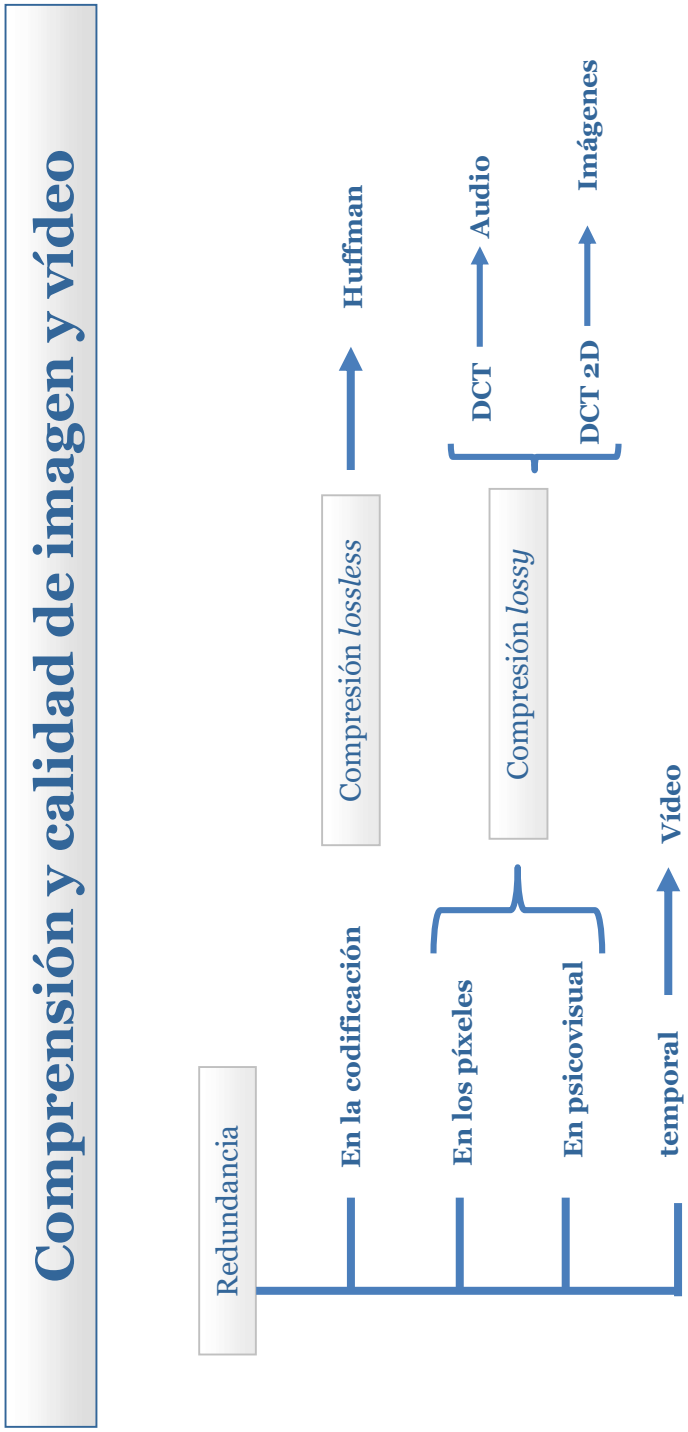
Compresión y calidad de imagen y vídeo

TEMA 12

Esquema



Ideas clave

12.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las **Ideas clave** que encontrarás a continuación.

En este tema vamos a ver cómo se comprimen las imágenes, incluyendo el uso de la DCT. Para acabar veremos cómo se comprimen secuencias de vídeo.

12.2. Redundancia en la imagen

La forma más sencilla de almacenar una imagen es almacenar todos los píxeles, fila a fila.

El problema de esta representación es que ocupa mucho espacio. Por ello, la mayoría de las imágenes se almacenan en formato comprimido.

Los métodos de compresión de imagen se dividen en dos grandes grupos:

- **Compresión sin perdidas** (*lossless*). Cuando la descompresión de la imagen recupera la imagen original sin cambios.
- **Compresión con pérdidas** (*lossy*). Cuando se tolera un pequeño grado de deterioro visual en la imagen descomprimida a cambio de un mayor ratio de compresión.

Como regla general, la compresión sin pérdidas se utiliza para imágenes sintéticas donde la solidez de las formas hace fácil percibir el deterioro de la imagen. Por el contrario, la compresión con pérdidas se utiliza para imágenes fotográficas donde el ojo humano no es capaz de diferenciar las pérdidas producidas por la compresión. La excepción a esta regla son las fotografías médicas, en las cuales no se tolera ninguna pérdida de información.

Para demostrar este criterio, la figura 1 muestra una imagen sintética y otra fotográfica y el resultado de comprimirla con calidad 10% usando GIMP. Podemos ver cómo en la imagen sintética sí se percibe un deterioro de imagen, mientras que en la imagen fotográfica el deterioro no es perceptible.

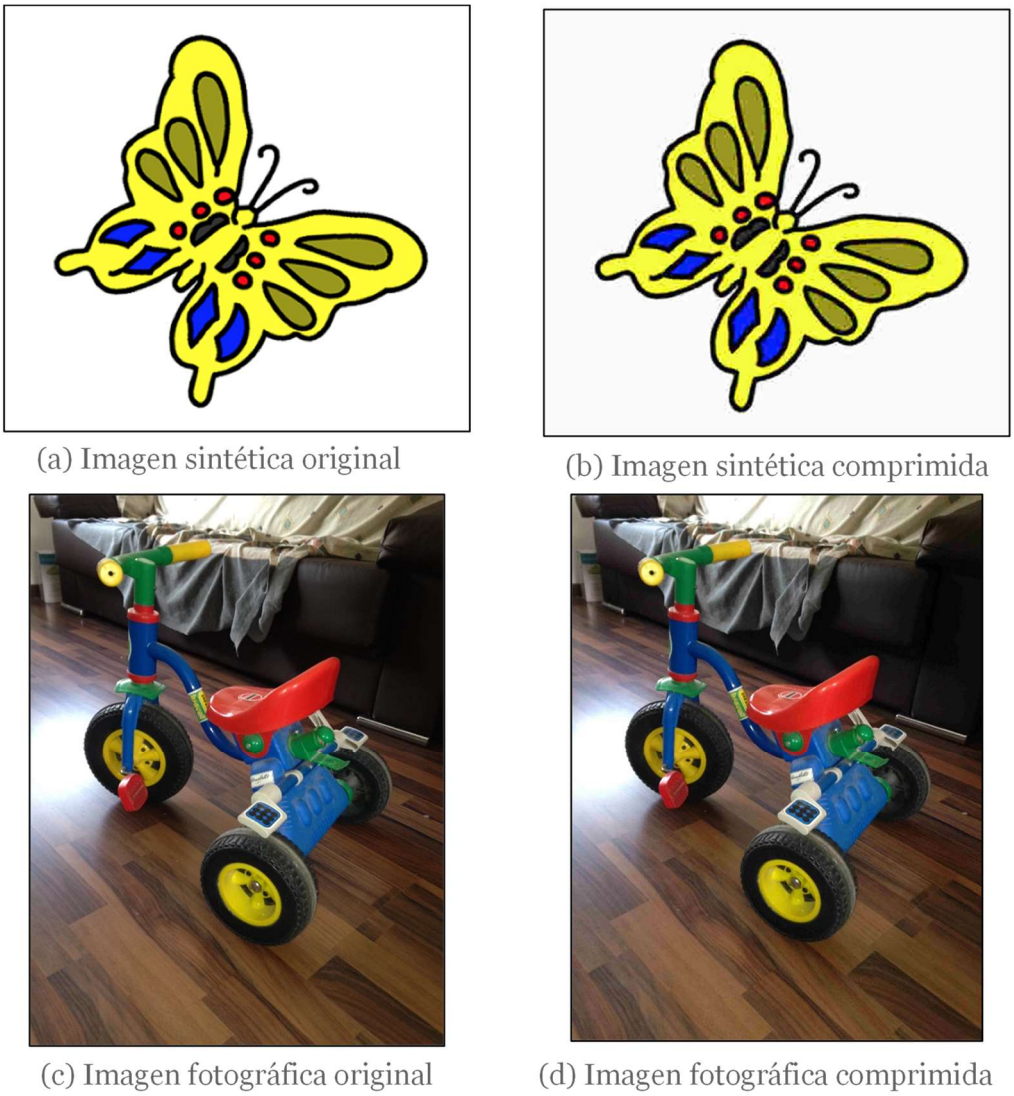


Figura 1: compresión de imagen sintética y fotográfica

Medidas de la compresión

Llamaremos **ratio de compresión** CR (*Compression Ratio*) a la proporción de reducción de datos que experimentan los datos comprimidos respecto a los datos originales. Si s es el tamaño de la imagen original y s_c es el tamaño de la imagen comprimida, el ratio de compresión será:

$$CR = \frac{s}{s_c}$$

La compresión de imágenes es un caso particular de la compresión de datos. En general, la información se puede comprimir si existe redundancia. Matemáticamente, la **redundancia** *R* se define como:

$$R = 1 - \frac{1}{CR}$$

De estas fórmulas podemos deducir que:

» Si no hay compresión, es decir, $s=s_c$, $CR=1$, con lo que la redundancia es $R=0$.
» Si hay una alta tasa de compresión, es decir $s \gg s_c$, $CR \to \infty$ y $R \to 1$.

Esto quiere decir que el rango que toman estas magnitudes es: $CR \in [1, \infty)$ y $R \in [0,1)$.

Compresión de imágenes con Octave

La operación *imwrite* de Octave puede recibir el parámetro *'quality'* para indicar la calidad de compresión JPEG de una imagen. Este valor debe darse en el rango 0..100.

Vamos a utilizar esta operación para comprimir la imagen *logo_unir.png* que aparece en la figura 2 (a). Para ello ejecutamos los siguientes comandos:

```
I = imread('logo_unir.png');
imwrite(I, 'logo_unir_50.jpg', 'quality', 50);
imwrite(I, 'logo_unir_10.jpg', 'quality', 10);
imwrite(I, 'logo_unir_0.jpg', 'quality', 0);
```

La figura 2 muestra los resultados obtenidos. Los tamaños de cada uno de los ficheros son los siguientes:

```
$ ll
-rw-r--r--@ 1 flopez  staff  34733 Feb  9 17:37 logo_unir.png
-rw-r--r--  1 flopez  staff   4759 Feb  9 17:38 logo_unir_0.jpg
-rw-r--r--  1 flopez  staff   6834 Feb  9 17:38 logo_unir_10.jpg
-rw-r--r--@ 1 flopez  staff  12447 Feb  9 17:38 logo_unir_50.jpg
```


Figura 2: ratios de compresión con Octave

La tabla 1 resume los niveles de calidad JPEG aplicados, el tamaño de cada fichero y el ratio de compresión CR alcanzado.

Nivel calidad JPEG	Tamaño fichero	Ratio de compresión (CR)
100	34733	34733/34733=1.000
50	12447	12447/34733=0.358
25	6834	6834/34733=0.196
0	4759	4759/34733=0.1370

Tabla 1: ratios de compresión

12.3. Tipos de redundancias en la imagen

Las técnicas de compresión de imágenes explotan tres tipos de redundancias:

- **Redundancia en la codificación**. Se produce porque los valores de los píxeles que representar la imagen no están uniformemente distribuidos.
- **Redundancia entre píxeles**. Se produce porque existen correlaciones entre píxeles cercanos.
- **Redundancia psicovisual**. Se produce cuando existe información que no es relevante para el sistema visual humano.

A continuación vamos a estudiar cada uno de estos tipos de redundancia.

Redundancia en la codificación

La redundancia en la codificación se produce cuando los símbolos a codificar tienen diferentes probabilidades. Por ejemplo, en una imagen a escala de grises donde predominan los colores claros, la probabilidad de los píxeles claros es mayor a la de los píxeles oscuros. Se llama **codificador entrópico** a un codificador que elimina la redundancia en la codificación.

En estos casos los símbolos se pueden comprimir representándolos con **códigos de longitud variable**. El **codificador de Huffman** es el más conocido para este tipo de redundancia.

Para conocer esta técnica tendrás que visualizar la lección magistral de este tema.

Los codificadores que aprovechan la redundancia en la codificación suelen ser *lossless*.

Redundancia entre píxeles

La redundancia entre píxeles se produce porque los píxeles cercanos tienden a contener valores cercanos.

A la redundancia entre píxeles también se le llama **redundancia espacial**. En el caso del vídeo también se ha observado que existe redundancia entre píxeles de la misma posición pero en instantes de tiempo (*frames*) contiguos. En este caso, a la redundancia se la llama **redundancia temporal**.

La redundancia entre píxeles se puede explotar de varias formas:

» **Diferenciación**. En vez de almacenar el valor de cada píxel, almacenamos la diferencia entre un píxel y el anterior.
» **Predicción**. Podemos predecir el valor de un píxel en función de los valores adyacentes anteriores y después almacenar la diferencia entre el valor predicho y el muestreado.

Para comprimir sonido también podíamos almacenar la diferencia entre muestras (codificación DPCM) y la predicción de una muestra (codificación ADPCM).

» **Ejercicio**.

Utilizar Octave para calcular la forma de almacenar de forma diferencial las siguientes 24 muestras de píxeles adyacentes y mostrar cada una de las secuencias de datos con el comando *stem*:

I = [12, 17, 14, 19, 21, 26, 23, 29, 41, 38, 31, 44, 46, 57, 53, 50, 60, 58, 55, 54, 52, 51, 56, 60];

Las 23 diferencias obtenidas son:

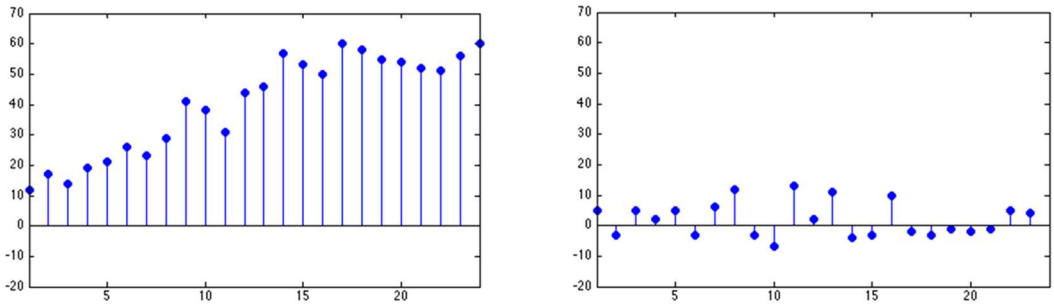
5, -3, 5, 2, 5, -3, 6, 12, -3, -7, 13, 2, 11, -4, -3, 10, -2, -3, -1, -2, -1, 5, 4

Tener en cuenta que para poder deshacer la operación, el primer valor de *I* hay que almacenarlo también, con lo que realmente tendríamos que almacenar estos 24 valores:

12, 5, -3, 5, 2, 5, -3, 6, 12, -3, -7, 13, 2, 11, -4, -3, 10, -2, -3, -1, -2, -1, 5, 4

La figura 3 muestra el resultado de pintar ambas secuencias con *stem*. Observando estas secuencias podemos sacar dos conclusiones:

- El valor de las diferencias es menor al valor a los valores originales.
- Las diferencias están **decorreladas**, es decir, la relación entre valores adyacentes ha sido eliminada.



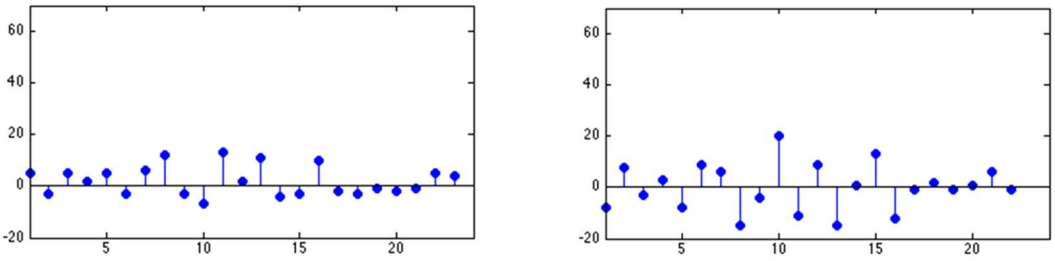
(a) Secuencia original (b) Secuencia diferencial

Figura 3: secuencia de datos almacenada de forma diferencial

Para ver que realmente las diferencias de la figura 3 (b) han sido decorreladas volvemos a aplicar el algoritmo de diferenciación, tras el cual obtenemos:

-8, 8, -3, 3, -8, 9, 6, -15, -4, 20, -11, 9, -15, 1, 13, -12, -1, 2, -1, 1, 6, -1

La figura 4 muestra la primera y segunda diferencia. Podemos apreciar que no hay una reducción clara en los valores de la segunda diferencia. Esto se debe a que las muestras han sido decorreladas.



(b) Primera diferencia (b) Segunda diferencia

Figura 4: segunda diferenciación

Un corolario de esta última observación es que no podemos comprimir imágenes ya comprimidas.

Los codificadores que aprovechan la redundancia entre píxeles pueden ser *lossless* o *lossy*, es decir, el algoritmo de descompresión puede recuperar los valores originales, o valores parecidos.

Redundancia psicovisual

Diferentes experimentos sobre la capacidad del sistema visual humano han concluido que el ojo humano:

- No es capaz de percibir determinadas informaciones almacenadas en las imágenes digitales.
- Percibe mejor la pérdida de algunas informaciones que otras.

Conocer cuándo el ojo humano no tiene agudeza suficiente para percibir diferencias en una imagen digital permite comprimir una imagen con la misma calidad. Conocer la información más difícil de percibir por el ojo humano permite crear sistemas que comprimen todavía más la imagen, a cambio de pequeñas perdidas de calidad.

Los codificadores que aprovechan la redundancia psicovisual son por naturaleza *lossy*, es decir, el algoritmo de descompresión no recupera exactamente la misma información.

12.4. La DCT

La DCT (*Discrete Cosine Transform*) es una representación en el dominio transformado muy usada porque permite comprimir una señal de tiempo discreto en un conjunto reducido de coeficientes. La DCT es una herramienta fundamental del estándar de compresión de vídeo MPEG y del estándar de compresión de imágenes JPEG. Además MP3 (MPEG-1 Layer 3) también utiliza una versión modificada de la DCT para comprimir la señal de audio.

Relación con otras transformadas

Se llama **transformada de longitud fija** a una transformación de una señal de tiempo discreto de longitud N en una representación frecuencial muestreada con N coeficientes.

La DFT y la DCT son ejemplos de transformadas de esta clase. Otros ejemplos de este tipo de transformadas son la transformada de *Haar*, la transformada de *Hadamard* y la transformada de *Hartley*.

Todas estas transformadas tienen una ecuación de análisis de la forma:

$$X[m] = \sum_{n=0}^{N-1} x[n]\phi_m^*[n]$$

Y una ecuación de síntesis de la forma:

$$x[n] = \frac{1}{N}\sum_{m=0}^{N-1} X[m]\phi_m[n]$$

Donde a $\phi_m[n]$ se le llama la **secuencia base** y sus valores son ortonormales entre sí.

La ortonormalidad significa que para todo $m,k \in [0,N-1]$:

$$\frac{1}{N}\sum_{n=0}^{N-1} \phi_m[n]\phi_m^*[n] = \begin{cases} 1, & m = n \\ 0, & m \neq n \end{cases}$$

En el caso de la DFT las secuencias base son las siguientes secuencias sinusoidales complejas:

$$\phi_m[n] = e^{j\frac{2\pi}{N}mn}$$

$$\phi_m^*[n] = e^{-j\frac{2\pi}{N}mn}$$

En el caso de la DFT los coeficientes $X[m]$ son en general complejos, incluso si la secuencia $x[n]$ es real. En cambio, la DCT es una transformada real pura porque tanto la señal como sus coeficientes se representan solo con números reales.

Definición de la DCT

La DCT es una transformada de longitud fija cuyas secuencias base $\phi_m[n]$ son cosenos a diferentes frecuencias. Existen varios tipos de DCT (habitualmente llamadas DCT-1, DCT-2, …, DCT-8) aunque nosotros vamos a estudiar solo la DCT-2 por la propiedad que presenta de compresión mediante compactación de coeficientes.

La ecuación de análisis de la DFT es:

$$X[m] = DCT\{x[n]\} = \beta[m] \sum_{n=0}^{N-1} x[n] \cos\left(\frac{(2n+1)\pi}{2N} m\right) \quad \text{con} \quad m = 0, 1, .., N-1 \quad (1)$$

Y la ecuación de síntesis de la DFT es:

$$x[n] = IDCT\{X[m]\} = \sum_{m=0}^{N-1} \beta[m] X[m] \cos\left(\frac{(2n+1)\pi}{2N} m\right) \quad \text{con} \quad n = 0, 1, .., N-1 \quad (2)$$

Donde $\beta[m]$ es un factor de escalado que está distribuido entre la ecuación de análisis y de síntesis tomando el valor:

$$\beta[m] = \begin{cases} \sqrt{\frac{1}{N}}, & m = 0 \\ \sqrt{\frac{2}{N}}, & m \neq 0 \end{cases} \quad (3)$$

Estos coeficientes están especialmente elegidos para cumplir la relación de Parseval:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{m=0}^{N-1} |X[m]|^2$$

» **Ejemplo 1**: cálculo de los coeficientes de la DCT.

Calcular los coeficientes de la DCT de la siguiente señal y comprobar que se cumple la relación de Parseval:

$$x[n] = \{1, -2, 1, 3\}$$

Aplicando la ecuación de análisis de la DCT (1) tenemos:

$$X[m] = \beta[m] \sum_{n=0}^{N-1} x[n] \cos\left(\frac{(2n+1)\pi}{2N} m\right)$$

$$= \beta[m] \left[\cos\left(\frac{\pi m}{8}\right) - 2\cos\left(\frac{3\pi m}{8}\right) + \cos\left(\frac{5\pi m}{8}\right) + 3\cos\left(\frac{7\pi m}{8}\right)\right]$$

Y evaluando en m=0, …, m=N-1 tenemos:

$$X[m] = \begin{cases} 3/2, & m = 0 \\ -3/\sqrt{2}, & m = 1 \\ 5/2, & m = 2 \\ \sqrt{2}, & m = 3 \end{cases}$$

Luego la energía en el dominio del tiempo es:

$$E_x = \sum_{n=0}^{N-1} |x[n]|^2 = 1 + 4 + 1 + 9 = 15$$

Y la energía en el dominio de la frecuencia es:

$$E_X = \sum_{m=0}^{N-1} |X[m]|^2 = \frac{1}{4}\left[\left(\frac{3}{2}\right)^2 + \left(\frac{-3}{\sqrt{2}}\right)^2 + \left(\frac{5}{2}\right)^2 + \left(\sqrt{2}\right)^2\right] = 15$$

Cálculo con Octave

Podemos usar el comando *dct(x)* para calcular la DCT de una secuencia y el comando *idct(X)* para obtener la transformada inversa.

Por ejemplo, para obtener los coeficientes de la DCT del 0 podemos hacer:

```
> x = [1 -2 1 3];
> X = dct(x)
 X =    1.5000  -2.1184   2.5000   1.4186
```

Análogamente podemos comprobar la relación de Parseval haciendo:

```
> sum(x.^2)
 ans =  15
> sum(X.^2)
 ans =  15.000
```

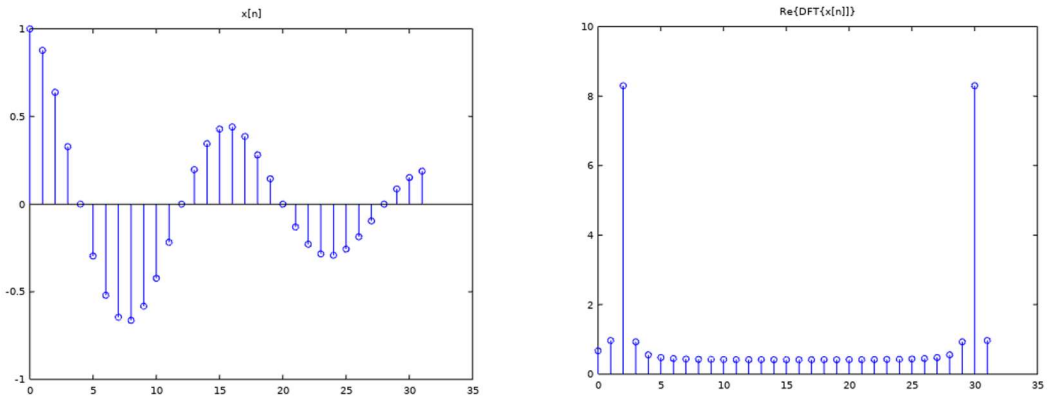
Compactación de coeficientes

La DCT se usa en aplicaciones de compresión por la propiedad de compactación de coeficientes. Esta propiedad hace que la mayoría de la energía se concentre en los primeros coeficientes.

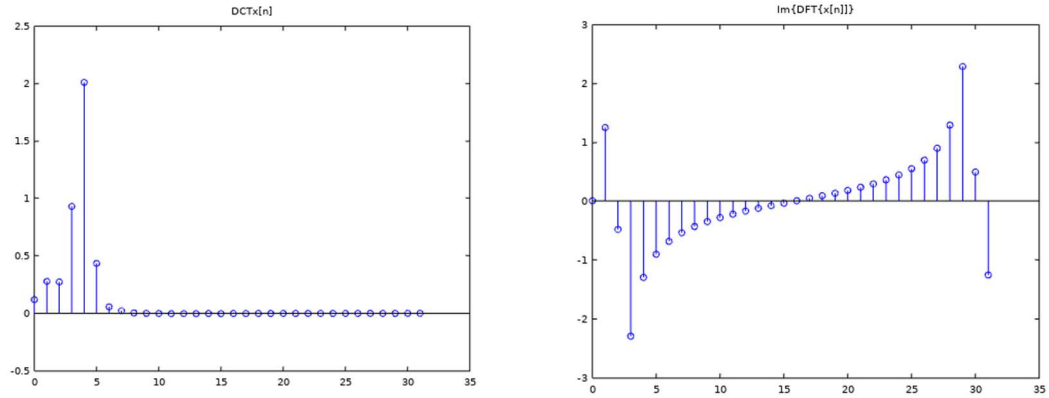
La figura 5 (a) muestra la señal:

$$x[n] = 0.95^n \cos\left(\frac{\pi}{8}n\right) \quad \text{con} \quad N = 32$$

Y la figura 5 (b) muestra sus coeficientes de la DCT donde podemos ver que la mayoría de la energía se concentra en los primeros 7 coeficientes.



(a) Señal $x[n]$ (c) Coeficientes Re{DFT{$x[n]$}}



(b) Coeficientes DCT{$x[n]$} (d) Coeficientes Im{DFT{$x[n]$}}

Figura 5: comparación entre coeficientes de la DFT y la DCT

Por comparación, la figura 5 (c) y (d) muestran la parte real e imaginaria de la DFT de la misma señal. En este caso solo es necesario almacenar los primeros 16 coeficientes, ya que los otros 16 coeficientes son simétricos conjugados, por ser x[n] real.

Observar que la DCT nos permite almacenar la señal guardando 7 coeficientes frente a los 16+16=32 coeficientes que necesitamos almacenar en la DFT.

Una de las propiedades más importantes de la DFT es que la convolución periódica en el tiempo coincide con el producto de los coeficientes en frecuencia, lo cual la hace ideal para implementar filtros. En cambio, los coeficientes de la DCT son más adecuados para comprimir la representación de señales.

Mecanismo de compresión

El algoritmo de compresión suele implicar:

- **Normalización**. Habitualmente la señal $x[n]$ de audio o imagen corresponde a valores enteros (típicamente de -128..127), mientras que los coeficientes son valores reales con cualquier rango. Para normalizar debemos encontrar el valor máximo y mínimo de los coeficientes y escalarlos al rango -128..127.
- **Redondeo**. Los coeficientes normalizados son valores reales. La compresión implica cuantificar los coeficientes: los primeros coeficientes se redondean al entero más cercano y los siguientes coeficientes se redondean a cero.
- **Partición**. Los datos se suelen particionar en secuencias de *N* elementos y la DCT se aplica a cada secuencia de forma individual.

El siguiente ejemplo muestra cómo se redondean los coeficientes y aun así la señal original se puede reconstruir. Esto se debe a que la mayoría de la información se concentra en los primeros coeficientes.

- **Ejemplo 2**: redondeo y reconstrucción.

  Aplicar la DCT a la siguiente señal, redondear los coeficientes y reconstruir la señal:

$$x[n] = \{12,10,15,10,15,13\}$$

  Si usamos Octave para calcular la DCT de esta señal tenemos:

```
> x = [12,10,15,10,15,13];
> X = dct(x)
 X =  30.61862  -1.85177   0.00000 -0.40825  0.00000  4.68020
```

  En este ejemplo no es necesario normalizar ya que todos los valores se encuentran en el rango -128..127, luego redondeamos los coeficientes:

```
> Xr = int8(X)
 Xr =   31  -2   0   0   0   5
```


Y finalmente reconstruimos la señal a partir de los coeficientes redondeados:

```
> x2 = idct(double(Xr))
 x2 =  12.2875  9.7980  15.1452  10.1662  15.5134  13.0239
```

Si ahora volvemos a redondear los valores de la señal reconstruida:

```
> int8(x2)
 ans =  12  10  15  10  16  13
```

Vemos que solo ha cambiado el quinto coeficiente en 1 unidad.

Altas y bajas frecuencias

En la ecuación de análisis de la DCT (1) podemos observar que el coseno que actúa como secuencia base aumenta de frecuencia a medida que aumenta *m*. Esto quiere decir que los primeros coeficientes representan las bajas frecuencias y los últimos coeficientes las altas frecuencias.

12.5. Compresión de imágenes con la DCT

La DCT es especialmente útil para compresión *lossy* cuando existen correlaciones (redundancia) entre valores contiguos. Hemos visto que las imágenes presentan alto nivel de correlación espacial entre píxeles cercanos en 2D. Por esta razón en esta sección vamos extender la DCT a 2D. El vídeo añade correlación temporal entre *frames* adyacentes con lo que presenta una redundancia 3D.

Ecuaciones de la DCT 2D

Dada una **matriz de píxeles** con *M* filas y *N* columnas *x*[*m*,*n*] obtenemos a una **matriz de coeficientes** *X*[*p*,*q*] con *M* filas y *N* columnas aplicando la siguiente ecuación de análisis de la DCT en 2D:

$$X[p,q] = DCT2\{x[m,n]\}$$
$$= \beta[p]\beta[q]\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[m,n]\cos\left(\frac{(2m+1)\pi}{2M}p\right)\cos\left(\frac{(2n+1)\pi}{2N}q\right) \quad (4)$$

Donde β es el factor de escalado (3) donde N puede ser M o N, respectivamente, es decir:

$$p = 0, 1, \ldots, M-1 \qquad \beta[p] = \begin{cases} \sqrt{\frac{1}{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & p \neq 0 \end{cases} \qquad \beta[q] = \begin{cases} \sqrt{\frac{1}{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & q \neq 0 \end{cases}$$
$$q = 0, 1, \ldots, N-1$$

Análogamente, la ecuación de síntesis en 2D es:

$$x[m,n] = IDCT2\{X[p,q]\}$$
$$= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1} \beta[p]\beta[q]\cos\left(\frac{(2m+1)\pi}{2M}p\right)\cos\left(\frac{(2n+1)\pi}{2N}q\right) \quad (5)$$

Con:
$$m = 0, 1, \ldots, M-1$$
$$n = 0, 1, \ldots, N-1$$

Algoritmo de la DCT 2D

La DCT es una transformación separable, lo cual significa que la transformada en 2D es equivalente a la transformada DCT en 1D a lo largo de una dimensión, seguida de otra transformación DCT en 1D a lo largo de la otra dimensión. Esta propiedad simplifica el cálculo de la DCT en 2D mediante dos operaciones de cálculo de la DCT en 1D.

Habitualmente la imagen se divide en matrices de 8x8 píxeles llamadas **bloques de píxeles**. Si la última fila o columna no son divisibles entre 8 se duplican estas filas o columnas el número de veces necesarias para hacerlas divisibles.

Cada bloque de píxeles se pasa por la ecuación de análisis para obtener un **bloque de coeficientes**. Dado que el bloque de píxeles contiene números reales se redondean al entero más cercano. Para deshacer esta operación se aplica la operación de síntesis de la DCT en 2D a cada bloque de coeficientes redondeados.

Cálculo con Octave

Octave dispone de la siguiente operación para aplicar a una matriz de píxeles *x* la ecuación de análisis de la DCT en 2D:

```
X = dct2(x);
```

Y para aplicar la ecuación de síntesis en 2D a la matriz de coeficientes *X* tenemos:

```
x = idct2(X);
```

» **Ejemplo 3**: DCT sobre un bloque.

Dada la imagen de 8x8 píxeles de la figura 6 (a) podemos obtener sus coeficientes de la DCT 2D haciendo:

```
x = imread('fi.png');
X = dct2(x)
```

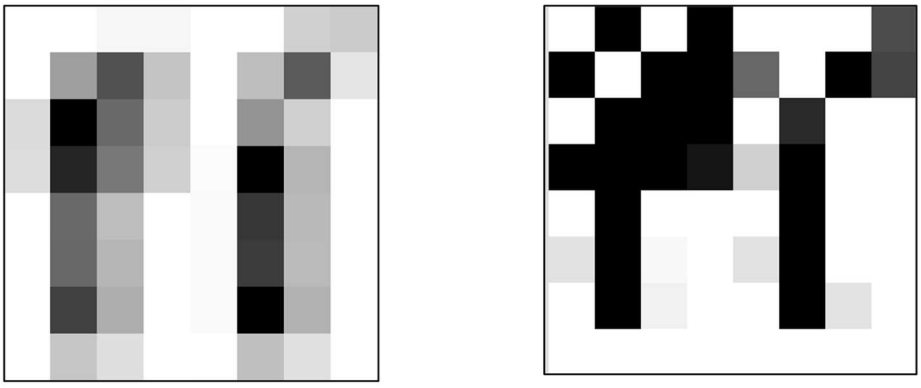
La tabla 2 (a) muestra el bloque de píxeles y la tabla 2 (b) su correspondiente bloque de coeficientes.

Los coeficientes están calculados fuera del rango -128..127 de un *byte*, con lo que para su almacenamiento usamos el comando *int8()* el cual los redondea y normalizada a este rango tal como muestra la tabla 2 (c):

```
Xn = int8(X);
```

Por último podemos recuperar el bloque de píxeles volviendo a normalizar al rango original de la tabla 2 (b) y aplicando la IDCT:

```
M = max(max(X));
m = min(min(X));
range = M-m;
Xr = (double(Xn)+128)*(range/255)+m;
xr = uint8(idct2(Xr));
```

(a) Imagen original (b) Imagen recuperada

Figura 6: resultado de aplicar la DCT 2D

255	253	245	245	255	255	208	203
255	159	81	196	255	190	91	229
219	0	106	204	254	148	208	255
220	37	120	208	251	0	181	255
255	105	189	255	249	56	185	255
254	104	182	255	249	60	186	255
255	66	174	255	249	0	178	255
255	198	223	255	253	192	224	255

(a) Bloque de píxeles original

1552.8	-41.9	17.8	-39.2	401.8	87.6	-23.6	139.3
-9.8	-16.7	-18.5	108.3	-55.7	-67.2	36.4	-67.4
152.2	62.4	-25.4	56.9	-142.5	-69.5	28.4	-90.6
62.0	91.0	-9.4	-48.0	-30.4	-5.2	-15.3	-2.4
96.0	38.3	-8.7	-27.8	-88.5	-7.5	-15.1	-1.8
-43.8	-22.9	-0.8	-49.0	30.3	59.6	-24.5	71.0
72.8	-43.5	-19.9	17.0	-70.3	29.8	21.6	-2.9
25.7	-21.1	-14.1	-4.7	-10.4	40.1	18.1	14.5

(b) Bloque de coeficientes

127	-42	18	-39	127	88	-24	127
-10	-17	-18	108	-56	-67	36	-67
127	62	-25	57	-128	-69	28	-91
62	91	-9	-48	-30	-5	-15	-2
96	38	-9	-28	-89	-8	-15	-2
-44	-23	-1	-49	30	60	-25	71
73	-43	-20	17	-70	30	22	-3
26	-21	-14	-5	-10	40	18	15

(c) Bloque de coeficientes normalizados y redondeados

255	0	255	0	255	255	255	76
0	255	0	0	103	255	0	68
255	0	0	0	255	43	255	255
0	0	0	23	208	0	255	255
255	0	255	255	255	0	255	255
226	0	247	255	226	0	255	255
255	0	241	255	255	0	227	255
255	255	255	255	255	255	255	255

(d) Bloque de píxeles restaurado

Tabla 2: DCT 2D de un bloque

La tabla 2 (d) y la figura 6 (b) muestran la señal restaurada *xr*. Podemos observar que:

» Los valores restaurados no son los mismos, aunque son suficientemente similares como para asociar psicovisualmente las imágenes de la figura 6 (a) y la figura 6 (b).
» Ninguno de los coeficientes de la tabla 2 (c) se ha hecho cero, lo cual reduce claramente la capacidad de compresión de este algoritmo. ¿A qué piensas que se debe? Hablaremos sobre esto.

Altas y bajas frecuencias

Al igual que ocurre con la DCT en 1D, a medida que aumentan los **índices de los coeficientes** *p,q* en la ecuación de análisis de la DCT en 2D (4) aumentan las frecuencias de los cosenos correspondientes. Esto quiere decir que la DCT en 2D transforma un bloque de píxeles 8x8 en una combinación lineal de 64 patrones en los que va aumentando la frecuencia horizontal y vertical según aumenta el índice de los coeficientes. La figura 7 muestra estos patrones de los coeficientes a los que se suele llamar **funciones base de la DCT**.

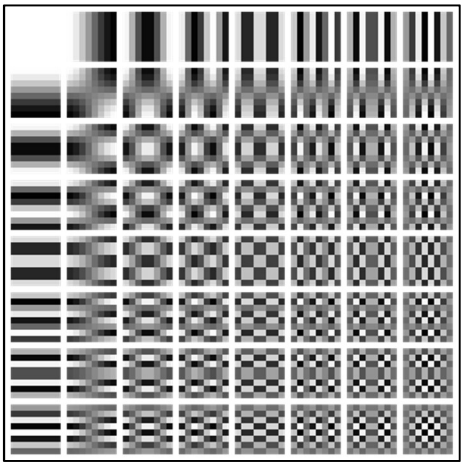


Figura 7: Patrones de los coeficientes de la DCT 2D para 8x8

12.6. Compresión de la redundancia temporal

Hasta ahora hemos vimos que la **redundancia espacial** se produce porque píxeles cercanos en la imagen tienden a tener valores similares. En el caso del vídeo, además de redundancia espacial dentro del *frame*, se produce una **redundancia temporal** debida a que *frames* contiguos en el tiempo tienden a ser similares.

(a) Frame *n* (b) Frame *n*+1 (c) Frame diferencia

Figura 8: Codificación diferencial de *frames* contiguos

La redundancia temporal puede ser eliminada mediante una codificación diferencial en la que los valores de un *frame* se almacenan como la diferencia entre un *frame* y el *frame* anterior.

La figura 8 (a) y (b) muestra dos *frames* consecutivos. Figura 8 (c) muestra el resultado de restar al valor del *frame* (b) el *frame* (a). En esta imagen diferencia 0 corresponde a blanco y 255 a negro. Podemos ver que la diferencia entre píxeles se aproxima a *0*, lo cual indica que el valor de las diferencias es menor al valor a los valores originales. Este resultado permite usar un codificador entrópico para comprimir la diferencia resultante.

En la figura 8**¡Error! No se encuentra el origen de la referencia.** (c), las uñas de la tortuga corresponden a un movimiento de la escena y los bordes del tortuguero corresponden a un movimiento de la cámara. Los cambios de iluminación también contribuyen a estas diferencias.

» **Ejercicio**.

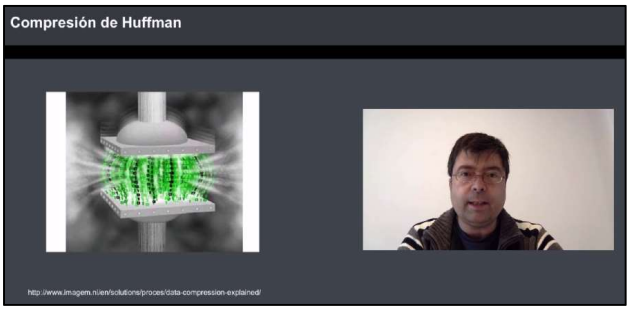
Identificar las rutinas Octave que, dados dos *frames* consecutivos, generan la imagen diferencia de la figura 8 (c).

Lo + recomendado

Lecciones magistrales

Compresión de Huffman

En esta lección magistral vamos a ver cuáles son los principios en los que se basa este popular método de codificación *lossless*.



Accede a la lección magistral a través del aula virtual.

No dejes de leer…

***Video-encoders*: ¿qué es un codificador de vídeo?**

En este artículo describe los codificadores de vídeo dentro de los circuitos cerrados de TV para videovigilancia.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web: http://www.axis.com/mx/es/learning/web-articles/technical-guide-to-network-video/what-is-a-video-encoder

Principios y finales en la codificación de la señal de vídeo digital

En este artículo trata más a fondo la evolución histórica y los principios de la compresión de vídeo.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.razonypalabra.org.mx/anteriores/n45/agarcia.html

No dejes de ver...

Compresión, formatos y Codecs de vídeo, todo lo que debes saber

Este tutorial describe los distintos tipos de *frames* y los estándares donde se utilizan.

Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
https://www.youtube.com/watch?v=vydPJ_iaQ8o

+ Información

A fondo

Fundamentos para el procesamiento de imágenes

Esqueda, J. J. (2005). *Fundamentos para el procesamiento de imágenes*. México: Universidad Autónoma de Baja California.

Puedes echar un vistazo a este libro en el que se describe a fondo el procesamiento y la compresión de imágenes.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://books.google.es/books?id=h4Gj8GuwPVkC

Bibliografía

Barni, M. (2006). *Document and image compression*. Reino Unido: Taylor & Francis.

Marqués, O. (2011). *Practical image and video processing using Matlab*. EE.UU.: Wiley.

Rabbani, M., Jones, P. (1991). *Digital image compression techniques*. EE.UU.: Spie.

Shukla. K. Prasad, M. (2011). *Lossy image compression: domain decomposition-based algorithms*. EE.UU.: Springer.

Test

1. ¿Dónde es más perceptible el efecto de un codificador *lossy*?

A. En imágenes claras.

B. En imágenes oscuras.

C. En imágenes sintéticas.

D. En imágenes naturales.

2. Si un compresor reduce una imagen al 40%, ¿qué nivel de redundancia tenía la imagen?

A. 40.

B. 0.4.

C. -1.5.

D. 0.6.

3. La similitud entre píxeles cercanos da lugar a:

A. Redundancia en la codificación.

B. Redundancia entre píxeles.

C. Redundancia psicovisual.

D. Todas las anteriores.

4. El hecho de que no todos los valores de píxel ocurren el mismo número de veces da lugar a:

A. Redundancia en la codificación.

B. Redundancia entre píxeles.

C. Redundancia psicovisual.

D. Todas las anteriores.

5. El codificador Huffman aprovecha:

A. La redundancia psicovisual.

B. La redundancia entre píxeles.

C. Códigos de longitud variable.

D. Ninguna de las anteriores.

6. ¿Cuándo consigue comprimir más un codificador diferencial?

A. Aplicándolo una vez.

B. Aplicándolo dos veces.

C. Aplicándolo tres veces.

D. Todas por igual.

7. ¿Cuál de estas no es una transformada de longitud fija?

A. DCT.

B. DFT.

C. DTFT.

D. Todas son de longitud fija.

8. ¿Qué transformada se usa habitualmente para comprimir?

A. DFT.

B. DCT -2.

C. DCT-3.

D. Los dos últimos.

9. ¿Dónde se produce la normalización en la compresión de imágenes con DCT?

A. Sobre el bloque de píxeles.

B. Sobre el bloque de coeficientes.

C. Sobre el bloque de errores.

D. En ninguno de los anteriores.

10. Los índices de los coeficientes de la DCT 2D representan:

A. Claros y oscuros.

B. Bajas frecuencias.

C. Altas frecuencias.

D. Bajas y altas frecuencias.

