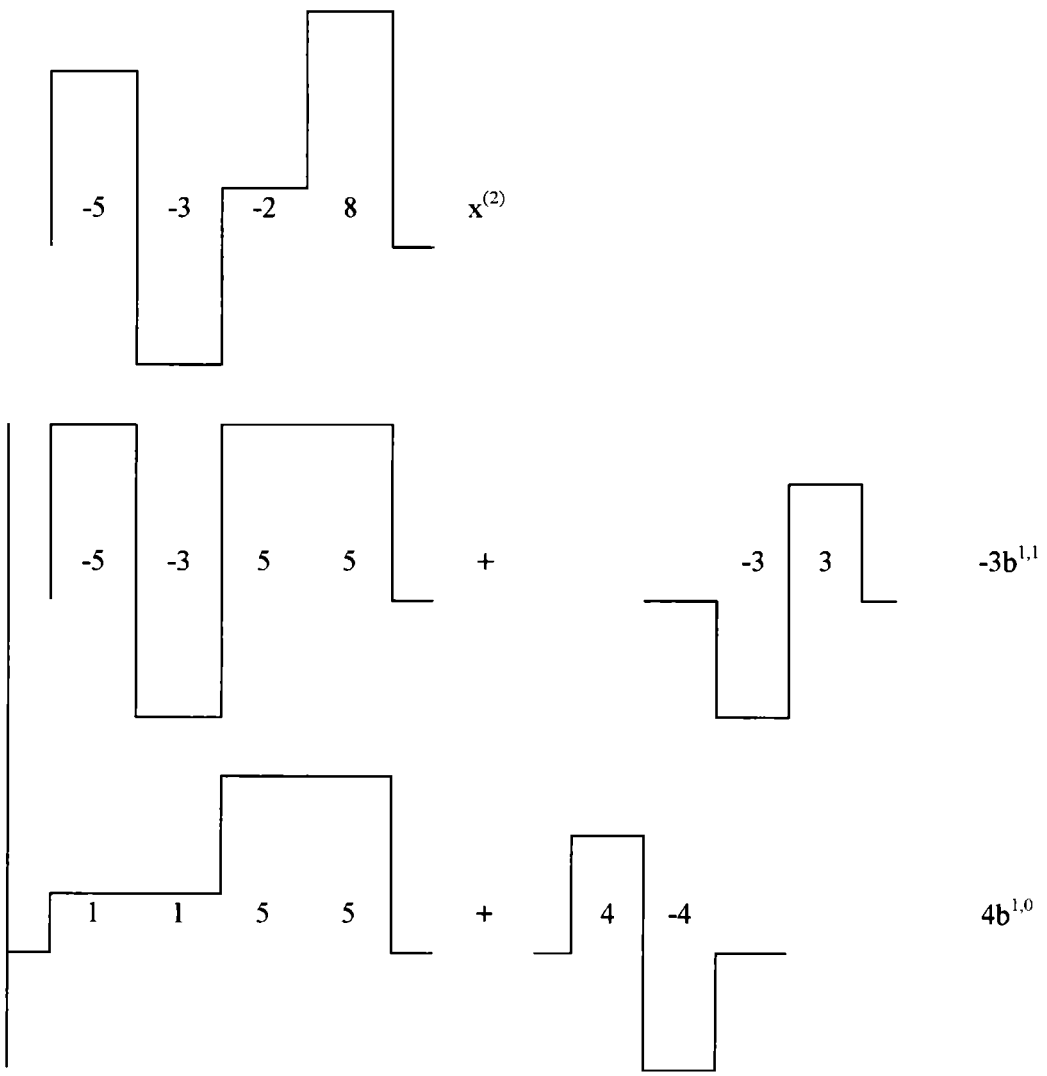
Los índices a y b de los coeficientes se refieren al nivel de la wavelet y su posición. El índice a dice cuánto se dilató la wavelet. Los niveles altos de a indican wavelet con bases afinadas, y son capaces de capturar cambios de información rápidamente. Los valores de b dicen dónde se aloca la wavelet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

La idea básica de las wavelets es filtrar la señal original con un filtro pasabajos para obtener una versión "*suavizada*", y luego encontrar las diferencias entre ésta y la

original. Luego, se suaviza otra vez y se busca las diferencias entre la nueva y la anterior, así continuar hasta que la señal se haya suavizado en un número simple.

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

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



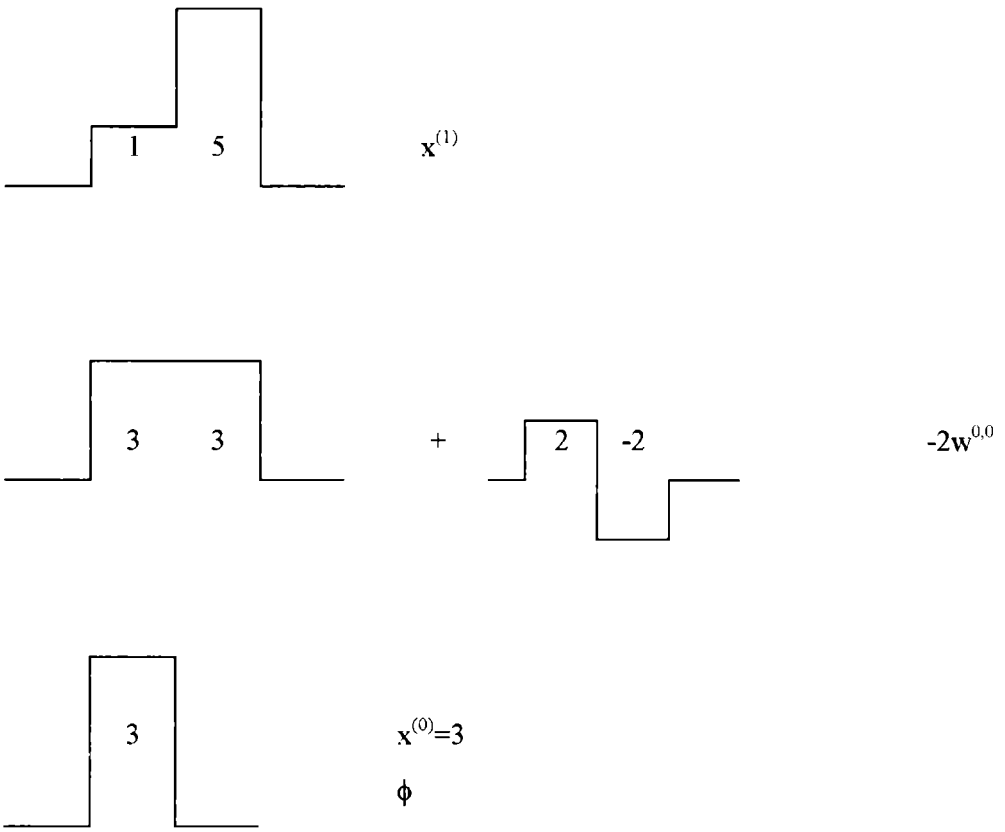


Figura 14 - La señal de entrada $x=[5,-3,2,8]^t$. Debajo, las series de wavelets.

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

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

La transformada de Haar repite un simple paso muchas veces. Se toman dos valores adyacentes en la señal y se lo reemplaza por dos nuevos números: su promedio y la amplitud de la wavelet que se le sumará al promedio para recrear los valores originales. En cada paso se reemplaza un par de valores p y q por su media m y la amplitud de la media w. Dado un par de valores adyacentes p y q:

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

Para recuperar p y q:

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

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

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

Construyendo los operadores

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

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

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

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

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

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

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

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

donde I es el operador identidad.

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

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

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

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

Ahora se define el algoritmo completo para encontrar los coeficientes wavelet.

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

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

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

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

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

Notación de Matriz

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

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

$$[L] = \begin{bmatrix} c1 & c0 & 0 & 0 & & & & \\ 0 & 0 & & & & & 0 & 0 \\ 0 & 0 & c1 & c0 & & & & \\ & & & & & & & \\ & & & & & & & \\ 0 & 0 & & & & c1 & c0 & 0 \\ 0 & 0 & & & & & 0 & 0 \\ c0 & 0 & & & & & 0 & c1 \end{bmatrix}$$

$$
[H] = \begin{bmatrix}
c0 & -c1 & 0 & 0 & & & \\
0 & 0 & & & & 0 & 0 \\
0 & 0 & c0 & -c1 & & & \\
 & & & & & & \\
 & & & & & & \\
0 & 0 & & & c0 & -c1 & 0 \\
0 & 0 & & & & 0 & 0 \\
-c1 & 0 & & & & 0 & c0
\end{bmatrix}
$$

Condiciones de coeficientes

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

Se debe notar que tanto L como H submuestrean sus entradas en un factor de 2. Por lo tanto la forma de la matriz de estos operadores tienen dimensiones n x 2n. Se pueden juntar, intercaladas, para crear una simple matriz A.

$$[A] = \begin{bmatrix} c0 & c1 & 0 & 0 & & & \\ 0 & 0 & -c1 & c0 & & & \\ & & & & & & \\ & & & & & & \\ 0 & 0 & & & c0 & c1 & 0 \\ c0 & 0 & & & 0 & 0 & -c1 \end{bmatrix}$$

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

$$[A]^T = \begin{bmatrix} c0 & 0 & & c0 \\ c1 & 0 & & 0 \\ 0 & -c1 & & \\ 0 & c0 & & \\ & & & \\ & & & \\ & & c0 & 0 \\ & & c1 & 0 \\ & & 0 & -c1 \end{bmatrix}$$

Multiplicando estas dos se ve:

$$[AA^T] = \begin{bmatrix} \alpha & 0 & \beta & 0 & 0 & 0 & \beta & 0 \\ 0 & \alpha & 0 & \beta & 0 & 0 & 0 & \beta \\ \beta & 0 & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & \beta & 0 & \alpha & 0 & 0 & 0 & 0 \\ \cdots & & & & & & & \cdots \\ 0 & 0 & 0 & 0 & \alpha & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & 0 & \beta \\ \beta & 0 & 0 & 0 & \beta & 0 & \alpha & 0 \\ 0 & \beta & 0 & 0 & 0 & \beta & 0 & \alpha \end{bmatrix}$$

donde, $\alpha = c0^2 + c1^2$

$\beta = c0*c2 + c1*c3$

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

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

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

Las wavelets de Haar son de orden cero, es decir, pueden hacer match sólo con funciones constantes a trozos. Se dice que el orden de la wavelet es el tipo de señal con la que pueden hacer match la función de escala. La regularidad de cada función es el número de veces que la función es diferenciable de manera continua. Se puede ir del

orden cero ofrecido por las wavelets Haar a una continuidad de primer orden imponiendo dos condiciones adicionales:

3) c3 - c2 + c1 - c0 = 0

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

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

Análisis de Multiresolución

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

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

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

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

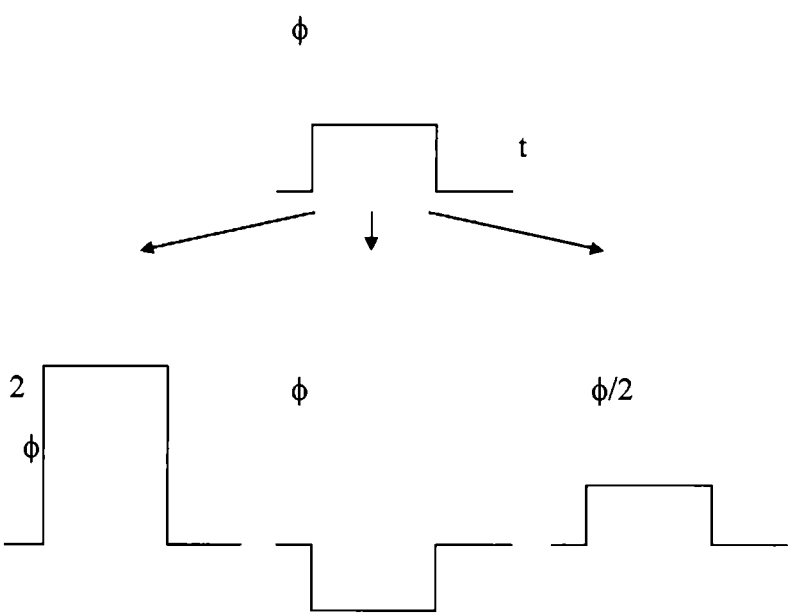
Si se combinan estos dos espacios por una suma cartesiana, se genera un tercer nuevo espacio V_1

$$V_1 = V_0 \oplus W_0$$

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

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

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



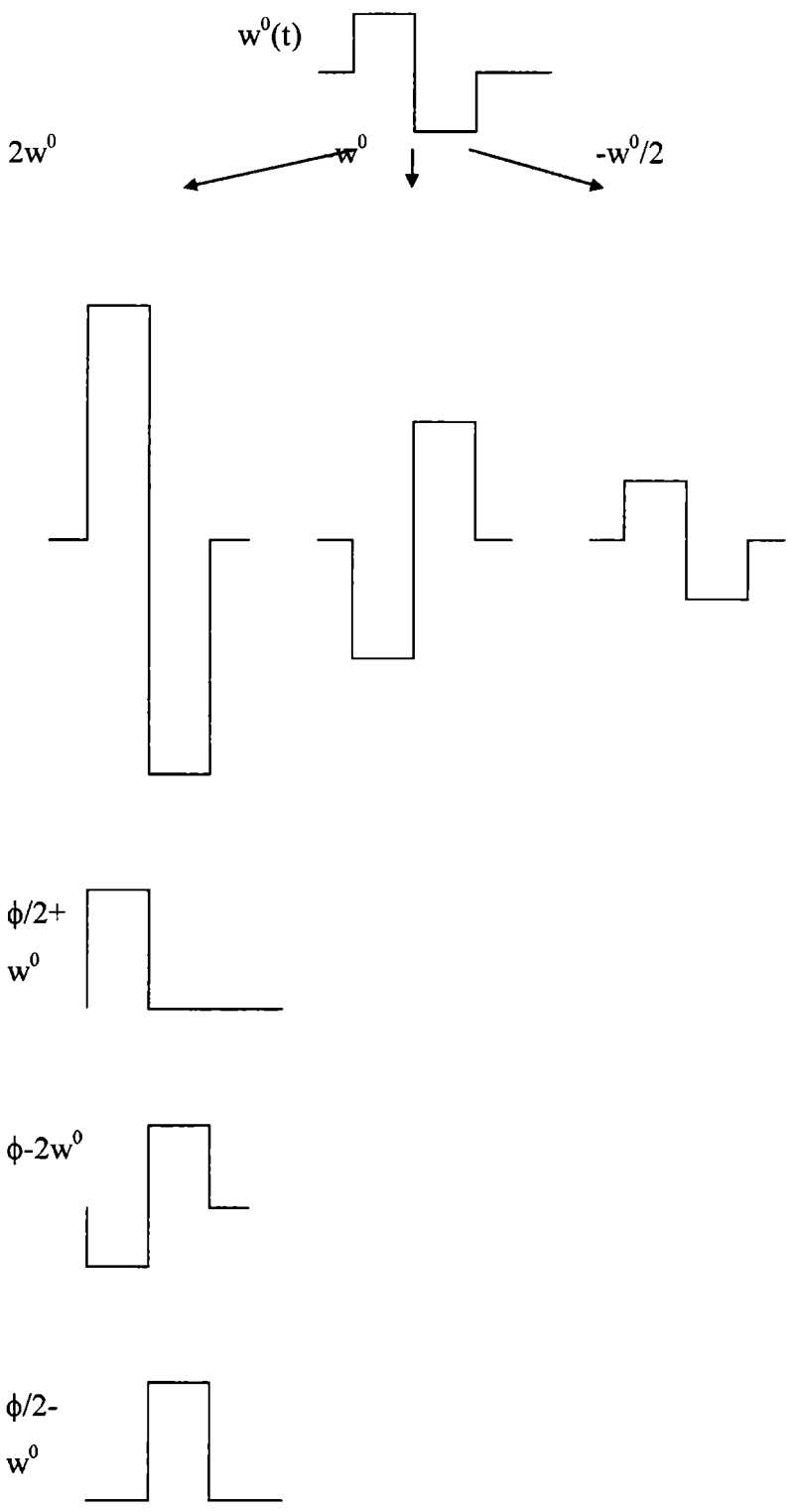


Figura 15

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

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

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

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

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

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

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

1. Contención: $V_m \subset V_{m+1}$.
2. Distinción: $\cap_m V_m = 0$.
3. Completitud: $\cup_m V_m = L^2(R)$.
4. Desplazamiento: si $f(t) \in V_m$, entonces $f(t-k) \in V_m$ para todos los $k \in Z$.
5. Escala: si $f(t) \in V_m$, entonces $f(2t) \in V_{m+1}$ para todas las $f \in L(R)^2$.
6. Base: Existe una función de escala $v(t) \in V_0$, tal que para todo m, el conjunto de funciones

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

forma una base ortonormal para V_m; esto es

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

Wavelets en el Dominio de Fourier

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

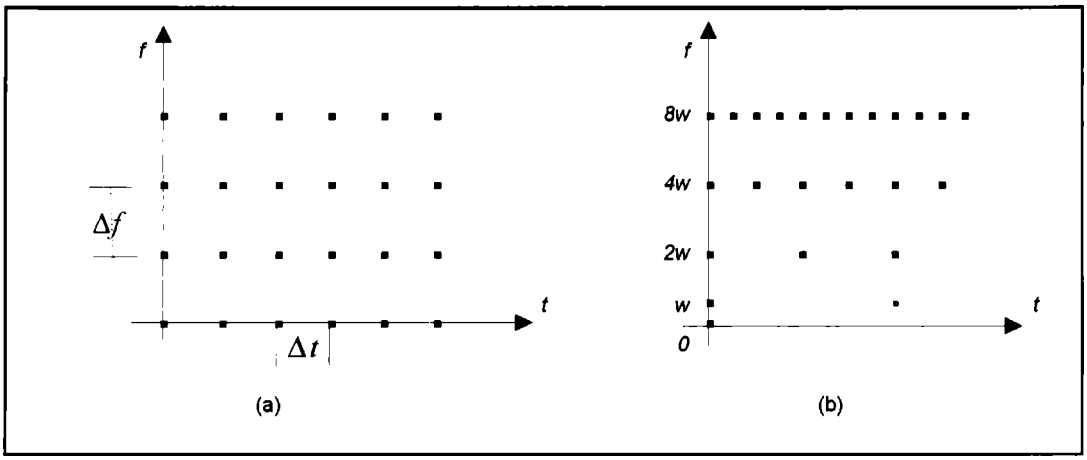


Figura 16 - (a) STFT ; (b) Wavelet.

t representa el tiempo, f la frecuencia.

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

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

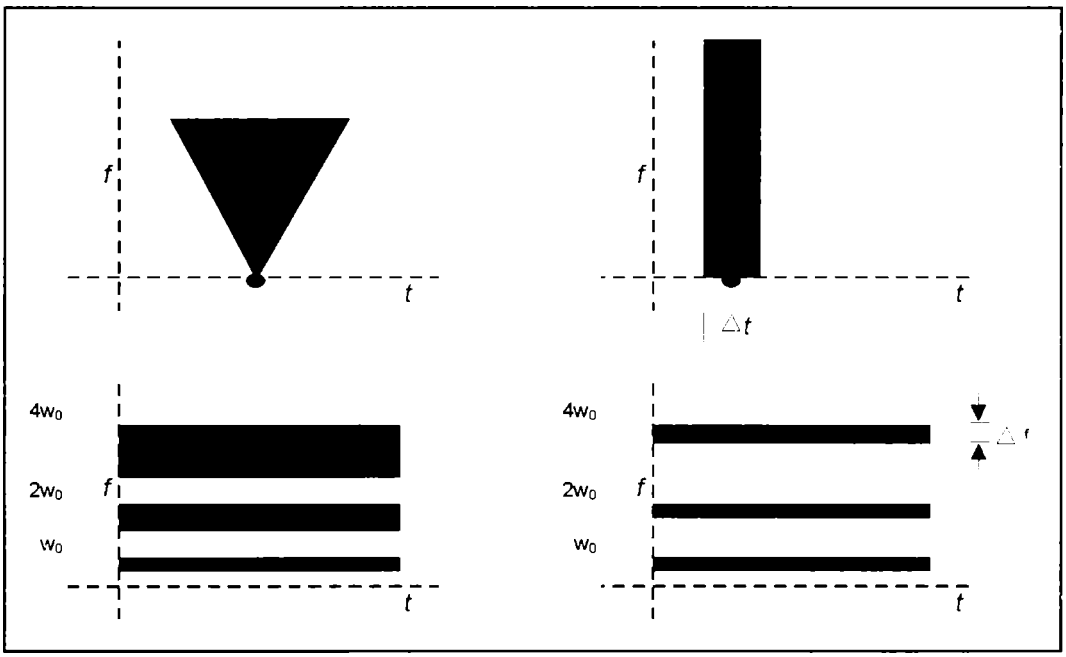


Figura 17 - Escalograma y espectograma de la función impulso.

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

Wavelets en dos Dimensiones

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

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

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

Descomposición wavelet rectangular

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

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

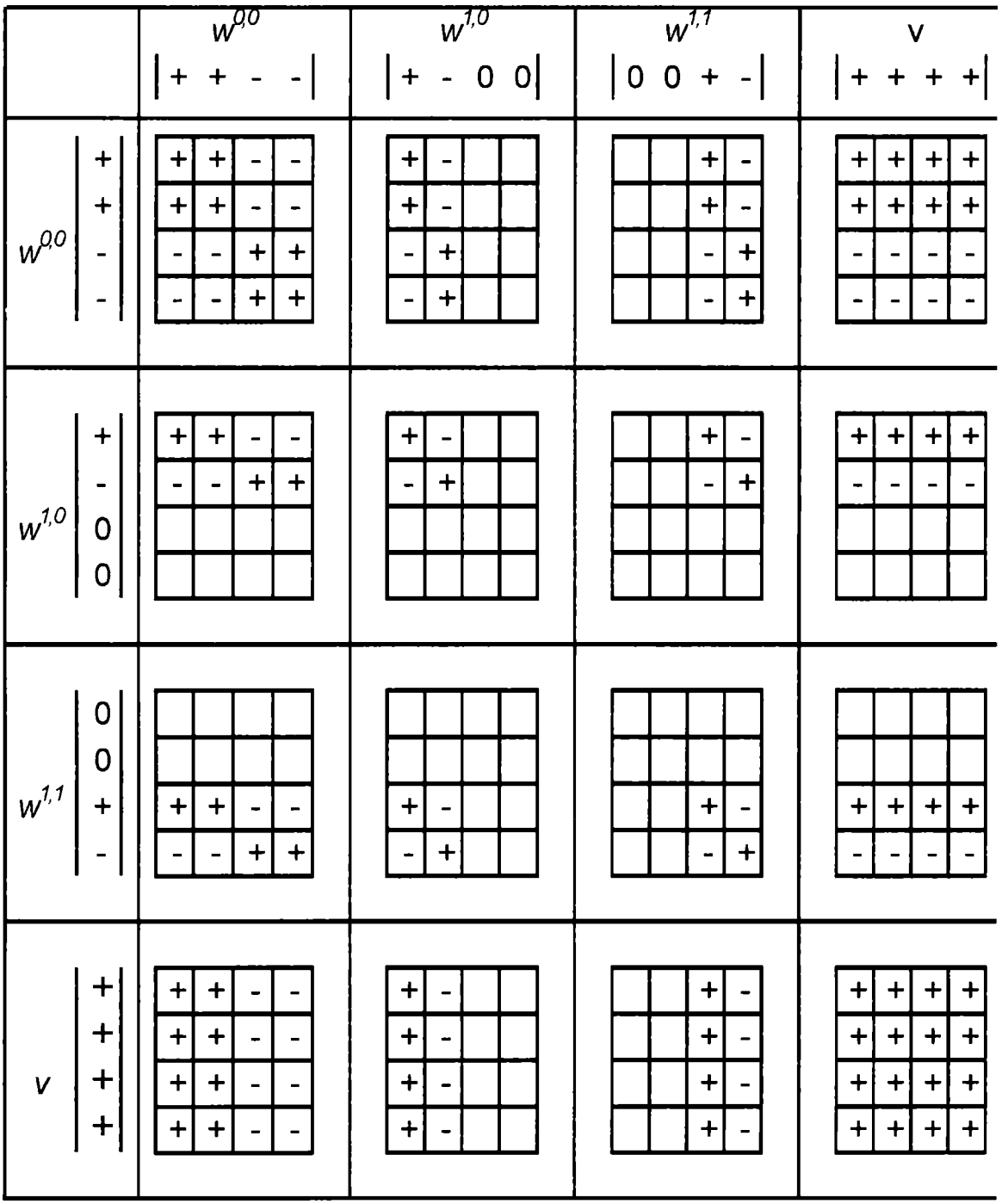


Figura 18

Las 16 funciones base para una descomposición wavelet rectangular de una matriz de 4x4. Cada función está hecha por el producto de tensor de la función del tope

de la fila con la función de la izquierda ; por ejemplo : la 3° función del tope de la fila está dada por $w^{1,1} \otimes w^{0,0}$.

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

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

Descomposición wavelet cuadrada

Se ha visto que las funciones bases construidas anteriormente tienen regiones cuadradas y rectangulares de elementos distintos de cero. Esto es útil cuando se quiere construir una descomposición con diferentes rangos de escala en dimensiones diferentes. Si se tratan todas las dimensiones de la misma manera, se puede construir una descomposición que tenga solo términos cuadrados; esto es llamado descomposición cuadrada (o no estándar).

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

Se comenzará con la función base más simple formada por v ⊗ v; esto es, una caja. El espacio que comprende a todas las trasladadas de v contiene todas las funciones constantes sobre cuadrados de tamaño entero, y se llamará $V_0^{(2)}$. Sería bueno moverse a un espacio $V_1^{(2)}$, que fuera constante a trozos sobre cajas de la mitad del tamaño entero. Para moverse de un espacio de menor resolución a uno de mayor resolución, se hace el producto cartesiano de $V_0^{(2)}$ con el espacio de detalle $W_0^{(2)}$, tal que

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

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

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

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

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

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

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

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

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

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

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

Wavelets y Compresión de Señales

La idea subyacente de cualquier esquema de compresión es eliminar la correlación presente en los datos. Como se mencionó anteriormente algunos tipos de correlación son [IEEE, 1995, 2]:

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

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

- ✓ la matriz de correlación no se conoce,

- ✓ calcular los vectores de esa matriz tiene complejidad cúbica,
- ✓ aún conociendo la base óptima, calcular la transformada es un algoritmo cuadrático,
- ✓ la base depende del conjunto de datos.

Esto dice que se necesita una transformada con las siguientes propiedades:

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

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

Se necesita una base local en espacio y en frecuencia que puede obtenerse:

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

Medida del Error

Para los esquemas de compresión con pérdida usualmente se quiere que la imagen descomprimida (recuperada) tenga la misma calidad visual que la original.

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

- Raíz del error cuadrático medio:

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

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

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

Transformada wavelet

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

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

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

Cuantificación

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

Codificación

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

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

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

$$\sum_i p_i = 1$$

El contenido de información o entropía ahora está dado por

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

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

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

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

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

Transformada Burrows Wheeler (BWT)

Introducción

En matemática, problemas dificiles a menudo pueden ser simplificados realizando un transformación en el conjunto de datos. Por ejemplo, programas de procesamiento digital de señales utilizan la transformada de Fourier para convertir datos de muestras de audio a conjuntos equivalentes de datos de frecuencia. Realizar un filtrado pasa bajos ahora sólo es cuestión de multiplicar los puntos por un factor de escala. Realizando la transformación inversa y obtenemos la nueva forma de onda de audio correspondiente al filtrado.

Michel Borrow y David Wheeler publicaron los detalles de una nueva transformada que abre las puertas a nuevas y revolucionarias técnicas de compresión. La transformada Borrow Wheeler (BWT) reorganiza un bloque de datos en una forma que está muy bien caracterizada para ser comprimida.

Michel Borrow y David Wheeler publicaron un reporte en 1994 comentando el trabajo que habían estado haciendo en el Centro de Investigación de Sistemas Digital. Su paper, "*A Block-sorting Lossless Data Compression Algorithm*", presentó un algoritmo de compresión de datos basado en una transformada nunca publicada previamente descubierta por Wheeler en 1983.

Mientras que este paper discute un conjunto completo de algoritmos para la compresión y descompresión, el corazón del paper es la transformada BWT.

El algoritmo BWT reordena un bloque de datos usando un algoritmo de ordenación. El resultado es un bloque de datos que contiene exactamente los mismos datos que el bloque original, con la única diferencia en el orden de los datos. La transformación es reversible, significa que el orden inicial de los elementos puede ser restaurado sin pérdida de fidelidad.

La transformada BWT trabaja sobre bloques de datos a diferencias de los algoritmos usuales que operan sobre uno o unos pocos bytes a la vez. La compresión que permite alcanzar esta transformada mejora a medida que aumenta el tamaño de bloque, cosa que generalmente no sucede con los algoritmos basados en diccionarios por la perdida de localidad del diccionario.

Debemos resaltar que el algoritmo en sí no comprime, solo reordena los datos de una manera que permite que otro algoritmo de compresión obtenga un alto grado de compresión. Obviamente muy superior al que obtendría trabajando sobre el bloque original.

Para ilustrar el funcionamiento del algoritmo se utiliza el siguiente conjunto de datos:

TRABAJO

Figura 1: Nuestros datos de entrada

El algoritmo toma un bloque de datos y realiza una matriz de NxN (N tamaño del bloque), donde la fila 0 es el bloque original, la fila 1 es la rotación de la fila 0 hacia la izquierda en un carácter y así sucesivamente.

S0	TRABAJO
S1	RABAJOT
S2	ABAJOTR
S3	BAJOTRA
S4	AJOTRAB
S5	JOTRABA
S6	OTRABAJ

Figura 2: Matriz de string rotados

Luego se ordenan lexicográficamente las filas.

	F L
S2	ABAJOTR
S4	AJOTRAB
S3	BAJOTRA
S5	JOTRABA
S6	OTRABAJ
S1	RABAJOT
S0	TRABAJO

Figura 3: Matriz de string rotados ordenados

En este momento hay dos puntos que conviene resaltar. Primero, los string han sido ordenados pero hemos llevado el rastro de la posición de cada string en la matriz original. Por lo tanto el String 0 (el string original) ha bajado hasta la fila 6 de la matriz.

Segundo, hemos marcado la primera y última columna de la matriz con las letras F (First) y L (Last). La columna F contiene los caracteres del string original en forma ordenada; por lo tanto el string original TRABAJO está representado en F por AABJORT.

Los caracteres en la columna L, aparentemente están completamente desordenados, pero en realidad ellos respetan una interesante propiedad: cada carácter en L es el prefijo del string que comienza en la misma fila en la columna F.

La salida de la BWT está compuesta por dos elementos: Una copia de la columna L y un entero indicando que fila de L contiene el primer carácter del string original (Indice primario). Por lo tanto la salida de BWT aplicada a nuestro string es RBAAJTO, y el índice primario 5.

El índice primario 5 fácil de encontrar ya que el primer carácter del buffer se encontrará en la columna L en la fila de la matriz que contenga a S1. Por lo tanto ubicar la fila de L en la que está el primer carácter del buffer equivale a ubicar la fila de la matriz que contiene S1.

En este momento surgen dos preguntas. Primero, no parece que esta sea una transformación reversible. Generalmente, una función sort() no viene con un unsort() asociado que restaure el orden original. Segundo, que es lo bueno que esta extraña transformación provee.

Para obtener el buffer original a partir de la columna L se requiere el uso de un vector de transformación, este es un arreglo que define el orden en que los string rotados fueron reordenados a través de las filas de la matriz.

El vector de transformación, digamos T, es un arreglo con un índice para cada fila. Para una fila dada j, T[j] se define como la fila donde se encuentra S_{i+1} suponiendo que S_i se encuentra en la fila j. En la Figura 3, la fila 6 contiene S_0, el string original de entrada, y la fila 5 contiene S_1, el string original rotado un carácter a la izquierda. Por lo tanto, T[6] contiene el valor 5. S_2 es encontrado en la fila 0, por lo tanto T[5] contiene el valor 0. Para nuestro ejemplo, el vector de transformación es [2, 3, 1, 4, 6, 0, 5].

La Figura 4 muestra como es usado el vector de transformación para recorrer las distintas filas. Para cualquier fila que contiene S_i, el vector provee el valor de la fila donde se encuentra S_{i+1}.

El vector de transformación da la clave para restaurar L a su orden original. Dado L y el índice primario, se puede restaurar el S_0 original. Dada una copia de L, usted puede calcular el vector de transformación para la matriz original de entrada. Y consecuentemente, dado el índice primario, usted puede recrear S_0.

La clave para recrear el vector de transformación es que usted puede hacerlo si conoce L y F. Además, si usted tiene L, tiene F (L ordenada lexicográficamente). Por lo tanto, usted puede generar el vector de transformación a partir de L.

Calculemos T para nuestro conjunto de trabajo

	LF
	R
	B
	A
	A
	J
	T
	O

Figura 4: Estado inicial en la reconstrucción de T dado L

Como vemos en la Figura 4, dada una copia de L no conocemos mucho acerca de la matriz original (movimos la L a la primera columna con propósitos de ilustración). Podemos obtener F ordenando los caracteres en L.

	LF
	RA
	BA
	AB
	AJ
	JO
	TR
	OT

Figura 5: Reconstrucción de T dado L

Ahora comenzamos el cálculo de T. El carácter R en la fila 0 de L se mueve a la fila 5 de la columna F lo que significa que el valor de T[5] es 0. El carácter B en la fila 1 se mueve a la fila 2 por lo que T[2] es 1. Pero que pasa con la letra A en la fila 2 ? La A aparece más de una vez en ambas columnas (en particular en la columna F).

La elección no es ambigua. A pesar de que el proceso de decisión no es intuitivo. Por definición, la columna F está ordenada y todos los string comenzando con A en la columna L también están ordenados. Por que ? Todos los string que comienzan con el mismo carácter están ordenados según su segundo carácter (más precisamente, por el todo el substring restante).

Ahora, cuando dichos caracteres aparezcan en L, estarán ordenados de acuerdo al mismo substring antes mencionado.

Este análisis se puede generalizar para cualquier carácter x:

xSS1
xSS2
xSS3

SS1x

SS2x

SS3x

Siguiendo este proceso obtenemos el siguiente vector de transformación: [2, 3, 1, 4, 6, 0, 5]. Les parece familiar este vector ?

Una vez que esta dificultad ha sido esclarecida recuperemos S_0 utilizando el siguiente algoritmo:

```
int T[] = { 2, 3, 1, 4, 6, 0, 5 };
char L[] = "RBAAJTO";
int indice_primario = 5;

void decode()
{
        int index = indice_primario;

        for(int i = 0; i < strlen(L); i++) {
                printf("%c", L[index]);
                index = T[index];
        }
}
```

La salida de este programa es TRABAJO.

Hemos contestado la primera de las dos preguntas que nos formulamos arriba. Ahora pasaremos a contestar la segunda que se refería a los beneficios que esta transformación puede ofrecernos. Para hacerlo veamos el siguiente cuadro, el cual corresponde a la aplicación de la transformada a este documento:

La tabla tiene 2 columnas:

- L: salida de la transformada (truncada)
- S_i: string asociado al carácter de salida (L es el último carácter del string S_i)

S_i	L
avid Wheeler.<13><10><13><10>El algoritmo	D
bajo "Compresi<243>n de datos que	**a**
bamos con algunos ejemplos.<13><10><13>	**o**
blicada en el paper "A Block-s	**u**
blicado por Michel Borrow y Da	**u**
bloque de datos que contiene e	
bloque de datos usando un algo	
bloque original, con la única	
bre una red virtual de procesa	**o**
buffer, scatter , gatter,etc.	
buffereados, sin buffer, scatt	
buida. Una comparaci<243>n con Soc	**i**
c, asinc, buffereados, sin buf	n
c, buffereados, sin buffer, sc	n

Concentremos nuestra atención en la sección de la salida correspondiente a todos los string que comienzan con el carácter b. No es sorprendente que el prefijo de los que comienzan con bloque * sea un blanco. O que el prefijo de los que comienzan con blicad* sea una u. O sea que la salida de la transformada tiene el siguiente formato:

prefijos del SR_0, prefijos de SR_1, ...

donde SR_i denota el string correspondiente a la fila i de la matriz ordenada.

Notar que son prefijos de un string y no de un carácter en particular. Por ejemplo, no es lo mismo trabajar sobre los prefijos de r que sobre los prefijos *de "rabajo ....*". Obviamente, el segundo conjunto está mucho más acotado.

La idea en este momento es utilizar compresores que exploten el formato de la salida de la BWT. Por ejemplo, los radios de compresión resultantes de aplicar un Huffman Semi-estático al archivo original y al archivo BWT-transformado correspondiente son los mimos ya que el histograma asociado a ambos es el mismo por lo que no tendría sentido aplicar la transformada.

Este es un punto bastante poco explotado, pero un ejemplo de uno de esos algoritmos es el que sugieren los autores, el MTF (Move To Front), seguido de un codificador de la entropía (por ejemplo Huffman).

El codificador MTF es otra transformación reversible, y tampoco comprime por sí mismo sino que, al igual que la BWT deja los datos en un formato muy apropiado para una posterior compresión. Este codificador mantiene todos, los 256, posibles caracteres en una lista. Cada vez que uno de ellos está por ser "*enviado*", se envía la posición de ese carácter en la lista y luego este es puesto al frente de la lista. Si la entrada del MTF es la salida de la BWT la salida del MTF será un conjunto de enteros pequeños, algunas corridas de 0 y algún que otro entero grande correspondiente, probablemente a la primera aparición de un carácter.

Como ejemplo, la salida MTF correspondiente a la sección de L en negrita de nuestro ejemplo ("*aouu o i*") es, [97, 111, 117, 0, 32+3, 0, 0, 2, 1, 0, 105+2]. Notemos que no se ha ganado demasiado debido al tamaño de la entrada. Estas transformadas mejoran a medida que aumenta el tamaño de la entrada. Como ejemplo, piense que pasaría si aplicáramos estas transformadas a un libro.

JPEG con Particionamiento Adaptivo

Generalización a particionamiento fijo utilizando JPEG estándar

La primer implementación es una generalización para particionar la imagen en bloques de tamaños fijos (4x4, 8x8, 16x16, 32x32), Se utilizan diferentes tablas de cuantificación y codificación: para particionamiento fijo con bloques de 8x8, se utilizó la matriz de cuantificación default y los distintos niveles de pérdida se lograron multiplicando esta matriz por un escalar o factor; las tablas de codificación Huffman para bloques de 8x8 son las provistas por el standard. Para los demás tamaños de bloques se construyó la siguiente matriz de cuantificación: =1+(1+i+j)*factor; 0<= i, j < N

Si bien existen técnicas para construir tablas de cuantificación adaptivas, el criterio utilizado es que generalmente los coeficientes decrecen en importancia desde del vértice superior izquierdo hacia el inferior derecho. Por esta razón la matriz se construye de tal forma que los coeficientes mas cercanos a la posición (0,0) conserven mayor precisión a la hora de ser decuantificados. Las tablas de códigos Huffman se construyeron realizando un recorrido sobre diversas imágenes sobre las que se aplicaron distintos grados de pérdida. Estos recorridos permitieron extraer las probabilidades para los símbolos, que se eligieron de acuerdo al tamaño del bloque.

Observaciones y Resultados de la Generalización

Se utilizo la imagen VLSI.BMP para realizar las pruebas, es una fórmula escrita en blanco sobre un fondo negro, a la cual se le extrajo una subimagen de 64x64 pixels perteneciente a la parte de color negro. A esta subimagen se le aplicó el algoritmo de particionamiento de tamaño fijo para los 4 tipos de bloques con factor 1 y se obtuvieron radios de 18, 36, 55 y 65 para N=4, 8, 16 y 32 respectivamente los cuales permanecían fijos a pesar de hacer variar los factores, en tanto que las imágenes

descomprimidas no presentaban cambios en el color negro, por lo menos visualmente. Luego se aplicó el mismo algoritmo a la imagen original completa y se obtuvieron los siguientes radios:

N=4		N=8		N=16		N=32	
Factor	Radio	Factor	Radio	Factor	Radio	Factor	Radio
2	6	2	13	2	6	5	13
8	8	7	24	6	11	9	21
18	10.4	9	27			10	23
25	10.8						

Los resultados óptimos en calidad visual se lograron con el procesamiento en bloques de 4x4, aunque los radios de compresión aumentaron con menos rapidez al aplicar factores mayores a 18. Como se observa en la tabla para factores 18 y 25 la diferencia en radio es mínima aunque con calidades similares en las imágenes descomprimidas. Con factores 7, 6 y 10 para bloques de 8x8, 16x16 y 32x32 respectivamente las distorsiones en la zona de la fórmula comenzaron a ser más notorias. Si bien se logró mayor compresión con tamaños de bloques más grandes se produjeron distorsiones visibles que afectaban las regiones de color negro alejadas de la fórmula.

Particionamiento Adaptivo Quadtree

Este particionamiento es utilizado en el procesamiento de imágenes, tratando de superar las dificultades presentadas en el particionamiento fijo (mencionado anteriormente). El algoritmo básico consiste en tomar un bloque cuadrado de imagen el cual es dividido en cuatro subloques cuadrados de igual tamaño. Esto se repite recursivamente desde la imagen original completa hasta que los cuadrados alcancen un tamaño deseado de acuerdo a un test de decisión, que dependerá de la aplicación. Se puede realizar el mismo particionamiento pero en forma button up comenzando el proceso con los subloques más chicos y uniéndolos para formar otros más grandes [Gersho].

Generalización utilizando particionamiento Quadtree

Con Quadtree se trata de adaptar las diversas zonas de la imagen a un determinado tamaño de bloque, de acuerdo a las variaciones de los niveles de grises dentro del mismo. Zonas grandes con cambios lentos en los niveles de detalle, son tratadas con bloques mas grandes y zonas con muchos detalles sean tratadas con bloques mas pequeños. Esta implementación se basa en un esquema propuesto por Chen (1989) en el cual la imagen es dividida en bloques iniciales de 32x32, los que a su vez son particionados completamente hasta obtener bloques de 4x4. Se aplica un test sobre 4 subloques hermanos de 4x4 para determinar si estos se procesan individualmente o bien son unidos para formar un posible bloque de 8x8. El mismo proceso se realiza para cada 4 bloques hermanos de 8x8, y 16x16.

El test es de la forma: *if abs(Mk(i)-Mk(j))>Tk para cualquier i=j*

Procesar los 4 subloques individualmente;

Else

Unir los subloques y marcarlos como un posible mas grande;

Los términos Mk(i) y Mk(j) (i,j=1..4) representan promedios de niveles de grises de bloques hermanos en el k-ésimo nivel del árbol y Tk, el límite de decisión. La idea es que para determinar cuan activa es una zona de NxN la dividimos en 4 bloques iguales, hallamos sus promedios y vemos que tan similares son. Si no existe ningún par de promedios cuya diferencia supere un límite establecido previamente entonces consideramos a la zona de NxN como poco activa (un límite sugerido es 10 para imágenes de 8 bits) y la procesamos como un solo bloque. En caso contrario la dividimos. Para representar la información del header solamente necesitamos transmitir un bit por cada subloque indicando si este se particiona o no, con lo cual en el peor de los casos codificaremos 21 bits por cada bloque de 32x32 equivalente a un costo de 0.02 bits por pixel. De acuerdo al tamaño de cada subloque del particionamiento final (4x4, 8x8, 16x16 o 32x32) el procesamiento es similar al aplicado por JPEG a cada bloque de 8x8 con la diferencia de que el coeficiente DC solo se codifica con respecto al mismo

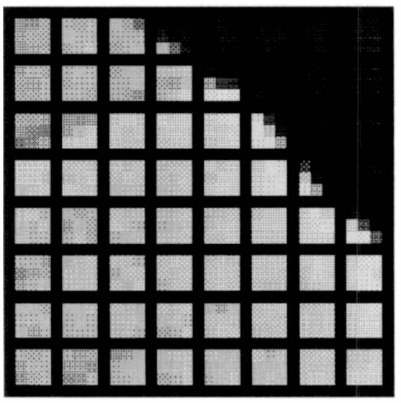
elemento del bloque anterior si es que ambos bloques tienen el mismo tamaño. Si el bloque anterior es de tamaño diferente se codifica el mismo valor del coeficiente cuantificado. Se pueden variar los factores de pérdida para los distintos tamaños de bloques de acuerdo al nivel de calidad deseado para las distintas zonas. Por ejemplo, si se desea aumentar la pérdida en las zona con mucha actividad lo mas conveniente es aumentar el factor de pérdida para los bloques mas chicos, debido a que es de esperarse que estos tamaños de bloques caigan en esas zonas. De la misma manera que si se quiere disminuir la calidad de la imagen en zonas grandes con poca actividad se debe ajustar los factores para los tamaños de bloques mas grandes. En general las zonas con poca actividad forman parte del fondo de la imagen y en consecuencia serán procesadas con bloques mas grandes (32x32, 16x16).

Tres ejemplos de secuencias de pasos que se siguen para particionar un bloque de 32x32

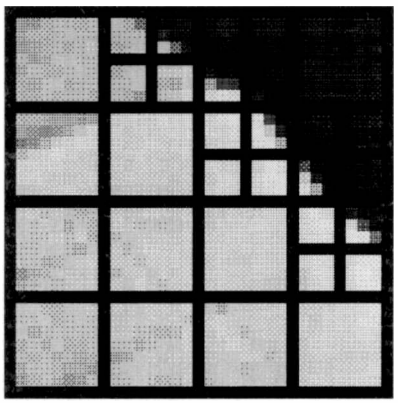
Imagen particionada con limites de 20,20,20

Bloque del sombrero

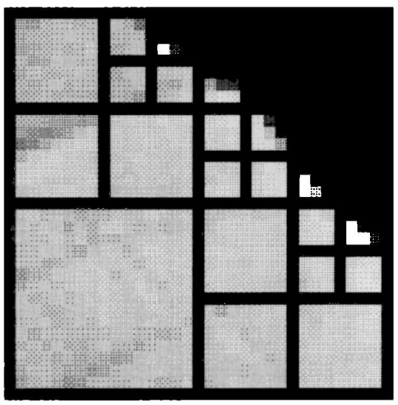
Paso1



Paso 2

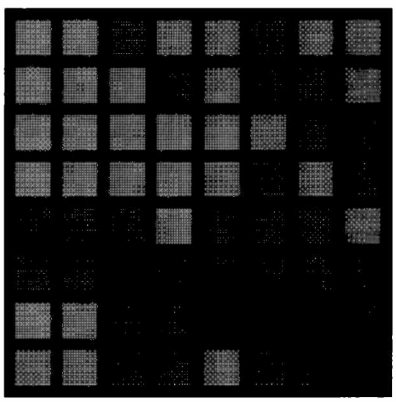


Paso 3

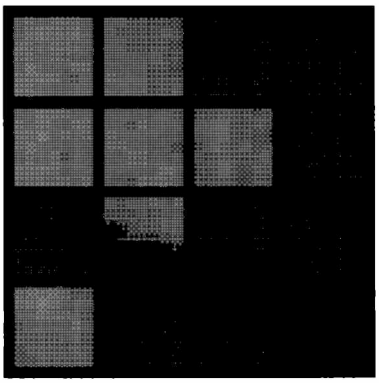


Bloque de espejo

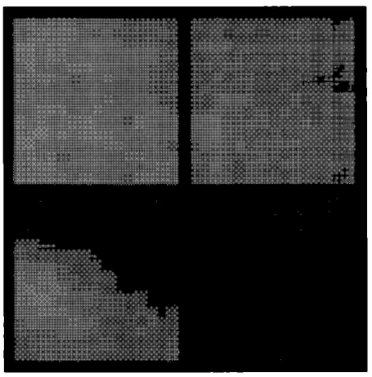
Paso 1



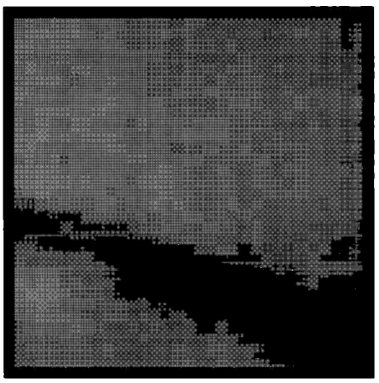
Paso 2



Paso 3

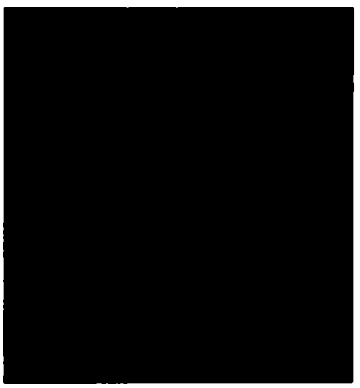


Paso 4

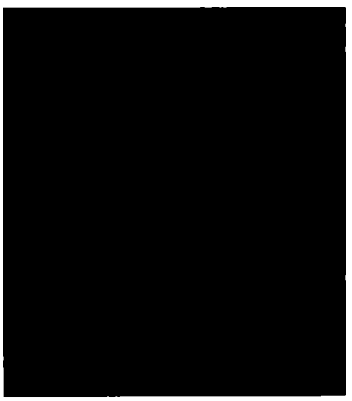


Bloque del omoplato

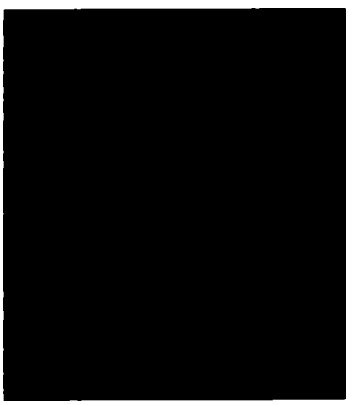
Paso 1



Paso 2



Paso 3

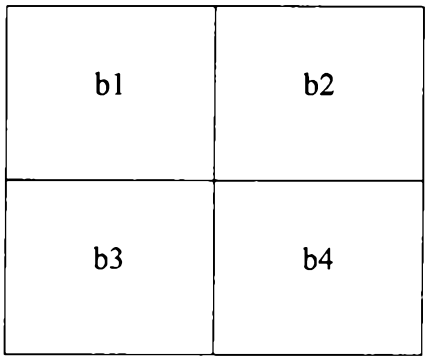


En síntesis esta técnica de compresión cuenta con los siguientes parámetros que pueden ajustarse: Límite de decisión para decidir si particionar o no un bloque de 8x8, 16x16, 32x32; factor de pérdida para bloques de 4x4, 8x8, 16x16 y 32x32.

El siguiente es un pseudo código lineal para realizar un particionamiento quadtree de un bloque de imagen de 32x32 pixels, dado los límites de decisión cuando se tiene los promedios de 4 subbloques adyacentes de nxn= 4,8,16:

```
n=4
While n<=16 do
  begin
      While (haya bloques adyacentes de nxn sin procesar) do
        begin
            Tomar los próximos subbloques adyacentes de nxn={b1,b2,b3,b4}
            Calcular los promedios de los valores de los pixels de los 4 subbloques
                  Promedio[0,0] = promedio del bloque b1
                  Promedio[0,1] = promedio del bloque b2
                  Promedio[1,0] = promedio del bloque b3
                  Promedio[1,1] = promedio del bloque b4
            if (particiona (limite para bloques de nxn, promedio ) then
                  procesar los 4 subbloques en forma independiente
            else
                  formar un solo bloque de 2nx2n
        end
      n=n*2
  end
```

Esquema simple de 4 subbloques:



El pseudocódigo de la rutina Particiona utilizada para decidir si un subbloque de nxn es particionado o no dados los promedios de los valores de los pixels de cada uno de sus 4 subbloques y un límite de decisión será:

```
Particiona (lim, promedios[2][2] )
        if ( |promedios[i,j] – promedios[x,y] | >lim then
                particiona = true
        else
                particiona = false

(i,j <> x,y) i,j,x,y = 0,1
```

Análisis Comparativo. Experiencias

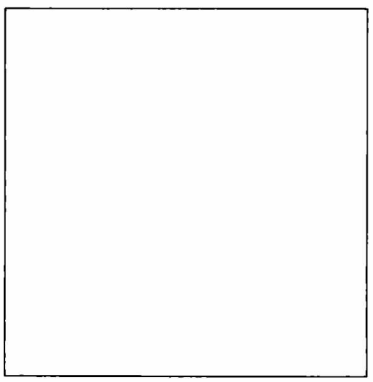
Compresión de datos utilizando Wavelets y comparación con el método JPEG

RESULTADOS

A continuación se muestran imágenes en tonos de gris y las respectivas imágenes resultantes de la compresión, descompresión wavelet y la imagen diferencia, de cuya comparación puede observarse la relación factor de compresión - calidad visual.

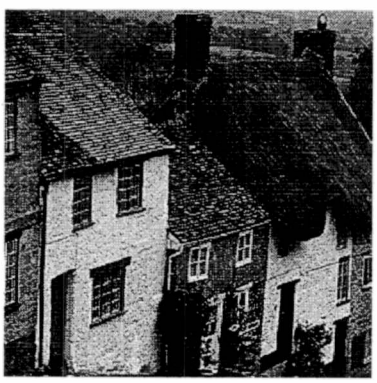
lena.bmp

lena7.bmp

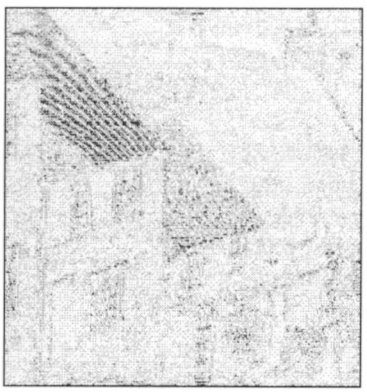


pérdida

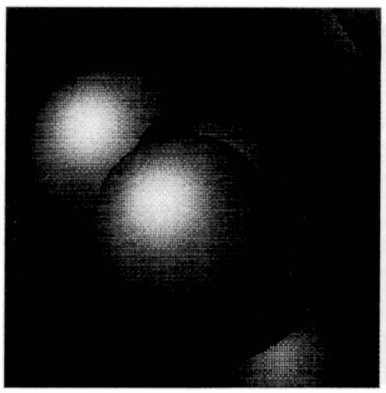
land.bmp

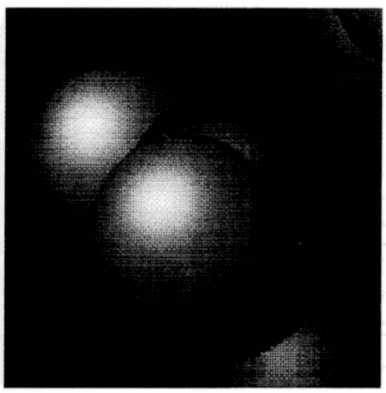


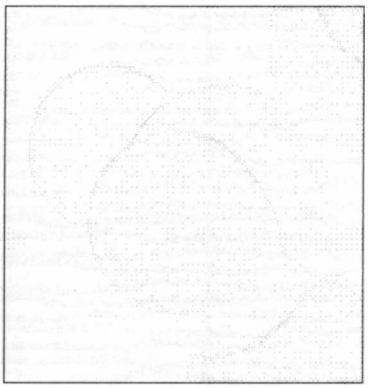
land7.bmp



pérdida


mole.bmp


mole7.bmp


pérdida

Factor de Compresión: bueno

(Con filtros de Daubechies)

IMAGEN	TAMAÑO ORIGINAL (BITS)	COMPRESIÓN		COMPRESIÓN S/HUFFMAN		COMPRESIÓN S/RUNLENGTH		COMPRESIÓN S/HUFFMAN Y RUNLENGTH	
		BITS	RADIO	BITS	RADIO	BITS	RADIO	BITS	RADIO
LENA	65.536	6359	10,3	7964	8,22	7358	8,90	21651	3,02
MOLE	65.536	4157	15,7	4711	13,91	5886	11,13	23277	2,81
LAND	65.536	10281	6,37	12103	5,41	10747	6,09	19203	3,41

(Con JPEG Base Line)

IMAGEN	TAMAÑO ORIGINAL (BITS)	COMPRESIÓN RADIO
LENA	65.536	8.80
MOLE	65.536	25.24
LAND	65.536	4.03

CONCLUSIONES

Analizando los resultados del cuadro anterior, concluimos que la aplicación de la transformada wavelet sobre los datos conduce a una pérdida menos perceptible de calidad visual debida a la cuantificación. Puede agregarse que si bien por este último paso se reduce notablemente el tamaño de la imagen, es la codificación entrópica, llevada a cabo con códigos RunLength y Huffman, la que permite un acercamiento de la longitud promedio de los mismos al límite marcado por la entropía.

El uso de la transformada wavelet para compresión de imágenes resulta en una buena relación entre factor de compresión y calidad visual, puesto que a través del filtrado son separadas las distintas componentes frecuenciales y luego son comprimidas de acuerdo a la importancia que tengan para el "*ojo humano*". La aplicación de la transformada sobre toda la imagen en lugar de por bloques, impide que los mismos se noten en la imagen resultante, como ocurre con la transformada del coseno.

Como trabajo futuro se sugiere la compresión de imágenes color y video, así como también la aplicación de cuantificación vectorial en lugar de escalar. También podría extenderse la aplicación a otros formatos de imágenes.

Algoritmo JPEG Adaptivo.

Los resultados obtenidos de la imagen VLSI.BMP y LENA.BMP se lograron con un particionamiento de 10 y 20 respectivamente para cada uno de los tres limites.

Con el particionamiento de la primer imagen se logró que las zonas de la fórmula fueran tratadas con bloques mas chicos (4x4 y 8x8), mientras la mayor parte de la zona oscura fuera procesada con bloques de 32x32 y en menor cantidad con los de 16x16. Con esta distribución de bloques solo se produjeron distorsiones en los lugares muy cercanos a la fórmula y solo al aplicar factores muy altos. En la siguientes tablas se muestran algunos resultados obtenidos para las imagen.

Si bien se puede aumentar el valor de los factores de pérdida para los bloques de 8x8, 16x16 y 32x32, sin distorsión visible para las zonas donde estos caen, estos incrementos no se ven reflejados en los tamaños de las imágenes comprimidas.

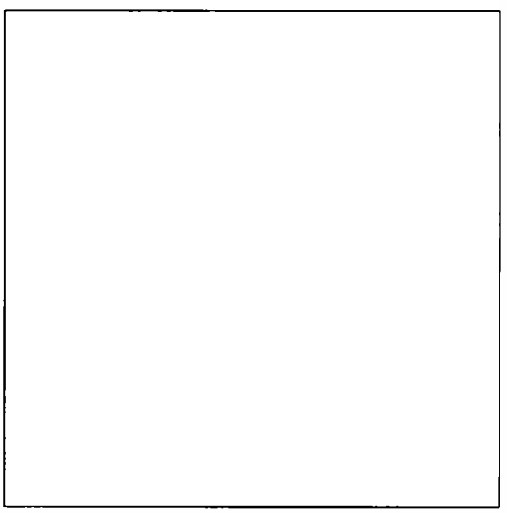
Tabla para VLSI.BMP

Descomprimida	Radio	Radio Factor 4x4	Radio Factor 8x8	Radio Factor 16x16	Radio Factor 32x32
pv1_1.bmp	6.4	2	2	2	2
pv1_2.bmp	9.9	9	2	2	2
pv1_3.bmp	9.98	9	9	2	2
pv1_4.bmp	11	15	9	2	2
pv1_5.bmp	13	20	9	2	2
pv1_6.bmp	14.4	25	9	2	2
pv1_7.bmp	14.5	25	9	9	9

Tabla para LENA.BMP (con valores de 20 para los límites de decisión en el particionamiento)

Descomprimida	Radio	Radio Factor 4x4	Radio Factor 8x8	Radio Factor 16x16	Radio Factor 32x32
plena4_1.bmp	4	2	2	2	2
plena4_2.bmp	8.3	9	2	9	9
plena4_3.bmp	6.4	5	3	5	5
plena4_4.bmp	9	10	3	10	10
plena4_5.bmp	10.8	15	2	15	15
plena4_6.bmp	11.18	15	3	15	15
plena4_7.bmp	11.4	15	4	15	15
plena4_7.bmp	13.16	20	4	15	15
plena4_7.bmp	13.3	20	5	15	15

plena4_1.bmp



pérdida

plena4_5.bmp

pérdida

plena4_9.bmp

pérdida

CONCLUSIONES

Se observó que en imágenes con grandes zonas con poca actividad en niveles de grises permitían radios de compresión mayores si eran procesadas con bloques mas grandes produciendo distorsiones poco visibles en las zonas poco activas, pero con el costo de pérdidas notables de calidad en zonas con poca actividad adyacentes a otras con mucho nivel de detalle. Con la reducción de los tamaños de bloques esas zonas conflictivas se fueron restringiendo pero con la desventaja de la disminución en los radios de compresión. Los excelentes resultados comparativos del esquema adaptivo respecto del fijo, nos han llevado a encarar la paralelización del algoritmo adaptivo, a fin de tener tiempos razonables para procesamiento en tiempo real.

Conclusiones Finales

La compresión wavelet es similar a la JPEG ya que aplica una transformada a la imagen, cuantiza los resultados de la transformada, y aplica compresión sin pérdida a los resultados cuantizados. La mayor diferencia es que en vez de aplicar la transformada a bloques discretos de 8x8 pixels, la transformada wavelet funciona sobre áreas de la imagen que se superponen. Como resultado, la compresión wavelet no produce los artefactos en bloques que se ven a altas relaciones de compresión JPEG; el efecto producido es, en cambio, una suavización general de la imagen.

La compresión wavelet parece ofrecer alrededor de una vez y media a tres veces más compresión que el JPEG para la misma calidad aparente de la imagen. Sin embargo, como siempre, hay una desventaja. La descompresión wavelet actual lleva de una vez y media a tres veces más tiempo de cómputo que la descompresión JPEG, por lo que puede haber muy poca o ninguna diferencia en velocidad entre las dos. Sin embargo, no hay razón para que los sistemas futuros de wavelet no puedan alcanzar la velocidad de descompresión del JPEG.

Existen algoritmos de compresión wavelets que pueden lograr radios de compresión para imágenes fijas de 300 a 1 y para video llega hasta 480 a 1, en MPEG como optimo se puede lograr obtener un radio de 200 a 1 por ello hoy en día se esta poniendo incapie en la comercialización de los algoritmos que utilizan wavelet para aplicaciones de video .

El JPEG está diseñado para comprimir tanto imágenes color como en escala de grises de escenas naturales del mundo real. Funciona bien sobre fotografías, trabajos artísticos naturalistas, y material similar; no tan bien sobre letras, dibujos simples o dibujos de líneas. El JPEG maneja sólo imágenes quietas, pero hay un estándar relacionado llamado MPEG para imágenes en movimiento.

En el estándar del JPEG se incluye un método de compresión sin pérdida, pero virtualmente nadie lo ha adoptado, principalmente porque tiene pocas características que lo hacen merecedor de ser recomendado por encima de otras técnicas. Sin embargo, la compresión JPEG con pérdida ha acumulado un amplio apoyo, y la mayoría de los browsers de Web gráficos ahora la comprenden.

Un archivo JPEG en escala de grises generalmente es alrededor de un 10 a un 25 % más chico que un archivo JPEG full color de calidad visual similar. Pero los datos no comprimidos de una escala de grises son sólo de 8 bits/pixel, o un tercio del tamaño de los datos color, por lo que la razón de compresión es mucho menor. El umbral de pérdida visible es a menudo alrededor de una compresión de 5:1 para una escala de grises.

El umbral exacto en el que los errores se vuelven visibles depende de sus condiciones de vista. Cuanto más pequeño sea un pixel individual, más difícil será ver un error; por lo tanto los errores son más visibles en una pantalla de computadora (a unos 70 puntos por pulgada) que en una impresora de alta calidad (300 o más puntos por pulgada). De esta manera, una imagen de resolución más alta puede tolerar más compresión... lo que es afortunado considerando que es mucho más grande desde el comienzo. Las relaciones de compresión citadas más arriba son típicas para una visualización por pantalla. También debe notarse que el umbral de error visible varía considerablemente de una imagen a otra.

Con la compresión JPEG con pérdida hay un "*trade-off*" o trueque entre el tamaño del archivo y la calidad de la imagen. Se pueden lograr razones de compresión muy altas - del orden de 100:1 o más - pero a expensas de artefactos bastante obvios con forma de bloque en la imagen. A razones de compresión más bajas la degradación de la imagen es mucho menos notable, y la mayoría de los observadores están de acuerdo en que las razones de alrededor 10:1 producen un resultado que es visualmente indistinguible del original.

La mayoría de los compresores JPEG le permiten seleccionar el nivel de trueque de tamaño del archivo versus calidad de la imagen seleccionando un seteo de la calidad. Parece haber una confusión generalizada acerca del significado de estos settings. "*Calidad 95*" NO significa "*mantener el 95% de la información*", como algunos han dicho. La escala de calidad es exclusivamente arbitraria; no es un porcentaje ni nada por el estilo.

Sería interesante si, habiendo comprimido una imagen con JPEG, se pudiera descomprimir, manipular (cortar un borde, por ejemplo), y recomprimirla sin otra degradación de la imagen más que lo que se perdió inicialmente. Desafortunadamente *este no es el caso*. En general, la recompresión de una imagen alterada pierde más información. Por lo tanto, es importante minimizar la cantidad de generaciones JPEG entre la versión inicial y la versión final de una imagen.

Si se descomprime y recomprime una imagen con el mismo seteo de calidad que se usó primero, ocurre muy poca o nula degradación adicional. Esto significa que se le pueden hacer modificaciones locales a una imagen JPEG sin una degradación material de las otras áreas de la imagen. (Sin embargo, las áreas que cambie sí se degradarán). Contrariamente a lo que indicaría la intuición, esto funciona mejor cuanto más bajo sea el seteo de calidad.

El JPEG es un formato útil para almacenaje de archivos y transmisión de imágenes, pero no se usa como un formato intermedio para secuencias de pasos de manipulación de imágenes. Para trabajar con la imagen use un formato de 24 bits sin pérdida (PPM, PNG, TIFF, etc.), luego el JPEG cuando esté listo para archivarla o enviarla a la red. Además de evitar la degradación, de esta manera se ahorrará mucho tiempo de compresión / descompresión.

El JPEG comprime muy bien cuando se trabaja con el tipo de imagen adecuado (fotografías e imágenes por el estilo). Para imágenes full color, los datos no comprimidos son normalmente de 24 bits/pixel. Los métodos de compresión sin pérdida mejor conocidos pueden comprimir este tipo de datos 2:1 en promedio. El JPEG puede lograr

compresiones de 10:1 a 20:1 sin pérdida visible, reduciendo el requerimiento de almacenaje efectivo a 1 o 2 bits/pixel. Es posible una compresión de 30:1 a 50:1 con defectos pequeños a moderados, mientras que para propósitos de muy baja calidad como presentaciones preliminares o índices de archivos, es posible una compresión de 100:1. Una imagen comprimida 100:1 con JPEG consume el mismo espacio que una imagen full color de una uña de pulgar en escala uno a diez, y aun así retiene mucho más detalle que esa uña.

Para comparación, una versión GIF de la misma imagen comenzaría sacrificando la mayor parte de la información de color para reducir la imagen a 256 colores (8 bits/pixel). Esto proporciona una compresión de 3:1. El GIF tiene una compresión "*LZW*" adicional incorporada, pero el LZW no funciona muy bien en datos fotográficos típicos, a lo sumo se puede obtener una compresión general de 5:1, y no es para nada poco común que con el LZW haya una pérdida neta (o sea, menos que una compresión general de 3:1). El LZW sí funciona bien en imágenes más simples como dibujos de líneas, que es la razón por la que el GIF maneja tan bien ese tipo de imágenes. Cuando se hace un archivo JPEG a partir de datos fotográficos full color, usando el seteo de calidad lo suficientemente alto como para evitar una pérdida visible, el JPEG típicamente será un factor de 4 o 5 más chico que el archivo GIF hecho a partir de los mismos datos.

Trabajos futuros

Trabajos Futuros

El estándar JPEG no especifica el comportamiento exacto de los compresores y descompresores, por lo que hay lugar para una implementación creativa. En particular, las implementaciones pueden negociar velocidad contra calidad de la imagen aproximaciones a la DCT más precisas o más rápidas pero menos precisas. Existen negociaciones similares para el downsampling / upsampling y para los pasos de conversión del espacio de color que no fue un tema tratado en esta tesis.

Un punto de investigación es la codificación de imágenes con wavelet guiada por quadtree. Realizar un híbrido entre wavelet y una codificación de imágenes quadtree tratando de mejorar la relación velocidad/fidelidad.

Es importante investigar la paralelización del método adaptivo propuesto, tratando de optimizar no sólo la compresión de datos sino además el tiempo computacional requerido.

Bibliografía

[Ajenjo, 1993] *"Tratamiento Digital de Imágenes"*, Alberto d. Ajenjo. ANAYA. 1993.

[Chui, 1992] *"An introduction to wavelet"*, Charles Chui. Academis Press 1992.

[Clarke, 1995] *"Digital Compression of Still Images and Video"*, Roger Clarke, Academic Press, 1995.

[Cody, 1992] *"The Fast Wavelet Transform"*, Mac. A. Cody, Dr. Dobb's. 1992.

[Cody, 1994] *"The Wavelet Packet Transform"*, Mac A. Cody, Dr. Dobb's. 1994.

[Dougherty, 1987] *"Matrix Structured Image Processing"*, E. Dougherty, C. Giardina, Prentice-Hall Inc., 1987.

[Edwards, 1991] *"Discrete Wavelet Transforms: Theory and Implementation"*, Tim Edwards, Stanford University, 1991.

[Feiner] *"Computer Graphics"*, Foley, van Dam, Feiner, Huges, Addison-Wesley,1990.

[Gersho] *"Vector Quantization and Signal Compression"*, Allen Gersho, Robert M. Gray.

[Glassner, 1996, VOL 1] *"Principles of Digital Image Synthesis"*, Vol. 1, Andrew Glassner, Morgan Kaufmann Publishers, 1996.

[Glassner, 1996, VOL 2] *"Principles of Digital Image Synthesis"*, Vol. 2, Andrew Glassner, Morgan Kaufmann Publishers, 1996.

[Gonzales, 1992] *"Digital Image Processing"*, R. Gonzales, R. Woods, Addison-Wesley, 1992.

[Hamming] *"Coding and Information Theory"*, Richard W. Hamming

[Held, 1994] *"Data Compression. Techniques and Applications. Hardware and Software Considerations", Gilbert Held, Wiley, Third Edition, 1994.*

[Heller] *"Linear-Phase M-Band Wavelets with Application to Image Coding"*, Peter M. Heller, Truong Q. Nguyen, Hemant Singh, W. Knox Carey.

[IEEE, 1995, 1] *"Wavelet for computer graphics: A primer, part 1"*, IEEE Computer Graphics and Application. May 1995.

[IEEE, 1995, 2] *"Wavelet for computer graphics: A primer, part 2"*, IEEE Computer Graphics and Application. May 1995.

[Jain] *"Fundamentals of Digital Image Processing"*, Anil K. Jain

[Jayant] *"Digital Coding of Waveforms: Principles and Applications to Speech and Video"*, N.S. Jayant.

[Kernighan] *"El Lenguaje de Programación C"*, Brian W. Kernighan - Dennis M. Ritchie

[Lindley] *"JPEG-Like Image Compression"*, Craig A. Lindley, Dr. Dobb's. 1995.

[Mallat] *"A Theory for Multiresolution Signal Descomposition. The Wavelet Representation"*, Stephan G. Mallat.

[Nelson, 1991] *"The Data Compresión Book"*, Mark Nelson, Prentice Hall, 1991.

[Odegard] *"Design of Linear Phase Cosine Modulated Filter Bank for Subband Image Compression"*, J.E. Odegard, R.a. Gophinat, C.S. Burrus.

[Robinson] *"Binary Tree Predictive Coding"*, John A. Robinson, University of Waterloo.

[Safar, 1991] *"Representación adaptiva de imágenes en bases de datos ortogonales óptimas, y aplicaciones a compresión"*, Félix Safar, Fabián Blasetti, Juan Pablo Villa, Daniel Cocimano, Sergio Katz. 1991.

[Sayood, 1996] *"Introduction to Data Compression"*, Khalid Sayood, Morgan Kaufmann Publishers, 1996.

[Shaddock, 1992] *"Creaciones Multimedia"*, Philip Shaddock, 1992, Anaya.

[SIGGRAPH, 1995] *"Wavelets and their Application in Computer Graphics"*, SIGGRAPH '94 Course Notes.

[Starck] *"Image Restoration with Noise Suppression Using the Wavelet Transform"*, Jean-Luc Starck. 1994.

[Sweldens] *"Wavelet Sampling Tecniques"*, Wim Sweldens, Robert Piessens.

[Ueda, 1995] *"Wavelet: An introduction and examples"*, Masami Ueda and Suresh lodha. 1995.

[Villa] Trabajo de Grado: *"Un algoritmo Paralelo Adaptativo de Compresión de Imágenes en tonos de gris"*, Juan Pablo Villa.

[Villaseñor, 1995] *"Wavelet Filter Evaluation for Image Compression"*, John D. Villaseñor, Benjamin Belzer, Judy Liao. 1995.

[Weels, 1994] *"Recent Advances in Wavelet Technology"*, R.O. Weels. 1994.