Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

1997

La compresión de impresiones dactilares mediante wavelets.

Tesis de Licenciatura.

Autor: ***Gonzalo A. Ramos***

Directores: ***Dr. Jorge Sanz, Lic. Eduardo Rodríguez.***

Resumen:

El problema de la compresión de imágenes está motivado por los grandes espacios que las imágenes ocupan en los medios de almacenamiento de hoy. Reducir los recursos ocupados por las imágenes (almacenamiento, ancho de banda en canales de transmisión, etc.) se vuelve de gran importancia, gracias al auge de aplicaciones multimedia, y la necesidad de transmisiones de datos en Internet.

Dentro de las imágenes en general, las impresiones dactilares (huellas digitales) juegan un papel de gran importancia en sistemas de identificación y documentación de personas. Dichos sistemas son utilizados en forma intensiva por organizaciones tanto gubernamentales como privadas.

Esta tesis aborda el tema de la compresión de las imágenes de huellas digitales mediante distintas técnicas. Una implementación del estándar de compresión WSQ del FBI [BRADLEY/93] (basado en wavelets) es realizada y su desempeño es comparado al método de compresión de imágenes de propósito general más usado en la actualidad, JPEG [MITCHEL/93]. Esta comparación es llevada a cabo sobre una base de datos con 1251 imágenes de huellas digitales [cd-rom2], capturadas a 500dpi y 8bpp. Para hacer justa la comparación (dado que se están confrontando un método especializado vs. uno de propósito general) se especializó JPEG a este tipo de imágenes [cd-rom-2], alterando su tabla de cuantización.

Los resultados obtenidos con tal cantidad de muestras establecen que para comprimir imágenes de huellas digitales, el método de compresión WSQ es muy superior en performance a JPEG.

Abstract:

The problem of image compression is motivated mainly by the great amount of storage space that they use. With the boom of multimedia applications, and the need for speed in data transmission over the Internet, means of reducing the resources needed by images (storage, bandwidth on communication channels, etc.) become an important issue.

Within images types, fingerprints are of great importance in person verification and documentation systems. Such systems are used intensively by both private and government organizations.

This thesis deals with the subject of the compression of fingerprint images with different techniques. An implementation of the FBI fingerprint compression standard, WSQ [BRADLEY/93] (based on wavelets) is made and its performance is compared with the general purpose image compression method more used today, JPEG [MITCHEL/93]. This comparison is run on a database of 1251 fingerprint images (scanned at 500dpi, 8bpp) [cd-rom2]. To make this a fair comparison (and because we are confronting a specialized method vs. a general purpose one), JPEG was adjusted to this particular set of images [cd-rom2], by altering its quantization table.

The result obtained with such a great number of samples show that for the task of compressing fingerprint images the WSQ method is superior that JPEG.

Indice.

Introducción.

Motivación

La compresión de imágenes surge como respuesta al problema causado por la gran cantidad de recursos que las imágenes pueden ocupar, ya sea en espacio de almacenamiento o canales de transmisión. Las imágenes médicas (tomografías, resonancia magnética nuclear, radiografías, etc.), geológicas, de satélite, mapas, películas, etc. son ejemplos típicos de imágenes que deben ser almacenadas en gran número, y que originan volúmenes de información muy grandes. La transmisión de fax, televisión, e Internet son situaciones donde las imágenes ocupan los recursos de un canal de transmisión de información. Las limitaciones físicas (y/o económicas) de dichos recursos, son compensadas mediante técnicas tales como la compresión de datos e imágenes.

Las técnicas de compresión de imágenes pueden separarse en dos grandes grupos: Aquellas denominadas de propósito general y aquellas especializadas para cierto tipo de imagen. Las técnicas de compresión de imágenes de propósito general tienen como objetivo el comprimir cualquier clase de imagen, que puede provenir de cualquier fuente (por lo general del mundo real, en contraste con la imágenes generadas artificialmente). Es obvio que las imágenes a las cuales se les aplican estos métodos pueden ser muy diferentes, como por ejemplo la imagen que es esta página y una fotografía, como la Figura 3 -9. Es razonable que el desempeño de estas técnicas no sea uniforme, es decir aplicadas a una imagen especifica, la técnicas especializada en esa imagen comprimirá en mayor medida que la de propósito general.

Este hecho impulsó el desarrollo de métodos especializados de compresión, que se aprovechan de las propiedades particulares de cierta familia de imágenes. Por ejemplo, la imagen de esta página solo posee 2 colores (la tinta y el papel), sin duda la proporción de blanco (papel) es mayor a la de negro (tinta) y además existen zonas enteramente blancas (los espacios entre renglones, etc.), entre las propiedades más evidentes. Un método que aprovecha ésto es la recomendación T.4 de la CCITT para la codificación de faxes del Grupo 3 [KAY/95].

Dado que este es un tema abierto (nuevos formatos y métodos de compresión son desarrollados constantemente), es importante el poder decidir cuando un nuevo método de compresión de imágenes debe reemplazar a otro existente y funcional. De la misma forma que no se recodifica toda una aplicación o sistema que ha estado funcionando durante años, cada vez que un nuevo paradigma de programación o un lenguaje nuevo es inventado, el cambio es justificable cuando los potenciales beneficios superan con creces a los problemas de transición.

Para esta tarea de decisión deben efectuarse mediciones, en la forma de prueba experimental o teórica, que muestren con claridad las diferencias de desempeño entre dos métodos que se enfrentan.

Contenido

La primera parte de este documento es una introducción a los diversos temas de los cuales está compuesta la compresión de imágenes. Se dan definiciones y preliminares matemáticos elementales sobre teoría de la información y procesamiento de señales, que permitirán la comprensión de los temas vistos en las secciones posteriores. El lector que esté avezado en estos temas puede pasar directamente a la segunda parte.

La segunda parte presenta un panorama del estado del arte en materia de compresión de imágenes en general. En ella se describen diversos métodos que representan los distintos enfoques con los que se ha atacado al problema de la compresión de imágenes a lo largo del tiempo, hasta nuestros días. Se analiza para cada uno sus características generales, ventajas, desventajas, su alcance y situación en el presente.

La tercera parte presenta informalmente a las huellas digitales, discute su uso y el sentido de que sean comprimidas. También se analiza cuál de los métodos vistos anteriormente es aplicable para obtener una compresión eficiente.

La cuarta parte de la tesis trata el tema de adaptar el método de compresión JPEG [MITCHEL/93] [WALLACE/91][Ijg] para comprimir huellas digitales más eficientemente que lo normal. Se hace esto con el objetivo de comparar en condiciones equitativas a uno de los métodos de compresión más populares (JPEG), con un estándar emergente, WSQ.

La quinta parte describe la implementación del estándar WSQ que se realizó. Se enumeran sus distintos componentes y las elecciones y diferencias de implementación en cada etapa. Finalmente se compara esta implementación con la implementación original.

En la sexta parte se exhiben los resultados de la comparación entre JPEG y WQS, utilizando como muestras las más de mil imágenes del [cd-rom2]. Se presentan entonces, conclusiones acerca de los resultados obtenidos.

Nota: En los 2 CD-ROM que acompañan a este documento se encuentra una versión electrónica del mismo así como todos los resultados experimentales e imágenes de los cuales se hace referencia explícita.

Preliminares

Información

Sea un suceso E que puede ocurrir con probabilidad P(E). Cuando E ocurre se dice que se han recibido $I(E)$ unidades de información. Donde

$$I(E) = \log \frac{1}{P(E)}$$

Fórmula 2-1: Definición de información de un suceso.

Esta definición otorga mayor contenido de información a los sucesos poco probables y menor información a los más probables. Por ejemplo: el evento 'el sol sale por el este' es un suceso altamente probable que da poca información, luego poca atención se le presta a su fuente. Si en cambio se recibe la información 'el sol ha salido por el oeste', ciertamente es un suceso con alto contenido de información (y algo alarmante, ya que los polos magnéticos se volvieron locos o la tierra gira en sentido contrario, entre las más 'tranquilizadoras' hipótesis...), a la cual ciertamente se debe prestar algo más de atención. También esta definición cumple la razonable propiedad que la cantidad de información de dos sucesos independientes es la suma de la información de ambos sucesos.

La base del logaritmo involucrada es un factor que determina la escala de las unidades con las que se trabajaran. En nuestro caso se usara la base 2, y la unidad de información de denominara *bit* (binary *unit*. Esto es distinto del *bit* usual de la base de numeración 2, *binary digit.* Donde el contexto no establezca con claridad a cual uno se refiere, se aclarará pertinentemente)

Se define a una fuente de información discreta *S,* como a una entidad que emite una secuencia de símbolos s_i pertenecientes a un alfabeto $\Sigma=\{s_0, s_1, ..., s_{n-1}\}$. Si estos símbolos son estadísticamente independientes (la aparición de uno de ellos no depende de las emisiones anteriores), se dice que la fuente es de memoria nula.

Si se denomina a la probabilidad de cada uno de estos símbolos $p_i=P(s_i)$, se define a la información media por símbolo o Entropía de la fuente a

$$H(S) = \sum_{i=0}^{n-1} p_i . \log_2 (\frac{1}{p_i}) \qquad \text{bits.}$$

Fórmula 2-2: definición de entropía de una fuente (de memoria nula).

Se demuestra en [ABRAMSON/86] que este valor es cota inferior de la longitud (en símbolos) media de cualquier codificación (que no pierda información) que ese construya para dicha fuente.

Es común ver modelada una imagen digitalizada como una fuente de información de memoria nula (lo cual es evidentemente inexacto, pues en general el valor de un pixel depende de sus vecinos). El alfabeto de esta fuente tiene entonces tantos símbolos como niveles de gris tenga la imagen en cuestión. Las probabilidades asociadas son estimadas por la frecuencia relativa de dichos símbolos.

Sobre la Profundidad de la imagen y la Tasa de compresión.

Típicamente los valores de los elementos de una imagen en tonos de gris son codificados mediante 8 bits, o sea 256 valores de gris para cada pixel (*picture element*). Dicha imagen se dice entonces que tiene 8 bits por pixel (bpp), este valor se denomina profundidad de la imagen.

En esta tesis se define como tasa de compresión al valor:

$$R: \frac{bits\_por\_pixel\_en\_imagen\_original}{bits\_por\_pixel\_en\_imagen\_codificada} : \frac{\#bytes\_en\_imagen\_original}{\#bytes\_en\_imagen\_codificada}$$

Fórmula 2-3: definición de tasa de compresión.

Por ejemplo, una codificación de una imagen a una profundidad de 1 bpp, se dice que ha logrado una compresión de 8 a 1.

Medidas de Error y Calidad

Las técnicas que se discutirán aquí deterioran (o visto de otra manera, introducen ruido) a la imagen codificada (comprimida). Esto significa la imagen reconstruida a través de la codificación utilizada difiere de la original. Estas diferencias se denominan ***anormalidades*** o ***artefactos***.

Figura 2-1: artefactos producidos por la codificación JPEG. La imagen de la izquierda es la original, mientras que la de la derecha exhibe las alteraciones (en la forma de bloques cuadrados) producidas por una compresión muy grande usando JPEG.

Cuando dos imágenes que retratan lo mismo son distintas, vale la pena preguntarse cuanto. Esto involucra encontrar una magnitud que (bajo cierto criterio) indique que tan parecidas (o que tan distintas) son dos imágenes.

Error Cuadrático Medio

Es uno de los más usados, se define como:

$$MSE = \frac{1}{N}.\sum_{i=1}^{N} E(u_i - \tilde{u}_i)^2 \cong \frac{1}{N}.\sum_{i=1}^{N} (u_i - \tilde{u}_i)^2$$

Fórmula 2-4: definición del error cuadrático medio.

donde *E(x)* es la esperanza matemática, *u* la imagen original (de N puntos) y $\tilde{u}$ la imagen 'degradada'. Generalmente la segunda expresión es un buen estimador del MSE.

Error Cuadrático Medio Normalizado

Definido como:

$$NMSE = 100.\frac{Var(u - \tilde{u})}{Var(u)}\%$$

Fórmula 2-5: definición del error cuadrático medio normalizado.

donde Var(x) es la varianza.

Error Cuadrático Medio Normalizado Invariante a Escalas

Definido como:

$$SINMSE = 100.\frac{Var(u - a.\tilde{u})}{Var(u)}$$ donde *a* minimiza a SINMSE.

Fórmula 2-6: definición del error cuadrático medio normalizado invariante a escalas (uf!)

se ve en [LIM/90] que

$$a = \frac{E(u - \tilde{u}) - E(u).E(\tilde{u})}{Var(\tilde{u})}$$

Fórmula 2-7.

dicha medida tiene la ventaja de ser invariante a variaciones de escala de la imagen (por ejemplo: multiplicar por un escalar).

Relación Señal Ruido.

Indica en cierta medida la proporción de señal original que permanece frente al ruido (distorsión) y se define en nuestro caso como:

$$SNR: 10.\log_{10}\left(\frac{valor\_pico\_a\_pico\_de\_la\_imagen\_original^2}{MSE}\right) \quad \text{dB.}$$

Fórmula 2-8: relación señal ruido.

para imágenes de 8bpp...

$$SNR = 10.\log_{10}\left(\frac{255^2}{MSE}\right) \text{ dB.}$$

Fórmula 2-9: SNR para imágenes de 8bpp.

El Sistema Visual Humano.

Las medidas de calidad presentadas no son por lo general precisas cundo se debe trabajar con imágenes que deben ser evaluadas visualmente. Se han sugerido entonces [JAIN/89][CAPMBELL/68] [PERKINS/89] algunas que intentan congeniar con el sistema visual humano (o un modelo de él). Dichas medidas exploran tanto características globales como locales de la imagen e involucran muchas veces la interacción de sujetos de prueba en lo que se denomina experimentos psico-fisicos [CAPMBELL/68] (donde intenta cuantizarse la percepción subjetiva del sujeto). Dada la complejidad numérica de las mismas y a los fines de una evaluación rápida, se trabaja con el MSE y la SNR.

Cuantización (escalar)

Un cuantizador es una función muchos-a-uno *Q(x)* que hace corresponder muchos valores de entrada en un conjunto bastante menor de valores de salida. Los cuantizadores son funciones 'escalera' caracterizadas por una serie de puntos $\{d_i\}_{i=0..N}$, llamados *puntos de decisión* y otro conjunto $\{r_i\}_{i=0..N-1}$, los *niveles de reconstrucción* (con $r_i \in Z$ generalmente). La función viene dada por:

$$Q(x) = r_i \Leftrightarrow x \in (d_i, d_{i+1}]$$

Fórmula 2-10.

Un cuantizador se dice ***regular*** si, además de incluir alguno de los extremos, $r_i \in (d_i, d_{i+1})$.

Un cuantizador donde $d_{i+1} - d_i \equiv cte.$ se llama cuantizador ***uniforme***, y ese valor se denomina ***paso*** del cuantizador.

Los ***bits*** (o resolución) de un cuantizador son la cantidad de bits necesaria para codificar el rango $\{r_i\}$ (por ejemplo, para el rango 0...255 son necesarios 8 bits). La cantidad de elementos en el rango, para un cuantizador de B bits la cantidad $L = 2^B$, son los ***niveles*** del cuantizador.

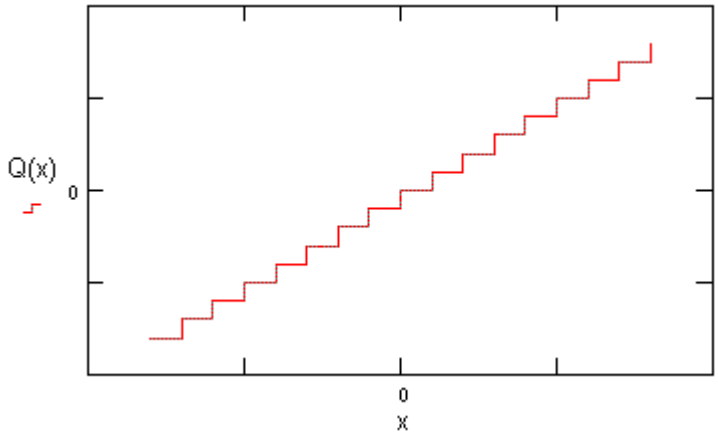


Figura 2-2: Gráfico de un cuantizador uniforme genérico.

Como puede verse, este es un proceso que pierde información, pues dado un valor r_i , no se puede decir de qué valor exacto de *x* provino. Dado que la señal (imagen) recuperada difiere de la original, se dice que el cuantizador ha introducido ruido a la misma. Es en esta etapa, por lo general, donde se produce la 'distorsión' de la imagen en los métodos de compresión con pérdida de información.

Una propiedad interesante es la disminución del valor de la entropía de la imagen o señal cuantizada, lo cual es razonable ya que se *pierde* información, esto sugiere un potencial (y que a todos los efectos ocurre) incremento de la compresión de la misma. Esto hace de la cuantización un método de compresión en sí mismo.

La cuantización de muestras de una señal (o imagen) para la transmisión o almacenamiento digital se llama ***PCM*** (Código por Modulación de Impulsos). Con esta terminología puede decirse que una imagen de 8bpp sea posiblemente el resultado de aplicar a las muestras de una imagen continua [ ], un cuantizador uniforme de 8 bits con cierto paso.

El cuantizador Optimo (Lloyd-Max)

La idea del cuantizador óptimo es la de encontrar un juego de parámetros (que definan un cuantizador) que minimice el error o distorsión (respecto a cierta medida de bondad) introducida al cuantizar una señal o imagen.

Un cuantizador Lloyd-Max minimiza el MSE para un número dado de niveles de cuantización, y una función de densidad para la señal (imagen) encuentra los intervalos $\{d_i\}_{i=0..N}$ correspondientes. Para distribuciones uniformes el cuantizador de Lloyd-Max devuelve un cuantizador uniforme [JAIN/89][GERSHO/92].

Es muy común cuantizar una variable aleatoria cuya distribución no es uniforme con un cuantizador uniforme. El cuantizador uniforme que minimiza el error cuadrático medio con respecto al paso, por ejemplo q, se denomina ***cuantizador uniforme optimo***. Por su simplicidad de diseño e implementación este cuantizador es usado habitualmente en todo tipo de aplicaciones, y a menos que se especifique de otra forma cuando uno se refiera a un cuantizador a lo largo de este trabajo, se estará refiriendo, a un cuantizador uniforme (óptimo).

Los algoritmos usados para el diseño de cuantizadores (como por ejemplo los algoritmos de Lloyd I y II) son de carácter iterativo, es decir, van refinando una solución tentativa inicial hasta que cierta condición es alcanzada.

El problema de la distribución de bits

Supóngase que se tiene un conjunto de k variables aleatorias, $X_1, X_2, ..., X_k$, con funciones de probabilidad conocidas. Supóngase también que los valores de dichas variables deberán ser cuantizados, con un cuantizador Q_i. Si se fija la cantidad de niveles N_i (una resolución de $b_i = log_2 N_i$ bits) del cuantizador, puede diseñarse un cuantizador optimo basado en una medida de distorsión particular $W_i(b_i)$. Esta función dice en cuánto se puede reducir la distorsión incrementando la resolución, a la manera de una relación costo-beneficio, donde el costo es la cantidad de bits asignados al cuantizador, y el beneficio es la calidad de la reconstrucción.

La distorsión general D del sistema puede calcularse como la suma de las distorsiones de cada cuantizador, por lo que la expresión queda:

$$D = D(b) = \sum_{i=1\cdots k} W_i(b_i)$$

Fórmula 2-11.

donde $b=(b_1, b_2,..., b_3)$ es el vector de distribución de bits.

El problema de distribución de bits surge de la necesidad de restringir los valores $b_1,..., b_k$ a una cota superior B (por ejemplo, por restricciones en el canal de transmisión de datos), por lo que se plantea formalmente como:

"Encontrar *b*, tal que se minimiza *D(b), sujeto a la condición que* $\sum_{i=1}^{k} b_i \leq B$ **"**

Existen muchos enfoques para resolver este problema, y éstos dependen de variados factores: si desean soluciones enteras, si se dispone *realmente* de las distribuciones de probabilidad, si se asumen distribuciones normales iguales, o distintas, si se aplicara una solución algorítmica (al estilo de Gradientes Conjugados) o algebraica (al estilo de los Multiplicadores de Lagrange) [LUENBERGER/84].

Lo que es muy claro es que se trata de un problema de optimización no lineal, sin un esquema standard probado de resolución (como el simplex en la programación lineal) [LUENBERGER/84].

Generalmente el problema de la distribución de bits estará presente como una importante etapa en el diseño de las técnicas de compresión que se verán más adelante.

Cuantización Vectorial.

Introducción

La cuantización vectorial (VQ) es una generalización de la cuantización ordinaria (escalar). Esta puede ser vista como una función que transforma a un vector en un conjunto ordenado de números reales. Uno de los principales usos de la VQ es el de la *compresión de datos*, aunque también encuentra aplicaciones en el campo del reconocimiento de patrones, clasificación, y transformaciones lineales.

Un cuantizador vectorial Q de dimensión k y tamaño N puede verse entonces como:

$$Q: R^k \rightarrow C,$$

Fórmula 2-12.

donde el ***código*** o libro de códigos $C = (y_1; y_2; \cdots; y_N)$ e $y_i \in R^k$ con i: 1…N, siendo R^k el espacio euclídeo de dimensión *k*. La resolución, del VQ se define como $r = (\log_2(N))/k$ que indica la cantidad de bits por componente del vector usados para representar al vector original. Esta da una idea de la precisión alcanzable con al VQ si el código está bien diseñado.

Esto define una partición del espacio R^k en *N* regiones o celda R_i, i:1…N.

$$R_i = \{x \in R^k / Q(x) = y_i\}$$

Fórmula 2-13.

Un VQ se llama ***regular*** si cada R_i es un conjunto convexo (la recta que une dos puntos cualesquiera pertenece al conjunto también…) y para cada $i, y_i \in R_i$.

Un VQ puede también descomponerse en dos operaciones: el codificador vectorial (que transforma al vector *x* en valor *i*: 1…N) y el decodificador vectorial (que dado el índice i devuelve el vector y_i)

También se define a un VQ como ***'polytopal'*** al VQ regular cuyas celdas están delimitadas por la superficie de los hiperplanos k-dimensionales.

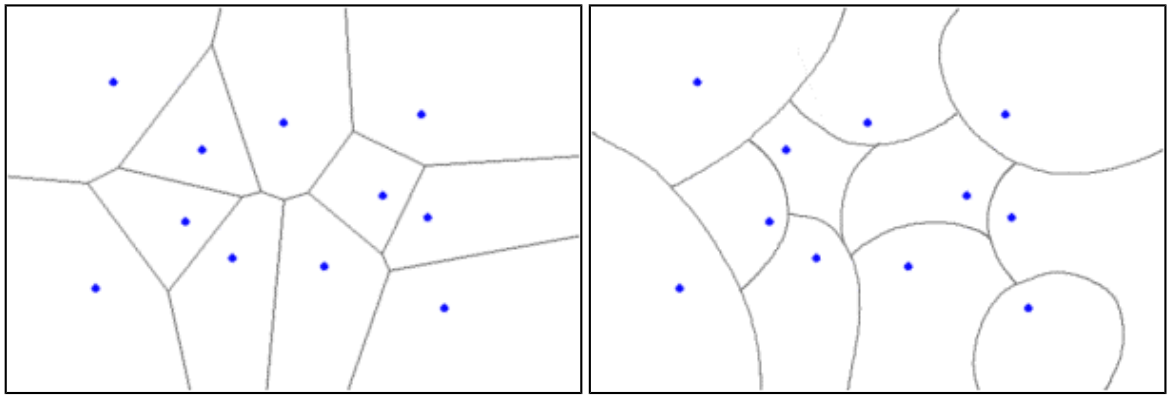


Figura 2-3: Los diagramas representan las regiones R_i de un VQ, el de la izquierda es regular (es mas, un diagrama de Voronoi), el de la derecha no es regular (está hecho a mano!). Los puntos indican a los representantes de cada región.

Nadie lo hace mejor.

La técnica de VQ puede al menos igualar la performance de cualquier sistema de codificación que opere con vectores (de muestras de señales o parámetros). Esto dice que no existe otra técnica de codificación vectorial que lo haga mejor que la VQ. Vale la pena demostrarlo.

"Dado un sistema de codificación (S) que hace corresponder un vector de señales a uno de N palabras binarias y reconstruye el vector aproximado de la palabra binaria, existe un VQ con un código de tamaño N que tiene exactamente la misma performance (o sea por cada vector de entrada produce la misma salida que el sistema de codificación dado"

Prueba: Numere el conjunto de palabras binarias producido por *S* como índices 1,2, …N. Sea el vector y_i la salida asociada a la i-ésima palabra binaria de *S.* Defina entonces el código C como el conjunto ordenado de los vectores y_i . Entonces un decodificador de un VQ tiene idéntica performance al decodificador *S* y un codificador de un VQ puede ser definido para ser idéntico al codificador de *S*. QED.

El truco: la VQ se define en forma tan general que abarca (o modeliza a) cualquier sistema de codificación que uno pueda pensar.

Factores de Diseño e Implementación.

Cuando se requiere diseñar un cuantizador (vectorial) para llevar a cabo determinada tarea, se desea que éste distorsione los datos originales lo menos posible (la diferencia entre una señal original y su versión reconstruida se denomina habitualmente como el *ruido* introducido por el cuantizador). Para esto es necesario introducir (como siempre) una medida de distancia que indica que tan parecidas son dos elementos. En este caso si $d(v,\tilde{v}) \geq 0$ mide el error o distorsión que ocurre al representar al vector *v* como el vector $\tilde{v}$, el codificador óptimo (para un código dado) selecciona y_i dado un *x* si

$$d(x,y_i) \leq d(x,y_j), \forall j \ .$$

Fórmula 2-14.

A este modo de trabajo se lo conoce como el criterio del ***vecino más cercano***. La partición de celdas está entonces completamente definida por el código y por *d().*

$$R_i = \{x \in R^k / d(x,y_i) \leq d(x,y_j), \forall j\} \ (*)$$

Fórmula 2-15.

(*) existen reglas para decidir sobre los vectores equidistantes, aquí se sacrifica para no complicar en notación matemática

A los cuantizadores definidos de esta manera se los llama *de Voronoi* o *de Vecino más Cercano.* El elegir una medida *d* que sea fácil de calcular (como el MSE) permite el uso de algoritmo de diseño de cuantizadores como el algoritmo de Lloyd, que iterativamente mejora un código inicial optimizando el codificador en función del decodificador, y luego el decodificador en función del codificador (criterio de parada mediante).

El desempeño básico de uno de estos VQ se delinea como:

- *Codificación*: dado un vector de entrada *x* encontrar un índice *i que* indique su clase (se busca en forma exhaustiva en el código aquella clase más cercana…)
- *Decodificación*: dado un índice *i,* encuentro el vector representativo *x'*. (una consulta a una tabla por la entrada *i*: Fácil!)

Evidentemente la codificación es la parte más costosa (computacionalmente hablando), que involucraría una búsqueda secuencial por una lista arbitraria de elementos. Es entonces necesario encontrar formas más eficientes (que la búsqueda secuencial) para encontrar al vecino más cercano (o la clase o celda asociada) de un vector dado.

La teoría de la codificación establece que la performance de un VQ es tan cercana al limite teórico optimo como uno quisiera, si uno tiene vectores de dimensión suficientemente grande. Lamentablemente el problema de la codificación crece exponencialmente con la dimensión del vector. Las aproximaciones a la solución de este problema son restringir la estructura del código. Esto resulta en códigos que no son óptimos, pero ofrecen buena performance. Para ejemplificar la idea se delinearan alguno de los…

VQ Estructurados en Arboles

La idea aquí es hacer de la búsqueda de una celda, una búsqueda en un árbol. La codificación entonces consiste en: dado un vector *x*, se realiza un test de este valor frente al elemento en el nodo raíz; de ahí se decide que rama tomar y el proceso se repite hasta llegar a una hoja (índice *i*).

Una implementación de esta idea es la de trabajar sobre VQs *'polytopales'*, donde los niveles de decisión del árbol están definidos como tests de pertenencia al semihiperplano definido por los limites de las celdas. La construcción de dicho árbol no es trivial, pero dadas las propiedades de los VQ polytopales se garantiza la optimalidad con una búsqueda menor a la búsqueda secuencial. Debe notarse que estos árboles pueden estar balanceados o no, con sus ventajas y desventajas.

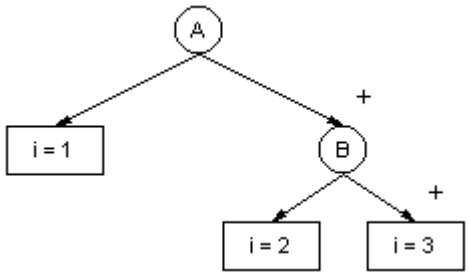


Figura 2-4: Estructura de árbol para una búsqueda eficiente del vecino más cercano. Las decisiones están basadas en si se está en un lado de un hiperplano (+) o no, en este caso hay dos planos: A y B

Otra implementación que sacrifica optimalidad combina para la construcción del árbol métodos de 'clustering' y otros. Aquí cada nodo contiene como información un vector (el test es simplemente el cálculo de la distancia) y el índice asociado viene dado por la cadena binaria que define el camino desde la raíz hasta la hoja. Se comienza entonces con un conjunto de entrenamiento, se calcula su centroide asociando este a la raíz. El 'cluster' es entonces dividido en dos subclases: el código en la raíz es perturbado (varias alternativas aquí), luego se aplica el algoritmo de clustering sobre el par de palabras clave. Una vez que el conjunto de entrenamiento se estabiliza en los dos subclusters, sus centroides son calculados, y éstos representan las palabras clave del segundo nivel del árbol. Dependiendo si se desea un árbol balanceado o no dependen los pasos a seguir.

El algoritmo converge cuando se llega a una longitud media de índice dado. Luego por lo general el árbol es podado eligiendo aquellos nodos que minimicen el incremento de la distorsión.

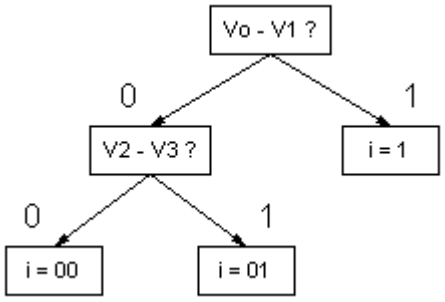


Figura 2-5: Ejemplo muy sencillo de árbol de codificación basada en métodos de clustering. V_i son los centroides de los distintos niveles de decisión.

Otras VQ

Existen otros métodos de VQ que intentan optimizar la performance de diversas formas, se mencionan por ejemplo: *VQ de Grilla (Lattice)*, *VQ predictiva, VQ de Estados Finitos y VQ Adaptiva.[GERSHO/92]*

Codificadores de Entropía

En general un codificador es una función $C:\Xi \rightarrow \Theta^+$, que hace corresponder a símbolos de un alfabeto Ξ, secuencias de símbolos de otro alfabeto Θ. Por ejemplo si

$$\Xi=\{0,1,2,3\} \qquad \text{y} \qquad \Theta=\{0,1\}$$

0 _____ 00
1 _____ 01
2 _____ 10
3 _____ 11

Figura 2-6: definición de un código.

Existen propiedades deseables de los códigos como el ser no singular, unívoco, instantáneo, etc. [ABRAMSON/86].

Código Huffman

La codificación Huffman construye códigos (instantáneos, unívocos, etc.) [ABRAMSON/86] cuyas palabras código tienen distinta longitud. Asigna a los símbolos más probables las palabras más cortas y a los menos probables las más largas. Se consigue así una longitud media del código sea menor que respecto a una codificación con palabras de longitud constante.

Al codificar un alfabeto se obtiene un árbol en el cual las hojas son los símbolos del alfabeto y la altura respecto a la raíz a la que se encuentran es la longitud de la palabra código asociada. La aridad del árbol está determinada por la cantidad de símbolos del alfabeto codificador, para {0,1} se obtiene un árbol binario. La decodificación, dado un árbol, es directa. Partiendo desde la raíz se reciben símbolos que determinan por que rama se debe seguir. Cuando se llega a una hoja, ese es el símbolo. Resultado: decodificación instantánea!.

Símbolo	probabilidad	palabra
A	0.5	0
B	0.25	11
C	0.25	10

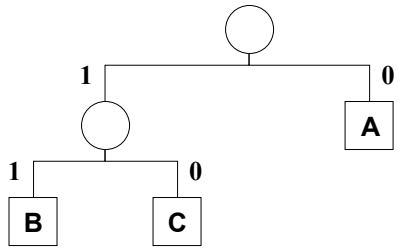


Figura 2-7: Tabla y árbol de Huffman asociado.

Cuando quiere codificarse una imagen binaria, por más probable que sea un símbolo -por ej. 1-, Huffman siempre le asigna una palabra de longitud uno. Esto significa que no hay compresión alguna. Una solución al problema es empaquetar grupos de bits y aplicar Huffman, otra es usar un esquema del tipo RLE (Codificación de Longitud de Corridas) que codifica la longitud de los tramos que consistan de símbolos iguales. Por ejemplo en una secuencia binaria donde predomine el valor cero (como ser el texto en un fax) se gana espacio representando a una serie consecutiva de ceros por un símbolo que represente la cantidad (longitud) de ceros.

Codificación Aritmética.

La codificación aritmética [RISSANEN/79] sigue una estrategia diferente a la de Huffman. En este caso la salida del codificador es un único número entre cero y uno. La codificación es como sigue: A cada símbolo se le asigna un intervalo en (0,1) proporcional a su probabilidad. Se ve el símbolo de entrada, al cual le corresponde cierto intervalo. Se asignan en este nuevo intervalo los subintervalos correspondientes a los símbolos del alfabeto, de acuerdo a su probabilidad. Definidos los nuevos intervalos, se toma el próximo símbolo de entrada y se repite el proceso. El número resultante es la cota inferior del ultimo intervalo obtenido.

Para la secuencia ABBA ...

Letra \ Intervalo	A	B	B	A
A	***[0 ; 0.5)***	[0 ; 0.25)	[0.25 ; 0.3125)	***[0.3125 ; 0.328125)***
B	[0.5 ; 0.75)	***[0.25 ; 0.375)***	***[0.3125 ; 0.34375)***	[0.328125 ; 0.3359375)
C	[0.75; 1)	[0.375 ; 0.5)	[0.34375 ; 0.375)	[0.33559375 ; 0.34375)

el código es 0.3125

Tabla 2-1: Codificación aritmética de cierta secuencia.

La decodificación sigue exactamente el camino inverso. El problema de decidir cuando se termina de decodificar puede solucionarse introduciendo un símbolo especial de fin de transmisión.

El truco es ahora encontrar una representación que exprese con la suficiente precisión y en forma compacta este número. Esto es posible y los rendimientos alcanzados son iguales sino mejores que Huffman.

A la hora de decidir que usar vale la pena recordar que hay una patente [uspt-4122440] sobre el método de codificación aritmética, lo cual restringe su uso a quien pague las regalías correspondientes.

Nota Sobre el Orden.

Los métodos de Huffman y codificación aritmética discutidos previamente aquí, consideran la secuencia de símbolos a codificar, como una fuente de orden cero (esto quiere decir que se asume que el símbolo recién emitido no depende en lo absoluto de los anteriormente recibidos), o de memoria nula, lo cual no es por lo general el caso. El trabajar asumiendo ordenes mayores lleva a mejoras resultados en lo que a compresión se refiere, pero la complejidad de las fórmulas y algoritmos crece en igual proporción.

Huffman Adaptivo.

Un pequeño inconveniente de la codificación presentada arriba [HUFFMAN/52], es que el decodificador debe conocer la tabla de Huffman para decodificar los datos. Dado que para la mayoría de los casos esto implica transmitir junto con los datos aproximadamente 256 bytes, las consecuencias de esto son insignificantes.

Pero si se desea mejorar la habilidad del codificador para comprimir, es necesario que este tenga mejores estadísticas sobre los datos que se desean transmitir. Para esto es necesario cambiar de un modelo de orden 0 (memoria nula) a uno de mayor orden, 1 por ejemplo. Esto significa que estaremos recolectando estadísticas sobre la aparición de un símbolo, dado que lo ha(n) precedido otro(s). Para describir las estadísticas de un modelo de orden 1 son necesarias 257 tablas, en vez de una. Para modelos de mayor orden el incremento es exponencial!. A menos que la cantidad de datos a codificar sea muy grande (realmente grande!), la sobrecarga de información hace al esquema impráctico. El problema se resume entonces a: Para comprimir mejor, se necesitan adquirir mejores estadísticas sobre los datos. Cuando se tienen mejores estadísticas se comprime mejor, pero las mejoras no sirven pues hay que transmitir demasiadas tablas.

La solución a esto es la ***codificación Huffman adaptiva***. Esta permite modelar mejor las estadísticas de los datos, sin tener que preocuparse por la transmisión de tablas. Esto se logra ajustando el árbol de codificación de Huffman dinámicamente, basándose en los datos vistos previamente y sin conocimiento alguno sobre apariciones futuras de los mismos.

El método puede esquematizare de la siguiente manera:

Codificador:

```
.inicialización
.mientras existan datos a codificar
      tomar dato
      codificar y emitir
      ajustar árbol
.fin
```

Decodificador:

```
.inicialización
.repetir
      tomar código
      decodificar y emitir dato
      ajustar árbol
.hasta que dato = fin de transmisión
```

Listado 2-1: Pseudocódigo del esquema Huffman Adaptivo.

La tarea clave que comparten ambos procesos es **`ajustar árbol.`** Definitivamente reconstruir el árbol cada vez que se recibe un símbolo nuevo no es optimo, hacer esto desembocaría en la codificación/decodificación más lenta de la historia. Existe una propiedad en los árboles Huffman [NELSON/92] llamada la ***propiedad de los hermanos***, gracias a la cual es posible modificar un árbol existente para tomar en consideración el cambio en la estadística (frecuencia relativa de aparición) de un símbolo.

Un árbol binario cumple la propiedad de los hermanos si sus nodos pueden ordenarse en forma creciente respecto de su peso (cantidad de veces que un símbolo aparece, en nuestro caso) y cada nodo aparece adyacente a su hermano (nodo que comparte al mismo padre) en dicha lista. Un árbol binario es un árbol que corresponde a un código Huffman si y solo si cumple la propiedad de los hermanos [NELSON/92] (esto se ve constructivamente). Luego, ajustar árbol se ocupa de preservar esta propiedad, reordenando los nodos que violen la propiedad de los hermanos.

Para ejemplificar, asúmase que se tienen tres símbolos A, B, C donde cada uno ha aparecido 2, 1 y 1 veces. Esto es, como en la Figura 2 -7. Sin una tabla asociada y con pesos asociados a los nodos esto se vería así:

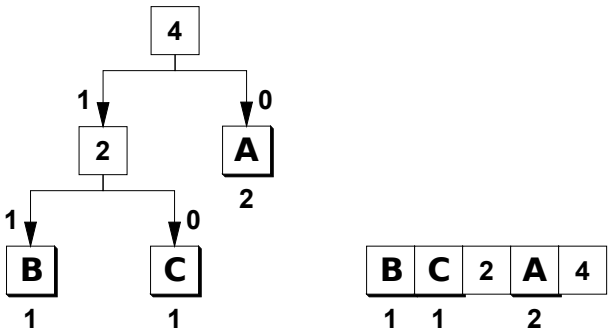


Figura 2-8: Arbol y lista ordenada con pesos en los nodos.

Si ahora se desea codificar una B, se observa el árbol y se emite el código (11). Luego se ajusta el árbol de la siguiente forma: se incrementa en 1 el peso de B, se verifica se sigue verificando la condición de los hermanos, como no es así, se modifica el árbol. Se explora la lista de nodos hasta el ultimo nodo con peso igual al de B menos 1 (en este caso C), y se los intercambia. Luego se propaga el cambio de peso hacia el padre del nodo y el proceso se repite hasta llegar a la raíz.

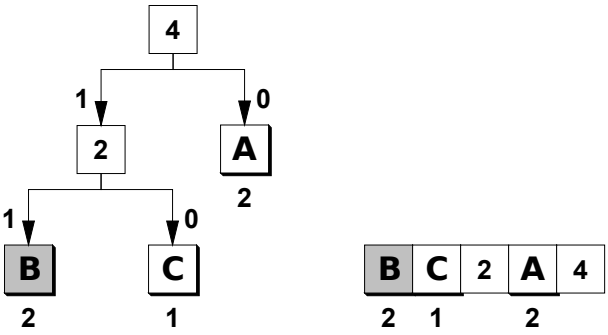


Figura 2-9: Arbol y lista. Luego de codificarse, B no cumple la condición de los hermanos. Viendo la lista, se lo debe intercambiar con C y propagar el peso.

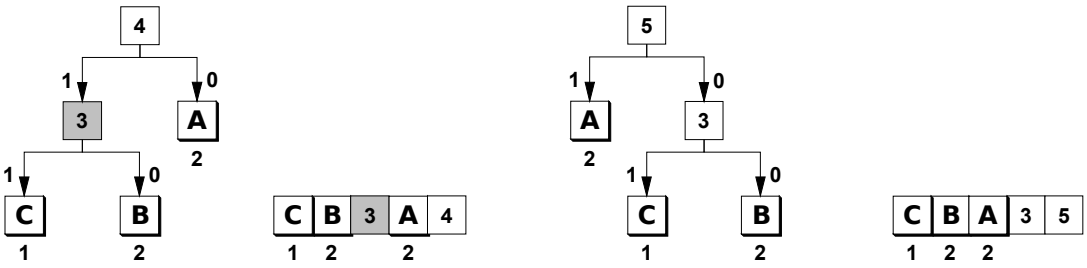


Figura 2-10: Luego de propagar el peso, se debe hacer un intercambio más para arribar a un nuevo árbol con nuevos códigos

De seguir incrementando el peso de B, este subiría de jerarquía en el árbol.

Existen contratiempos en la implementación del método, como por ejemplo: dada la suficiente cantidad de apariciones de símbolos se sobrepasa la capacidad del número entero que almacena el peso. Para estos inconvenientes hay distintas soluciones. Una de ellas bastante simple es reinicializar el proceso a partir de cierto instante, esto se justifica en el hecho de que la estadística de los datos es similar cerca del presente, o dicho de otra manera, la estadística de los datos suele ser distinta que la del pasado distante.

La performance en los niveles de compresión alcanzados por este método, es levemente inferior, sino igual, a la obtenida a través de Huffman tradicional. El único punto de controversia que puede inclinar a usar un método en vez de otro, es el tiempo de sobrecarga en el procesamiento y reestructuración del árbol de codificación.

Códigos basados en diccionarios.

Siendo métodos de compresión en si mismos (nacidos con la intención de reducir el tamaño de algún "texto"), los códigos basados en diccionarios, no son utilizados en forma directa en la compresión de imágenes, sino que se utilizan en forma intermedia como los codificadores de entropía vistos recién [][]. Por eso se discute este tema aquí y no en el apartado principal de métodos de compresión. Hacia el final de esta sección se nombran algunos métodos de compresión de imágenes [Png] que los utilizan.

A diferencia de los métodos vistos hasta ahora, que representan a un símbolo de un alfabeto como una palabra binaria cuya longitud promedio es menor a este (logrando así la compresión), los métodos basados en diccionarios se basan en otro principio: éstos codifican secuencias de símbolos (de longitud variable) de un alfabeto como un símbolo (token). Si el tamaño del símbolo es menor al de la secuencia que representan, se consigue la compresión.

Un ejemplo muy sencillo de este esquema es el codificar un texto escrito como índices a cierto diccionario de la lengua, aquí cada palabra estará representada como un token <no. página, no. palabra>.

Utilizar este ejemplo para codificar un texto en un idioma distinto no seria muy útil, lo cual hace pensar con razón en las desventajas (y ventajas!) de un diccionario ***estático.*** Un método de codificación ***adaptivo*** comienza la codificación con un diccionario base (o hasta vacío) y lo va actualizando a medida que encuentra nuevas frases que transformar en símbolos (tokens).

La popularidad de estos métodos se debe a la difusión de los trabajos de Jacob e Abraham Lempel, que desde la publicación de sus trabajos [LEMPEL-ZIV/77][LEMPEL-ZIV/78] originaron toda una avalancha de algoritmos, y programas (para nombrar algunos LZ77, LZ78, PkZip , ARC , ARJ , Lharc, etc.). Uno de los más famosos métodos es el LZW, una variante del LZ78 realizada por Terry Welch [WELCH/84], que además ostenta una de las más famosas patentes [uspt-4558302][uspt-4464650] otorgadas a métodos de compresión de datos. más de esto después…

Compresión por Ventana Deslizante.

En estos métodos el diccionario está formado por frases de longitud fija que se encuentran en una ventana de tamaño fijo definida en el texto previamente analizado. El tamaño de esta ventana está generalmente entre 2Kbytes y 16Kbytes y la longitud de frase máxima entre los 16 a 64 bytes. El *'primero'* de estos algoritmos fue LZ77 [LEMPEL-ZIV/77], para ejemplificar se verá como funciona.

La ventana de LZ77 consta de dos partes: la primera es un bloque grande de texto visto y procesado previamente (diccionario), y la segunda un bloque mucho más pequeño con texto leído pero no procesado (buffer). La codificación consiste en reemplazar frases en el texto de entrada por punteros a frases en el diccionario, esto se hace tratando de encontrar ocurrencias de texto en el buffer en el diccionario. Cada puntero (o token) consta de dos parámetros: desplazamiento (hacia atrás) hasta el comienzo de la frase, y el largo de la frase.

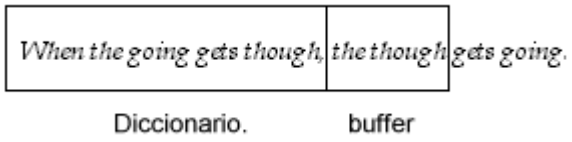


Figura 2-11: Un ejemplo de una ventana.

En la Figura 2 -11, el buffer contiene la frase "<space>*the though*<space>", buscando en el diccionario se encuentra que "<space>*the*<space>" está 23 caracteres hacia atrás, coincide en 5 caracteres, el token es : <23,5>.

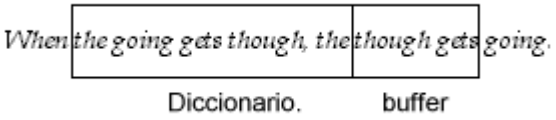


Figura 2-12: La ventana luego de codificar <23,5>

Siguiendo con el ejemplo, el próximo token generado será <12,6>, para la palabra "*though*".

En su primera encarnación (que es esquematizada en este ejemplo en forma cruda), LZ77 adolecía de problemas tales como mal comportamiento en caso de textos de entrada especiales, ineficiencia en las búsquedas en el diccionario, etc. Pero hacia mediados de los 80's ciertas mejoras y refinamientos fueron presentados [NELSON/92] que hicieron de LZ77 un valido competidor de LZ78 (y LZW), que hasta el momento era el dominador absoluto (en el Compress de UNIX, por ejemplo).

Uso restringido.

Dado que el algoritmo LZW se encuentra cubierto por una patente [uspt-4558302] , que autoriza a su dueño el colectar regalías en implementaciones que usen LZW (por ejemplo v42*bis*, el formato de intercambio gráfico GIF, etc.), su utilización por parte del mundo libre de la programación se hace de alguna manera tortuosa.

Pero existen alternativas muchos de los descendientes de LZ77 [Zlib][Deflate][Gzip], fueron diseñados teniendo especial cuidado que pudiesen desarrollarse sin que ninguna patente los cubra. Por ejemplo el formato gráfico PNG (Portable Network Graphic) [Png] se basa en estos algoritmos y con el tiempo se piensa remplazará a GIF [MURRAY/96][Gff].

Puntos Básicos sobre Procesamiento de Imágenes.

Siendo hijo directo del procesamiento de señales, se define (en el lenguaje de todos los días) al procesamiento de imágenes a aquella disciplina que agrupa la teoría, métodos, y aplicaciones que tratan específicamente (o en sus conclusiones) con las imágenes, y lo que es posible hacer con ellas. Algunos de sus más importantes campos de estudio son:

- Representación y modelado de imágenes.
- Mejoramiento de imágenes.
- Restauración de imágenes.
- Análisis de imágenes

Definiciones y Notación

Aquí van algunas definiciones y conceptos que serán útiles a lo largo de este documento. Algunas de éstas serán presentadas para el caso real, unidimensional y discreto (siendo la principal intención el usar computadoras…), siendo su generalización a dos (o más) dimensiones sencilla. Cuando se haga referencia a el caso continuo se aclarará convenientemente.

Señales

Se define como ***señal discreta*** (o secuencia) a cualquier vector de dimensión finita o infinita, por ejemplo

$v = v(k)$; donde $k \in Z$ o $k \in \{0 \cdots n\}$ es una señal discreta

Una ***señal continua*** es simplemente una función, por ejemplo

$v = v(x)$; donde $x \in R$ o $x \in [a,b]$ es una señal continua

donde el intervalo $[a,b]$ ($a,b \in R$), puede ser abierto en uno o todos sus bordes.

Una señal puede ser multidimensional, en ese caso se la representa como un vector de señales unidimensionales, por ejemplo

v = v(k, l) donde $v(k,l) = (v_0(k), v_1(l))$ es una señal de dos dimensiones

Una imagen es generalmente representada digitalmente como una señal bidimensional.
Una señal muy usada es el ***impulso*** $\delta(n)$ definido como

$$\delta(n) := \begin{cases} 1, \text{si } n := 0 \\ 0, \text{sino} \end{cases}$$

Fórmula 2-16.

Ver que cualquier señal puede expresarse como combinación lineal de impulsos desplazados.

$$x(n) = \sum_{k \in Z} x(k).\delta(n-k)$$

Fórmula 2-17.

Sistemas lineares

Defínase como ***sistema*** a un proceso por el cual una señal de entrada *x(n)* es transformada en una señal de salida *y(n)*.

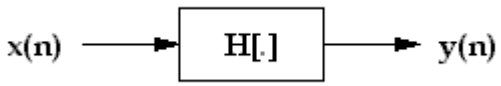


Figura 2-13: Un sistema *y(n)* = H[*x(n)*].

Un sistema es ***lineal*** si cualquier combinación lineal en la entrada se traduce en la misma combinación a la salida:

$$H[a.x(n)] = a.H[x(n)] = a.y(n)$$

Fórmula 2-18.

Un sistema es ***invariante a desplazamientos (LSI)*** si

$$H[x(n-m)] = y(n-m)$$

Fórmula 2-19.

Respuesta al impulso

Usando puntos anteriores se expresa a la salida *y( )* de un sistema lineal invariante a desplazamientos , para cierta entrada *x( )* como

$$y(n) = S[x(n)] = S[\sum_{k \in Z} x(k)\delta(n-k)] = \sum_{k \in Z} x(k).S[\delta(n-k)]$$

Fórmula 2-20.

luego, el sistema queda completamente caracterizado por la ***respuesta al impulso*** del sistema

$$h(n) = S[\delta(n)]$$

Fórmula 2-21.

con lo que la relación entrada salida queda

$$y(n) = \sum_{k \in Z} x(k).h(n-k)$$

Fórmula 2-22.

Convolución.

Dadas dos señales *x(n)* y *h(n)* , se define como la convolución de éstas a

$$y(n) = (x*h)(n) = \sum_{k \in Z} x(k)h(n-k)$$

Fórmula 2-23.

para aquellos valores donde las señales no están definidas se las considera cero.

Puede verse a la convolución como un sistema lineal caracterizada por una de las señales, para la definición anterior:

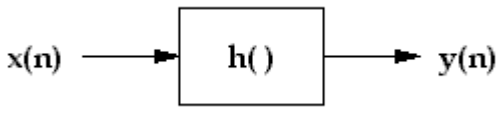


Figura 2-14: La Convolución vista como un sistema lineal, caracterizado en este caso por la señal *h(n).*

La Convolución (como se ha definido aquí, en los reales) es una operación conmutativa, asociativa y distributiva respecto a la suma. Es fácil ver que el impulso es el elemento unitario.

Función, señal, secuencia separable

Una función (señal o secuencia) multidimensional *f(x,y,...,z)* se dice ***separable*** si

$$f(x,y,\ldots,z) = f_1(x).f_2(y).\ \ldots\ .f_n(z)$$

Fórmula 2-24.

Este tipo de secuencias es muy importante en el campo del procesamiento de imágenes ya que muchos de los resultados que se aplican a secuencias unidimensionales se extienden de manera directa a dos o más dimensiones. Por ejemplo, en el caso concreto de una imagen (2-D) como entrada a un sistema LSI [ ] separable, los elementos son los siguientes

Una imagen *x(n,m),* y un sistema LSI caracterizado por su respuesta al impulso *h(n,m)*. Se verá como el hecho de la separabilidad permite aplicar el sistema 2-D como dos sistemas 1-D (lo cual facilita y reduce considerablemente muchos cálculos). Por ejemplo, sea el sistema $y(m,n) = (x*h)(m,n)$ donde la respuesta al impulso $h(m,n) = h_0(m).h_1(n)$ y la señal *x(m,n)* representa una matriz (imagen) 2-D. Luego

$$y(m,n) = \sum_k \sum_l x(k,l).h(m-k,n-l)$$
$$= \sum_k h_0(m-k).\sum_l x(k,l).h_1(n-l)$$

Fórmula 2-25.

donde para un *k* fijo, *x(k,l)* es el vector (señal 1-D) de los elementos de la fila k-ésima de la matriz.

Sea entonces

$$y_1(k,n) = \sum_l x_k(l).h_1(n-l)$$
$$y(m,n) = \sum_k h_o(m-k).y_1(k,n)$$

Fórmula 2-26.

donde y_1 es una convolución en 1-D, y para un cierto *n*, *y(m,n)* también. Gráficamente :

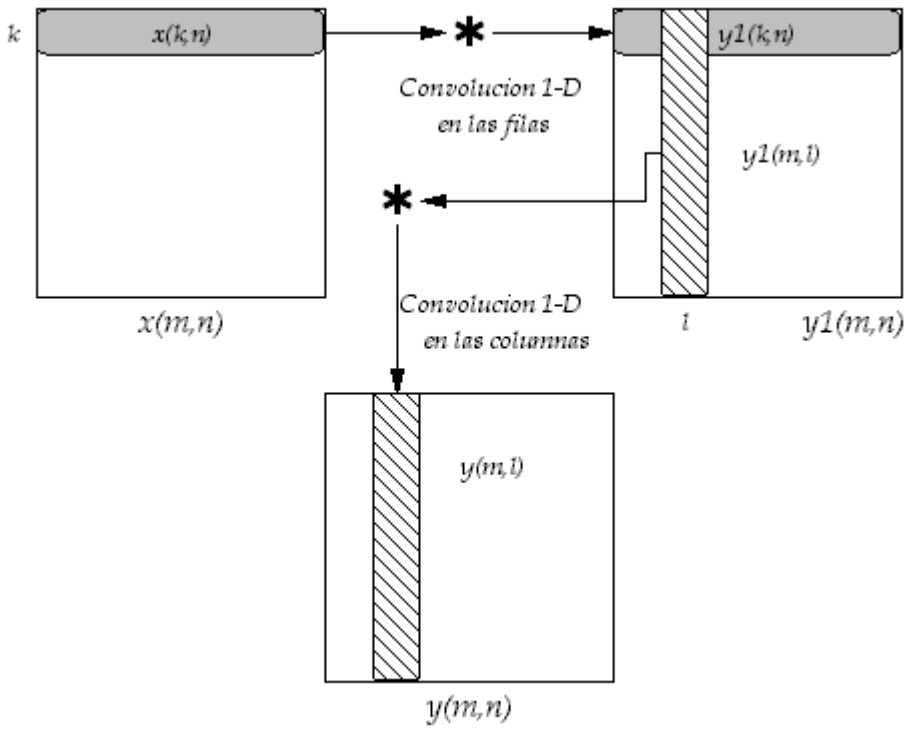


Figura 2-15: Ventajas de la separabilidad. Una convolución bidimensional (de orden . [2]) es equivalente a dos convoluciones unidimensionales sucesivas (orden 2.)

Transformadas.

Un concepto fundamental en el procesamiento de imágenes es el de transformada, la misma está compuesta por dos funciones. La primera de éstas se denomina ***transformada directa*** y la segunda ***transformada inversa*** (a veces también se llaman par transformada-antitransformada). Básicamente son dos sistemas con la siguiente propiedad

$$X(w) = DT[x(y)] \quad , \text{ transformada}$$
$$x(y) = IT[X(w)] \quad , \text{ antitransformada}$$

Fórmula 2-27.

Existen un sinnúmero de transformadas útiles para el procesamiento de imágenes, se hará referencia a algunas de ellas más adelante (algunos métodos de compresión están basados en una aplicación directa de alguna transformada). Vale la pena introducir ahora a la

Transformada de Fourier

El par de la transformada de Fourier en el espacio discreto (cuando se aplica a una secuencia) es

$$X(w) = \sum_{n \in Z} x(n).e^{-i.w.n}$$

$$x(n) = \frac{1}{2.\pi}. \int_{w=-\pi}^{\pi} X(w).e^{i.w.n} dw$$

Fórmula 2-28: Transformada de Fourier

donde si se expresa al exponente complejo e^{iwn} = ***cos(w.n) + i.sen(w.n)*** se percibe intuitivamente al valor *X(w)* como la amplitud asociada a la frecuencia *w*, o que magnitud de frecuencia *w* se encuentra presente en *x(n)*. Se dice habitualmente que la transformada de Fourier lleva a una función (secuencia, señal) del espacio del tiempo *(n)* al espacio de las frecuencias *(w)*.

Notar que la función *X(w)* es continua definida sobre un continuo, entonces cuando se trabaja con secuencias finitas, se utiliza la Transformada Discreta de Fourier (DFT)

$$X(k) = \sum_{n:0...N-1} x(n).e^{-i.(\frac{2.\pi}{N}).k.n}$$

$$x(n) = \frac{1}{N} \sum_{k:0...N-} X(k).e^{i.(\frac{2.\pi}{N}).k.n}$$

Fórmula 2-29: Transformada Discreta de Fourier

Algunas propiedades de la transformada:

-es separable
-es lineal
-la convolución en el espacio normal es equivalente al producto en el espacio de las frecuencias: $x * y(n) \leftrightarrow X.Y(w)$. Ya que la respuesta al impulso *h( )* caracteriza completamente a un LSI (se convoluciona la entrada con *h( )*), también es equivalente decir que el sistema está unívocamente determinado por la ***respuesta en frecuencias*** *H(w) = **F**[h(x)]*.

(Notación: generalmente la transformada de *x(n)* es *X(w),* la de *h(n)* es *H(w)*, y así sucesivamente. Las minúsculas son las funciones originales, y las mayúsculas las transformadas).

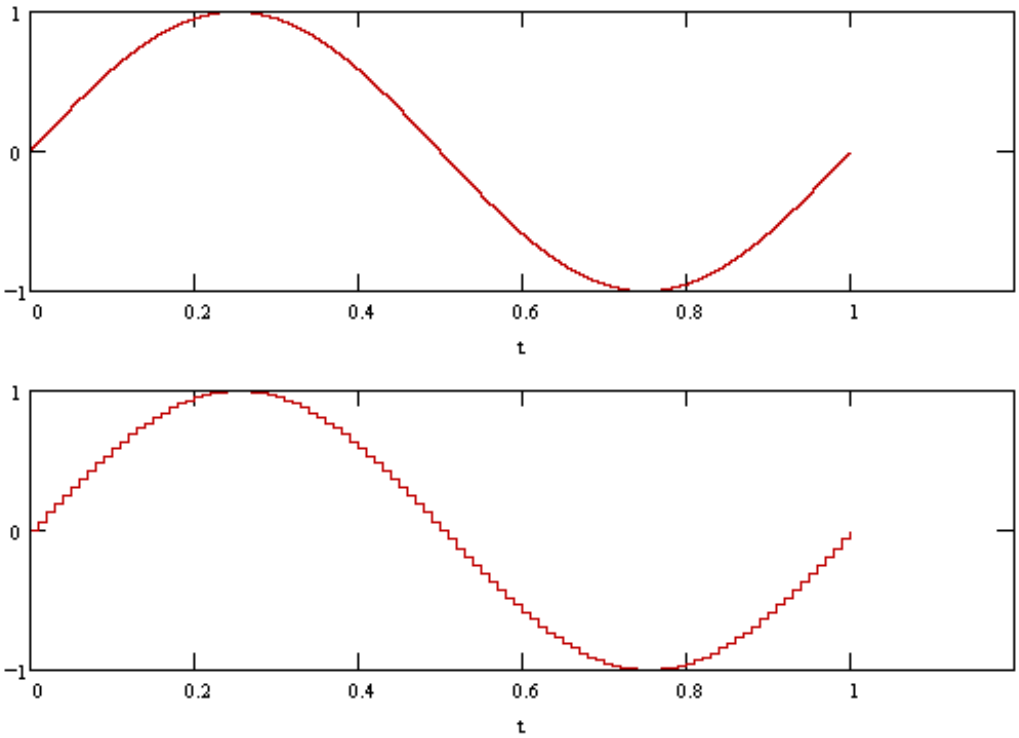
-es periódica de período 2π. Es habitual trabajar con $w \in (-\pi, \pi)$.
-La transformada discreta puede implementarse de forma rápida - $O(n.\log_2(n))$ operaciones -. Esto se denomina Transformada Rápida de Fourier (FFT).

Resolución de una imagen - El teorema de muestreo.

Cuando uno quiere procesar una señal de la vida real en una computadora dos mundos colisionan: el analógico (continuo) con el digital (discreto). Es necesario, entonces hacer un muestreo o digitalizar la señal original para

obtener una representación discreta de la misma, y que la información que contenga sea útil. El parámetro principal que controla el muestreo es la frecuencia, la cantidad de muestras por unidad de medida en el espacio original. Cuando se hable de señales unidimensionales es usual hablar de tantas muestras por segundo, cuando uno se refiera a imágenes serán tantos ***pixeles*** por unidad de longitud (por ejemplo ***dpi*** se refiere a puntos por pulgada). Por ejemplo

Figura 2-16: Tres imágenes de Amber. De derecha a izquierda, original 5x5cm (esto no es totalmente cierto: ya que una computadora procesa este documento está viendo la mejor resolución que pude conseguir en una impresora, 720dpi), con un muestreo a 50dpi, con un muestreo a 10dpi. Como siempre la verdad se encuentra en algún lugar entre medio…



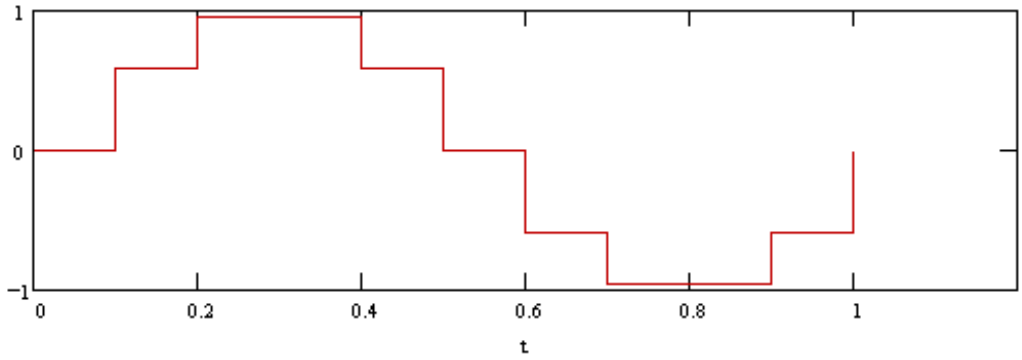


Figura 2-17: Una señal de un ciclo por segundo. De arriba hacia abajo: original, 100 y 10 muestras por segundo.

Si se observan las distintas encarnaciones de las señales puede decirse que se ha perdido información, que dada una de las señales muestreadas jamás se podrá recuperar la señal original, ¿cierto?. Bueno, no tanto. De eso es lo que el teorema del muestreo se trata. Algunas definiciones primero:

Una función *f(x)* es de ***banda limitada*** (tiene ancho de banda limitado) si su transformada de Fourier es cero fuera de una región acotada del espacio de las frecuencias. A esta región se la denomina ***soporte*** de la transformada.

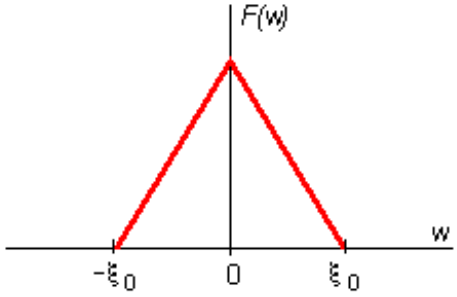


Figura 2-18: Un esquema de la transformada Fourier de una función de banda limitada 1-D.

el soporte de la función es el intervalo $[-\xi_0, \xi_0]$. A esta longitud, o sea al valor $2.\xi_0$, se lo llama ***frecuencia de Nyquist***.

Si para realizar el muestreo toma muestras de la función original en intervalos de Δx, se define a la ***frecuencia de muestreo*** como $\xi_s = \frac{1}{\Delta x}$.

El *Teorema de Muestreo* [JAIN/89][CASTLEMAN/96] dice : "*Una función f(x) de* ***banda limitada*** *que es muestreada uniformemente con un espacio entre muestras de* Δx *puede recuperarse sin error a partir de las muestras, si la* ***frecuencia de muestreo*** *es mayor que la* ***frecuencia de Nyquist***"

Métodos de Compresión

Introducción

En esta sección se describirán una serie de técnicas cuyo propósito es el de comprimir una imagen. La lista pretende ser representativa de los métodos utilizados hoy día, y por lo tanto no es ni pretende ser exhaustiva. De cada método se dará una pequeña introducción, se describirá la base conceptual y/o teórica que lo sostenga, se explicará como trabajan las fases de compresión/descompresión, y finalmente se señalarán ventajas desventaja y como se usa hoy día.

Existen dos características fundamentales en los métodos que se describirán, se dirá que unos de ellos son ***con pérdida*** y otros serán ***sin pérdida***. Que un método sea con pérdida significa que el método, al comprimir una imagen, introduce en esta distorsión, esto quiere decir lo siguiente: la imagen original es comprimida por cierta técnica generando una imagen 'comprimida'; al descomprimir esta imagen (para visualizarla) se observa, si se analiza muy detalladamente, que ésta no es exactamente igual a la inicial. Afortunadamente el ojo humano casi no percibe diferencia entre las dos imágenes.

DPCM (Codificación Diferencial por Modulación de Pulso)

Esta técnica de compresión (una de las primeras) explota la dependencia o correlación entre valores adyacentes (en el espacio si se habla de imágenes, en el tiempo si uno se refiere a señales). Para señales continuas es de esperarse que para dos valores consecutivos, la variación sea pequeña, entonces porqué no codificar un valor que varia poco (las diferencias, que por lo tanto tienen menos información) que uno que varia mucho (la señal)?

Las siguientes figuras ejemplificaran el método en una dimensión. La de la izquierda muestra una sinusoide, sobre ella se toman 101 puntos que se han cuantizado con un cuantizador uniforme de paso 1 (aunque no parezca los valores que se ven son enteros, y el rango de los mismos oscila entre -128 y 127). Los valores tomados por dicha señal se denominaran $f_{i:0..100}$.

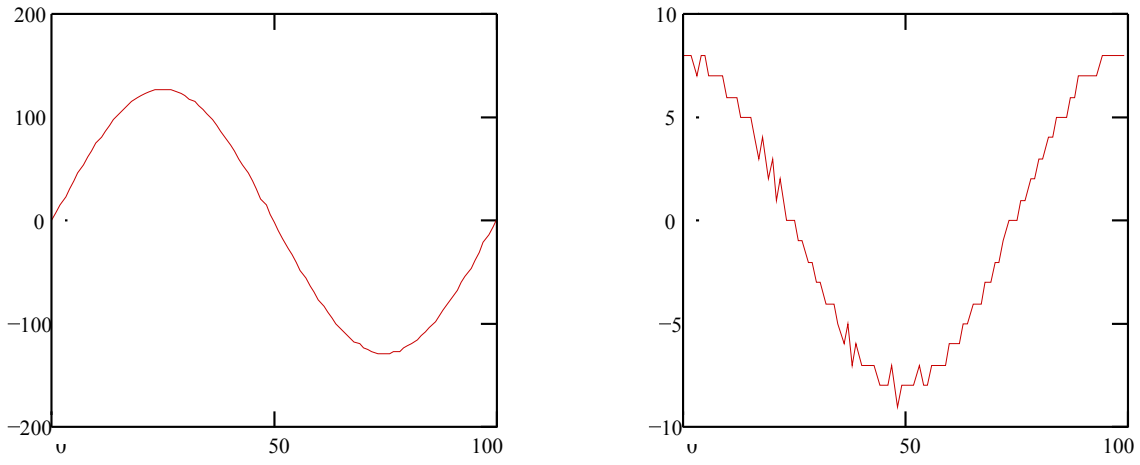


Figura 3-1: (izq.) señal original cuantizada. (der.) señal transformada con regla diferencial.

La figura de la derecha muestra una nueva señal construida con la siguiente regla: $d_j = f_{j+1} - f_j$. Esto da como resultado una señal cuyos valores oscilan entre -9 y 8. La diferencia en la codificación es evidente: mientras que la primera señal se codifica con 8 bits por punto, a la segunda le bastan 5 bits, la compresión obtenida es de 1.6: 1.

Codificación Predictiva.

Buscando mayores valores de compresión puede extenderse la idea anterior como sigue. Dado un dato u_i (pulso, pixel, elemento) a transmitir se intenta predecir su valor en función de sus predecesores, si la predicción es buena este valor u'_i será muy parecido a u_i. Es de esperarse que $\varepsilon_i = u_i - u'_i \approx 0$, y este es el valor a transmitir ya que son necesarios pocos bits para esta tarea.

Dependiendo que tan buena sea la predicción, se tendrá una serie de valores cercanos a cero, sobre los cuales se podrá realizar una mayor compresión de datos (usando menos bits en el cuantizador, o usando un esquema de codificación de entropía que saque provecho de esta concentración alrededor del cero).

Como no se dispone de información sobre los valores no transmitidos (recibidos) la predicción debe estar basada sobre datos ya transmitidos (recibidos). Este hecho de no usar valores no transmitidos (recibidos) o datos del futuro es una característica (y restricción) muy importante y se llama ***causalidad.***

Para el caso de señales bidimensionales el esquema es similar, pero la formula del predictor varia según que pixeles adyacentes se consideran.

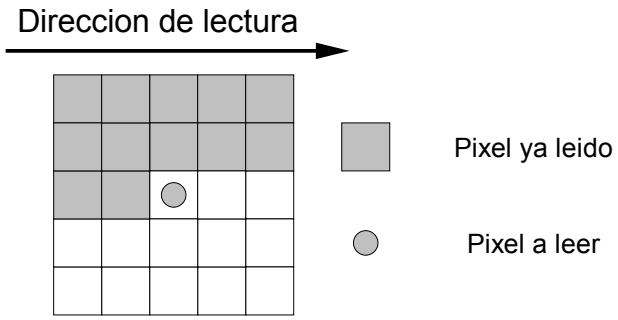


Figura 3-2: Ejemplo de predictor, la ventana muestra la posible configuración de un predictor, no se especifica una formula pues existen muchas (por ej. : simplemente promediar los valores). Por supuesto existen ventanas mayores o menores que la exhibida aquí.

Puede verse entonces el caso más básico de DPCM mencionado anteriormente, como una codificación predictiva, donde el predictor es simplemente tomar el valor anterior transmitido. El esquema general se ve como sigue:

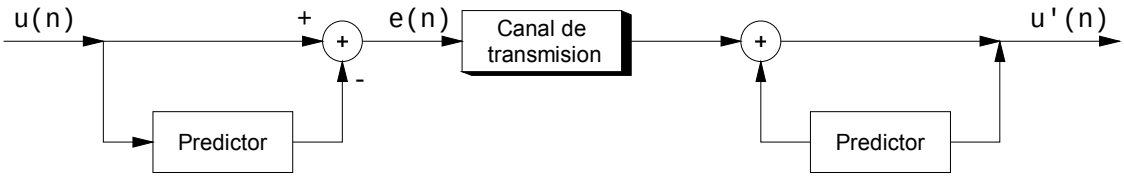


Figura 3-3: Esquema general de codificación/decodificación predictiva.

El Standard: JPEG

Introducción

Motivados por las ventajas [WALLACE/91] que para todos significaría un standard internacional para la compresión de imágenes (el intercambio de información por nombrar una), la CCITT y la ISO trabajando en conjunto

crearon JPEG [MITCHEL/93] (por Joint Photographic Experts Group). Este grupo trabajó en la especificación de técnicas de compresión de imágenes de tonos continuos color y en escalas de gris (como son comúnmente las imágenes 'fotográficas') sin pérdida de información, basadas en técnicas predictivas/adaptivas, con un codificador Huffman, y una referida a la compresión con pérdida de información. A ésta ultima se la llama 'vulgarmente' JPEG.

Nota: Aunque existen dentro del propio standard varios sabores (o modos de operación) como ser codificación progresiva, jerárquica, etc.) aquí se describirá la más sencilla de sus variantes, referido en el original como '*the Baseline Method'*.

El método

Como tantas técnicas de compresión, JPEG está basada en una transformada, que tiene como objetivo decorrelacionar los pixeles de la imagen.

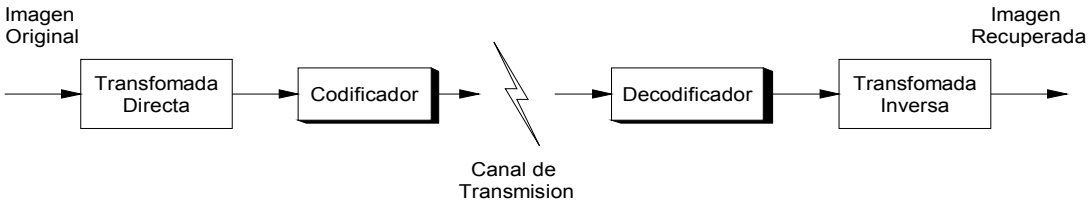


Figura 3-4: Esquema típico de codificación por transformada.

Se utiliza aquí la Transformada Discreta del Coseno (DCT) [WALLACE/91] [JAIN/89] y se define como

$$DCT(i,j) = C(i).C(j).\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} Pixel(x,y).\cos(\frac{(2x+1)i\pi}{2N}).\cos(\frac{(2y+1)j\pi}{2N})$$

$$C(x) = \begin{cases} \frac{1}{\sqrt{N}} & \text{si } x = 0 \\ \sqrt{\frac{2}{N}} & \text{sino} \end{cases}$$

Fórmula 3-1: DCT, aplicada a una matriz de NxN elementos. El rango de *i, j es de 0..N-1*

$$Pixel(x,y) = \frac{1}{\sqrt{2N}}.\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(i).C(j).DCT(i,j).\cos(\frac{(2x+1)i\pi}{2N}).\cos(\frac{(2y+1)j\pi}{2N})$$

Fórmula 3-2: Transformada Inversa Discreta del Coseno.

Codificación

A la entrada del codificador, la imagen fuente es dividida en bloques de 8x8 pixeles y cada uno de los valores de la imagen son trasladados desde el rango $[0, 2^p-1]$ hacia $[-2^{p-1}, 2^{p-1}-1]$, normalmente esto mapea el rango [0-255] hacia [-128, 127]. Luego cada uno de estos bloques son las entradas a la transformada directa (FDCT).

La codificación se resume con el siguiente esquema:

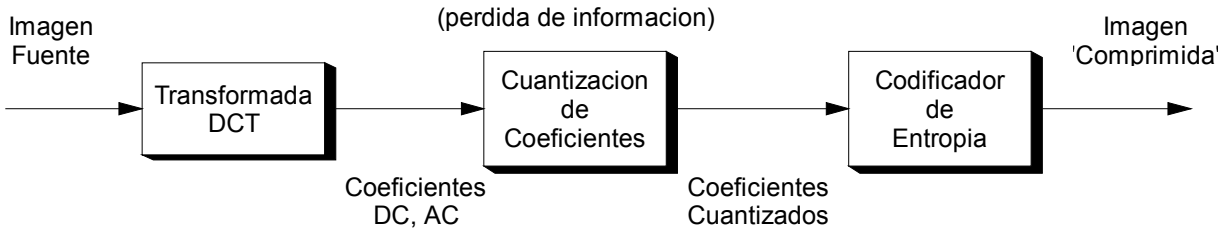


Figura 3-5: Esquema muy general del codificador JPEG

La primera fase es la aplicación directa de la Transformada del Coseno. Tiene particular interés el coeficiente DCT(0,0), o DC para abreviar, ya que la función base allí es una constante (prácticamente el doble del promedio de los valores del bloque); todos los demás coeficientes son llamados AC (AC/DC!). Esta transformada puede implementarse en algoritmos 'rápidos' (similarmente a la FFT).

Se ve también que los valores de la matriz de coeficientes de la transformada tienen su orden de magnitud mayor en DC y van decreciendo a medida que la distancia a este aumenta (hasta agradablemente rápido a veces).

Luego los coeficientes son cuantizados. Es aquí donde la pérdida de información empieza a ocurrir (aunque que se asume aritmética de precisión 'infinita'), por lo que una elección adecuada de la 'matriz de cuantización' es necesaria.

Después de la cuantización los coeficientes DC son codificados mediante DPCM, lo cual tiene sentido considerando que este coeficiente se refiere a la contribución de la imagen original, por lo que se espera que no varíen entre sí demasiado.

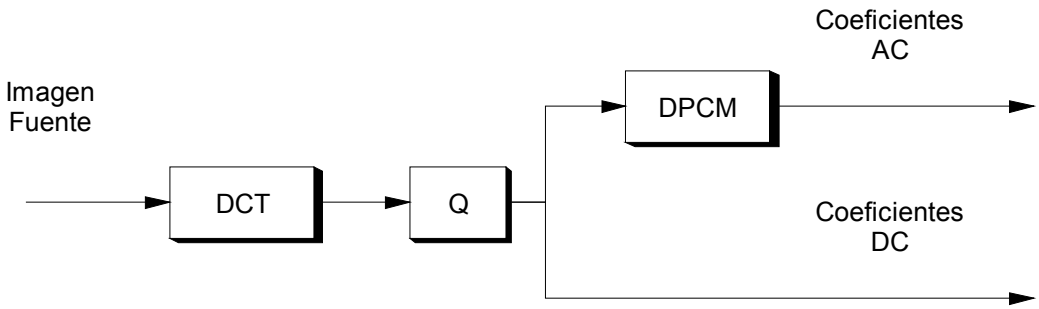


Figura 3-6: Etapa inicial del codificador JPEG, antes del codificador de entropía.

Es ahora donde la verdadera codificación se lleva a cabo. JPEG realiza dos pasos: primero rearregla los coeficientes en una secuencia en zig-zag para luego realizar un run-length-coding de secuencias de ceros (aprovechándose de la estructura especial de la matriz de valores cuantizados); luego los coeficientes son enviados a

un ‘Codificador de Entropía’, el cual involucra codificación por Huffman o codificación aritmética (dependiendo de la implementación)

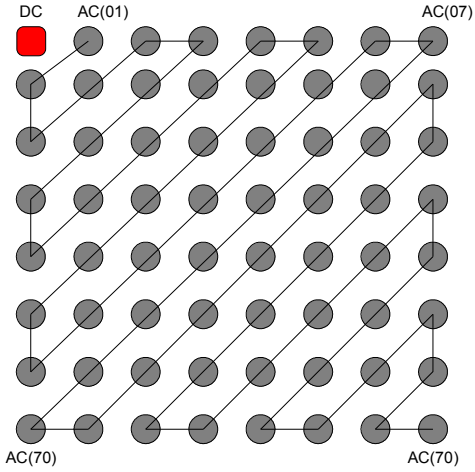


Figura 3-7: Secuencia zig-zag

JPEG permite controlar la cantidad de compresión deseada mediante un parámetro que afecta al cuantizador del codificador. Esto impacta directamente en las corridas de ceros que aparecerán en la matriz.

Decodificación

La decodificación consiste en la aplicación de los pasos inversos sobre los datos recibidos por el canal de información, o sea:

- Alimentar los datos al 'Decodificador de Entropía'
- Transformar la secuencia de zig-zag a 'normal'
- Decodificar los valores DC (previamente codificados con DCPM)
- Decuantizar los coeficientes
- Aplicar la transformada inversa

gráficamente (el codificador/decodificador de entropía no es relevante aquí):

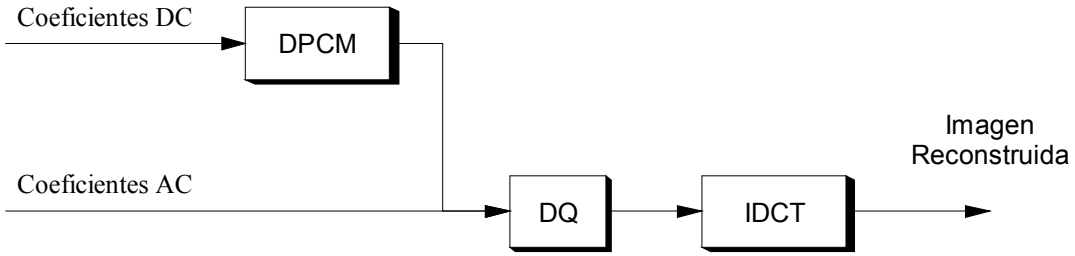


Figura 3-8: Etapa final del decodificador JPEG, luego del decodificador de entropía.

Las tablas de Cuantización y Huffman.

Cada elemento *F(u,v)* del bloque transformado es cuantizado con un cuantizador uniforme de paso *Q(u,v),* lo que define una matriz de cuantización *Q*. Aunque el standard deja lugar para que el usuario defina como mejor le parezcan las tablas de Cuantización y de Huffman, también propone un conjunto de tablas probadas, las cuales pueden observarse en la documentación. Debe tenerse entonces la precaución, a la hora de diseñar el codec, que ambos extremos conozcan las tablas, esto se logra estableciendo tablas fijas (como las que figuran en el standard), o incluyendo (transmitiendo) las tablas en la 'imagen comprimida'.

Conclusiones.

El standard JPEG [MITCHEL/93][WALLACE/91] tiene la virtud de lograr tasas de compresión altos para imágenes de tipo fotográfico (entre 10-20 a 1) sin que se puedan apreciar una gran degradación en la imagen restaurada. Para niveles de compresión más grandes sin embargo, la degradación se torna evidente (debido a un efecto de 'bloqueo' inherente a la estrategia inicial de subdividir la imagen en bloques de 8x8), como así también la aplicación de la técnica a imágenes que no provengan del 'mundo natural'. Para cualquier método de compresión que ha aparecido o aparezca, sin embargo, JPEG es la piedra de toque a la hora de comparar sus desempeños (estandarización obliga...).

El método es ampliamente usado hoy día, debido a las eficientes implementaciones [Ijg] que existen.

Compresión Fractal

Introducción

Este esquema de compresión, fue propuesto por M. Barnsley [BARNSLEY/89] quien fundó una empresa (Iterated Systems) [Iterated] en base a ella. Vale la pena aclarar que ciertos detalles de su implementación están protegidos por patentes [uspt-4941193]. Una muy buena referencia al tema de compresión fractal, así como código, se encuentra en [FISHER/92][FISHER/95].

Método

'Los fractales son estructuras cuya dimensión de Haussdorf es fraccionaria'. Confundido?, bien. Pruebe entonces: los fractales son estructuras que exhiben auto-similaridad a distintas escalas. Esto se observa en las siguientes figuras:

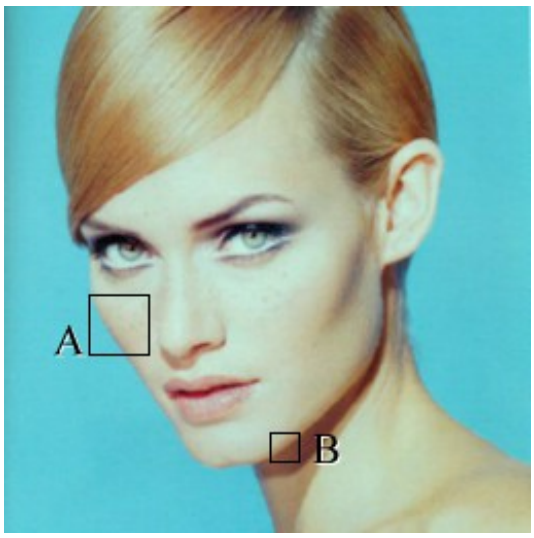


Figura 3-9: Aunque los bloques A y B corresponden a distintas zonas y a distintas escalas de la imagen éstos son similares, permitidas variaciones de tono reflejos y rotaciones.

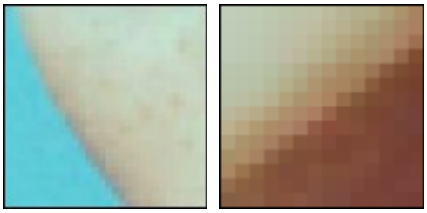


Figura 3-10: Los bloques A y B ampliados donde de aprecia la similitud de los mismos. (Aun es prematuro aventurar que Amber Valletta es un fractal)

El método está basado en lo que se denomina Sistemas de Funciones Iterativas (o IFS en el original), lo cual se refiere a aplicar una misma transformada una y otra vez sobre un dato inicial. Se pone como 'única' restricción que dicha transformada sea contractiva. Esto significa que dos puntos en la imagen dominio estarán más cerca en la imagen transformada (dada alguna noción de distancia).

Puede verse que por las características de la transformada el sistema converge a un 'punto fijo', el cual es llamado atractor para este IFS. Uno se pregunta entonces si dado que la imagen a comprimir sea el atractor, almacenar ésta como un conjunto de transformadas lleva a una compresión de la imagen, la respuesta es: SI.

Una transformada de las que se habla se define como:

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

Fórmula 3-3.

donde ***z*** es el nivel de gris, s_i controla el contraste y o_i el brillo de la transformación. Los subíndices se refieren a la partición que se hace sobre el dominio de la imagen, por lo que en realidad más que una transformada se tienen un cierto número de las mismas, por ejemplo N.

Cada transformada está definida como $w_i : D_i \rightarrow R_i$, donde D_i es el dominio y R_i el rango. La transformada definida así se denomina PIFS, por Sistemas Particionados Iterativos de Funciones.

Entonces la compresión viene dada por el número de transformaciones necesarias para que el PIFS obtenido tenga como atractor a la imagen en cuestión.

Codificación

Dada una imagen f a codificar, se quieren encontrar una colección de transformadas w_1, ..., w_N con

$$W = \bigcup_{i=1}^{N} w_i \text{ tal que } f = W(f) = w_1(f) \cup w_2(f) \cup \ldots w_N(f)$$

Fórmula 3-4.

Entonces se busca una partición de ***f*** en piezas a las cuales se les aplica las w_i y se obtiene de vuelta ***f***. En general y de una manera más realista se espera llegar a una ***f' = W(f')*** con ***d(f', f)*** pequeña. O sea que se construye una transformada ***W*** cuyo punto fijo ***f'*** se parece mucho a ***f***.

Es suficiente entonces aproximar partes de la imagen con piezas trasformadas, para hacer esto basta minimizar

$$d(f \cap R_i, w_i(f)) \qquad ; \qquad i = 1, \ldots, N$$

Fórmula 3-5.

El corazón del problema es: encontrar piezas D_i a las que se les aplica una w_i y se obtiene algo muy parecido a la imagen sobre R_i.

Ejemplo

A fines ilustrativos se explica como esto podría ser hecho. Sea una imagen de 256x256 pixeles con 256 niveles de gris. Sean entonces $\boldsymbol{R_1, R_2, ..., R_{1024}}$ los rectángulos de 8x8 que resultan de dividir la imagen por 32 en cada dimensión; y sea *D* el conjunto de todos los sub-rectangulos de 16x16 posibles en la imagen (éstos son 241 x 241 = 58081 cuadrados). Entonces para cada $\boldsymbol{R_i}$ se busca $D_i \in D$ tal que minimiza Fórmula 3 -5. Hay 8 formas de mapear un cuadrado en otro, o sea esto es hacer 8 x 58081 = 464648 comparaciones por cada uno de los 1024 $\boldsymbol{R_i}$, más de 47 millones de comparaciones (también hay que considerar las diferencias de tamaño de los $\boldsymbol{D_i}$ respecto a los R_i). Otro detalle en consideración es la elección de los parámetros de contraste y brillo $\boldsymbol{o_i}$ y $\boldsymbol{s_i}$ (por ejemplo usando cuadrados mínimos). Una vez obtenidos $\boldsymbol{w_1, ..., w_{1024}}$ la codificación está 'concluida'. Los resultados de esta codificación son: cada transformada se representa por 8+8 bits para especificar $\boldsymbol{D_i}$, 7 bits para $\boldsymbol{o_i}$, 5 bits para $\boldsymbol{s_i}$, 3 bits para determinar rotación y 'flip' del mapeo (8+8+7+5+3=31 bits); con 1024 transformadas son en total 3968 bytes, que frente a los 256x256x256 = 65536 bytes originales dan una tasa de compresión de 16.5 : 1.

Nada mal considerando que la partición realizada es bastante simple. Sin embargo este tipo de partición hace que aparezcan efectos de 'bloqueo' (en la terminología habitual esto denota que los limites entre los bloques en los que se particiona la imagen se distinguen a simple vista, dando la impresión de un 'mosaico') o artefactos [ ] en la imagen por lo que otros métodos más elaborados existen para ello, por ejemplo particiones en Quad-Trees , Particiones HV , partición triangular [FISHER/92], etc.

En este ejemplo el número de particiones es fijo, lo cual no tiene por que ser cierto. Métodos que realizan particiones adaptivas así como etapas posteriores de codificación por entropía son mejoras para incrementar tanto la calidad como el nivel de compresión obtenidos

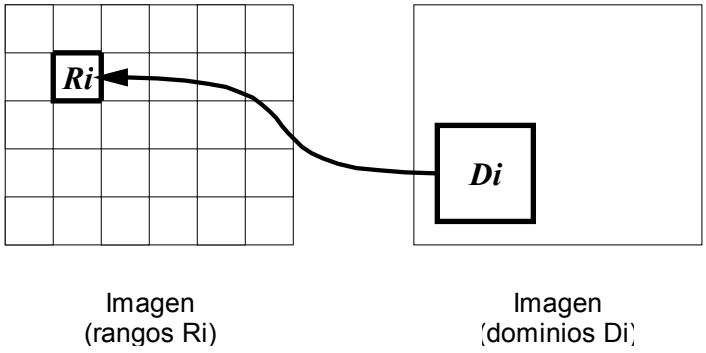


Figura 3-11: cómo opera una transformada en la compresión fractal.

Decodificación

La decodificación de la imagen consiste sencillamente en aplicar el PIFS a una imagen inicial (generalmente aleatoria), una y otra vez hasta que cierto criterio de convergencia se cumpla (por ejemplo: variación entre dos iteraciones consecutivas menor que cierta cota.)

Conclusiones

De entre los nuevos métodos de compresión de datos, la compresión fractal resulta atractiva por su sencillez y altos niveles de compresión alcanzados (llegando a compresiones de 100:1 para algunas imágenes fotográficas).

La imagen codificada como un fractal tiene una particularidad muy interesante: *no tiene resolución* asociada. Una imagen digital convencional es el resultado de otorgar un valor a un pixel, que representa cierta zona rectangular

del espacio donde se tomo la imagen. Dependiendo de en cuantas zonas se dividió el espacio y del tamaño de cada zona se tiene una imagen de cierta *resolución*, medida generalmente en puntos por unidad de longitud. Para poder mostrar la imagen en dispositivos que tienen distintas resoluciones entre sí (como monitores e impresoras) se utilizan procedimientos de mapeo que por lo general perjudican a la misma. Una imagen vista como un fractal es, por el contrario, escalable y contraíble al infinito (en la práctica esto está limitado por la resolución de la imagen que se transforma y de que tan bien interpreta la transformada a la imagen como un fractal, evidentemente no es posible ver los átomos de la piel...).

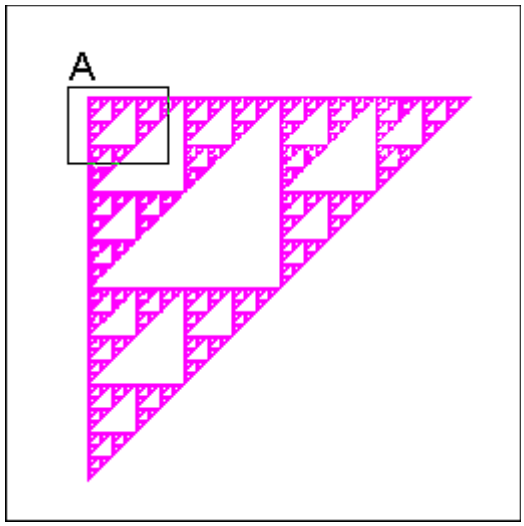


Figura 3-12: Una clásica figura fractal (triángulo de Sierpinski) para demostrar la independencia de resolución, tamaño 256x256 pixeles:

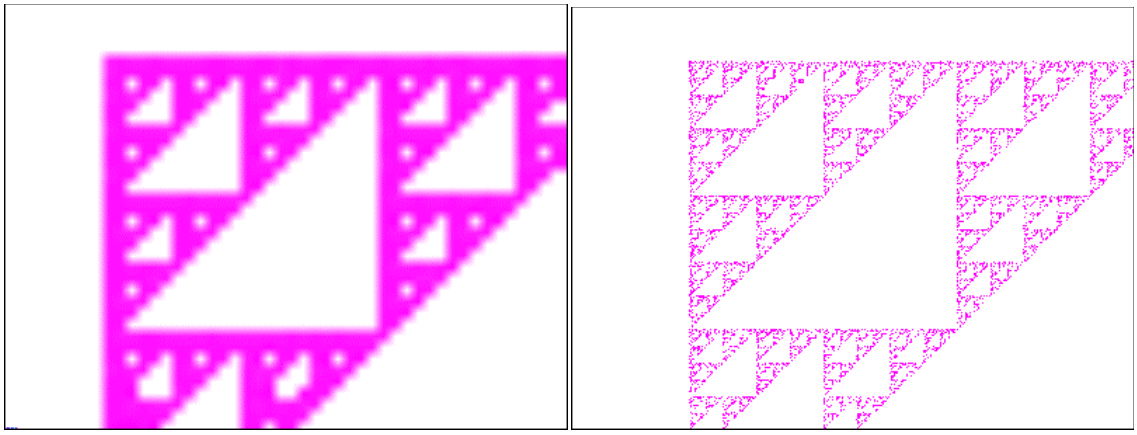


Figura 3-13: Ampliación del bloque 'A' de la figura anterior, tamaño: 400x300 pixeles. La imagen de la izquierda es una ampliación ordinaria de la imagen que usa la interpolación para aumentar la resolución de la imagen. La imagen de la derecha muestra la misma zona pero el ajuste de resolución se hace a través de las propiedades fractales de la imagen.

Aunque el proceso de codificación resulta computacionalmente costoso, la decodificación es realmente directa y rápida. El método resulta realmente propicio para imágenes que poseen características similares en si mismas. Una implementación de una transformada rápida fractal es discutida en [MCGREGOR/94] [MCGREGOR/96], donde el número de comparaciones es reducido en gran medida usando en vez de una búsqueda exhaustiva sobre los dominios, una búsqueda sobre una estructura de árbol.

Asistido por un chip de bajo costo que acelere la transformada-antitransformada este esquema de compresión de datos puede desplazar próximamente a los 'viejos' standard.

Actualmente Iterated Systems está y sigue desarrollando con gran fuerza esta tecnología para las áreas de transmisión de datos en Internet.

La pirámide Laplaciana

Introducción

Esta técnica [BURT/83] genera a partir de una imagen un código que consiste en una versión a escala reducida y una secuencia de 'imágenes' que contienen la información necesaria para transformar esa imagen reducida en otra aproximación en un paso de escala superior. La teoría detrás de este método puede considerarse antepasada directa de las técnicas de análisis multiresolución [MALLAT/89][MALLAT/87], y la codificación por wavelets que se verá más adelante [BRADLEY/93][].

El Método

Sea una imagen g_0 ,obténgase la secuencia g_i, aplicando un filtro pasabajos w. Cada elemento de la sucesión tiene la mitad del ancho de banda de su predecesor, por lo tanto la imagen puede submuestrearse por 2 sin ninguna pérdida de información (gracias al Teorema de Muestreo [JAIN/89][]). A esta secuencia se la llama Pirámide Gaussiana, pues - en el límite - ésa es la forma que recuerda el filtro que la genera.

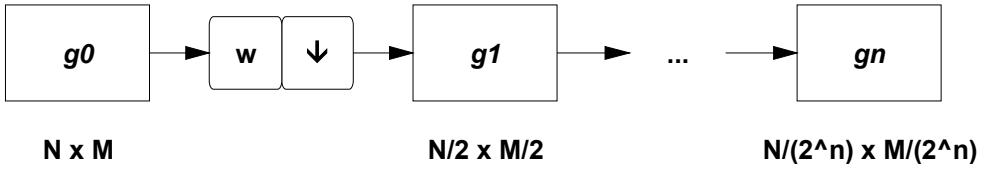


Figura 3-14: secuencia de imágenes g_i , el tamaño de las imágenes va disminuyendo.

A partir de esta secuencia se construye otra $\{L_i\}_{i:0..n}$, con

$$L_i = g_i - g_{i+1}$$

Fórmula 3-6.

(si se tienen $g_0 \ldots g_n$, se define $L_n = g_n$). Luego el código consiste en la secuencia de las L_i, llamada pirámide Laplaciana pues la diferencia entre las dos imágenes es similar a aplicar los operadores Laplacianos (detectores de bordes) tan comunes en procesamiento de imágenes [JAIN/89][CASTLEMAN/96]. Este código tiene la ventaja que se ha reducido la correlación de cada elemento y por lo tanto se puede esperar compresión de datos.

Codificación

Básicamente generar la secuencia $L_0, \ldots, L_n$ es codificar. Si uno se detiene aquí puede considerarse como una cota inferior de codificación a la entropía de cada imagen (nivel de la pirámide). Prácticamente uno se acerca a estos valores con técnicas como codificación de longitud variable (como por ejemplo, Huffman [] [HUFFMAN/52]).

Hasta aquí, (suponiendo precisión aritmética 'ilimitada') no existe pérdida de información. Para obtener mayores tasas de compresión uno debe estar dispuesto a sacrificar la calidad de los datos.

La entropía se reduce drásticamente cuantizando [][GERSHO/92] la imagen, con lo cual puede comprimirse aún más la imagen. La elección de la técnica de cuantización dicta los nuevos niveles de compresión así como que tanto uno se aparta de una reconstrucción perfecta. Se lograron buenos resultados con cuantizadores uniformes.

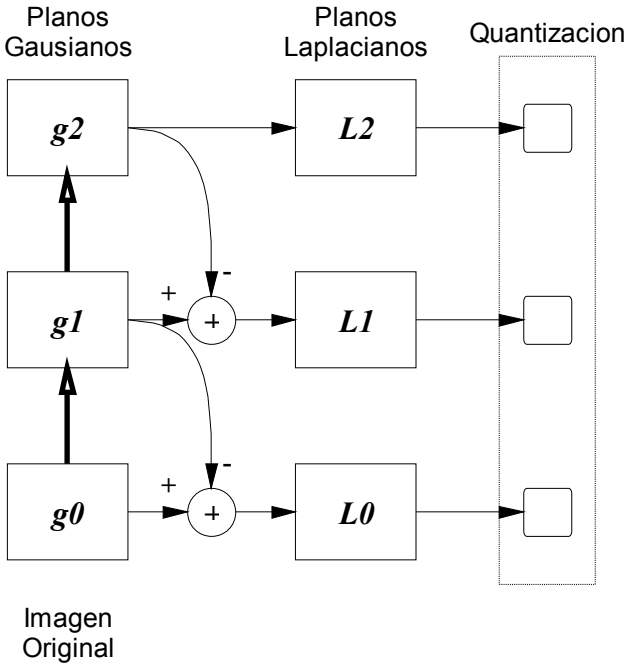


Figura 3-15: Etapa de codificación.

Decodificación

La decodificación consiste en sumar todas las imágenes del código entre sí (una vez salvadas las diferencias de tamaño). Esto desprende naturalmente de **Fórmula 3 -6.**

Cuando se trabaja con la versión de imágenes cuantizadas se trabaja directamente sobre los valores cuantizados.

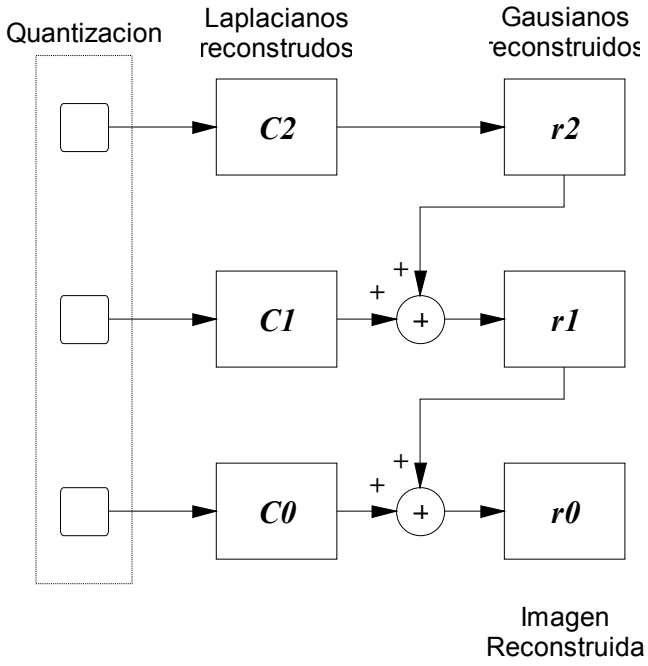


Figura 3-16: Etapa de reconstrucción.

Conclusiones

Este esquema de codificación genera una cantidad total de coeficientes mayor a la original. Por un lado puede verse esto como una desventaja frente a otros métodos descriptos luego [] [] que preservan este número. Por otro lado la 'redundancia' da robustez al código frente a problemas en el medio de transmisión de la señal. A su vez el método es sencillo y fácil de implementar y computar.

Figura 3-17: La pirámide Gaussiana, de mayor a menor tamaño, g_0, g_1, g_2 y g_3.

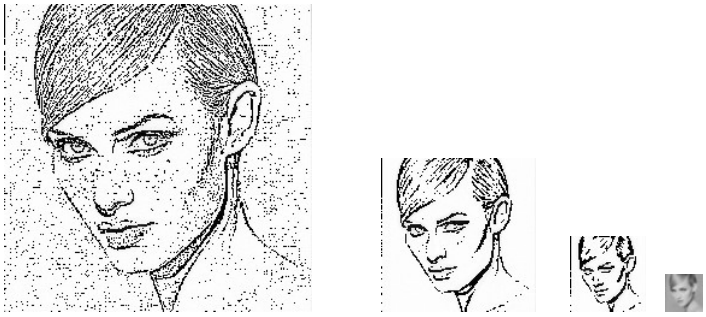


Figura 3-18: La pirámide Laplaciana, de mayor a menor tamaño, l_0, l_1, l_2 y $l_3 = g_3$.

También este método se presta particularmente bien para la transmisión progresiva de información: inicialmente se transmiten los niveles altos de la pirámide (de baja entropía) y el receptor obtiene una imagen cada vez con mayor detalle, decidiendo hasta cuando es suficiente. La transmisión progresiva de una imagen es una técnica utilizada por distintos métodos de codificación de imágenes (GIF, JPEG, Fractal, Wavelets entre otros) y puede ser vista en acción en muchas páginas de la WWW.

Codificación Sub-Banda.

Introducción

La idea básica de la codificación sub-banda [GERSHO/92][WESTERINK/89] es la de separar la banda de frecuencia de una señal en sub-bandas y entonces codificar cada banda con un codificador y una distribución de bits que saquen provecho de la distribución estadística de la misma. Al reducir el ancho de banda de cada sub-banda es posible submuestrear (sin pérdida de información) la señal reduciendo aun más la información a codificar. Aunque se mencionó que las sub-bandas se codifican separadamente, es posible codificarlas en forma conjunta, diseñando lo que se denomina un Cuantizador Vectorial (VQ).

Mas adelante se verá que existe una relación directa de la codificación por wavelets y la codificación sub-banda.

El método

Para fines didácticos considérese un banco de filtros pasabanda ideales

$$H_m(f) = \begin{cases} 1 & \text{si} \quad \dfrac{(m-1)f_s}{2M} < f < \dfrac{m.f_s}{2M} \\ 0 & \text{sino} \end{cases}$$

Fórmula 3-7: Una familia de filtros pasabanda ideales.

donde m: 1...M ; M es la cantidad de bandas y f_s es la frecuencia de muestreo (que debe cumplir las condiciones del teorema de muestreo, para la posterior reconstrucción de la señal). Cada uno de los filtros tiene ancho de banda f_s/M; las salidas obtenidas de aplicar el filtro m-ésimo a la señal X, son las sub-bandas X_m.

La naturaleza de los filtros implica que

$$X = \sum_{m=1}^{M} X_m$$

Fórmula 3-8.

Habiéndose reducido el ancho de banda de cada sub-banda M veces, se puede reducir en ese factor la cantidad de muestras por sub-banda (sub-muestreo). Luego la cantidad total de muestras es igual a la cantidad total de la señal original. El próximo (si se desea) paso es cuantizar las sub-bandas. La información se transmite, descuantiza, aumenta en muestras (interpolación) y se suma para obtener la señal original. Gráficamente:

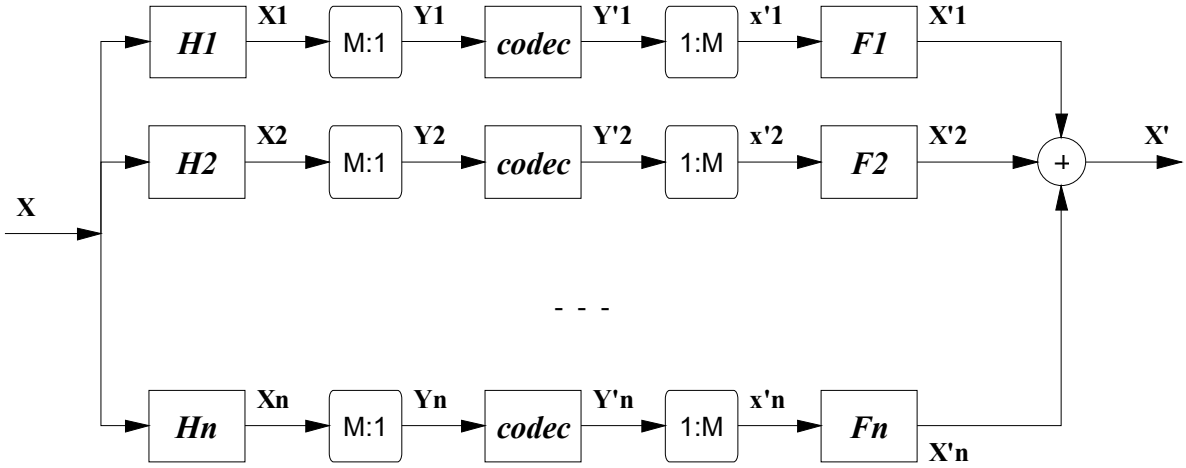


Figura 3-19: Esquema de descomposición y recomposición en el esquema de bub-bandas. Nota: "codec" significa (cuantización -) codificación - transmisión - decodificación (- decuantización).

El caso del banco de dos canales.

Para el caso de un banco de filtros de dos canales (sub-bandas), la señal de entrada se divide en una banda de baja y otra de alta frecuencia respectivamente. Para lograr esto se utilizan dos filtros H_0 y H_1. Se demuestra en [WESTERINK/89], que si se asume transmisión perfecta (y sin pérdida de información)

$$y_i = y'_i$$
$$y_0 = (x * h_0) \downarrow 2$$
$$y_1 = (x * h_1) \downarrow 2$$

Fórmula 3-9.

donde *(x*h)* denota la convolución de *x* con *h*. Entonces el proceso arriba descripto puede notarse como:

$$x'(n) = (y_0 \uparrow 2) * f_0 + (y_1 \uparrow 2) * f_1$$

Fórmula 3-10.

luego su transformada Fourier es

$$X'(\omega) = \frac{1}{2} \cdot \left[F_0(\omega) \cdot H_0(\omega) + F_1(\omega) \cdot H_1(\omega) \right] \cdot X(\omega) + \frac{1}{2} \cdot \left[F_0(\omega) \cdot H_0(\omega+\pi) + F_1(\omega) \cdot H_1(\omega+\pi) \right] \cdot X(\omega+\pi)$$

Fórmula 3-11.

En el caso de filtros 'perfectos', el segundo termino es cero (los filtros no se solapan - aliasing -) y el primero es igual a uno. Se logra entonces reconstrucción perfecta.

En el caso de que no se disponga de tales filtros (por ej. filtros FIR), puede hacerse:

$$\begin{cases} F_0 = 2 \cdot H_1(\omega + \pi) \\ F_1 = -2 \cdot H_0(\omega + \pi) \end{cases}$$

Fórmula 3-12.

con lo que el termino de 'aliasing' se hace cero, y el sistema se transforma en LSI (Lineal Invariante a Desplazamientos).

QMFs

Son ampliamente usados pares de filtros (pasabajos -pasabanda-) llamados QMF (del ingles Quadrature Mirror Filters: Filtros de Espejo en Cuadratura). Por ejemplo tales filtros pueden elegirse como:

$$\begin{cases} H_0(\omega) = H(\omega) \\ H_1(\omega) = H(\omega + \pi) \end{cases}$$

Fórmula 3-13: caracterización típica de un banco QMF

Donde $H_0(\omega)$ es un filtro pasa bajos. De esta relación se percibe el porqué de la denominación QMF (basta con hacer las gráficas). Las ventajas de usar este tipo de filtros se relacionan con hacer pequeñas las distorsiones de fase o amplitud introducidas en este esquema [WESTERINK/89][GERSHO/92].

Extensión a 2 dimensiones.

Hasta aquí todo es muy bonito, pero en solo una dimensión. Una forma natural (y sencilla) de realizar la extensión es considerando un banco de filtros separables []:

$$H_{ij}(\omega_0, \omega_1) = H_i(\omega_0) \cdot H_j(\omega_1)$$
$$F_{ij}(\omega_0, \omega_1) = F_i(\omega_0) \cdot F_j(\omega_1)$$

Fórmula 3-14: Filtros expresados en forma separable

La señal (una imagen, por ejemplo), se divide en cuatro sub-bandas. Las mismas consideraciones sobre sub-muestreo son tenidas en cuenta aquí. Gráficamente:

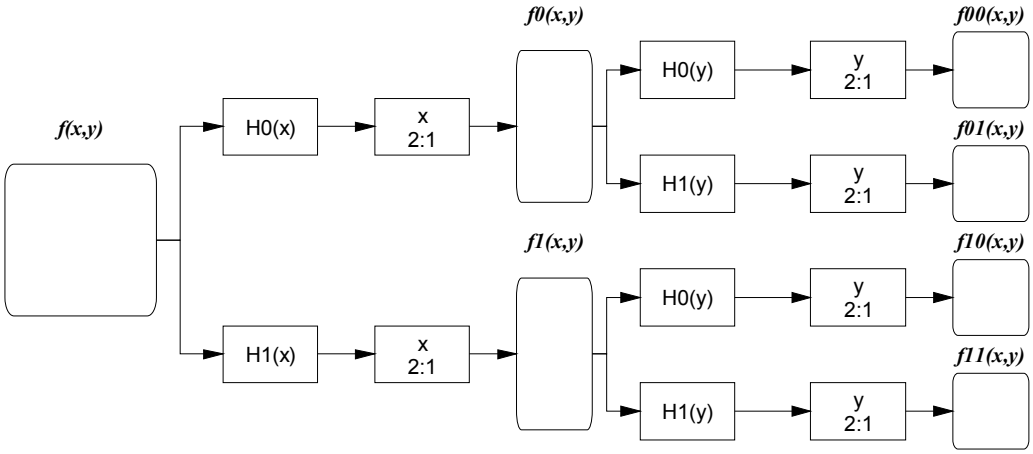


Figura 3-20: El esquema de descomposición sub-banda.

Se considera cada una de las cuatro imágenes producto de la aplicación de los cuatro filtros H_{00}, H_{01}, H_{10}, H_{11}, respectivamente. Los filtros se aplican alternativamente sobre filas y luego columnas.

Codificación y Decodificación

Eligiendo que 'Codec' se utilizara, y habiendo decidido los filtros pertinentes, la aplicación es directa siguiendo la Figura 3 -20.

Conclusiones

Para sacar provecho del método, el objetivo es encontrar filtros que introduzcan la mínima cantidad de error (o mejor aun: reconstrucción perfecta) junto con un esquema de codificación que reduzca en gran medida la información necesaria a transmitir. Elegir un cuantizador, por ejemplo, y analizar el impacto en el filtrado (y viceversa), son asuntos claves en el diseño.

Trabajos mencionados en [WESTERINK/89] sugieren que la cuantización vectorial de los datos (con Lloyd-Max) dan muy buenos resultados.

Transformadas una vez más: Wavelets.

Introducción.

Existe numerosa bibliografía para brindar una rigurosa introducción [HILTON/94][MALLAT/89] [STRANG/89] a las Wavelets, Ondeletas, u 'Onditas' -en el más puro español-. Dado que el objetivo final de esta tesis es implementar un compresor basado en wavelets se dará una introducción un poco más detallada al asunto.

Disgresión

La idea detrás de la transformada de una función es la de expresarla como una suma ponderada de bloques base (funciones base). Trabajando en el espacio discreto:

$$f(x) = \sum_i c_i \cdot b_i(x)$$

Fórmula 3-15: una función como combinación lineal.

En general es 'agradable' trabajar con funciones base ortogonales (o mejor aun: ortonormales), por lo que el proceso de transformación puede verse como la proyección ortogonal de la función sobre el espacio generado por las bases. Estando de acuerdo sobre que bases está uno trabajando, los coeficientes obtenidos poseen toda la información de la función. En este es fácil ver que

$$c_i = \langle f, b_i \rangle$$

Fórmula 3-16.

donde <,> denota producto interno.

El representante más sencillo de este esquema usa como bases la función delta desplazada, con lo que

$$f(x) = \sum_i c_i \cdot \delta(x - i)$$
$$c_i = f(i)$$

Fórmula 3-17.

Los coeficientes dan solo información sobre como se comporta la función al variar el tiempo (x), esto se denomina estar ***localizado en el tiempo***.

Un caso extremo es el de elegir como función base a sinusoides, esto es, la representación Fourier de la función. Aquí se obtiene información sobre el comportamiento de la función en el espacio de frecuencias, lo que se denomina estar ***localizado en las frecuencias***.

Hacen su aparición : Wavelets.

En el punto anterior se vio que la función delta no provee información sobre el comportamiento en el espacio de frecuencias, porque su soporte es infinitamente pequeño. Contrariamente a esto, las sinusoides no dan información sobre un instante particular en el tiempo, ya que su soporte es infinito.

Para los fines de la compresión de señales (donde los sucesos de baja frecuencia son 'globales' y los sucesos de alta frecuencia están 'localizados' en un intervalo corto de tiempo), se busca algo entre dos mundos: un conjunto de bases de soporte finito, de diferentes anchos. Las funciones 'anchas' examinan con cierta globalidad la señal y analizan con precisión la información sobre bajas frecuencias. Por otra parte, las bases 'cortas' examinan intervalos pequeños de la señal, dando información precisa sobre detalles localizados en el tiempo.

Funciones base que tienen esas características pueden construirse a partir de una función base ψ llamada wavelet madre -dicha función además cumple ciertas propiedades 'agradables' de soporte, decaimiento, etc. [MALLAT/89]). Dicha función es dilatada y trasladada para obtener las funciones base deseadas $\{\psi_i\}$, las wavelets.

La descomposición wavelet de una función f() viene dada por:

$$f(x) = \sum_{v}\sum_{k} c_{vk}\,\psi_{vk}(x)$$

$$\psi_{vk}(x) = 2^{\frac{v}{2}}\,\psi(2^{v}x - k)$$

Fórmula 3-18.

donde el factor $2^{v/2}$ es necesario para hacer la base ortonormal. Se ve también que la función madre es dilatada en factores de 2.

Puede verse en [MALLAT/89][MALLAT/87] que la transformada wavelet es pensada e implementada como la aplicación de un par de filtros, haciéndolo en cierta medida equivalente a la codificación sub-banda [WESTERINK/89][GERSHO/92].

El método.

En sí mismo la transformada no realiza ninguna compresión de datos. La verdadera ganancia viene de saber aprovechar los valores decorrelacionados de los coeficientes, producto de la transformación.

Viendo las sub-bandas puede hasta verse que la magnitud de los coeficientes es bastante más grande que la de los datos originales, y por otra parte otras sub-bandas poseen regiones muy cercanas a cero. Siendo la transformada wavelet una que conserva la energía de la señal, pequeñas modificaciones en los coeficientes se traducirán como pequeñas variaciones en la señal, por lo que las regiones cercanas a cero pueden aumentar su tamaño y sacar entonces provecho de un esquema de *run-length-coding*.

Es muy importante el diseño de un cuantizador [] que introduzca la menor distorsión posible en las sub-bandas donde el detalle es importante, y que no se atan cuidadoso en las zonas casi nulas. Técnicas para esto van desde la simple cuantización uniforme, cuantización vectorial, quad-trees, etc.

Para la codificación de los coeficientes distintos de cero, codificadores de entropía como Huffman [HUFFMAN/52][NELSON/92][] son ampliamente usados.

La extensión del método a dos dimensiones es muy similar, sino idéntica, a lo visto en el apartado sobre codificación sub-banda.

Codificación.

La compresión se realiza aplicando primero la transformada wavelet, luego cuantizando los coeficientes y después codificando los valores cuantizados:

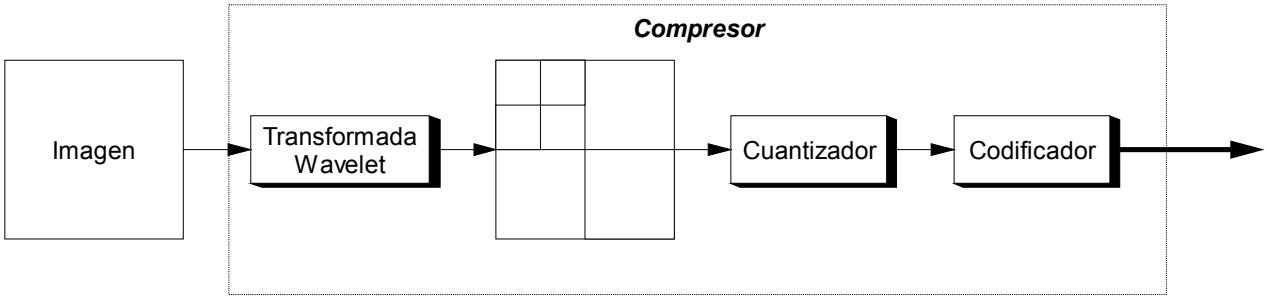


Figura 3-21: Esquema del codificador wavelet.

Mayor compresión se obtiene aplicando la transformada a cada una de las sub-bandas obtenidas de un paso anterior.

Decodificación.

La reconstrucción de la imagen se logra invirtiendo los pasos descriptos arriba:

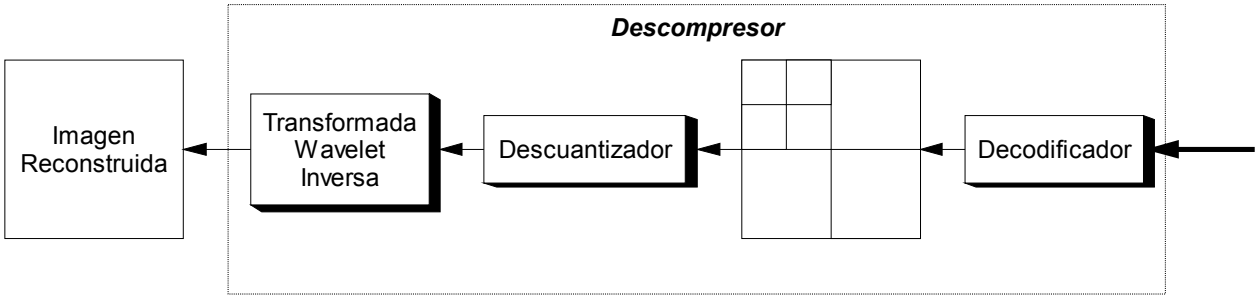


Figura 3-22: Esquema del decodificador wavelet.

Resultados.

Los resultados obtenidos con este esquema de compresión son iguales, sino mejores que aquellos logrados con JPEG [MITCHEL/93], debido al importante progreso en la investigación en el campo de wavelets.

Puntos claves para el diseño de un compresor eficiente (bajo cierto criterio), están en la elección de los filtros (o wavelets) para cierta familia de imágenes y el método de codificación que mayor provecho saque de las estadísticas de los datos cuantizados.

Mas adelante [], siguiendo el objetivo de esta tesis, se realiza una implementación del standard para compresión de huellas digitales [] [BRADLEY/93] empleado por el FBI, y se lo compara experimentalmente con JPEG.

Las huellas digitales.

Definimos a las huellas digitales como la impresión o reproducción física de los dibujos formados por las crestas papilares de las yemas de los dedos de las manos. Aunque se sabe que el hombre de Aurignac 'firmaba' sus dibujos con la imagen de su mano en la prehistoria, y algunos documentos oficiales eran firmados en la China del siglo VIII con las manchas de uno o más dedos, fue recién en el siglo XVII (llamando la atención de Marcelo Malpighi 1628-1694) que las huellas digitales fueros estudiadas de manera científica. Muchos trabajos e investigaciones pasaron [COBOS/78] hasta que en el transcurso de los años 1888-1891, sir Francis Galton (1822-1911) llega a la conclusión, debido a abrumadora evidencia experimental, que:

1- Las huellas digitales no varían en el tiempo, y
2- Las huellas digitales son únicas a cada individuo.

A partir de estas hipótesis, propone lo que podría considerarse el primer método de clasificación de huellas digitales.

Figura 4-1: Una huella digital.

Elementos de una huella digital

Las minucias

Dada la propiedad particular de que una huella digital pueda asociarse unívocamente a una persona, las mismas se han usado con fines de identificación, con gran fuerza hasta nuestros días.

Dos huellas digitales se diferencian a partir de sus ***minucias***, elementos o puntos característicos. Estas minucias se manifiestan como diseños particulares que aparecen en las ***crestas*** (las líneas) de la impresión dactilar. Cada una de estas minucias puede, a su vez, ser un tipo especifico, dependiendo del comportamiento de las líneas que conforman la huella. Una lista no exhaustiva de las mismas es: terminación, bifurcación, empalme, ojal, vuelta, secante, punto (isla), etc.

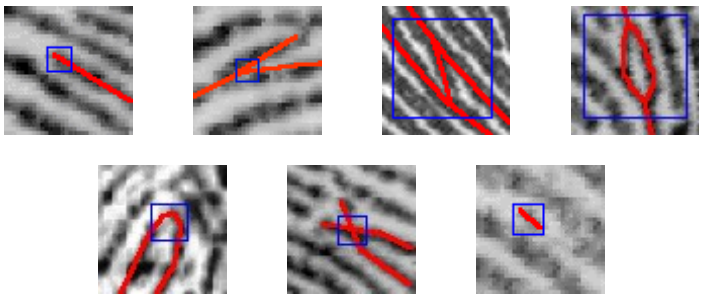


Figura 4-2: Algunas minucias…de izquierda a derecha: terminación, bifurcación, empalme, ojal, vuelta, secante, y punto (isla)

Existen también minucias especiales, que son consideradas de gran importancia para la tarea de clasificación de la huella digital, así como para la medición de la ubicación de las demás minucias. Estas son el *corazón (o core)* y el (los) *delta(s)*. El corazón es informalmente aquel punto que marca el centro de la impresión dactilar, y puede reconocerse la mayoría de las veces como el centro de una especie blanco de tiro. El delta está definido por una convergencia de crestas que recuerda a la letra griega Δ.

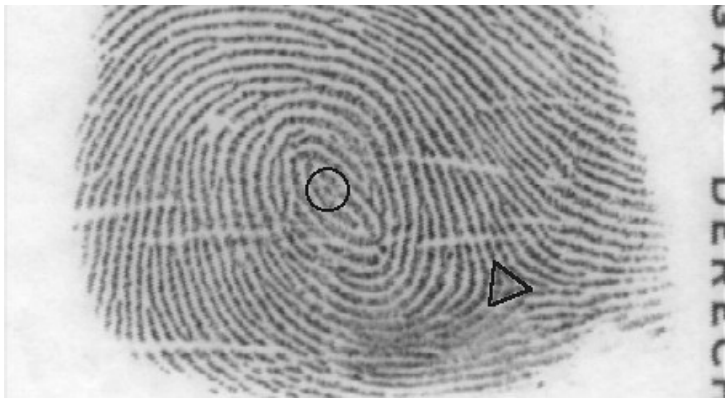


Figura 4-3: El corazón (circulo) y el delta (triángulo) de la Figura 4 -1

A partir de estos puntos importantes es posible listar a las minucias como puntos caracterizados por un tipo, un ángulo y una distancia. Opcionalmente también se agrega orientación que posee, esto es, el ángulo de la cresta a la cual pertenece (respecto a la horizontal).

Algo de clase.

Otro elemento que caracteriza a una impresión dactilar es lo que se denomina su ***clase***. La clase está relacionada con las características globales de la huella digital, a diferencia de las minucias, que señalan fenómenos locales. Existen cuatro tipos fundamentales: arco, presilla interna, presilla externa y verticilo. A su vez, y dependiendo del sistema de clasificación [COBOS/78] gran cantidad de sub-tipos. Estos sub-tipos son las base de sistemas de clasificación como el sistema Henry y el sistema Vucetich .

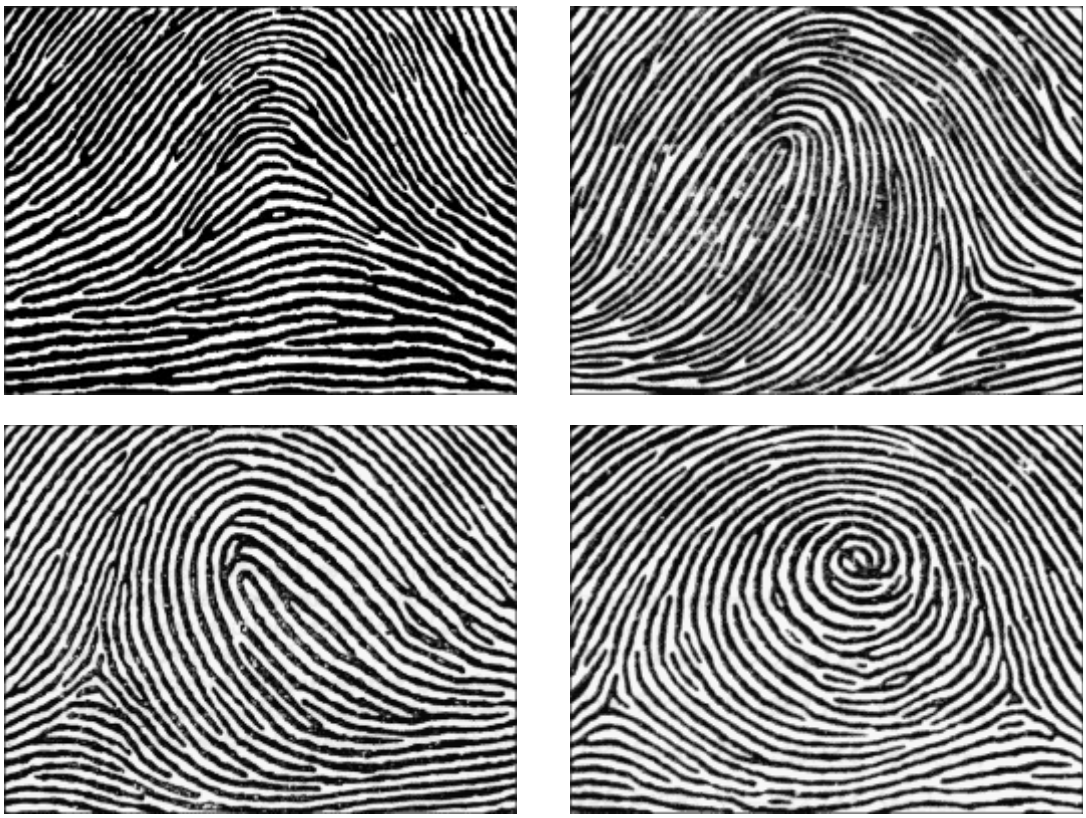


Figura 4-4: Los cuatro tipos básicos de clases. De izquierda a derecha - arriba a abajo: Arco, Presilla Interna, Presilla Externa y Verticilo.

Otra métrica que pueden extraerse de una huella digital, es la cantidad de crestas que cruzan a la línea imaginaria que une dos minucias, información que se considera valiosa para el proceso de identificación.

La Calidad de una Huella Digital.

Para que una huella digital sea útil, deben poder distinguirse en ella los elementos que la caracterizan como un objeto único. Esto es, deben poder detectarse las crestas y minucias con claridad. Esta calidad está directamente relacionada con la forma y la técnica de captura de la impresión dactilar. Estas técnicas se dividen en dos grandes grupos:

a) Técnicas que involucran sustancias químicas

Este método generalmente funciona de la siguiente manera: una sustancia ***A*** es aplicada sobre el dedo del cual se quiere obtener la impresión, luego el dígito es apoyado sobre una superficie ***B*** que reacciona a la sustancia ***A***, logrando que aparezca en ésta la huella digital. Ejemplos de esto es el tradicional método de entintar un dedo y apoyarlo sobre un papel. Otros métodos utilizan sustancias que no manchan la piel, pero originan una reacción en el papel especial sobre el cual los dedos son apoyados.

Un caso especial es el de la obtención de huellas ***latentes*** sobre objetos, como las que uno deja al tomar una copa de cristal. Estas pueden ser capturadas mediante polvos químicos que se adhieren a la huella digital (que está constituida por las secreciones naturales de la piel en la palma de la mano), luego ésta es recuperada mediante una película adhesiva o mediante iluminación especial y fotografía.

b) Otras (por lo general involucran dispositivos de captura electrónicos)

Estas otras técnicas obtienen la huella dactilar por métodos no químicos, por lo general ópticos, en su mayoría involucrando dispositivos electrónicos de algún tipo. El más popular de éstos el método basado en el principio óptico de la ***reflexión interna total*** [HALLIDAY/92]. El dispositivo está compuesto por un prisma , iluminación adecuada y una forma de capturar una imagen (generalmente una micro-camara de TV) . Básicamente la imagen se obtiene apoyando el dedo del cual se quiere obtener la huella digital sobre la superficie plana de un prisma. Dadas las características del prisma, solo las crestas que son apoyadas sobre el mismo son visibles desde el otro lado del prisma, que es lo que la cámara captura.

Para todas estas técnicas existen básicamente tres formas de obtener una huella digital, de la cual se obtienen:

a) Huellas Roladas (o Volteadas)

Esta forma consiste en rodar el dedo sobre la superficie de captura, esto da origen a imágenes como las que se encuentran en [cd-rom2]. Este tipo de huella es requerido por casi todos los organismos gubernamentales pues incluye una gran superficie de imagen, donde pueden observarse gran cantidad de minucias. Tiene la desventaja que es propenso a producir una mala calidad de imagen si el dedo no es 'rolado' en forma apropiada, por ejemplo: borroneo.

b) Huellas Pulsadas (o Planas)

Las huellas digitales pulsadas son obtenidas simplemente mediante la acción de apoyar en forma plana, el dedo sobre la superficie de captura. Así se obtienen generalmente imágenes de mejor calidad que (a), pero puede llegar a perder minucias importantes como los deltas de algunos dedos.

Figura 4-5: Huella pulsada (en un dispositivo basado en reflexión total)

c) Huellas Latentes

Estas huellas son las obtenidas sin la cooperación del dueño de la huella dactilar, generalmente en la escena de algún crimen, o como requerimiento de alguna pericia criminalística. Generalmente la imagen obtenida es

una porción de una huella digital y no una imagen completa. Dependiendo también de la superficie de donde se obtuvo, la misma suele estar sujeta a deformaciones geométricas no lineales (o en simple castellano, estiramientos, torceduras, etc.).

Clasificación, Identificación y Verificación.

Dada una huella digital y sus características, uno puede realizar con ella las siguientes tareas:

a) Clasificar.

Siendo una de las primeras actividades desarrolladas, fundamentalmente para el ámbito policial, la clasificación de las impresiones dactilares se basa en las *clases* en las cuales una huella digital es catalogada []. Estos esquemas de clasificación proponen un método de organizar las huellas digitales de una persona de tal forma que su búsqueda y recuperación puedan ser hechas de forma humanamente manejable. Cada registro o ficha está constituida por la información de uno o más (generalmente los 10) de los dedos de una persona. Según el tipo y sub-tipo de cada dedo, a cada ficha le corresponderá un color y una clave alfanumérica.

Dados, entonces, los datos dactiloscópicos de cierto individuo la clasificación ayuda a reducir en gran medida el volumen de registros o fichas que deben analizarse a fin de encontrar si existe una ficha similar (de la misma clase o lo suficientemente cercana) en el conjunto total de registros (archivo). Esto ayuda grandemente en el proceso de *identificación.*

b) Verificar.

La verificación responde a la pregunta: Cuando dos huellas digitales son iguales? Esta tarea es tradicionalmente llevada a cabo mediante la localización, inspección y comparación de las minucias (y/u otras métricas, como ser la cuenta de crestas) de las dos impresiones digitales que se están comparando. En lo que a técnicas criminalísticas se refiere, si dos impresiones dactilares coinciden en cierta cantidad de minucias (por ejemplo, la Policía de la Provincia de Bs. As. Requiere por lo menos 17), éstas se consideran iguales. Este proceso se torna complejo cuando no se posee una huella digital completa (latente) y se la desea comparar con una huella digital completa previamente registrada.

c) Identificar.

La identificación responde a la pregunta: A quien pertenece una huella digital dada? Esta tarea involucra por lo general a las dos anteriores aplicadas sobre un archivo con gran cantidad de registros (o fichas, históricamente hablando).

El factor no humano.

Desde la aparición de las computadoras, las tareas descriptas en [] han sido, de alguna u otra manera, sujetas a la automatización. Pero dadas las consecuencias que los resultados de dichos algoritmos pueden producir la detención y condena de personas, es siempre deseable (sino obligatorio) que la decisión final siempre este sujeta a la inspección y aprobación de un ser humano calificado (perito dactilóscopo).

La extracción de características.

La extracción de características en forma automática (mediante una computadora) de una huella digital es una disciplina en si misma, que enfrenta por lo general un dilema: qué características extraer. Este comentario (a riesgo de parecer circular) es cierto, es posible extraer las minucias con las cuales se trabaja normalmente o, sino, trabajar con toda una colección de métricas no percibibles por los sentidos humanos, tal vez como medidas invariantes [HU/62] provenientes de la teoría de procesamiento de imágenes, entre otras. Se comentará brevemente la primera alternativa, ya que la segunda posee la desventaja de que la inspección humana del proceso no es concluyente.

El método tradicional de extracción de minucias trabaja de la siguiente forma: dada una imagen de una huella digital primero se la binariza, es decir se reduce la cantidad de colores a 2, por ejemplo blanco para el fondo y negro

para las crestas. Segundo, se aplican algoritmos de 'adelgazamiento' de líneas, hasta lograr que las crestas tengan 1 pixel de ancho; y tercero de este 'esqueleto' se localizan y extraen un subconjunto de las minucias, específicamente sólo las terminaciones y las bifurcaciones (con sus correspondientes ángulos de orientación). Un ejemplo de esto puede verse en [JAIN/97]. Luego de la segunda parte pueden también aplicarse algoritmos para detectar minucias especiales como el corazón y el (los) delta(s).

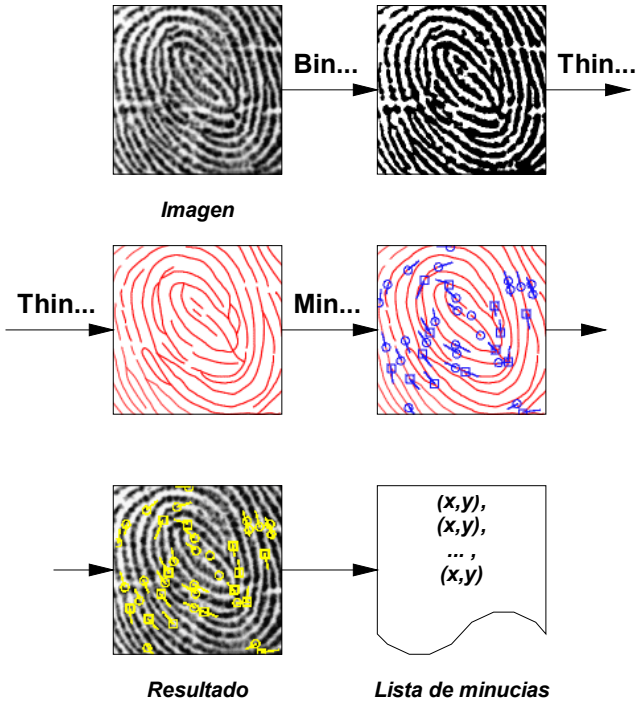


Figura 4-6: Esquema del proceso de extracción de características mediante una computadora. Dada una imagen se binariza, adelgaza (thinning), se detectan las minucias, y luego se construye una lista de puntos.

Dada una lista de minucias, ésta se puede representar básicamente de dos maneras: como un conjunto de puntos tomando como origen de coordenadas polares a un punto central (generalmente el corazón) o como un grafo planar (sin ejes que se crucen) donde cada eje tiene asociado como peso la cantidad de crestas que atraviesa. La primera de éstas representaciones es la más directa pero adolece de la desventaja de ser sensible a las rotaciones de la imagen (aunque existen maneras de mitigar este problema, como utilizar también otro punto para definir un eje base). La representación mediante un grafo es insensible a rotaciones pero es una estructura de datos que requiere más cuidado en su manejo y construcción.

La verificación automática.

Lo Suficiente para la Maquina, Lo Necesario para el Hombre.

Un método de verificación automático decide, dados dos conjuntos de minucias, si éstos provinieron de la misma huella digital o no, asignando (generalmente) un puntaje a la decisión que da cierto nivel de confianza al resultado. Como se expresó en el comienzo de [] una decisión como ésta deberá estar de acuerdo con un ser humano. Lo interesante es que el proceso de verificación ya no opera con una imagen, sino con un conjunto de puntos; mientras que el perito humano necesita la imagen para tomar su decisión. Es entonces deseable que siempre se disponga del par *(imagen, lista de minucias)* como objeto de referencia.

La verificación.

La verificación (o *matching*) automática de huellas digitales es abordada de diversas formas [BALDI/93] [ISENOR/86][JAIN/97], siendo la más sencilla la inspección y comparación una a una de las minucias presentadas en dos listas. El problema se torna complejo cuando una de las listas representa la información de una huella latente, en ese caso es justificable el representar las minucias como un grafo y buscar isomorfismos en subgrafos, aunque es bien sabido que este es un problema de complejidad NP completo [GAREY/83].

Otras técnicas de verificación de huellas digitales se apartan del modelo de extraer las minucias tradicionales (terminaciones y bifurcaciones) y utilizan métodos tales como correlación, y componentes principales [FIELDING/91].

Sistemas Automáticos de Identificación de Huellas Digitales (AFIS)

El primer objetivo de adaptar la tecnología informática a las huellas digitales fue el de automatizar, acelerar y hacer más eficiente el ya existente método de recuperación de fichas dactilares de los grandes archivos en departamentos policiales. El resultado de este esfuerzo son los sistemas conocidos como AFIS *(**A**utomated **F**ingerprint **I**dentification **S**ystem)* [Nec][Printrak]. Estos sistemas de gran envergadura incluyen las tres tareas previamente descriptas en []:

1) *Clasificación*: para acotar las búsquedas en las grandes bases de datos.

2) *Identificación*: a partir de imágenes de huellas digitales pulsadas, roladas o latentes.

3) *Verificación*: procesos generalmente optimizados e implementados en hardware, lo que acelera enormemente los tiempos de búsqueda.

Los sistemas AFIS son valuados hoy en día en millones de dólares debido al equipo (hardware, software, mantenimiento y operación) altamente especializado que involucran. Pero, básicamente, los AFIS siguen haciendo los mismos procesos elementales descriptos en los puntos anteriores [], solo que aplicados a enormes bases de datos y con exigentes requerimientos de velocidad de procesamiento. El ciclo de funcionamiento del AFIS se describe de la siguiente forma:

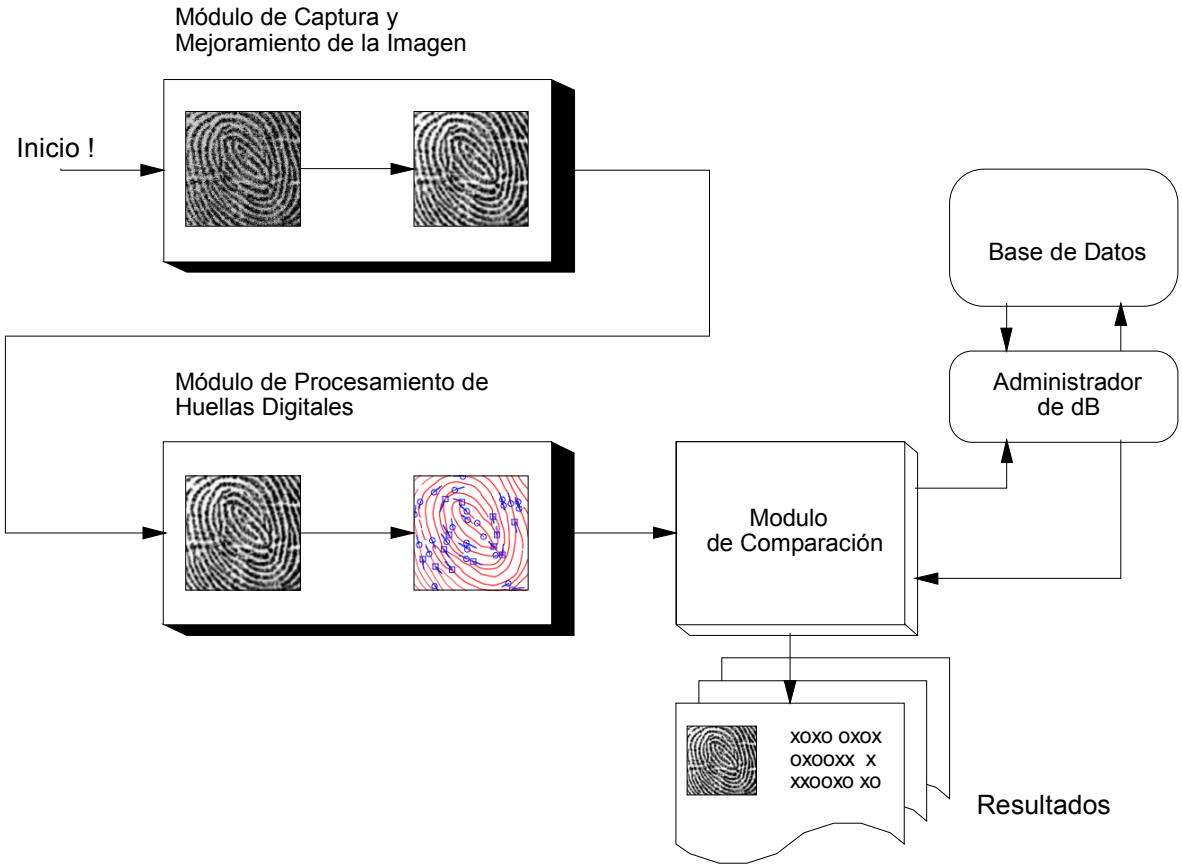


Figura 4-7: Diagrama de funcionamiento de un AFIS.

Primero se obtiene la imagen de la huella digital que se quiere identificar [], luego ésta es procesada para resaltar las crestas del fondo y así lograr luego, una más efectiva extracción de características. Luego de extraer las características, (incluidos datos sobre su clasificación) éstas son alimentadas al modulo comparador que realza una búsqueda sobre la base de datos de las huellas digitales que más se parecen a la cual se desea identificar. El resultado de esta búsqueda es por lo general una lista de impresiones dactilares candidatas. La decisión final es tomada por un operario humano.

Otras Aplicaciones

Hoy las computadoras también son usadas para asistir en el procesamiento, clasificación y verificación de las impresiones dactilares, en otros sistemas, además de los AFIS. Ejemplos de éstas son:

- *Bienestar Social:*
  Reducción importante de los fraudes, mediante la identificación y comprobación de las personas antes y durante el otorgamiento de beneficios.

- *Administración de Cárceles:*
  Para el control de entradas y salidas de reos, evitando así la posible confusión de identidad de personas y potenciales métodos de escape (¡alguien entra y otro sale!)

- *Control de Acceso:*
  Una de las aplicaciones más populares de esta tecnología, llevada a cabo con pequeños equipos de verificación biométrica [Identix][Startek] que controlan accesos a lugares físicos (como bóvedas bancarias) o lógicos (como el acceso a un cajero automático -ATM-).

- *Control de Asistencia/Presentismo:*
  Otra de las aplicaciones de mayor popularidad, donde la verificación de huellas digitales reemplaza al sistema de tarjetas perforadas para el control de horarios de entrada y salida de los empleados de cierto edificio. Tal sistema elimina la posibilidad de 'préstamo' de tarjetas para fraguar presentismo.

Es interesante destacar, que los sistemas de seguridad basados en la comparación de imágenes de huellas digitales, no son inviolables: existe la tecnología para reproducir una huella digital mediante látex. Esto motiva por lo general sistemas donde se combinan elementos extra, para asegurarse que no se intenta engañar al sistema. Aun así, estos sistemas siguen ofreciendo una alta seguridad frente a los sistemas convencionales.

El problema del almacenamiento: por qué comprimir.

Se ha visto [] la necesidad que las aplicaciones de identificación (como por ej. los AFIS) deban almacenar las imágenes de las huellas digitales.

Hoy día, también, las huellas digitales son necesarias para numerosos tramites documentales (por ejemplo, documento de identidad, pasaporte, registro de armas, prontuarios policiales, etc.). Esto genera volúmenes de archivos en papel de enorme tamaño. A este problema se agrega el natural deterioro de un medio como el papel, que es muchas veces víctima de factores como la humedad, suciedad, temperatura y (porque no mencionarlos) roedores. El almacenamiento digital de dicha información salva la mayoría de esos inconvenientes, brindando una colección de imágenes perdurable en el tiempo y factible de ser usada (transmitida , recibida, procesada) con mejor performance que antes.

Existe todavía el problema del espacio de almacenamiento, por ejemplo: en la República Argentina una ficha dactilar común y corriente posee las impresiones roladas de los diez dígitos de las manos. Cada una de éstas está en una casilla de unos 4x4 centímetros, digitalizadas a 500 dpi (la resolución mínima que es mundialmente utilizada para este tipo de imágenes) da como resultado una imagen de 775x775 pixeles. Esto es, una imagen que, a 256 tonos de gris por pixel, tiene un tamaño bruto de aproximadamente 586.5Kilobytes. Entonces diez de están ocupan 5.7 Megabytes, y unos veinte millones de fichas (con diez impresiones cada una) ocupan 109.2 Terabytes (sin contar información textual extra en las fichas...).

En la actualidad, un dispositivo óptico de almacenamiento masivo típico [Storage] tiene una capacidad de unos 5 Tb por unidad. Es entonces cuando comprimir se convierte en una alternativa interesante tanto como para reducir el impacto en el medio físico de almacenamiento, como en los costos de los sistemas.

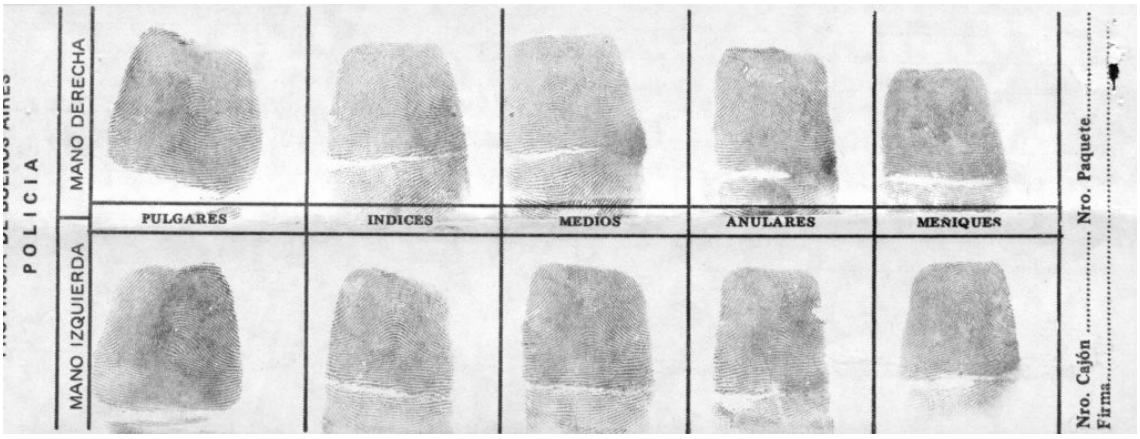


Figura 4-8: Ejemplo de una ficha dactilar utilizada por la Policía de la Pcia. de Bs. As. (Cortesía Policía de la Pcia. De Bs. As.)

El problema de comprimir: hasta cuando.

Cuando se trabaja con métodos de compresión con pérdida de información se está sacrificando exactitud en la información almacenada a favor de mayores tasas de compresión. En el área de las huellas digitales es importante que dichas imágenes conserven las características que las hacen útiles: las minucias. Por ejemplo, las minucias más usadas tanto en sistemas automáticos (como las AFIS) como en manuales (realizados por un perito dactilóscopo) son las bifurcaciones y las terminaciones. Lamentablemente ésta no es una medida de error de fácil medición. El perito humano puede reconocer una minucia en imágenes con un nivel de degradación alto, donde un sistema automático falla rotundamente.

La Medida de un Buen Ojo.

Se llevó a cabo el siguiente experimento para determinar el límite a partir del cual una imagen de una huella digital se degrada más allá de un uso práctico. Se tomaron cuatro imágenes representativas, esto es, imágenes cuyas relaciones error vs. tasa de compresión estuviesen equitativamente distribuidas sobre el espacio de todas las relaciones [Figura 7 -1]. Para cada una de éstas un sujeto entrenado, con la experiencia de decidir cuando una minucia es distinguible y cuando no lo es, observa una serie de imágenes que representan los puntos de la relación error vs. tasa de compresión. Cuando el observador nota que la imagen se ha degradado tanto que las minucias pierden la definición necesaria como para ser consideradas como tal, el proceso se detiene y se toma nota de los datos del punto inmediatamente anterior al que detuvo al proceso.

Así se llegó a la conclusión que el límite a partir del cual las minucias se degradan tanto, que se tornan inservibles es de 30 +- 2 dB. (utilizando PSNR [] como medida del error).

La importancia de un standard.

Dada la familia de aplicaciones y actividades [] en las cuales las huellas digitales son utilizadas, se torna importante la definición de un estándar para el intercambio de la información. Las ventajas de trabajar con un standard permiten que los usuarios no se preocupen por una marca de hardware y/o software especifico, sino por la adhesión o no al estándar.

El estándar de compresión de huellas digitales utilizado por el FBI [BRADLEY/93] logra tasas de compresión del orden de 15:1, con lo que la ganancia en espacio de almacenamiento se torna más que atractiva. Pero existen otros métodos de compresión que logran tasas en el orden de 10:1, y cada uno de ellos propone un standard en

sí mismo (con distintos objetivos de diseño: eficiencia, facilidad de implementación en hardware, decompresión rápida, etc.). Por lo tanto se hace importante el decidir cual de ellos se debe utilizar.

Compresión de huellas digitales sin wavelets.

Como alternativas a la compresión de huellas digitales con wavelets [] (el objeto de esta tesis) existen dos opciones que sobresalen sobre las demás: Compresión Fractal [FISHER/92][BARNSLEY/89], que exhibe tasas de compresión muy altas y JPEG [MITCHEL/93][WALLACE/91], es estándar de compresión de propósito general más usado hoy día. La primera de éstas se basa en explotar la auto similitud de una imagen a distintas escalas, dado que las huellas digitales no poseen dicha propiedad (salvo a una única y característica escala) es de esperarse que este método no sea mejor a wavelets.

La siguiente ilustración muestra los resultados de comprimir la imagen 0007.bmp [cd-rom2] usando el esquema de compresión fractal comercializado por Iterated Systems [Iterated] comparada frente al standard del FBI [BRADLEY/93]. Puede observarse que su desempeño no es bueno, tal como se anticipó.

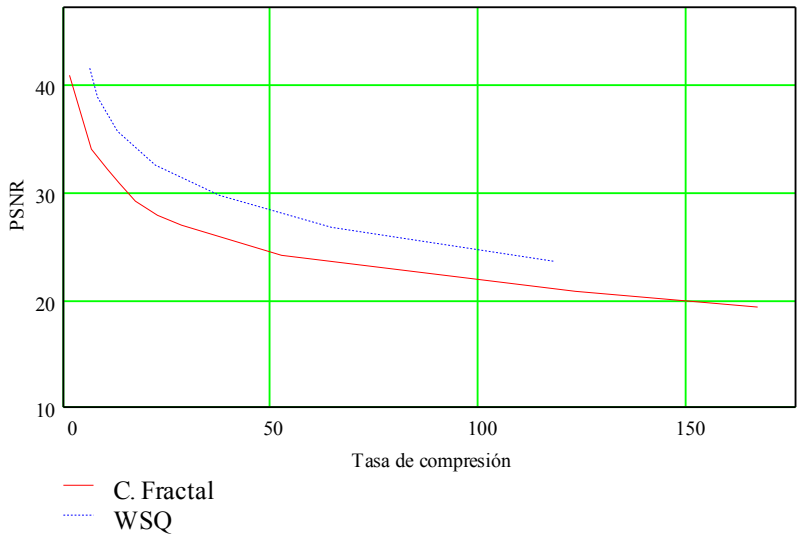


Figura 4-9: Las curvas muestran el desempeño de los distintos métodos de compresión aplicados a la imagen 0007 de [cd-rom2].

Por otro lado, JPEG tiene la ventaja de ser un standard ampliamente difundido para el intercambio de información fotográfica. Pero el método adolece, en su diseño, de un problema que hace que para grandes tasas de compresión los artefactos [] resultantes de la división en ventanas de 8x8 pixeles [WALLACE/91] degraden la imagen más allá de su utilidad.

En la sección [] se realizará una comparación más exhaustiva entre JPEG y el standard de compresión por wavelets.

Ajustando JPEG.

JPEG es El Estándar para la compresión de imágenes de 'tono continuo' (un paisaje, un rostro, etc.) en general. En su diseño, este método fue ideado para tratar cierto tipo de imágenes y ciertas estructuras de datos que gobiernan su comportamiento, establecidas y parametrizadas. El objetivo de este articulo es encontrar una parametrización distinta a la utilizada generalmente.

Esta ultima es la propuesta como ejemplo por el propio estándar, curioso es que sin que se obligue a usarla, sea la más (sino prácticamente la única) utilizada. Esto puede interpretarse que la obtención de estos parámetros no es una actividad trivial.

Con esta nueva parametrización se espera obtener mejores resultados del método frente a una familia de imágenes que se aparta de aquellas para las cuales fue diseñado.

Objetivos.

El principal objetivo de este trabajo es, aunque parezca contradictorio, NO MODIFICAR JPEG, o sea dejar el estándar intacto y trabajar sobre la re-parametrización de algunas de sus estructuras de datos (mas específicamente y como se verá más adelante, la tabla de cuantización). Esto es posible gracias a que existen implementaciones disponibles del método [Ijg].

Es importante que el estándar permanezca como tal, a fines de comparar su potencia frente a otros métodos que en el futuro puedan aparecer. En las siguientes secciones se analizará los componentes que intervienen en el método y qué se puede hacer con ellos para mejorar su desempeño.

Una mirada un poco más en detalle.

En [] se realiza una descripción de cómo trabaja la codificación / decodificación en JPEG, donde se explica que la transformada utilizada es la Transformada Discreta del Coseno (DCT). En un primer vistazo al diagrama en bloques Figura 3 -4 aparecen dos componentes: la Transformada/Antitransformada, y el Codificador/Decodificador. No hay nada que se pueda hacer con el primero de ellos, pues las parametrizaciones posibles allí son cambiar la transformada (NO! el estándar cambiaría) o cambiar el tamaño de la ventana sobre el cual se aplica la transformada (NO! Idem!). Analicemos el segundo componente: el Codificador/Decodificador.

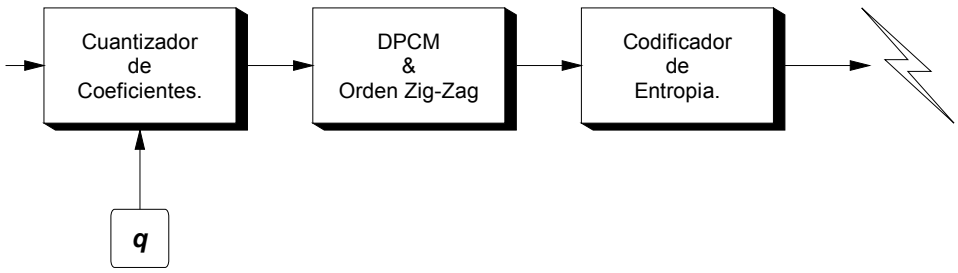


Figura 5-1: Mirando dentro del Codificador

Aquí las dos etapas distintivas son: Cuantización y Codificación de Entropía. El codificador de entropía opera sobre una tabla de símbolos que consiste en coeficientes y corridas de ceros. Estos símbolos son generados por una sub-etapa que explora la ventana en forma diagonal para sacar el mayor provecho posible de los ceros en las zonas de alta frecuencia. Esta etapa previa a la codificación trata a los coeficientes DC de manera distinta, usando DPCM. Se utiliza la codificación Huffman de memoria nula (aunque la codificación aritmética es contemplada en el estándar [WALLACE/91], no es usada por el ya consabido problema de patentes!), donde se puede elegir entre una tabla fija (propuesta por el estándar) o generar una en tiempo de ejecución, con características óptimas.

La cuantización ocurre antes de todo esto y opera mediante un conjunto de cuantizadores uniformes de paso $Q_i(q)$: $q \rightarrow [1...255]$ (uno por cada elemento de la ventana transformada) el parámetro q es un entero en el rango

(0..100]. Mientras más ancho el paso del cuantizador, más información se pierde, más se comprime la imagen, más ruido se introduce a la misma. ***Cualquier ajuste de parámetros que debiéramos hacer debe ocurrir aquí.***

Una nota de Color.

Nada se ha dicho sobre si la imagen es monocromática o color, JPEG trata a cada imagen como varias imágenes monocromáticas, cada una de éstas representa una componente del modelo que se elige para representar al color. La implementaron más utilizada convierte cualquier imagen al espacio YCrCb o luminancia/cromiancia (3 componentes), donde la información sobre los colores se encuentra en las dos ultimas bandas. Desde el espacio RGB estas componentes se definen como:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16874 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Fórmula 5-1: Conversión del espacio RGB -> YCrCb

Para imágenes monocromáticas ***R=B=G***, con lo que solo la componente ***Y*** se torna relevante. Existen distintos conjuntos de tablas de cuantización y tablas de Huffman para las componentes de luminancia y crominancia. Como la familia de imágenes que se estudiara es monocromática solo interesará la componente de luminancia, y por lo tanto se trabajará con solo una tabla de cuantización.

La Tabla de Cuantización.

Encontrado el punto sobre el cual se trabajará, se verá con un poco más de detalle. Desde las primeras etapas del proceso se usa una ventana de 8x8 elementos, luego a la salida de la DCT se tiene una matriz de 8x8; luego la matriz de cuantización tendrá ese tamaño. Para el canal de luminancia el estándar propone:

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 5-1: Tabla de cuantización propuesta por el estándar JPEG para la banda de luminancia de una imagen.

Encontrando una nueva tabla de cuantización.

De Muchas Opciones, la Más Codiciosa.

Descripto en [JpegFAQ] como un '*arte negro*', el diseño de tablas de cuantización apropiadas para JPEG no es algo trivial. El principio básico para elegir cada uno de los 64 pasos de cuantización es: elija aquel valor limite, para el cual de aumentarse se observa una "degradación visual notable" en la calidad de la imagen reconstruida.

Esto puede interpretarse como un problema de minimización de una función de costo o error, donde la misma depende de variables tan diversas como las características de las imágenes, el contexto (CRT, iluminación

ambiente, etc.) y distancia en la cual son mostradas, la función de sensibilidad del ojo para determinadas frecuencias [CAPMBELL/68][PERKINS/89], etc. Si todos estos parámetros pueden ser controlados y medidos, se pueden desarrollar procedimientos para la solución de dicho problema [MITCHEL/93], obteniendo tablas de cuantización como las expuestas a modo de ejemplo (y usadas como una ley!) en la especificación del estándar JPEG [MITCHEL/93].

Otra forma de abordar el problema de la minimización es considerar una función de error que dependa de menos factores. Se realizará esta aproximación aquí por considerarla atractiva y posible de realizar con menos recursos que los mencionados en el párrafo anterior.

Básicamente, el método puede considerarse una adaptación al esquema de distribución de bits encontrado en [GERSHO/92] y en [CAPMBELL/68], llamado (y con razón) el ***algoritmo codicioso*** (o de análisis marginal [PERKINS/89]) En forma general el método funciona de la siguiente forma: "*para un número de bits dado, darlos a quienes más lo necesitan*". Bien, ¿quienes los necesitan más?. Pues aquellos que, de no tenerlos, introducirían tanto ruido que producirían una "degradación notable" en la imagen reconstruida (visto de esta forma el método también puede ser visto como el ***algoritmo extorsivo***!).

Una nueva variante de este algoritmo es presentada aquí. Consiste en distribuir niveles de cuantización en vez de bits de resolución, esperando de esta manera obtener una respuesta más realista de las necesidades de cada sub-banda. El enfoque tradicional, al distribuir cantidades enteras de bits de resolución, incrementa la cantidad de niveles de decisión del cuantizador en un factor de 2, como se observa en el siguiente gráfico:

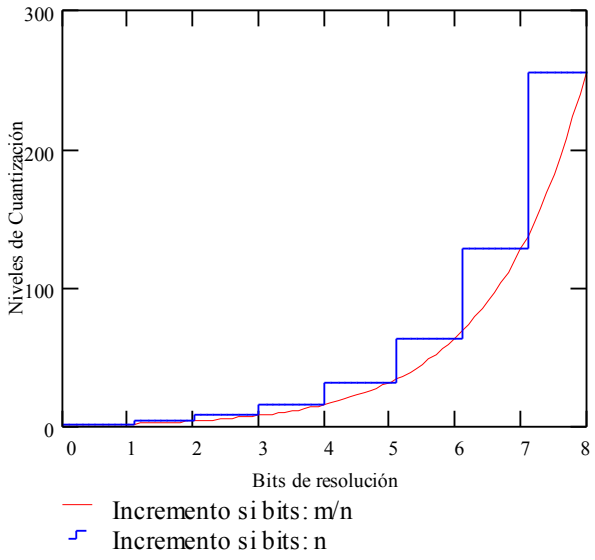


Figura 5-2: Incremento de los niveles de un cuantizador uniforme para una cantidad entera de bits de resolución y para una cantidad fraccionaria.

Esto motiva la decisión de distribuir niveles de cuantización, lo que conlleva indirectamente a la asignación fraccionaria de bits de resolución. De esta manera se espera obtener mejores soluciones a aquellas obtenibles de utilizar cantidades enteras de bits como restricción.

Para poder describir el método en mayor detalle, defínase:

$\boldsymbol{R_i}$: rango de la sub-banda ***i***, (máximo valor aparecido – mínimo valor aparecido).

$\boldsymbol{pdf_i(x)}$: estimador de la función de probabilidad de la sub-banda ***i***.

$p_i(n) = \left[\frac{R_i}{n} \right]$: paso del cuantizador de la sub-banda ***i***, con ***n*** niveles de cuantización.

$q_i(x,n) = \left[\frac{x}{p_i(n)} \right]$: función de cuantización de la sub-banda ***i***, con ***n*** niveles de cuantización,

- $b_i = log_2(n)$ -

donde **[.]** denota a la *parte entera.*

$w_i(n)$: estima el error al cuantizar uniformemente la sub-banda ***i***, con ***n*** niveles de cuantización.

$$w_i(n) = \sum_{x \in R_i} pdf_i(x).(x - q_i(x,n).p_i(n))^2$$

donde |.| denota tamaño y asumiendo que $\sum_{x \in R_i} pdf_i(x) = 1$

W(i, step): estima la distorsión general si se aumenta la cantidad de niveles del cuantizador de la sub-banda ***i*** en ***step***.

$$W(i, step) = w_i(n_i + step) + \sum_{j:0...63,\, j \neq i} w_j(n_j)$$

Luego el algoritmo, en su descripción más simple puede enunciarse como:

```
para todo i
       n_i = valor inicial.

step = 1.

mientras (condición de parada no se cumpla)
       /*cada iteración entrega 1 bit*/

       j=argmax(i: W(i, step))

       si j existe
              n_j  =  n_j + step
              step = 1
       sino

              step = step + 1
       finsi
```

Listado 5-1: Pseudocódigo del algoritmo de distribución de bits.

Al final del proceso, se tiene una matriz de 8x8 con la cantidad de niveles (n_i) que le corresponde a cada sub-banda, la matriz de cuantización estará formada entonces por $q_i(n_i)$.

Los últimos detalles del método son: determinar el valor inicial que se otorga a cada cuantizador, controlar que ***step*** no crezca indefinidamente y determinar la condición de parada.

El valor inicial de niveles de cada cuantizador se calcula de la siguiente manera: dado el rango de la sub-banda ***i*** y el paso del cuantizador propuesto por el estándar [MITCHEL/93] se obtiene la cantidad de niveles que la tabla de cuantización estándar otorgaría al conjunto de datos. Este valor es luego multiplicado por un factor α . Para evitar mínimos locales, al comenzar desde una buena zona inicial probada. O sea:

$$n_i = \alpha . \frac{R_i}{qnorm_i} \qquad ; \qquad \text{donde}$$

n_i : cantidad de niveles del cuantizador ***i***

α: factor entre 0 y 1

R_i : rango de la sub-banda ***i*** (máximo valor – mínimo valor)

$qnorm_i$: paso del cuantizador ***i*** propuesto por el estándar.

La condición de parada se define como la relación entre el error obtenido al cuantizar utilizando la nueva tabla, frente al error cometido utilizando la tabla propuesta por el estándar [MITCHEL/93][WALLACE/91]. Un factor β permite controlar por cuanto se está permitido superar este limite. Esto es:

Error por nueva tabla >= β. Error por tabla estándar

Donde

β: factor entre 1 y 1.5

Para que ***step*** no crezca indefinidamente se establece la heurística que fija que no puede ser mayor que 128.

Recolectando Estadísticas.

Para al cálculo de la distorsión, o error, es necesario conocer la distribución de probabilidad de las bandas. Sin suposiciones sobre si responden o no a tal modelo, la familia de imágenes a la cual se quiere adaptar la tabla de cuantización es analizada con el fin de obtener ciertos datos estadísticos, principalmente su histograma, que será un estimador de la pdf.

El proceso que se encarga de dicha tarea realiza los siguientes pasos:

```
inicializar el histograma H_i en cero, para i:0..63
para cada imagen I
       para cada sub-bloque de 8x8 pixeles. I'
              Calcula  C = DCT (I')
              para cada coeficiente de C_i , i:0..63
                     actualiza el histograma H_i
```

Listado 5-2: Pseudocódigo del proceso de recolección de estadísticas de la DCT.

Se utiliza la definición dada en Fórmula 3 -1, especializada para una matriz de 8x8 elementos. La DCT es entonces de la forma

$$F(u,v) = \frac{1}{4} C(u).C(v).\left[\sum_{x:0...7}\sum_{y:0...7} f(x,y).\cos\frac{(2.x+1).u.\pi}{16}.\cos\frac{(2.y+1).v.\pi}{16}\right]$$

donde

$$C(u): \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0 \\ 1 & \text{sino} \end{cases}$$

Fórmula 5-2.

Dado que la entrada a la transformada es corrida del intervalo [0, 255] hacia [-128, 127], los coeficientes (resultado de la transformada) estarán en el intervalo [-1024, 1016]. Entonces el histograma tendrá 2040 ‘casillas’ que representaran a los coeficientes.

Resultados.

Las Estadísticas.

Los siguientes gráficos muestran los histogramas de las bandas más representativas, de izquierda a derecha y de arriba abajo, la correspondiente a los coeficientes DC (0,0), y tres más correspondientes a las zonas de más alta frecuencia (6,6), y frecuencias medias (3,4) (4,2).

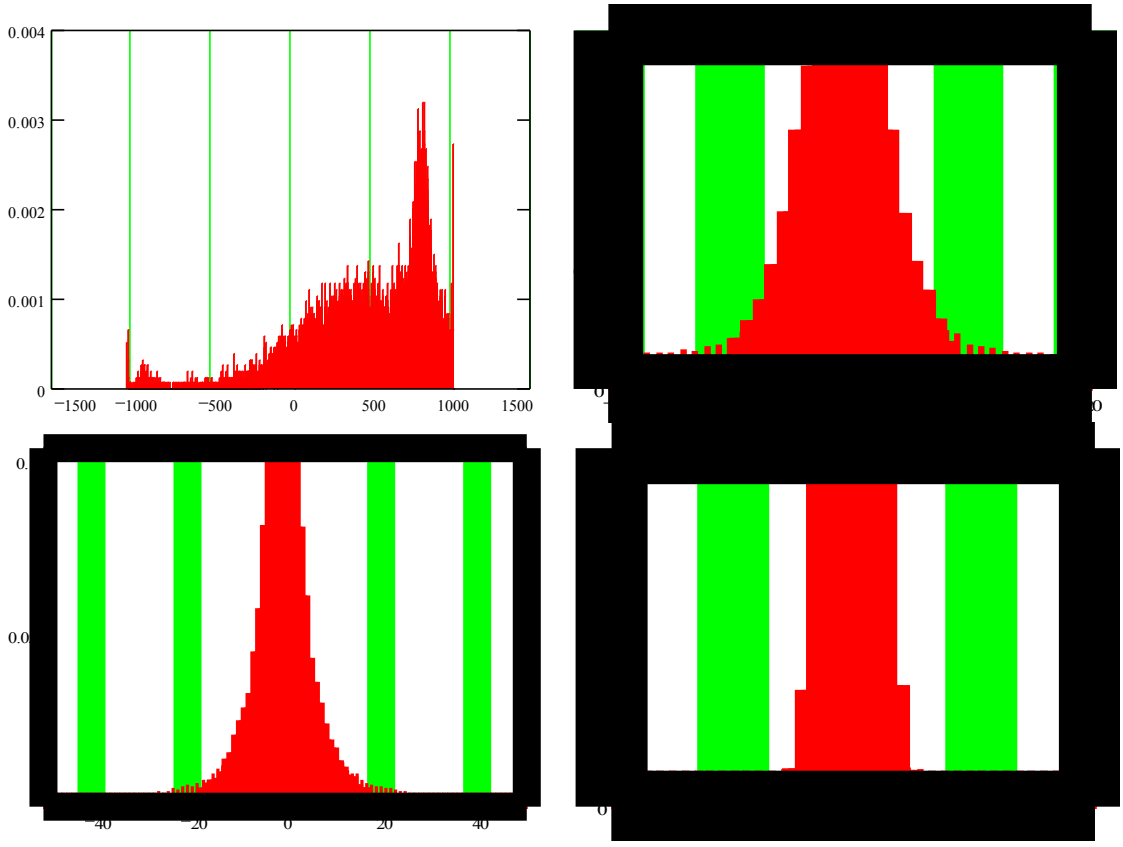


Figura 5-3: Histogramas de algunas bandas, resultantes de la DCT utilizada en JPEG.

Como es de esperarse, el histograma de los coeficientes DC es similar al histograma correspondiente a una huella digital del conjunto, aunque vale la pena recordar que las estadísticas no provienen de una sola imagen, sino de un conjunto. También son razonables los histogramas obtenidos de las bandas de media y altas frecuencias, centradas en valores cercanos a cero y con varianza decreciente al avanzar hacia la zona de más alta frecuencia. Sin embargo, dada su forma, cabe preguntarse si estos coeficientes siguen alguna distribución probabilística dada; de ser así, esto permitiría modelar a las imágenes con mayor precisión.

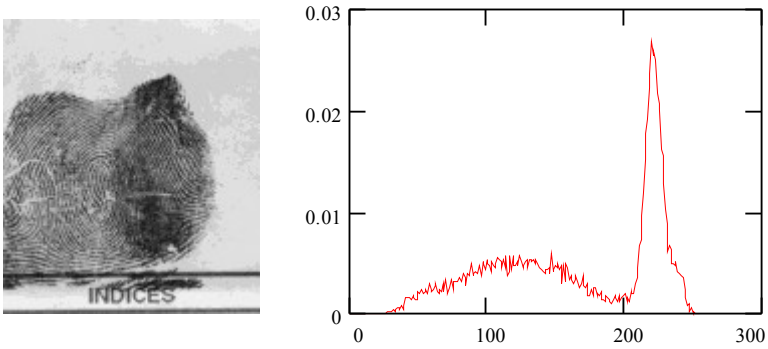


Figura 5-4: Una imagen genérica y su histograma. Notar la semejanza con el histograma de la banda DC (0,0).

Todos los resultados (histogramas) se encuentran en [cd-rom1].

Las Tablas.

Para la ejecución se hicieron corridas con diversos valores de α y β sobre un subconjunto de 10 imágenes de [cd-rom2], y se estableció que los valores que resultaban en tablas que mejoraban el desempeño son $\alpha = 0.6$ y $\beta = 1.0$

Después de ejecutar el algoritmo codicioso la mejor tabla obtenida fue:

Niveles de cuantización por banda

090	104	047	015	006	003	003	002
087	038	021	008	004	002	002	001
034	022	012	005	002	001	001	001
017	010	005	003	001	001	001	001
008	005	003	002	001	001	001	001
005	003	002	001	001	001	001	001
003	002	002	001	001	001	001	001
003	001	001	001	001	001	001	001

Tabla 5-2: Tabla de niveles de cuantización por banda.

Tabla de Cuantización

023	016	014	022	031	045	039	043
017	017	020	026	027	037	033	043
020	019	021	029	039	051	037	033
019	021	030	024	049	034	030	026
024	026	030	030	033	026	020	024
034	038	033	044	027	020	021	019
049	050	035	043	028	017	015	013
052	084	059	031	018	013	012	012

Tabla 5-3: Tabla de cuantización obtenida con el método 'codicioso'

Las Odiosas Comparaciones.

Las siguientes gráficas muestran el grado de mejora en la relación tasa de compresión – error de la nueva tabla frente a la propuesta por el estándar con algunas imágenes del conjunto de prueba.

Se utilizó como medida de calidad de una imagen a la relación señal ruido PSNR [].

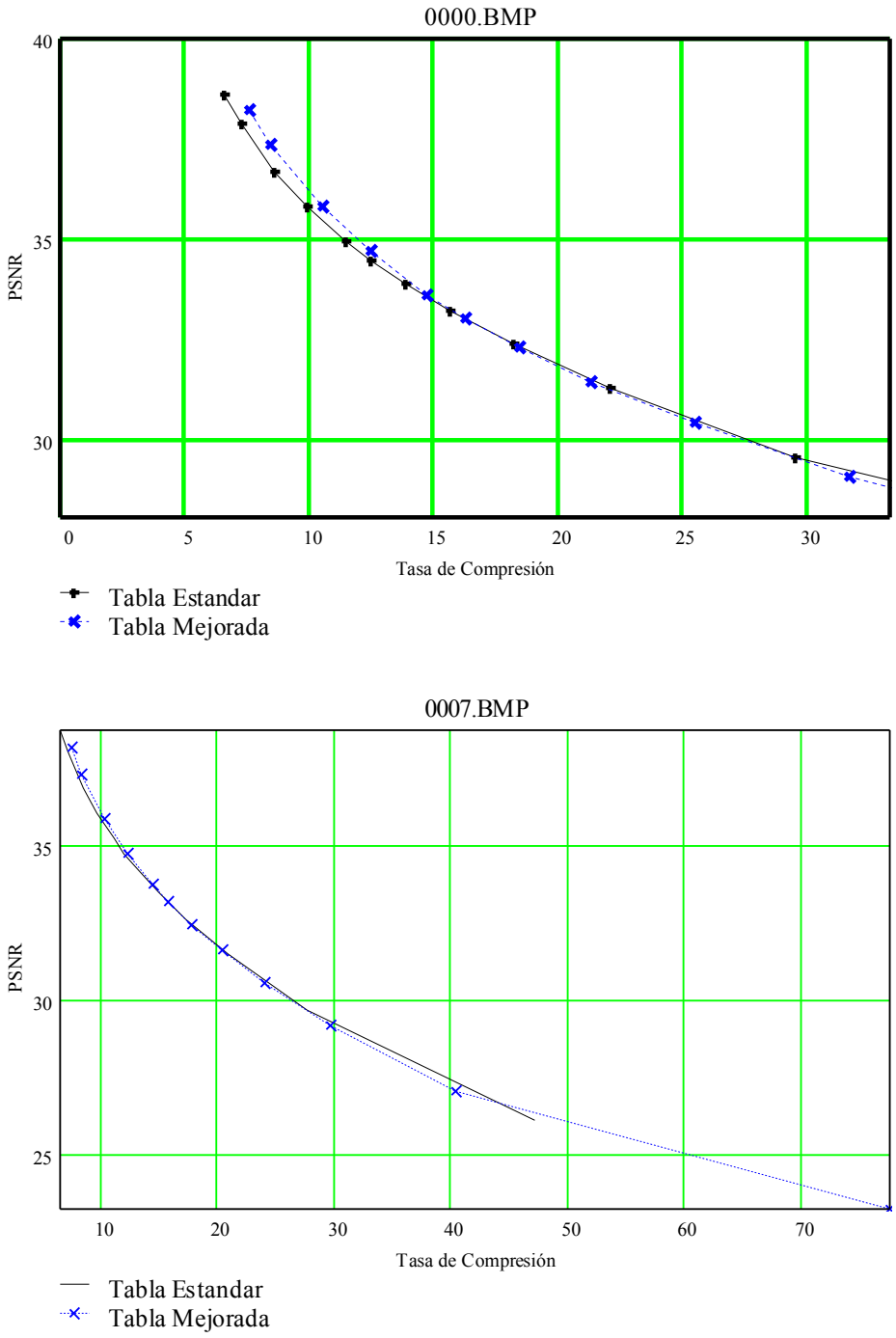


Figura 5-5: Las dos gráficas muestran que tanto se mejora usando la nueva tabla de cuantización.

De la observación de la Figura 5 -5, se ve que la mejora se nota en el rango de calidad aceptables de las imágenes (mayor a 30 dB.). Para valores más bajos las curvas son prácticamente coincidentes. Un detalle interesante de observar, sin embargo, es como el rango de compresión se ve incrementado con la nueva tabla. Esto es para el mismo parámetro de cuantización q, se comprime mucho mas, lo cual se considera como una mejora en el rendimiento.

Una Implementación del Standard WSQ.

WSQ se refiere a Cuantización Escalar por Wavelets, en este documento se describe la implementación de una variante del método adoptado por el FBI [FBISPEC]. Se analizan las características que involucran los procesos de transformación, cuantización y codificación de entropía, no considerando en forma particular el formato de los archivos que el standard establece.

El Método.

El algoritmo WSQ cae dentro de la familia de métodos de compresión de datos con pérdida de información) basados en transformadas. En éstos se distinguen claramente tres etapas:

- ***Aplicar una transformación lineal*** que conserve la energía. Esto decorrelaciona (de alguna manera) los datos y concentra la energía de la imagen en un número menor coeficientes. Al conservar la energía los valores cercanos a cero pueden hacerse cero, con lo que se logran tener matrices dispersas eficientemente codificables con un esquema adecuado (por ej. : zigzag en JPEG), implicando un impacto mínimo en la imagen reconstruida.
- ***Cuantizar los coeficientes obtenidos.*** Esta etapa involucra por lo general al problema de distribución de bits. También es aquí donde se produce la pérdida de información que categoriza a esta familia de métodos.
- ***Codificar la salida del cuantizador***. Aquí se utiliza por lo general un esquema mixto de codificación de entropía (por ej. Huffman) y métodos que saquen provecho de la topología de ceros antes mencionada (Quad-Trees, RLE, etc. [GERSHO/92]).

El Algoritmo.

Tipos de datos básicos:

Se implementa la clase _submatrix, que encapsula las propiedades y funcionalidad de una matriz (y submatriz también!). Su uso principal es el de almacenar la imagen, y los coeficientes de las transformaciones, cuantizaciones, etc. Una submatriz es definida en función de una matriz padre, y no posee espacio propio para sus elementos. Estos existen en la matriz padre (punteros mediante).

La Transformada directa e inversa:

La transformada directa (FDWT) se implementa usando un banco de filtros de fase lineal de dos canales y reconstrucción perfecta. Para tratar los problemas de aplicar los filtros cerca de y en los bordes se realiza una extensión simétrica de la imagen. Esto permite la transformación de imágenes de dimensiones impares [BRADLEY/93]. Esta separación en dos sub-bandas es aplicada tanto a filas como a columnas con lo que se obtiene una descomposición 2-D en cuatro sub-bandas. Esta transformación es aplicada varias veces en cascada a ciertas sub-bandas para obtener así 64 sub-bandas en total. Los motivos de esta elección en la descomposición se mencionan brevemente en [BRADLEY/93].

La transformada inversa (IDWT) sigue el camino de atrás hacia adelante, dadas cuatro sub-bandas, les son aplicadas a éstas el banco de filtros inversos en el orden opuesto al aplicado en la trasformada directa, para el párrafo anterior esto es primero a las columnas y luego a las filas.

(Nota: Parte del código que implementa el kernel de la transformada mencionada aquí fue escrito por Craig Watson del NIST, que a su vez derivo su código del escrito por Tom Hopper del FBI. COPYRIGHT a sus respectivos autores... !).

El banco de filtros es implementado como la estructura (clase) SWTBank que encapsula los filtros a ser usados en las transformadas.

Las transformadas son realizadas básicamente por dos funciones:

```
int split_bands( _submatrix<float> &img, SWTBank &swtb );
```

donde `img` es la imagen a transformar, y `swtb` es el banco de filtros a aplicar.

Esta función aplica la FDWT a las filas de la matriz, el resultado (dos vectores que concatenados tienen la misma dimensión que una fila) es devuelto sobre la misma matriz, gráficamente:

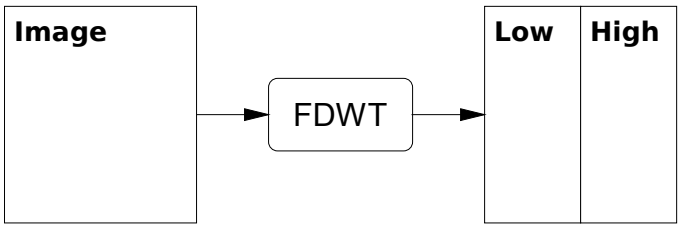


Figura 6-1: como trabaja *split_bands*

```
void join_bands( _submatrix<float> &img, SWTBank &swtb );
```

donde `img` es la imagen que contiene las bandas bajas y altas concatenadas (típicamente, producto de `split_bands`), y `swtb` es el banco de filtros a aplicar.

Esta función aplica la IDWT a las filas de la matriz (o sea a los dos vectores en cada una de ellas), el resultado es devuelto sobre la misma matriz, gráficamente:

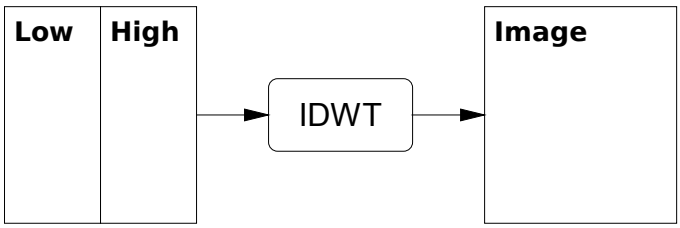


Figura 6-2: como trabaja *join_bands*

La descomposición en sub-bandas:

Dos estructuras de datos soportan esta tarea,

```
_submatrix wtree[21];
```

Este vector de submatrices comanda el orden de aplicación de las transformadas, en otras palabras: que submatriz transformar (antitransformar) y cuando hacerlo (antes de quien?, después de cual?)

```
_submatrix sbtree[64];
```

Este vector de submatrices dirá al final de su construcción, la ubicación de la sub-banda número i-ésima, con i: 0...63.

Las rutinas que inicializan estas estructuras son:

```
  VOID MAKE_W_TREE(_SUBMATRIX<FLOAT> &IMG, _SUBMATRIX<FLOAT> WTREE[21]);
void make_sb_tree(_submatrix<float> &img, _submatrix<float>
sbtree[64]);
```

(Nota: para aquellos programadores preocupados por los requerimientos de memoria, no hay 85 matrices por ahí, sino referencias a UNA sola matriz: la imagen).

La descomposición en sub-bandas puede verse gráficamente en los anexos.

Se respetan los valores de los coeficientes de los filtros propuestos por [BRADLEY/93].

El esquema en cascada que descompone una imagen en 64 sub-bandas y luego la reconstruye a partir de éstas es llevado a cabo por las funciones:

```
int dtransf(_submatrix<float> &img, SWTBank &swtb,
            _submatrix<float> w_tree[21]);

int itransf(_submatrix<float> &img, SWTBank &swtb,
            _submatrix<float> w_tree[21]);
```

Cuantización

Las 64 DWT sub-bandas son cuantizadas separadamente con cuantizadores uniformes. El diseño de los mismos se relaciona con el nivel de compresión deseado y depende directamente la información de las sub-bandas. Los pasos de los cuantizadores vienen dados por la formula:

$$Q_k = \begin{cases} \dfrac{1}{q} & 0 \le k \le 3 \\ \dfrac{10}{q.A_k.\log_e(\sigma_k^{\,2})} & 4 \le k \le 59 \end{cases}$$

Fórmula 6-1: cómo se calcula el paso Q para cada sub-banda.

Las sub-bandas 60 a la 63 no son transmitidas. Los valores de las constantes A_k pueden verse en [BRADLEY/93].

El cuantizador para cierta sub-banda *k* mapea el coeficiente resultante de la transformación wavelet de punto flotante ***w*** a un valor entero ***p***.

$$p(w) = \begin{cases} floor\left(\frac{w - Z/2}{Q}\right) + 1 \; ; \; w > Z/2 \\ ceil\left(\frac{w + Z/2}{Q}\right) - 1 \; ; \; w < -Z/2 \end{cases}$$

con

$$Z = 1.2Q$$

Fórmula 6-2: cuantización.

p=0 para los valores no contemplados.

Luego a cierto valor entero ***p*** le corresponderá un representante de punto flotante ***a***

$$a(p) = \begin{cases} (p - C).Q + Z/2 & , \quad p > 0 \\ (p + C).Q - Z/2 & , \quad p < 0 \end{cases}$$

Fórmula 6-3: decuantización.

a =0 para p=0. La especificación indica un valor de C = 0.44.

El código implementa una clase que representa al cuantizador, y tiene como funciones miembro principales a:

```
int p(int k, float a);
float a(int k, int p);
```

La codificación de esta sección es fácil y directa de acuerdo a las especificaciones mencionadas aquí y en [BRADLEY/93].

La distribución de bits

En la sección anterior se vio que la cuantización está dirigida por un parámetro general ***q,*** que toma valores en (0; 1]. El radio de compresión que se logra con un ***q*** dado es incierto, la pregunta que se plantea aquí es si es posible hallar un ***q*** para cierto radio de compresión deseado.

El problema a resolver entonces es el de minimizar la distorsión introducida al cuantizar las sub-bandas, donde las variables son bits asignados a cada cuantizador (o el paso asignado a cada cuantizador Q_k, o sea según [BRADLEY/93] el parámetro ***q***).

Esta implementación no aborda este problema, para posibles soluciones puede verse la aproximación que usa la implementación de WSQ del NIST o en [BRADLEY/93][BRADLEY-2/93][BRISLAWN/92][BRADLEY/92]

Codificación de Entropía

Primero se codifican los coeficientes resultado de la cuantización en una secuencia de códigos de escape (a un código de escape lo sigue cierto dato que *'escapa'* de lo común), cuya principal característica es la de codificar las corridas de ceros con unos pocos símbolos. Estos símbolos son el alfabeto sobre el cual se aplicara la codificación de entropía real, en este caso Huffman.

En [BRADLEY/93] define claramente este alfabeto, en esta implementación se aparta ligeramente de lo indicado en el standard. Aquí se utiliza un símbolo más, teniendo entonces la oportunidad de representar un coeficiente adicional (en este caso -74).

Símbolo	Significado
0	corrida de 1 cero
1	corrida de 2 ceros
...	...
99	corrida de 100 ceros
100	Esc + corrida de ceros (8 bits) (*)
101	Esc + corrida de ceros (16 bits) (*)
102	Esc + coeficiente positivo (8 bits) (*)
103	Esc + coeficiente negativo (8 bits) (*)
104	Esc + coeficiente positivo (16 bits) (*)
105	Esc + coeficiente negativo (16 bits) (*)
106	-74
107	-73
...	...
179	-1
180	no usado, pues está el símbolo 0
181	+1
...	
254	+74

Tabla 6-1: Tabla de símbolos Huffman

El standard también dice que se aplican distintas codificaciones a bloques de sub-bandas, a cada una correspondiéndole una tabla de Huffman distinta (con un máximo de ocho tablas por imagen).

Dado que esto implica la eventual transmisión de varias tablas y que cada una de ellas tiene un tamaño sino grande, tampoco pequeño, se espera que esto influya (en alguna medida) en las razones de compresión.

Para esta implementación en particular se elige como esquema codificador de entropía lo que se denomina Huffman Adaptivo [].

Siguiendo el esquema descripto en [], se aclara aquí el tema de la inicialización del árbol. Para no ocupar a la estructura de datos con información innecesaria, se utiliza un símbolo especial <esc>, que se utiliza para agregar símbolos al árbol. Otro símbolo especial <end_of_data> es utilizado para marcar el fin de los datos.

El método puede describirse de la siguiente manera:

Codificación:

```
Se comienza con un árbol de Huffman con dos símbolos (<esc>, <end_of_data>)
Repetir ... (hasta que no hay más elementos a codificar)
Se lee el <elemento> a codificar
Si no pertenece al árbol
       se lo transmite como secuencia <esc> <elemento>
       se agrega <elemento> al árbol
       se actualizan los pesos de los nodos <esc> y <elemento>
Si pertenece al árbol
       se transmite el código de <elemento>
       se actualiza el peso del nodo <elemento>
Se reestructura el árbol de ser necesario
```

Decodificación

```
Se comienza con un árbol de Huffman con dos símbolos (<esc>, <end_of_data>)
Repetir...(hasta recibir un <fin de datos> o alguna otra condición agradable)
Se lee <código>
Si <código> = <esc>
       se lee directamente <elemento>
       se agrega <elemento> al árbol
       se actualizan los pesos de los nodos <esc> y <elemento>
Si <código> = <elemento> ( != <esc>)
       se actualiza el peso del nodo <elemento>
Se reestructura el árbol de ser necesario
```

Listado 6-1: Pseudocódigo más refinado de la codificación Huffman Adaptiva.

La especificación original divide a las sub-bandas en 2 bloques, de la 0 a la 18 y de la 19 a la 59 (recordar que no se transmiten las sub-bandas 60..63), calculando dos tablas de Huffman basadas en las frecuencias de los símbolos que aparecen en cada una.

En esta implementación, y basándose en razones experimentales las sub-bandas se dividen en tres bloques: 0-18, 19-52, y 53-59. Dado que la codificación adaptiva no necesita tablas, codificar independientemente estos bloques no implica mayor cantidad de datos a transmitir. La elección de esta partición de los bloques se baso en los rangos y distribución de los coeficientes que se encuentran en los mismos.

La implementación de la codificación adaptiva de Huffman está basada en el código encontrado en [NELSON/92]. Esta agrega algunos símbolos mas, para el control del algoritmo.

La transmisión de datos.

A los fines de completar la documentación, se describe el formato del archivo generado por la implementación aquí descripta. Por que se eligió este y no otro (como el descripto en [BRADLEY/93]) se debe principalmente al principio del consumo de mínima energía, en otras palabras, contiene la mínima información necesaria, y al autor le resulto fácil y rápido de codificar !.

El archivo comienza con la estructura WSQHEADER, que contiene información global a la imagen: firma (***signature***) que identifica al formato, ancho (***width***), alto (***height***), factor de corrimiento (***shift***), y factor de escala (***scale***). Sobre éstos dos últimos: antes de ser transformada, a la imagen le aplicada la siguiente formula

$$I' = \frac{I - shift}{scale}$$

Fórmula 6-4: preprocesamiento de la imagen

con el objeto de hacer que tenga media aproximadamente cero y valores en el rango -128 a +128.

signature(20)	
width(2)	*height(2)*
shift(4)	*scale(4)*

Figura 6-3: WSQHEADER

Luego siguen en orden las bandas codificadas, la información de cada banda es precedida por una cabecera BANDHEADER, que contiene la siguiente información: ***id***, 'S' para todas las sub-bandas (permite agregar otro tipo de información en el futuro y además como una firma, luego es verificada); ***num***, el número de sub-banda que precede; ancho(***width***), alto (***height***); y ***q***, el paso del cuantizador para la sub-banda.

id(1)	*num(1)*
width(2)	*height(2)*
q(4)	

Figura 6-4: BANDHEADER

Después de esta cabecera viene la secuencia de bits de la codificación Huffman, alineada en bytes.

La descomposición en sub-bandas.

El siguiente gráfico muestra la descomposición en sub-bandas de la imagen de una huella digital. Los puntos negros en la imagen representan los coeficientes que son distintos de cero.

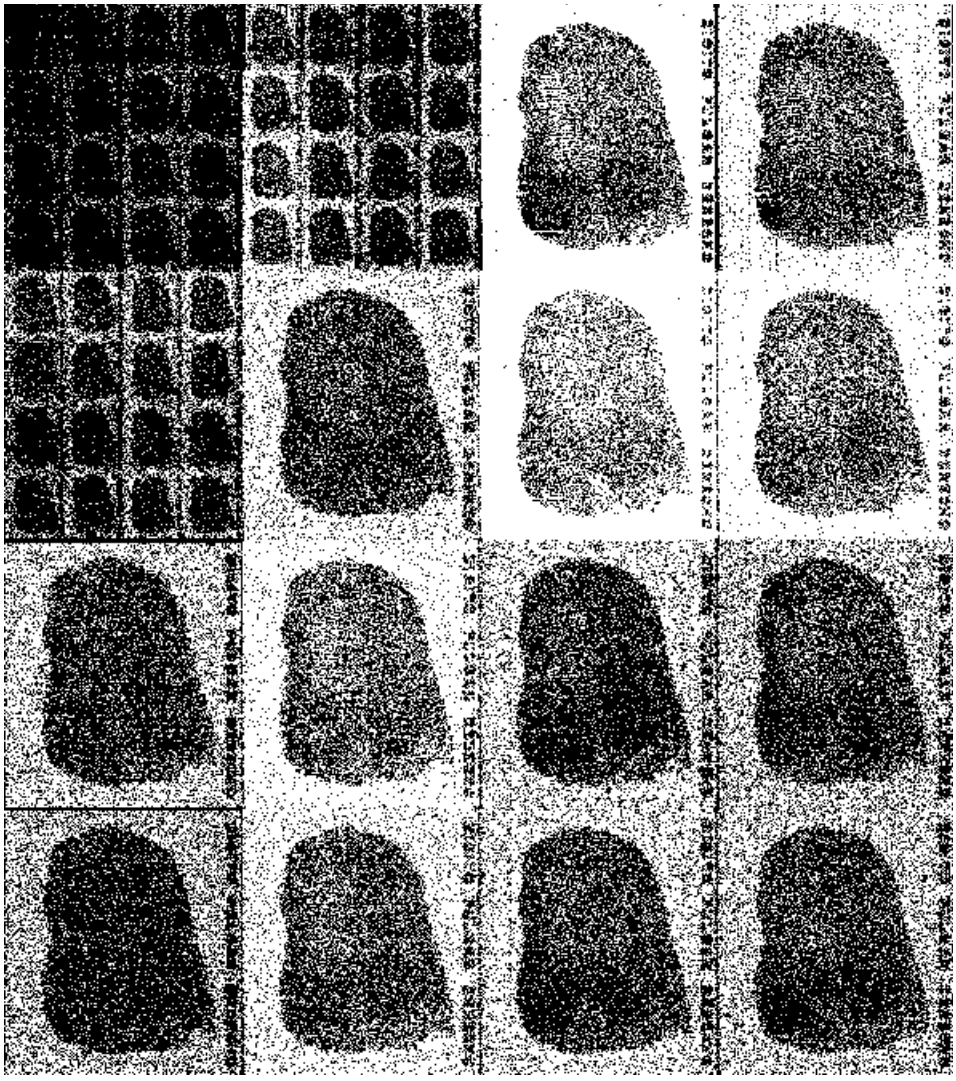
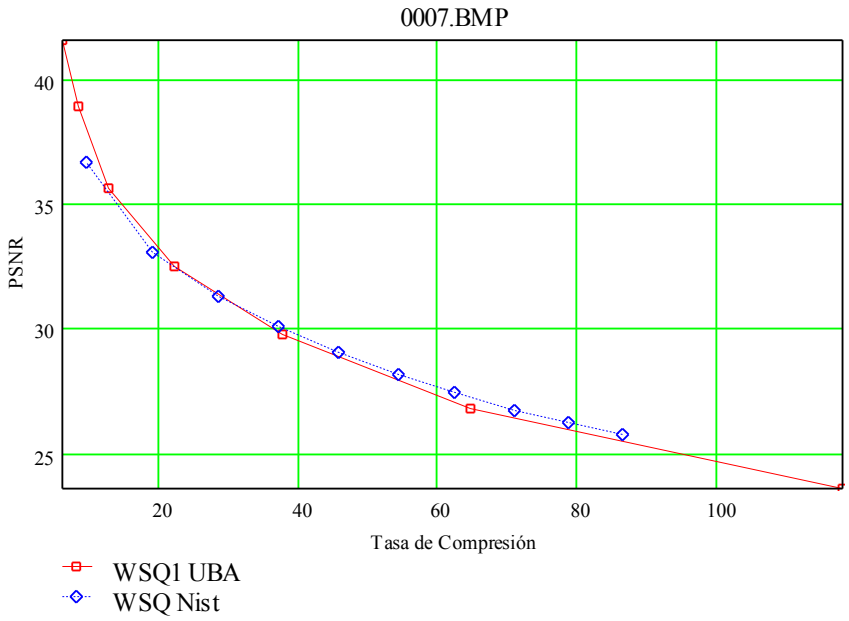


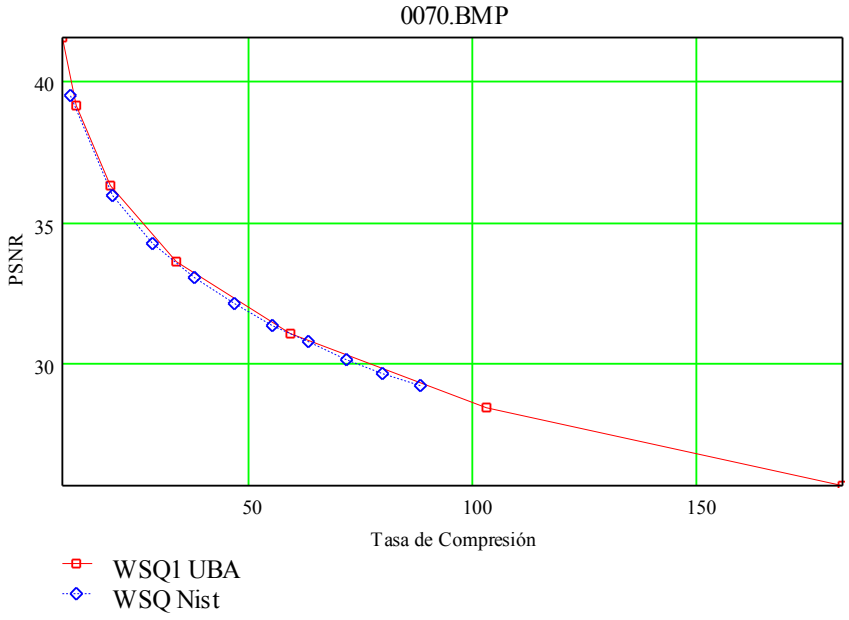
Figura 6-5: Esquema de sub-bandas en WSQ. Los puntos negros indican coeficientes distintos de cero.

Comparación con la implementación original.

Al haberse apartado en algunos detalles de implementación (como usar codificación Huffman adaptiva) del estándar original, cabe preguntarse si el desempeño del método ha mejorado o empeorado donde es más importante: tasa de compresión versus error.

A continuación se muestran tres gráficas mostrando esto. A la implementación original se la denominara WSQ NIST, ya que para los experimentos se utilizo la implementación realizada por el NIST, que adhiere al estándar donde la hecha aquí no. La implementación realizada en esta tesis se denominara WSQ1 UBA.





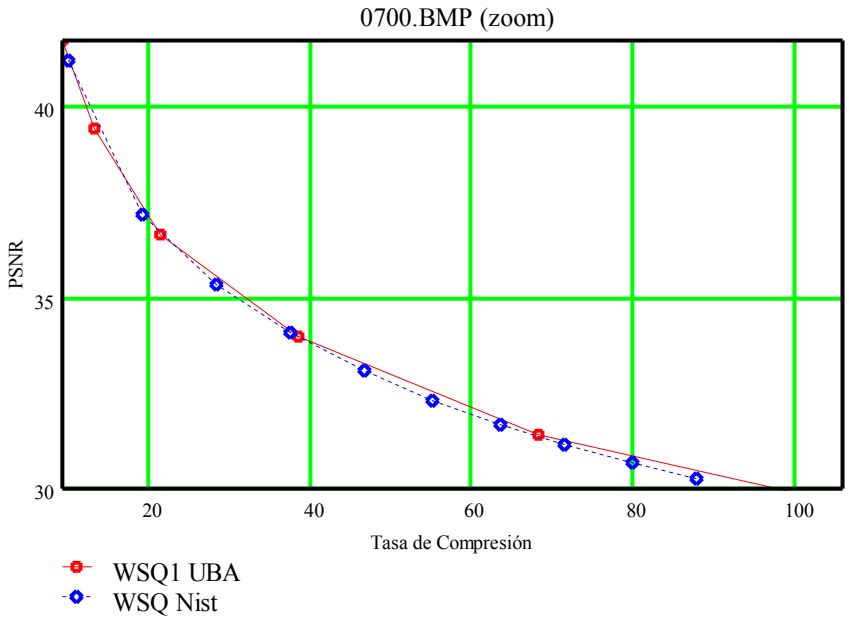


Figura 6-6: Tasa de compresión vs. PSNR para tres imágenes.

Las tres gráficas muestran la relación compresión - error para tres imágenes representativas, la ultima de ellas, es una ampliación de la zona donde la relación señal ruido es mayor a 30 dB. Como puede observarse no existe gran diferencia entre un método y el otro. Sin embargo puede apreciarse que la implementación UBA logra en general comprimir un poco más para PSNRs mayores a 30 dB.

Evaluación Final: JPEG vs. WQS

Se compararon las curvas tasa de compresión – PSRN resultantes de procesar 1251 imágenes provenientes de [cd-rom2]. Para crear las mismas se corrieron procesos en lotes, donde en el caso de WSQ1 [] el parámetro de compresión q toma los siguientes valores: 0.005, 0.01, 0.02, 0.04, 0.08, 0.15, 0.25. Para las curvas asociadas a JPEG (con la tabla de cuantización obtenida en []) los valores de q fueron: 5, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 75. Los puntos obtenidos de este proceso se observan en el siguiente gráfico:

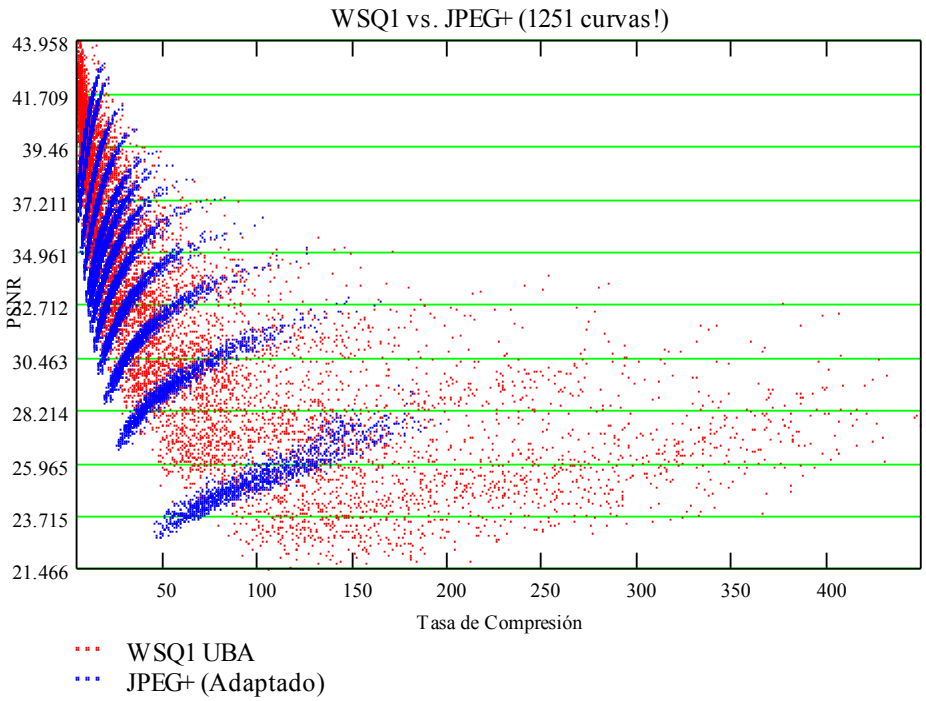


Figura 7-1: Resultado de comprimir 1251 imágenes con JPEG+ y WSQ1, cada punto representa una imagen comprimida con cierto método y determinado parámetro de compresión *q*.

Es interesante ver como los puntos resultantes de la compresión JPEG se agrupan en 'clusters', donde cada uno de ellos corresponde a determinado parámetro q de compresión (son 12 de ellos).

De las 1251 pares de curvas se observo cual de ellas se encontraba por encima de la otra. Que una curva este encima de la otra significa que el método del cual proviene, para el mismo nivel de compresión, produce menos error, o reconstruye una imagen de mayor calidad. Los resultados de esta observación fueron:

	# Curvas
WSQ1 > JPEG+	1246
WSQ1 < JPEG+	0
Se cruzan (*)	5
Total	**1251**

Tabla 7-1: Resultado de comparar todas las curvas.

(*) Las curvas que se cruzan son descartadas, al no poder compararlas. Un ejemplo de esto:

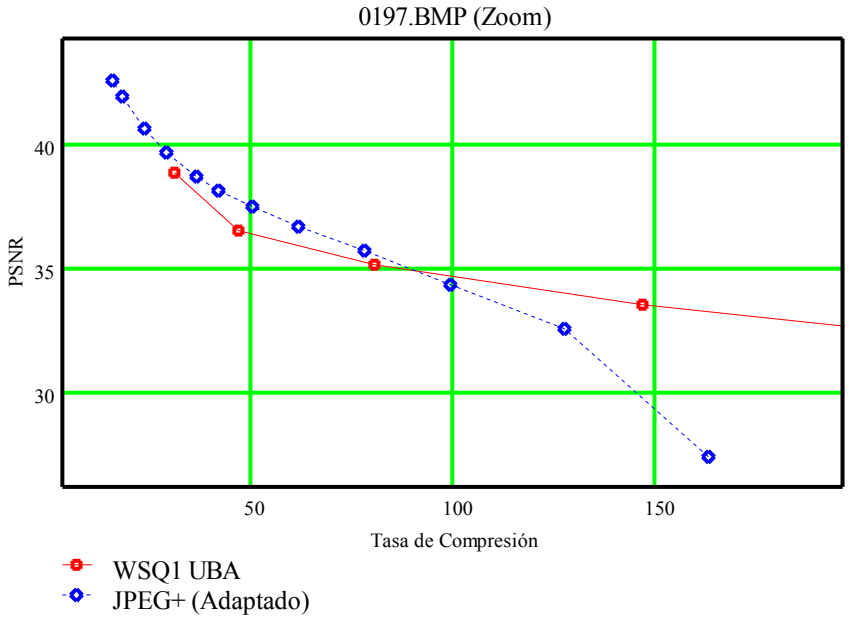
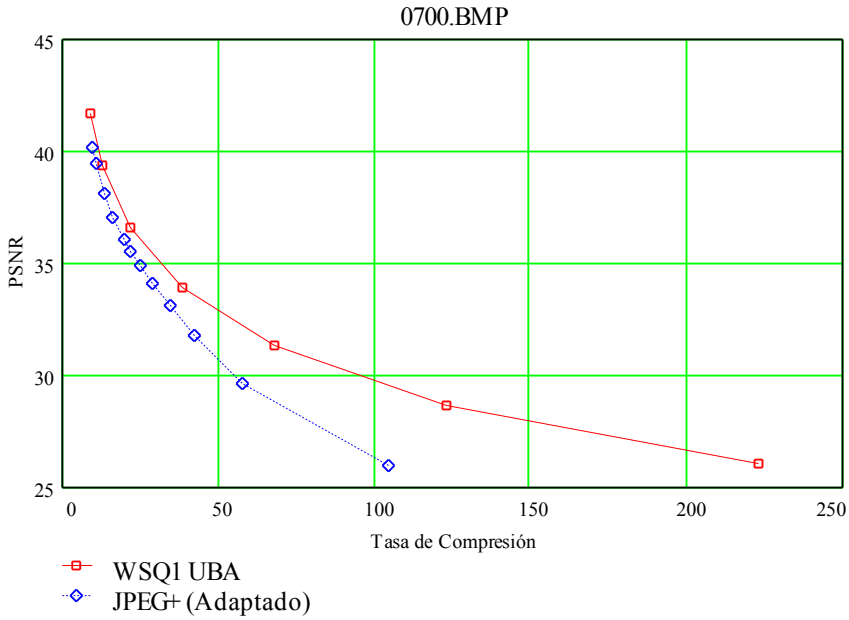


Figura 7-2: Gráfica de ciertos casos particulares, donde no se puede decidir si una curva está totalmente sobre la otra.

A continuación se muestran algunas de las curvas que muestran cómo WSQ tiene mejor desempeño que JPEG:



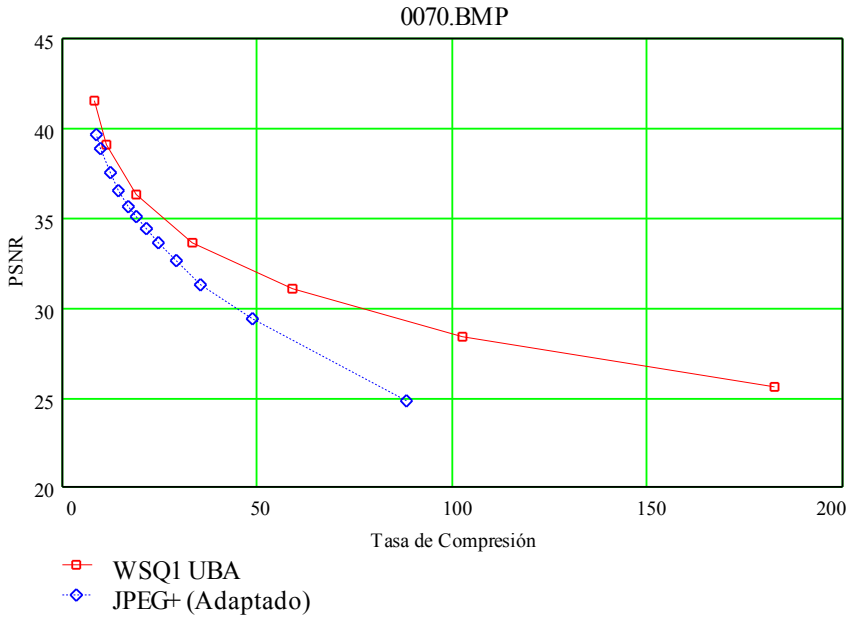


Figura 7-3: Gráfico que muestra cómo WSQ1 comprime mejor que JPEG+ para cierta imagen, tomando como medida de calidad a la PSNR.

Resultados y Conclusiones.

Se ha verificado experimentalmente con una cantidad de muestras significativamente grande (alrededor de 1250) la eficiencia del estándar WSQ para la compresión de imágenes de huellas digitales, frente a otros métodos de compresión. En particular, se realizo una comparación frente a JPEG, el estándar más usado para la compresión de imágenes de nuestros días.

El problema se abordó de la siguiente manera: primero se realizó una investigación de los distintos principios y métodos para la compresión de imágenes, con una perspectiva histórica, llegando hasta nuestros días. Esto brindó el conocimiento y criterio necesario para poder discernir ventajas y desventajas de las distintas alternativas.

Luego se estudio el tipo particular de imagen con el que se trabajaría: las impresiones dactilares. Los elementos que las componen, su utilidad, las aplicaciones que las utilizan hoy y la necesidad de la compresión de las mismas. Además, se estableció la necesidad de la conservación de la imagen de la huella dactilar una vez codificada (extraídas sus características). Se construyo una base de datos de 1251 imágenes de huellas digitales, para ser usada en los experimentos subsiguientes.

Después, habiendo elegido a JPEG como medida de comparación para el nuevo estándar WSQ, y siendo JPEG un método general, se analizo el problema de especializarlo para el tipo particular de imagen (huellas digitales) con las cuales se trabajaría. Se implemento una nueva variante de un esquema de distribución de bits, consiguiendo cierta mejora en el rendimiento de JPEG sobre las imágenes de impresiones dactilares.

Siguiendo esto, se implementó la especificación del método WSQ. Esta implementación se apartó del estándar original en algunos detalles no críticos, como ser el modulo de codificación de entropía y las estructuras de los archivos. Se comprobó luego que la eficiencia en la compresión fue mejorada, comparada con la implementación original.

Por ultimo se comparó el resultado de comprimir las 1251 imágenes con distintos niveles de compresión, utilizando la nueva implementación de WSQ y JPEG mejorado. Sobre las curvas generadas, el 99.6% mostró que WSQ supera en eficiencia (menor error vs. tasa de compresión) a JPEG.

Futuro Trabajo

Dada la diversidad de temas abordados en esta tesis (compresión de imágenes, procesamiento de señales, wavelets, optimización, teoría de la información, cuantización, por nombrar los más relevantes), son muchas las posibles extensiones a realizar.

Dado que se ha llegado a la importante conclusión de que vale la pena utilizar el método especializado de compresión WSQ, frente a otro de propósito general (JPEG) para la compresión de imágenes de huellas digitales, es relevante, entonces, el investigar si esta técnica puede trasladarse a otros tipos de imágenes que constituyen grandes bases de datos. Un ejemplo de dichas imágenes son los '*mugshots*' o fotos de rostros de personas.

Otra línea de trabajo se relaciona directamente con la propiedad de que, mediante wavelets, es posible realizar una compresión '*selectiva*' de ciertas zonas de una imagen. Esto es, dado los coeficientes de la transformada wavelet están asociados a determinado lugar, a cierta escala de la señal, es posible conservar los coeficientes de los lugares '*importantes*' de la señal y descartar (o menospreciar) los otros. Esto sugiere un método de compresión de huellas digitales que opera de la siguiente forma: primero se detectan de forma automática las minucias de la huella, esto define *zonas de importancia*; luego la cuantización (la etapa de pérdida de información) favorece a las zonas de importancia, asignándole más bits (o niveles); por ultimo se realiza la codificación de entropía. Es de esperarse que este método tenga aun mejor performance que el estándar [BRADLEY/93], salvados los inconvenientes de sobrecarga en los tipos de datos y los tiempos de ejecución.

En esta tesis también se estableció la importancia de poseer una medida de error cuya magnitud se vea reflejada directamente sobre las consecuencias que la persona que utiliza las imágenes perciba. Esto permite realizar una comparación más realista sobre dos fuentes de ruido. Dichas medidas deben involucrar elementos que describan el sistema de percepción visual humano y también deben poder valuar en cierto sentido la '*semántica*' de la imagen. Construir dicha medida es evidentemente una tarea compleja, por lo que generalmente se consideran medidas más sencillas (como en esta tesis). Un trabajo interesante, para comparar más ajustadamente distintos enfoques de compresión de imágenes de huellas digitales, es utilizar un proceso automático de extracción de minucias y analizar la relación de cantidad de minucias detectadas en función de la degradación de la imagen, producto de la compresión. Dicha medida de error constituiría una herramienta muy útil para este y futuros trabajos sobre huellas digitales.

Es interesante también continuar el trabajo sobre la optimización de JPEG, ya que existen otras áreas (como por ejemplo las de las imágenes médicas), donde es crucial el reducir los recursos de almacenamiento, existen exigencias muy altas de calidad de imagen y por simplicidad en la integración del sistema, algún estándar debe ser usado. Para lograr esto, hay diversas tareas que pueden realizarse: por ejemplo, investigar métodos más sofisticados de distribución de bits, utilizando por ejemplo técnicas de optimización no lineal al problema. Otras áreas de trabajo incluyen el modelar los histogramas de la Transformada Discreta del Coseno (DCT), como se sugirió en [ ] y el repetir los experimentos descriptos en [MITCHEL/93] con un conjunto de imágenes representativas (como por ejemplo [cd-rom2]) para obtener tablas de cuantización 'optimas'. Los resultados obtenidos podrían utilizarse también para una comparación más exacta de JPEG frente a otros métodos rivales.

Por ultimo, para dotar al estándar WSQ de compresión de imágenes de huellas digitales de las características que lo definan como un producto '*comercial*', es necesario el trabajar sobre la implementación para obtener una ejecución rápida del algoritmo, así como investigar otros aspectos que puedan mejorar su desempeño en la relación tasa de compresión vs. error. Esto ultimo incluye el investigar si existen otro tipo de funciones base (wavelets) que logren una mejor compresión, trabajar sobre la codificación de entropía (métodos estáticos vs. dinámicos, métodos de mayor orden) y el utilizar técnicas de Cuantización Vectorial [GERSHO/92].

Agradecimientos.

Al Oficial Inspector Hector Ribau, Jefe C.P.E.D de la Policía de la Pcia. De Buenos Aires, por haber facilitado las fichas dactiloscópicas con las que se confeccionó la base de datos para los experimentos e invaluable información sobre dactiloscopia y sus técnicas.

Al Dr. Jorge Arnaboldi por haber facilitado los recursos de hardware para realizar la digitalización y la grabación en CD-ROM de la base de datos de prueba.

Referencias.

Bibliografía

[ABRAMSON/86]: Norman Abramson,
"Teoría de la información y Codificación",
Parainfo 1986.

[BALDI/93]: P. Baldi, Y.Chauvin,
"Neural Networks for Fingerprint Recognition",
Neural Computation Journal, Vol. 5, No. 3, pp.402-418 1993.

[BARNSLEY/89]: M. Barnsley,
"Fractals Everywhere",
Academic Press, San Diego, 1989

[BRADLEY/93]: T. Hopper, C.Brislawn, J. Bradley.
"WSQ Gray-Scale Fingerprint Image Compression Specification"
Federal Bureau of Investigations. Feb . 1993

[BRADLEY-2/93]: J. N. Bradley & C. M. Brislawn,
"Proposed First Generation WSQ Bit Allocation Procedure"
Los Alamos National Laboratory, 1993

[BRISLAWN/92]: J. N. Bradley and C. M. Brislawn,
"The Wavelet/Scalar Quantization Compresión Standard for Digital Fingerprint Images"
Los Alamos National Laboratory, 1992.

[BRISLAWN/92]: C. M. Brislawn,
"Classification of Symmetric Wavelet Transforms"
Los Alamos National Laboratory, 1992.

[BURT/83]: P. J. Burt and E. H. Adelson,
"The Laplacian Pyramid as a Compact Image Code",
IEEE Trans. Comm. 1983

[BRADLEY/92]: J. N. Bradley and C. M. Brislawn,
"The Wavelet/Scalar Quantization Compresión Standard for Digital Fingerprint Images",
Los Alamos National Laboratory, 1992.

[CAPMBELL/68]: F.W. Campbell and J.G. Robson,
"Application of Fourier Análisis to the Visibility of Gratings",
J. Physiol. Vol. 197, pp. 55-566, 1968.

[COBOS/78]: M. Cobos, M. Abraham,
"Manual de Dactiloscopía"
Ed. Plus Ultra, Bs. As. 1978.

[FIELDING/91]: K.H. Fielding, J.L. Horner, C.K. Makekau,
"Optical Fingerprint Identification by
Binary Joint Transform Correlator"
Optical Engieneering, 30(12):1958-1961, 1991.

[CASTLEMAN/96]: K.R. Castleman,
"Digital Imagene Processing"
Prentice Hall, ISBN 0-13-21467-4, 1996.

[COLLINS/85]: C.G. Collins,
"Fingerprint Science: How to roll, classify, and use fingerprints",
Custom Pub. Co. , Costa Mesa , CA, ISBN 0942728181, 1985.

[CORNSWEET/71]: T.N.Cornsweet,
"Visual Perception",
New York, Academic Press, 1971.

[FISHER/92]: Y. Fisher,
"Fractal Image Compression",
Course Notes of SIGGRAPH'92, 1992.

[FISHER/95]: Y. Fisher (Editor),
"Fractal Image Compression: Theory and Application to Digital Images",
Springer Verlag, NY 1995 - ISBN 0-387-94211-4, 1995.

[FOX/66]: Fox,
"Discrete Optimization via Marginal Analysis"
Management Science, November 1966.

[GAREY/83]: M.R. Garey, D.S. Johnson,
"Computers and intractability: a guide to the theory of
{NP-completeness}",
Freeman, New York, NY, - 1983

[GERSHO/92]: Allen Gersho, Robert M. Gray,
"Vector Quantization and Signal Compresión",
Kluwer Academic Press - 1992.

[HALLIDAY/92]: D. Halliday, R. Resnick, K.S. Krane,
"Physics vol. II",
John Wiley & Sons, 4ta. Edición, ISBN 0471804576, 1992.

[HARDWICK/90]: S.A. Hardwick, T. Kent, V.G. Sears,
"Fingerprint detection by fluorescence examination"
Publicación No. 3/90,
Police Scientific Development Branch, Home Office, 1990.

[HILTON/94]: M. L. Hilton, B. D. Jawerth, A. Sengupta,
"Compressing Still and Moving Pictures with Wavelets",
Multimedia Systems, vol. 2 No. 3 1994.

[HU/62]: M.K. Hu,
"Visual pattern recognition by moment invariants",
IEEE Trans. Infor. Theory, vol 8, pp179-187, feb. 1962

[HUFFMAN/52]: D.A.Huffman,
"A Method for the Construction of Minimum Redundancy Codes",
Proc. IRE, vol. 40 no. 10, pp. 1098-1101, September 1952.

[ISENOR/86]: D.K. Isenor, S.G. Zaky,
"Fingerprint Identification Using Graph Matching",
Pattern Recognition Journal Vol. 19 pp. 113-122 1986.

[JAIN/89]: A. K. Jain,
"Fundamentals of Digital Image Processing"**,**
Prentice-Hall 1989

[JAIN/81]: A. K. Jain,
"Image Data Compression: A Review"**,**
IEEE Proceedings vol. 69 No. 3 March 1981

[JAIN/97]: A. Jain, L. Hong, R. Bolle,
"On-Line Fingerprint Verification",
IEEE PAMI, Vol. 19 No. 4 April 1997.

[KAWAGOE/84]: M. Kawagoe, A. Tojo,
"Fingerprint Pattern Classification"**,**
ETL, Vol. 17, pp. 295-303 1984.

[KAY/95]: D.C. Kay, J.R. Levine,
"Graphics File Formats",
McGraw-Hill , 1995.

[LEMPEL-ZIV/77]: J. Ziv and A. Lempel,
"A Universal Algoritm for Sequential Data",
IEEE Trans. on Inf. Theory, vol IT-23, No. 3, May 1977.

[LEMPEL-ZIV/78]: J. Ziv and A. Lempel,
"Compression of Individual Sequences via Variablr-Rate Coding",
IEEE Trans. on Inf. Theory, vol IT-24, No. 5, Sept. 1978.

[LIM/90]: J.S. Lim,
"Two-Dimensional Signal and Image Processing",
Prentice Hall, Englewood Cliffs, NJ. 1990

[LUENBERGER/84]: D.A. Luenberger,
"Linear and Nonlinear Programming",

Addison-Wesley -1984.

[MALLAT/89]: S. G. Mallat,
"A Theory for Multiresolution Signal Decomposition: The Wavelet Representation",
IEEE trans on PAMI vol II, No 7 July 1989

[MALLAT/87]: S. G. Mallat,
"Scale Change Versus Scale Space Representation",
1987

[MCGREGOR/94]: D.R. McGregor, R.J. Fryer, W.P.Cockshott & P. Murray,
"Fast Fractal Transform Method for Data Compression"
Departamental Research Report: RR/94/156 [IKBS-17-94]

[MCGREGOR/96]: D.R. McGregor, R.J. Fryer, W.P.Cockshott & P. Murray,
"Faster Fractal Compression"
Dr. Dobb's Journal, No. 234, pp. 34-42, Jan 1996

[MENZEL/91]: R.E. Menzel,
"An introduction to lasers, forensic lights, and fluorescent fingerprint detection techniques"
Salem, Or. : Lightning Powder Co., ISBN: 0962230561, 1991.

[MITCHEL/93]: William B. Pennebaker and Joan L. Mitchell,
"JPEG Still Data Compresión Standard"
Van Nostrand Reinhold, 1993, ISBN 0-442-01272-1, 1993.

[MURRAY/96]: J.D. Murray & William vanRyper
"Encyclopedia of Graphics File Formats"
O'Reilly & Asociates, Inc. ISBN 1-56592-161-5, May 1996.

[NELSON/92]: Mark Nelson,
"The Data Compression Book"
M&T Publishing, 1992. ISBN 1-55851-216-0

[PERKINS/89]: Michael G. Perkins, Tom Lookabaugh,
"A Psycophysically Justified Bit Allocation Algorithm for Subband Image Coding Systems",
Proceedings of ICASSP, 1989.

[RISSANEN/79]: J.Rissanen and G. Langdon,
"Arithmetic Coding",
IBM J. Res. Develop. 23, March 1979.

[STRANG/89]: G. Strang,
"Wavelets and Dialtion Equantions: A Brief Introduction",
SIAM Review vol. 31 No. 4 1989

[WALLACE/91]: Gregory K. Wallace,
"The JPEG Still Picture Compresión Standard",
Communications of the ACM (vol. 34 no. 4), pp 30-44, April 1991.

[WEGSTEIN/82]: J.H. Wegstein,
"An automated fingerprint identification system"
sponsored by the Federal Bureau of Investigation, U.S. Department of Justice,
Washington, D.C. : U.S. Dept. of Commerce,

National Bureau of Standards, 1982.

[WELCH/84]: T.A.Welch,
"A technique for High-Performance Data Compression",
Computer Magazine of the Comp. Group News of the IEEE Comp. Group Soc.
Vol 17, No. 6, Jun 1984.

[WESTERINK/89]: P.H. Westerink,
"Subband Coding of Images"
Technical University of Delft, Delft, Netherlands, Oct 1989.

[WILSON/94]: C.L. Wilson, G.T. Candela, C.I. Watson,
"Neural-Network Fingerprint Classification"
J. Artificial Neural Networks, vol. 1, no. 2, pp. 203-228, 1994.

Referencias en Internet.

Nota: En algunas de las referencias a continuación se mencionan algunas empresas. Queda ***expresamente declarado*** que no existe ningún tipo de preferencia, y/o adhesión con las mismas. Dichas menciones sólo sirven al propósito de asentar la existencia de productos y tecnologías de cierta clase que son mencionados a lo largo de esta tesis.

[Deflate]: **ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html**
Especificacion del formato de compresión de datos deflate.

[Gff]: **http://www.ora.com/centers/gff/index.htm**
"Graphics File Format Home Page"
Mantenida por O'Reilly & Associates.

[Gzip]: **ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html**
Especificacion del formato de archivo gzip.

[Identix]: **http://www.identix.com**
Empresa dedicada a la fabricación y venta de equipos biometricos
basados en huellas dactilares

[Ijg]: **ftp://ftp.uu.net/graphics/jpeg**
"The Independent JPEG Group's software"

[Iterated]: **http:/www.iterated.com**
Empresa que desarrolla y comercialializa productos basados en compresión fractal.

[JpegFAQ]: **http://www.cis.ohio-state.edu/hypertext/faq/jpeg-faq/part1/faq-doc-8.html**
"*JPEG Image Image Compression Frecuently Asked Questions"*
Preguntas y respuestas sobre JPEG.

[Nec]: **http://www.nec.com**
Empresa que desarrolla (entre muchos otros productos) el sistema AFIS 21.

[Png]: **ftp://ftp.uu.net/graphics/png/documents/**
Especificación del formato de compresión de imágenes PNG.

[Printrak]: **http://www.printrakinternational.com**
Empresa que desarrolla hardware y software de sistemas AFIS.

[Startek]: **http://www.startek.com.tw**
Empresa dedicada a la fabricación y venta de equipos biometricos

basados en huellas dactilares

[Storage]: **http://www.storage.ibm.com**
http://www.maxoptix.com
http://www.sun.com/storage
Algunos puntos de referencia. Distintas firmas que comercializan medios de almacenamiento masivo de información.

[Zlib]: **ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html**
Especificación del formato de compresión de datos zlib.

Patentes mencionadas.

[uspt-4941193]: M.F. Barnsley, A.D. Sloan
"Methods and Apparatus for Image Compression by Iterated Function System"
Iterated Systems Inc., NY. Jul 1990

[uspt-4558302]: T.A. Welch,
"High Speed Data Compression and Decompression Apparatus and Method"
Sperry Corporation, NY. Jun 1983

[uspt-4464650]: W. Eastman, A. Lempel, J. Ziv, M.Cohn,
"Apparatus and Method for Compressing Data Signals and Restoring the Compressed Data"
Sperry Corporation, NY. Jun 1981

[uspt-4122440]: G.G. Langdon (Jr.), J.J. Rissanen
"Methods and Means for Arithmetic String Coding"
IBM Corp., NY. Jun 1977