UNIVERSITAT POLITÈCNICA DE VALÈNCIA

El sistema JPG y la transformada coseno

Apellidos, nombre	**Benítez López, Julio** (jbenitez@mat.upv.es)
Departamento	**Departamento de Matemática Aplicada**
Centro	**Universitat Politècnica de València**

1 Resumen de las ideas clave

En este artículo vamos a introducir la base matemática (álgebra matricial) del formato JPG: la transformada coseno.

2 Objetivos

Tras asimilar los contenidos de este documento, el alumno debe poder explicar la transformada coseno y entender los fundamentos básicos del sistema JPG usado en fotografía.

3 Introducción

Todos nosotros sabemos lo que es un archivo JPG: cuando hacemos una fotografía con el móvil o con cualquier cámara digital, esta es guardada con esta extensión. JPG es uno de los formatos gráficos más populares de los usados actualmente. ¿Por qué? No es nada evidente al ojo humano, pero el formato JPG elimina información para hacer que los ficheros JPG sean pequeños. Cada vez que hacemos una foto y la guardamos en este formato, algunos datos se pierden; pero esta pérdida de información es indistinguible al ojo humano. JPEG (del inglés *Joint Photographic Experts Group*) es el nombre de un comité de expertos que creó en los años 90 un estándar de codificación de imágenes.

4 Breve repaso de la transformada discreta de Fourier

A lo largo del artículo, todos los vectores de $\mathbb{C}^n$ se consideran columnas. Si A es una matriz, entonces A^t denota su transpuesta.

La **tranformada discreta de Fourier** de un vector $\mathbf{h} \in \mathbb{C}^n$ es el vector

$$\widehat{\mathbf{h}} = \frac{1}{n}\overline{F}\mathbf{h}, \qquad \text{(Ecuación 1)}$$

siendo F la matriz $n \times n$ dada por

$$F = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix}, \qquad \omega = \mathrm{e}^{2\pi \mathrm{j}/n}.$$

Se puede comprobar que $F^{-1} = \frac{1}{n}\overline{F}$, por lo que la Ecuación 1 explica que la **antitransformada discreta de Fourier** de un vector $\widehat{\mathbf{h}} \in \mathbb{C}^n$ es el vector

$$\mathbf{h} = F\widehat{\mathbf{h}}. \qquad \text{(Ecuación 2)}$$

Si $\mathbf{h} = [h_0\ h_1\ \cdots\ h_{n-1}]^t$ y $\widehat{\mathbf{h}} = [b_0\ b_1\ \cdots\ b_{n-1}]^t$, entonces la igualdad anterior se escribe como $h_k = \sum_{r=0}^{n-1} b_r \mathrm{e}^{2\pi r \mathrm{j} k/n}$, lo que motiva a decir que si $\mathbf{h}$ es una señal en el dominio temporal, entonces $\widehat{\mathbf{h}}$ es la misma señal, pero en el dominio de las frecuencias.

La transformada discreta de Fourier tiene muchas propiedades, pero sin embargo solo listamos dos de estas, ya que son las únicas que usaremos en este artículo.

Propiedad 1. $\|\mathbf{h}\| = \sqrt{n}\|\widehat{\mathbf{h}}\|$ (la norma de un vector $\mathbf{z} = [z_1 \ \cdots \ z_n] \in \mathbb{C}^n$ se calcula con $\|\mathbf{z}\|^2 = |z_1|^2 + \cdots + |z_n|^2$).

Propiedad 2. $\widehat{\mathbf{h}+\mathbf{g}} = \widehat{\mathbf{h}} + \widehat{\mathbf{g}}$ y $\widehat{\lambda\mathbf{h}} = \lambda\widehat{\mathbf{h}}$ (para cualquier $\lambda \in \mathbb{C}$).

5 Compresión digital y la transformada discreta de Fourier

Veamos un ejemplo numérico. Dada la señal $\mathbf{h} = [1 \ \ 2 \ \ 3 \ \ 4 \ \ 4.5 \ \ 4 \ \ 3 \ \ 2 \ \ 1]^t \in \mathbb{C}^9$, si calculas la transformada de Fourier discreta de $\mathbf{h}$ mediante la Ecuación 1 (es buena idea que implementes, por ejemplo, en Octave la Ecuación 1 y la Ecuación 2), obtienes (redondeando) que si $\widehat{\mathbf{h}} = [b_0 \ b_1 \ \cdots \ b_8]^t$, entonces

$$b_0 \simeq 2.72,\ b_1 \simeq -0.81 - 0.30\,\mathrm{j},\ b_2 \simeq -0.02 - 0.02\,\mathrm{j}, b_3 \simeq -0.03 - 0.05\,\mathrm{j},\ b_4 \simeq -0.00 - 0.01\,\mathrm{j}$$

y $b_5 = \overline{b_4}$, $b_6 = \overline{b_3}$, $b_7 = \overline{b_2}$, $b_8 = \overline{b_1}$. Como puedes observar, las componentes b_2, b_3, b_4, b_5, b_6 y b_7 son muy pequeñas.

¿Qué ocurre si anulamos estas componentes? Si definimos el vector $\widehat{\mathbf{g}} = [b_0 \ b_1 \ 0 \ \cdots \ 0 \ b_8]$ y calculamos su transformada inversa discreta de Fourier (usando la Ecuación 2), obtenemos

$$\mathbf{g} \simeq [1.094 \ \ 1.857 \ \ 3.023 \ \ 4.049 \ \ 4.454 \ \ 4.049 \ \ 3.023 \ \ 1.857 \ \ 1.094]^t.$$

Podemos ver que $\mathbf{h}$ y $\mathbf{g}$ son bastante parecidos. De hecho, debido a las propiedades de las transformada discreta de Fourier se tiene que $\|\mathbf{h}-\mathbf{g}\| = \sqrt{n}\|\widehat{\mathbf{h}-\mathbf{g}}\| = \sqrt{n}\|\widehat{\mathbf{h}}-\widehat{\mathbf{g}}\|$, aquí, $n = 9$ es el tamaño de los vectores. Esta última igualdad nos dice que *si cambiamos un poco las señales en el dominio de las frecuencias, entonces cambian poco en el dominio temporal.* Observa la Figura 1 en donde se muestra de forma esquemática el proceso descrito en el ejemplo.

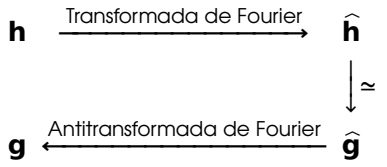


Figura 1: Si $\widehat{\mathbf{h}} \simeq \widehat{\mathbf{g}}$, entonces $\mathbf{h} \simeq \mathbf{g}$.

Y... ¿qué utilidad tiene esto? Podemos ver que mientras $\mathbf{h}$ tiene 9 entradas no nulas, el vector $\widehat{\mathbf{g}}$ tiene solo 3 entradas no nulas, (además la segunda y última componente de $\widehat{\mathbf{g}}$ son conjugadas una de la otra). Es decir, hemos comprimido en un 66 % la señal original $\mathbf{h}$ (perdiendo una mínima información). Además sabemos de manera precisa la distorsión $\|\mathbf{h}-\mathbf{g}\|$ usando $\|\mathbf{h}-\mathbf{g}\| = \sqrt{n}\|\widehat{\mathbf{h}}-\widehat{\mathbf{g}}\|$. Observa que para que este proceso de compresión sea útil en el sentido $\mathbf{h} \simeq \mathbf{g}$, hemos de asegurar que $\widehat{\mathbf{h}} \simeq \widehat{\mathbf{g}}$; es decir, las componentes de $\widehat{\mathbf{h}}$ que anulemos tienen que ser "pequeñas".

Veamos otro ejemplo. Considera $\mathbf{h} = [1 \ \ 2 \ \ 3 \ \ 4 \ \ 5 \ \ 7]^t \in \mathbb{C}^7$. ¡Más regular no puede ser! Sea $\widehat{\mathbf{h}}$ su transformada discreta de Fourier. Si calculamos el vector cuya componente k es el módulo de la componente k de $\widehat{\mathbf{h}}$ obtenemos $[4 \ \ 1.15 \ \ 0.64 \ \ 0.51 \ \ 0.51 \ \ 0.64 \ \ 1.15]^t$. Vemos que en $\widehat{\mathbf{h}}$ no hay componentes que sean "pequeñas", por lo que si anulamos algunas componentes en el dominio de las frecuencias, entonces debido a $\|\mathbf{h}-\mathbf{g}\| = \sqrt{n}\|\widehat{\mathbf{h}}-\widehat{\mathbf{g}}\|$, procaríamos grandes cambios en el dominio temporal.

Podríamos pensar en anular la 4^a y 5^a componente de $\widehat{\mathbf{h}}$, obteniendo $\widehat{\mathbf{g}}$. En este caso, se puede comprobar que $\mathbf{g} = [2 \;\; 1.198 \;\; 3.445 \;\; 4 \;\; 4.555 \;\; 6.802 \;\; 6]^t$. Observa que $\mathbf{g}$ y $\mathbf{h}$ no se parecen.

¿Porqué estos dos ejemplos tienen un comportamiento tan distinto? La situación se aclara si pensamos que las señales son **periódicas**. La seal del ejemplo anterior debe ser

$$\left[\cdots \;\; 5 \;\; 6 \;\; 7 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 1 \;\; 2 \;\; \cdots \right].$$

En la Figura 2 se ve mejor esta señal.

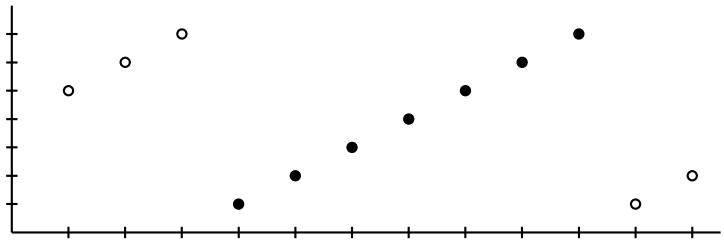


Figura 2: Los puntos negros son la señal finita; pero en realidad, la señal es infinita (y periódica). Observa que hay "saltos".

Esta es la razón de que la señal del último ejemplo se "comporte mal" cuando hallamos su transformada discreta de Fourier. En realidad, las palabras "mal comportamiento" son erróneas: de hecho se comporta como debería comportarse: debido a que la señal tiene saltos considerables, las altas frecuencias son significativas.

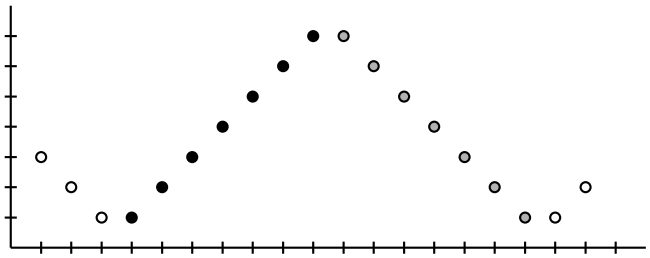


Figura 3: Extensión simétrica de la señal finita.

Hay una manera de evitar estos saltos: simetrizar la señal. La mejor manera es verlo en la Figura 3. Esta es la idea de la transformada coseno que veremos en la sección siguiente.

6 La transformada coseno

Sea $\mathbf{h} = [h_0 \; h_1 \; \cdots \; h_{n-1}]^t \in \mathbb{R}^n$ (observa que $\mathbf{h}$ es un vector **real**). En primer lugar definimos la extensión simétrica[1] de $\mathbf{h}$ como $\mathbf{y} = [h_0 \; h_1 \; \cdots \; h_{n-1} \; h_{n-1} \cdots \; h_1 \; h_0]^t \in \mathbb{R}^{2n}$. Ahora hallamos la transformada de Fourier discreta de $\mathbf{y}$: Sea $\widehat{\mathbf{y}} = [b_0 \; b_1 \; \cdots \; b_{2n-1}] \in \mathbb{C}^{2n}$ esta transformada;

[1] Dependiendo de cómo se haga la extensión, se obtienen distintas transformadas coseno.

es decir

$$\widehat{\mathbf{y}} = \frac{1}{2n}\overline{F}\mathbf{y}, \qquad \text{(Ecuación 3)}$$

donde la matriz $F = (F_{rs})$ está definida ahora (recuerda que estamos calculando la trasformada de Fourier para vectores de $2n$ componentes y no de n) como

$$F_{rs} = \omega^{rs}, \quad r, s = 0, \ldots, 2n-1, \qquad \omega = \mathrm{e}^{(2\pi \mathrm{j})/(2n)} = \mathrm{e}^{\pi \mathrm{j}/n}.$$

Puesto que aparece $\overline{F}$ en la Ecuación 3, denotaremos $\xi = \overline{\omega} = \exp(-\pi \mathrm{j}/n)$. Sea $k = 0, 1, \ldots, 2n-1$. De la Ecuación 3 se deduce que

$$\begin{aligned} b_k &= \frac{1}{2n}(k\text{-ésima fila de } \overline{F})\ \mathbf{y} \\ &= \frac{1}{2n}\left[(\xi^0 + \xi^{k(2n-1)})h_0 + (\xi^k + \xi^{k(2n-2)})h_1 + \cdots + (\xi^{k(n-1)} + \xi^{kn})h_{n-1}\right]. \end{aligned}$$

El coeficiente de h_r es $\xi^{kr} + \xi^{k(2n-1-r)}$. Para simplificar este coeficiente, recordemos que $\xi = \exp(-\pi \mathrm{j}/n)$ y por tanto $\xi^n = -1$ y $\xi^{2n} = 1$.

$$\begin{aligned} \xi^{kr} + \xi^{k(2n-1-r)} &= \xi^{kr} + \xi^{-k(1+r)} \\ &= \xi^{-k/2}\left(\xi^{kr+k/2} + \xi^{-kr-k/2}\right) \\ &= \xi^{-k/2}\left(\exp\left(\frac{-\pi \mathrm{j}(kr + k/2)}{n}\right) + \exp\left(\frac{\pi \mathrm{j}(kr + k/2)}{n}\right)\right). \end{aligned}$$

Usando la fórmula $\cos\alpha = (\mathrm{e}^{\mathrm{j}\alpha} + \mathrm{e}^{-\mathrm{j}\alpha})/2$ se tiene

$$\xi^{kr} + \xi^{k(2n-1-r)} = 2\xi^{-k/2}\cos\left(\frac{\pi(kr + k/2)}{n}\right) = 2\omega^{k/2}\cos\left(\frac{\pi k(2r+1)}{2n}\right).$$

Por tanto,

$$b_k = \frac{\omega^{k/2}}{n}\sum_{r=0}^{n-1} h_r \cos\left(\frac{\pi k(2r+1)}{2n}\right). \qquad \text{(Ecuación 4)}$$

Esta igualdad es el germen de la transformada coseno, que se definirá a continuación. La aparición de $\sqrt{n}$ y $\sqrt{2}$ en esta definición es simplemente debido a la costumbre.

Sea $\mathbf{h} \in \mathbb{R}^n$. El vector $\mathbf{c} \in \mathbb{R}^n$ dado por $\mathbf{c} = C\mathbf{h}$, donde $\theta = \pi/(2n)$ y

$$C = \frac{1}{\sqrt{n}}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \sqrt{2}\cos\theta & \sqrt{2}\cos(3\theta) & \cdots & \sqrt{2}\cos((2n-1)\theta) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{2}\cos((n-1)\theta) & \sqrt{2}\cos(3(n-1)\theta) & \cdots & \sqrt{2}\cos((2n-1)(n-1)\theta) \end{bmatrix}, \qquad \text{(Ecuación 5)}$$

es la **transformada coseno** de $\mathbf{h}$. La matriz C que aparece se le llama la **matriz coseno**.

La función de Octave presentada en la Figura 4 calcula la transformada coseno del vector $\mathbf{h}$. Observa que en este código, la matriz C es la matriz C de la definición anterior.

El cálculo de la antitransformada es importante. Obviamente, de $\mathbf{c} = C\mathbf{h}$ se deduce $\mathbf{h} = C^{-1}\mathbf{c}$ (siempre que C sea invertible); pero vamos a calcular la antitransformada de un modo más eficiente. Para este fin, observa que de la Ecuación 4 se deduce $b_n = 0$ y $\overline{b_r} = b_{2n-r}$ para $r = 1, \ldots, n-1$. Si no quieres saber cómo se decuce la expresión de la transformada inversa, puedes ir directamente hasta la Ecuación 8.

```
function c = coseno(h)
h=h(:);
[n m]=size(h);
theta=pi/(2*n);
N=(0:n-1)'*(1:2:2*n-1);
M=cos(theta*N);
D=diag([1 sqrt(2)*ones(1,n-1)]);
C=D*M/sqrt(n);
c=C*h;
```

Figura 4: Función que calcula la transformada coseno

Sea $\mathbf{c} \in \mathbb{R}^n$ conocido y queremos saber si existe y cómo se puede calcular $\mathbf{h} = [h_0 \ \cdots \ h_{n-1}]^t \in \mathbb{R}^n$ tal que $C\mathbf{h} = \mathbf{c}$. Suponiendo que exista tal $\mathbf{h}$, definimos su extensión $\mathbf{y} \in \mathbb{R}^{2n}$ como en el principio de esta sección y sea $\widehat{\mathbf{y}} = [b_0 \ \cdots \ b_{2n-1}] \in \mathbb{R}^{2n}$ la transformada discreta de Fourier de $\mathbf{y}$. Como $\mathbf{y} = F\widehat{\mathbf{y}}$, entonces para $k = 0, \ldots, n-1$,

$$\begin{aligned} h_k &= \omega^0 b_0 + \omega^k b_1 + \cdots + \omega^{k(n-1)} b_{n-1} + \omega^{kn} b_n + \omega^{k(n+1)} b_{n+1} + \cdots + \omega^{k(2n-1)} b_{2n-1} \\ &= b_0 + \left[\omega^k b_1 + \omega^{k(2n-1)} b_{2n-1}\right] + \cdots + \left[\omega^{k(n-1)} b_{n-1} + \omega^{k(n+1)} b_{n+1}\right] \\ &= b_0 + \left[\omega^k b_1 + \omega^{-k}\overline{b_1}\right] + \cdots + \left[\omega^{k(n-1)} b_{n-1} + \omega^{-k(n-1)}\overline{b_{n-1}}\right] \\ &= b_0 + \left[\omega^k b_1 + \overline{\omega^k b_1}\right] + \cdots + \left[\omega^{k(n-1)} b_{n-1} + \overline{\omega^{k(n-1)} b_{n-1}}\right] \\ &= b_0 + 2\mathrm{Re}\left(\omega^k b_1\right) + \cdots + 2\mathrm{Re}\left(\omega^{k(n-1)} b_{n-1}\right). \end{aligned}$$

En el último paso hemos usado que $z + \overline{z}$ es la parte real de $z \in \mathbb{C}$. Como b_0 es real (esto se deduce de la Ecuación 4), se tiene

$$h_k = \mathrm{Re}\left(b_0 + 2\omega^k b_1 + \cdots + 2\omega^{(n-1)k} b_{n-1}\right). \qquad \text{(Ecuación 6)}$$

Ahora pondremos cada b_r en función de $\mathbf{c}$ para poder expresar (usando la Ecuación 6) cada h_k en función de $\mathbf{c}$. Definimos $\mathbf{b} = [b_0 \ b_1 \ \cdots \ b_{n-1}]^t$ (observa que $\mathbf{b}$ es la "primera mitad" de $\widehat{\mathbf{y}}$). Por la Ecuación 4 y por la definición de la matriz coseno, se tiene

$$\mathbf{b} = \frac{1}{\sqrt{n}} \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \omega^{1/2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega^{(n-1)/2} \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1/\sqrt{2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\sqrt{2} \end{bmatrix}}_{:=D} C\mathbf{h} = \frac{1}{\sqrt{n}} D\mathbf{c}.$$

Ya hemos puesto $\mathbf{b}$ en función de $\mathbf{c}$. Retomamos los cálculos dejados en la Ecuación 6,

$$\begin{aligned} b_0 + 2\omega^k b_1 + \cdots + 2\omega^{(n-1)k} b_{n-1} &= \\ &= \begin{bmatrix} 1 & 2\omega^k & \cdots & 2\omega^{k(n-1)} \end{bmatrix} \mathbf{b} \\ &= \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 2\omega^k & \cdots & 2\omega^{k(n-1)} \end{bmatrix} D\mathbf{c} \\ &= \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & \sqrt{2}\omega^{k+1/2} & \cdots & \sqrt{2}\omega^{k(n-1)+(n-1)/2} \end{bmatrix} \mathbf{c}. \end{aligned}$$

No perdamos de vista que en la Ecuación 6 aparece la parte real de $b_0 + 2\omega^k b_1 + \cdots + 2\omega^{(n-1)k} b_{n-1}$ y que $\mathbf{c}$ es un vector real. Por tanto, vamos a tomar la parte real de $\omega^{kr+r/2}$,

para $r = 1, \ldots, n-1$. Recuerda que $\omega = \exp(\pi \mathrm{j}/n)$ y $\theta = \pi/(2n)$.

$$\omega^{kr+r/2} = \exp\left(\left(kr + \frac{r}{2}\right)\frac{\pi \mathrm{j}}{n}\right) = \exp\left(\pi \mathrm{j}\frac{2kr+r}{2n}\right) = \cos((2k+1)r\theta) + \mathrm{j}\,\mathrm{sen}((2k+1)r\theta).$$

Por tanto, de la Ecuación 6, se deduce que

$$h_k = \frac{1}{\sqrt{n}}\begin{bmatrix} 1 & \sqrt{2}\cos((2k+1)\theta) & \cdots & \sqrt{2}\cos((2k+1)(n-1)\theta) \end{bmatrix}\mathbf{c}.$$

De forma matricial,

$$\mathbf{h} = \frac{1}{\sqrt{n}}\begin{bmatrix} 1 & \sqrt{2}\cos\theta & \cdots & \sqrt{2}\cos(\theta(n-1)) \\ 1 & \sqrt{2}\cos(3\theta) & \cdots & \sqrt{2}\cos(3(n-1)\theta) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \sqrt{2}\cos((2n-1)\theta) & \cdots & \sqrt{2}\cos((2n-1)(n-1)\theta) \end{bmatrix}\mathbf{c}. \qquad \text{(Ecuación 7)}$$

Resumamos lo que hemos hecho hasta ahora: Dado $\mathbf{c} \in \mathbb{R}^n$, hemos demostrado que si existe $\mathbf{h} \in \mathbb{R}^n$ tal que $C\mathbf{h} = \mathbf{c}$, entonces tal $\mathbf{h}$ es único y viene dado por la Ecuación 7. Es decir, hemos encontrado la expresión para la inversa de la transformada coseno.

La matriz cuadrada que aparece en la Ecuación 7 es ¡la traspuesta de la matriz coseno C! (mira la Ecuación 5). Esto permite calcular la antitransformada coseno sin invertir ninguna matriz: Si $\mathbf{h} \in \mathbb{R}^n$ y $\mathbf{c}$ su trasformada coseno, entonces

$$\mathbf{h} = C^t\mathbf{c}. \qquad \text{(Ecuación 8)}$$

Observa que computacionalmente, es mucho menos costoso usar $\mathbf{h} = C^t\mathbf{c}$ que $\mathbf{h} = C^{-1}\mathbf{c}$. Con una modificación obvia en la última línea de la función mostrada en la Figura 4, se puede escribir una función de Octave que calcula la antitransformada coseno.

Como $C^{-1} = C^t$, si $\mathbf{h} \in \mathbb{R}^n$ y $\mathbf{c}$ su transformada coseno, entonces $\|\mathbf{c}\|^2 = \mathbf{c}^t\mathbf{c} = \mathbf{h}^tC^tC\mathbf{h} = \mathbf{h}^tC^{-1}C\mathbf{h} = \mathbf{h}^t\mathbf{h} = \|\mathbf{h}\|^2$, luego $\|\mathbf{c}\| = \|\mathbf{h}\|$. Esta igualdad es importante pues nos dice que si hacemos un cambio a $\mathbf{h}$, entonces $\mathbf{c}$ también se ve afectado de la misma manera, y viceversa.

7 La transformada discreta coseno bidimensional

La idea fundamental del formato JPG es aplicar la transformada coseno a una matriz (y no a un vector) pues una imagen se puede modelar como una matriz. Por tanto, vamos a ver cómo se puede definir de forma natural la transformada de una matriz.

$$\begin{array}{ccc} \mathbb{R}^m & \xrightarrow{C} & \mathbb{R}^m \\ {\scriptstyle M}\downarrow & & \\ \mathbb{R}^n & \xrightarrow{C} & \mathbb{R}^m \end{array}$$

Sea M una matriz $n \times m$. Podemos pensar que esta matriz es el operador de $\mathbb{R}^m$ a $\mathbb{R}^n$ que actúa como $\mathbf{x} \mapsto M\mathbf{x}$. Mira la la gráfica de la derecha: A la izquierda están los vectores que van a ser transformados por C y a la derecha los vectores que ya han sido transformados por C.

Si queremos definir la "transformada de M por medio de C" podemos pensar en los dos diagramas siguientes y exigir que de igual ir desde la esquina superior izquierda a la esquina

inferior derecha por los dos caminos distintos. Denotaremos a partir de ahora $\mathscr{C}M$ la transformada de la matriz M por medio de C.

$$\begin{array}{ccc} \mathbb{R}^m & \xrightarrow{C} & \mathbb{R}^m \\ & \downarrow{\scriptstyle \mathscr{C}M} & \\ \mathbb{R}^n & & \mathbb{R}^n \end{array} \quad = \quad \begin{array}{ccc} \mathbb{R}^m & & \mathbb{R}^m \\ \downarrow{\scriptstyle M} & & \\ \mathbb{R}^n & \xrightarrow{C} & \mathbb{R}^n \end{array}$$

O con fórmulas, $CM = (\mathscr{C}M)C$. Como C es invertible, entonces podemos despejar $\mathscr{C}M$. Esto motiva a definir la transformada coseno de la matriz M como $\mathscr{C}M = CMC^{-1}$. Pero, como hemos visto, $C^{-1} = C^t$, luego

> La transformada coseno de $M = CMC^{-1} = CMC^t$.

Por supuesto, también tenemos que saber dar el paso opuesto: es decir, dada una matriz ya transformada N, ¿cómo hallar la matriz M de la cual "proviene"? Esto es fácil si de $CMC^{-1} = N$ despejamos M:

> La antitransformada coseno de $N = C^{-1}NC = C^tNC$.

Las siguientes funciones calculan la transformada coseno (directa e inversa) de matrices cuadradas

```
function C = matrizc(n)
theta = pi/(2*n);
N = (0:n-1)'*(1:2:2*n-1);
M = cos(theta*N);
sq = sqrt(2);
D = diag([1 sq*ones(1,n-1)]);
C = D*M/sqrt(n);
```

```
function N = cosb(M)
[n m] = size(M);
C = matrizc(n);
N = C*M*C';
```

```
function M = cosbi(N)
[n m] = size(N);
C = matrizc(n);
N = C'*N*C;
```

Veamos cómo se puede usar Octave para entender el formato JPG. La figura de la derecha (de dominio público) puede descargarse en

`http://www.wpclipart.com/recreation/games/chess/`

Si hemos cargado esta figura con el nombre `rey.jpg`, entonces la orden `A = imread("rey.jpg");` crea la matriz `A` en donde se almacena el fichero gráfico. No olvides el punto y coma para que no se muestre el resultado (en este ejemplo `A` es de tamaño 500×500). Como solo hay píxeles blancos y negros, las entradas de `A` toman el valor 0 (negro) o bien 255 (blanco). La función `imread` crea una matriz de enteros (que tienen una aritmética más restringida que la de los complejos).

Para poder manipular la matriz `A`, convertimos sus entradas a números complejos mediante `A = double(A);`. Con `AC = cosb(A);` hallamos la transformada coseno de la matriz `A`. Con `ind = find(abs(AC)<1000);` buscamos las entradas de `AC` que son menores en valor absoluto a 1000. Observemos que hay `size(ind)` = 249708 entradas de este tipo de un total de $500^2 = 250000$. Anulamos estas entradas con `AC(indices) = 0;` (anulamos el 99.883 % de las entradas) y antitransformamos con `B = cosbi(AC);`

Convertimos `B` a una matriz de enteros con `B = uint8(B);` Y por último, creamos el fichero JPG correspondiente con `imwrite(B,"reybis.jpg")`.

En la Figura 5 se muestran tres imágenes. La primera es la recién obtenida, la segunda se ha conseguido anulando las componentes de AC menores en valor aboluto a 500 y en la tercera, se han anulado las componentes de AC menores en valor aboluto a 100.

Incluso, si anulamos las entradas de AC menores en valor absoluto a 20 logramos una compresión del 90.8 % produciendo una imagen prácticamente indistinguible de la original.

Figura 5: En el primer gráfico se ha logrado una compresin del 99.83 %. En el segundo del 99.76 %. En el tercero del 98.17 %.

8 El formato JPG

Ya que uno de los pasos del proceso JPG es anular algunas entradas de la matriz transformada de la imagen, vamos a analizar tales cambios. Y para esto usaremos matrices 4×4.

Denotemos por E_{rs} la matriz de orden 4 cuya entrada (r, s) es 1 y el resto de sus entradas son ceros. Sea G_{rs} la antitransformada coseno de E_{rs}. Un cambio en la entrada (r, s) de la transformada de una imagen se ve reflejada si observamos la imagen asociada a G_{rs}.

$$\begin{array}{ccc} \text{Imágenes} & \Longleftrightarrow & \text{Frecuencias} \\ G_{rs} & \Longleftrightarrow & E_{rs} \end{array}$$

Para generar la primera imagen de la Figura 6 se ha usado el código

```
ac=zeros(4,4);
ac(1,1)=600;
a=uint8(cosbi(ac));
imwrite(a,"c11.jpg")
```

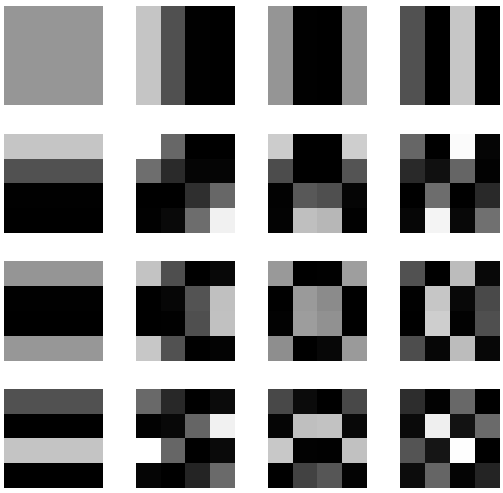


Figura 6: Toda imagen 4×4 se descompone como combinación de estas 16 imágenes.

Se ha usado el factor 600 en este código para que las gráficas sean claras.

Quizá algunas fórmulas ayuden a entender tanto la Figura 6 como la idea básica de la compresión JPG. Sea A una matriz cuadrada de tamaño $n \times n$ y $\mathscr{C}A$ su transformada coseno. Esta transformada se puede expresar como $\mathscr{C}A = \sum_{r,s} \lambda_{rs} E_{rs}$. Observa que λ_{rs} es la entrada (r, s) de la matriz $\mathscr{C}A$. Como la antitransformada coseno de una matriz X es $C^t X C$, entonces

$$A = C^t(\mathscr{C}A)\,C = C^t\left(\sum_{r,s} \lambda_{rs} E_{rs}\right) C = \sum_{r,s} \lambda_{rs} C^t E_{rs} C = \sum_{r,s} \lambda_{rs} G_{rs}.$$

Por tanto, la imagen original (representada por medio de la matriz A) se descompone como combinación de las matrices G_{rs} y las imágenes asociadas a estas matrices son las que aparecen en la Figura 6. Además vemos que cuanto más abajo y más a la derecha están estas imágenes, son "menos uniformes". Veamos la razón de esta "menos uniformidad".

Vamos a denotar por $[M]_{uv}$ la entrada (u, v) de una matriz cualquiera M. Como

$$[G_{rs}]_{uv} = \left[C^t E_{rs} C\right]_{uv} = \sum_{k=0}^{n-1} [C^t]_{uk} [E_{rs} C]_{kv},$$

antes tenemos que ser capaces de calcular $[E_{rs}C]_{kv}$:

$$[E_{rs}C]_{kv} = \sum_{m=0}^{n-1} [E_{rs}]_{km} [C]_{mv} = \begin{cases} [C]_{sv} & \text{si } r = k, \\ 0 & \text{si } r \neq k. \end{cases}$$

Luego

$$[G_{rs}]_{uv} = \sum_{k=0}^{n-1} [C^t]_{uk} [E_{rs}C]_{kv} = [C^t]_{ur} [C]_{sv} = [C]_{ru} [C]_{sv}.$$

Las componentes de la matriz C están escritas en la Ecuación 5. Y podemos ver que *cuanto mayores sean r y s, las frecuencias son mayores.* Y a mayor frecuencia, más oscilación de la onda. Esta es la explicación de que en la Figura 6, cuanto más cerca estemos de la posición "noroeste", la uniformidad es menor.

Otra idea del sistema JPG es que a escalas pequeñas, los cambios deben ser pequeños. Por tanto, si tenemos un trozo pequeño de una imagen, entonces las altas frecuencias pueden eliminarse sin que perjudique la calidad de la imagen. Con esto, ya tenemos el esquema básico del método JPG.

Imagen original $\longrightarrow$ Dividir la imagen en trozos pequeños $\longrightarrow$
$\longrightarrow$ Eliminar las altas frecuencias de cada trozo pequeño $\longrightarrow$ Juntar los trozos.

El proceso JPG divide la imagen original en bloques de 8×8 píxeles (un tamaño realmente pequeño). El siguiente paso es "suavizar" cada bloque, es decir eliminar las altas frecuencias. ¿Cómo eliminamos las altas frecuencias de cada trozo 8×8?

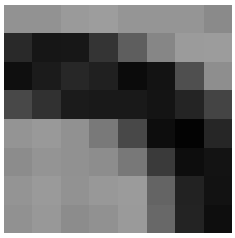
Sea A una matriz 8×8 (que corresponde a un bloque cuadrado de 8 píxeles). Primero calculamos $\mathscr{C}A = CAC^t$. Como esta operación hay que hacerla para cada bloque, resulta que la matriz C solo hay que calcularla una vez (de hecho ya está implementada y no hay que calcularla nunca, pues viene "de fábrica"). Para cada entrada (r, s) de $\mathscr{C}A$ el sistema JPG calcula $[\mathscr{C}A]_{rs}/q_{rs}$ y redondea el resultado al entero más próximo obteniendo una matriz R con muchos ceros. Esta matriz R es la que se guarda. Como resulta que deseamos eliminar las frecuencias mayores, los números q_{rs} deben ser mayores a medida que r y s aumentan. Este es el paso donde se pierde información; pero irrelevante. Un ejemplo es

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 40 & 57 & 69 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \qquad \text{(Ecuación 9)}$$

Esta matriz ha sido obtenida de forma empírica y ha sido usada con buenos resultados (pero pueden usarse otras matrices concretas)[2]. Para representar la imagen comprimida, (es decir para recuperar la matriz original, aproximadamente) se multiplica R por Q elemento a elemento y este producto es el que se antitransforma.

Actividad: Considera la siguiente matriz (que genera la imagen de la derecha)

$$A = \begin{bmatrix} 148 & 146 & 148 & 148 & 148 & 147 & 146 & 148 \\ 27 & 30 & 40 & 60 & 90 & 138 & 145 & 146 \\ 20 & 22 & 21 & 22 & 20 & 24 & 84 & 146 \\ 66 & 60 & 38 & 23 & 24 & 20 & 23 & 72 \\ 148 & 147 & 146 & 125 & 47 & 22 & 22 & 23 \\ 148 & 148 & 144 & 146 & 144 & 44 & 20 & 20 \\ 148 & 145 & 148 & 147 & 146 & 99 & 25 & 22 \\ 147 & 146 & 148 & 146 & 148 & 122 & 20 & 21 \end{bmatrix}.$$



1) Usa los comandos `uint8` y `imwrite` para generar esta imagen.
2) Calcula la transformada coseno bidimensional de A y guárdala en B.
3) Guarda la matriz Q definida en la Ecuación 9.
4) Calcula la matriz cuya entrada (r, s) es B_{rs}/Q_{rs} (acuérdate que en Octave, la división entrada a entrada se hace con `./`). Por último redondea las entradas de esta matriz al entero más próximo obteniendo la matriz R (usa el comando `round`).

Es en este último paso cuando se ha perdido información. Pero observa que la matriz R solo tiene 21 entradas no nulas. Así, en vez de almacenar 64 entradas numéricas, solo tenemos que almacenar 21. ¡Una reducción considerable!

Ahora tenemos que "deshacer el proceso" para recuperar la imagen (o matriz) original. Eso sí, un poco distorsionada.

5) Multiplica entrada a entrada R por Q.
6) Aplica la antitransformada coseno bidimensional a este último resultado.
7) Redondea las entradas de la última matriz.
8) Obtén el fichero JPG y compáralo con la imagen original.

[2] Esta matriz ha sido copiada de las recomendaciones de la *International Telecommuncation Union*, en el anexo K de `http://www.w3.org/Graphics/JPEG/itu-t81.pdf`.