7 El formato JPEG

7.1 Introducción

En un principio la mayoría de las aplicaciones informáticas eran muy simples gráficamente hablando y se ejecutaban en modo texto sin necesidad de presentar imágenes por pantalla. Con el paso del tiempo y, bien por necesidad o bien por simplificar las aplicaciones de cara al usuario, se han impuesto los sistemas operativos gráficos y con ellos las aplicaciones de tratamiento de imágenes en formato digital.

El principal problema para la mayoría de estas aplicaciones era la gran cantidad de memoria o espacio en disco requerida para almacenar imágenes digitales. La versión digitalizada de una imagen de color simple, a la resolución de una pantalla de TV contiene del orden de un millón de bytes. Si almacenamos estos bytes de información sin ningún tipo de compresión nos encontramos con que el tamaño de una imagen simple se dispara, por no hablar de imágenes de vídeo que no son más que sucesiones de imágenes estáticas.

Para solucionar este problema surgieron los métodos de compresión de imágenes digitales, que consiguen compresiones para imágenes típicas de 1:10 a 1:50 sin afectar la calidad de la imagen muy significativamente.

Además de conseguir desarrollar métodos de compresión eficaces, se hizo necesaria la creación de un estándar eficaz de compresión que evitara que cada fabricante introdujera su propio formato y el consiguiente problema de cara al usuario a la hora de compartir imágenes y usarlas en sus aplicaciones.

Es aquí donde entra en escena el formato JPEG, acrónimo que significa Joint Photographics Experts Group, en relación con la unión entre el CCITT y la ISO para buscar un formato común. Así, JPEG es tanto un estándar ISO como una recomendación del CCITT. Los textos de ambos son idénticos.

7.2 Requerimientos del formato

El objetivo de JPEG era desarrollar un método de compresión de imágenes que cumpliera ciertos requisitos:

1) ser un método plenamente configurable en cuanto a compresión y en cuanto a calidad de la imagen comprimida en relación con la imagen original
2) ser aplicable a prácticamente cualquier tipo de imagen sin importar sus dimensiones, número de espacios de colores, relación de aspecto…, etc.
3) Poseer una complejidad computacional relativa que permita su implementación en muchos tipos de CPU así como su implementación a nivel hardware a un coste viable para aplicaciones que requieran gran velocidad de proceso.
4) Tener los siguiente modos de operación:

    - Codificación secuencial (secuencial encoding): cada componente de la imagen es codificada en un "scan" simple de izquierda a derecha y de arriba abajo.
    - Codificación progresiva (progressive encoding): la imagen es codificada en varios scans para aplicaciones en las que el tiempo de transmisión es algo y el visualizador va presentando las imágenes con calidad progresiva según va decodificando cada scan.
    - Codificación sin pérdida (lossless encoding): la imagen es codificada de manera que se pueda recuperar exactamente cada muestra de la imagen original aunque resulte en una compresión muy baja comparada con los métodos con pérdida.
    - Codificación jerárquica (hierarchical encoding): la imagen es codificada a varias resoluciones de manera que las versiones de la imagen de resolución más baja puede ser obtenidas sin tener que decodificar la imagen a su resolución completa

El método más usual y prácticamente el que ha quedado como único de estos cuatro es el de codificación secuencial, que por otra parte es más que suficiente para cualquier aplicación.

7.3 Codificación secuencial

Las figuras 1 y 2 describen mediante bloques los procesos de codificación y decodificación respectivamente.

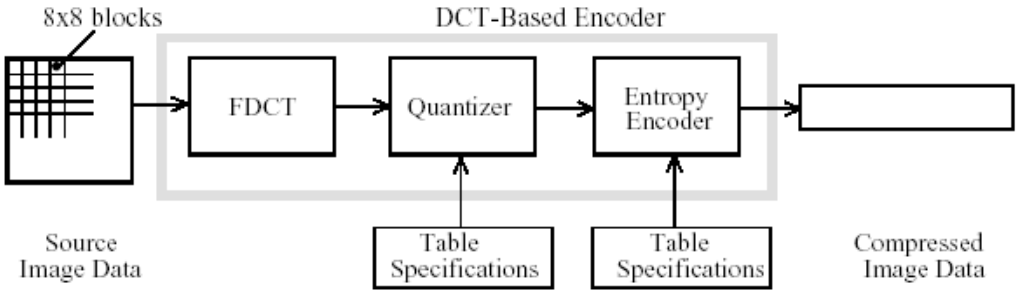


Figure 1. DCT-Based Encoder Processing Steps

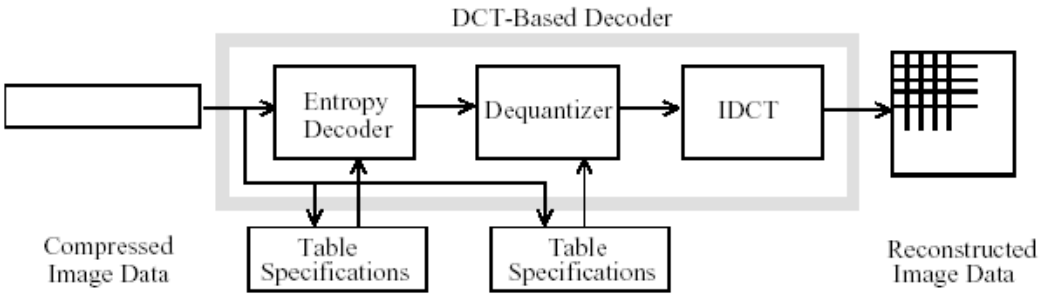


Figure 2. DCT-Based Decoder Processing Steps

Pasemos ahora a describir cada uno de los bloques que integran la codificación.

7.3.1 La Transformada Discreta del Coseno (DCT)

El formato JPEG basa su gran capacidad de compresión en una operación matemática denominada transformada discreta del coseno o DCT, que es aplicada a los puntos de la imagen, realmente a matrices de 8x8 píxels.

Esta operación tiene su inversa, la IDCT que reconstruye la matriz original a partir de la matriz resultado de aplicarle la DCT directa o FDCT (forward DCT). Las definiciones matemáticas de ambas operaciones son:

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

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

Donde: C(u), C(v) = $1/\sqrt{2}$ para u,v = 0;
C(u), C(v) = 1 e.o.c.

Como se puede intuir, la DCT está relacionada con la Transformada Discreta de Fourier (DFT). Cada bloque de 8x8 puntos de la imagen original es una señal discreta de 64 puntos que es función de las dos coordenadas espaciales x e y. La FDCT toma esta señal como entrada y la descompone en sus 64 componentes base ortogonales, cada una de las cuales contiene una de las 64 frecuencias espaciales 2D que componen el espectro de la señal de entrada. El resultado de la FDCT es el conjunto de las 64 amplitudes de las señales base. Estas amplitudes se conocen como coeficientes DCT y sus valores están determinados únicamente por los 64 puntos de la señal de entrada.

Los coeficientes de la DCT definen pues la variación relativa en 2D de las frecuencias espaciales contenidas en los 64 puntos de la señal de entrada. El coeficiente de frecuencia cero en ambas dimensiones es conocido como “coeficiente DC” o coeficiente de continua, y los 63 coeficiente restantes son los “coeficientes AC” o coeficientes de alterna.

Debido a que en una imagen fotográfica los valores de cada píxel varían suavemente a lo largo de los puntos de la imagen, la mayoría de los valores significativos en el espacio de la frecuencia se concentran en las bajas frecuencias, con lo que para un bloque de 8x8 píxels de una imagen típica, la mayoría de los valores de las frecuencias espaciales son cero o casi cero, con lo que no necesitan ser codificados, con el consiguiente ahorro de espacio.

En el decodificador se realiza el proceso inverso utilizando la IDCT, que toma los 64 coeficientes DCT y reconstruye los 64 puntos de la señal original sumando las señales base. Matemáticamente la DCT es una relación única entre vectores de 64 puntos de la imagen original y el dominio de la frecuencia.

Si la FDCT y la IDCT fueran calculadas con precisión perfecta se podría recuperar la señal original sin pérdida alguna, ya que idealmente, la DCT no introduce ninguna pérdida, sino que transforma la señal de entrada a un dominio, el de la frecuencia, donde puede ser codificada con mayor eficiencia. Esto es debido al caso particular de las imágenes habituales de tipo fotográfico en las que las variaciones de un punto a otro en la imagen se producen muy suavemente, de aquí que este formato sea el ideal para codificar imágenes de este tipo y sin embargo no dé resultados tan buenos con imágenes en las que la variación de un punto a otro sea mas brusca.

7.3.2 Cuantización

Después de aplicar la DCT a la señal original, y para conseguir una compresión mayor y ajustable, cada coeficiente DCT es cuantizado por su correspondiente coeficiente de cuantización tomado de una tabla de cuantización, que debe ser especificada por la aplicación (o el usuario) al codificador. Cada elemento de esta tabla es un entero entre 1 y 255.

El objetivo de la cuantización es conseguir mayor compresión representando los coeficientes DCT con una precisión no mayor de la necesaria para una determinada calidad de imagen, además de poder elegir precisamente esa calidad de imagen dándole mayor o menor valor a los coeficientes de la tabla de cuantización.

La cuantización de cada coeficiente DCT se realiza simplemente dividiendo este por el coeficiente de cuantización correspondiente y redondeando posteriormente al entero más cercano, por lo que esta operación es eminentemente con pérdidas, al no poder reconstruir exactamente los coeficientes originales simplemente realizando la operación inversa. La descuantización se realiza lógicamente multiplicando los coeficientes cuantizados por los coeficientes de cuantización utilizados en la cuantización.

Con esta operación, si por ejemplo un valor está muy cercano al cero, al cuantizarlo por un valor considerable y posteriormente redondear al entero más cercano, se transformaría en un cero, con lo que sería otro valor más que nos ahorramos al codificar.

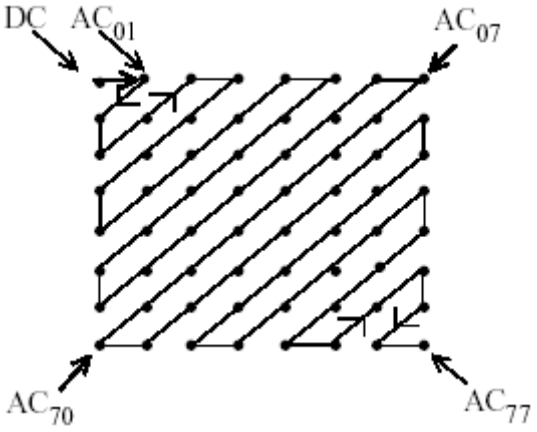
7.3.3 Codificación de coeficientes DC

El siguiente paso es codificar los 64 coeficientes uno a uno, pero hay que tener en cuenta que el coeficiente DC se trata de manera especial. Este coeficiente representa el valor medio de los 64 puntos. Debido a que habitualmente existe un gran correlación entre los coeficientes DC de bloques de 8x8 adyacentes, los coeficientes DC cuantizados se codifican como la diferencia entre el actual y el del bloque anterior.

Este tratamiento especial de los coeficientes DC se debe a que estos contienen habitualmente la mayor parte de la energía total de la imagen, con lo que codificando la diferencia se consigue comprimir aún más.

7.3.4 Secuencia zig-zag

Tras aplicar la DCT al bloque de 8x8 solo algunos coeficientes permanecen distinto de cero, en concreto los correspondientes a las frecuencias más bajas, y estos se hallan localizados en la esquina superior izquierda de la matriz de coeficientes DCT, luego a la hora de codificarlos se utiliza la ordenación en secuencia zig-zag como indica la figura, con lo que conseguimos que los coeficientes de baja frecuencia (los únicos que poseen valores distinto de cero generalmente) vayan delante en la secuencia, y posteriormente una secuencia de ceros hasta completar los 64 coeficientes.



7.3.5 Codificador de entropía

El proceso final de la codificación basada en la DCT es la codificación de entropía. En este paso se consigue aún mayor compresión sin pérdida codificando los coeficientes DCT cuantizados de una manera más compacta basándose en sus características analizadas de forma estadística.

JPEG especifica dos tipos de codificaciones de entropía: la codificación Huffman y la codificación aritmética. La codificación secuencial utiliza codificación Huffman, aunque se especifican codecs para ambos métodos.

La codificación de entropía se puede entender como un proceso de dos pasos. En el primer paso se convierte la secuencia en zig-zag de coeficientes cuantizados en una secuencia intermedia de símbolos. En el segundo paso se convierten estos símbolos en un flujo de datos en el que estos símbolos no son identificables externamente. La forma y la definición de los símbolos intermedios dependen del método de codificación.

La codificación Huffman requiere que la aplicación defina uno o más conjuntos de tablas de códigos Huffman. Las mismas tablas que se usen para comprimir la imagen se necesitan a la hora de descomprimir, por lo que deben ser almacenadas en el fichero de imagen jpeg.

Por el contrario, el método aritmético de codificación no necesita ninguna tabla como entrada, ya que es capaz de adaptarse a las estadísticas de la imagen cuando la codifica. Se ha comprobado que la codificación aritmética produce un compresión del orden del 5-10% mejor que la codificación Huffman, pero tiene como inconveniente una mayor complejidad computacional que ha hecho que la codificación Huffman se haya adoptado para la mayoría de las aplicaciones, sobre todo para implementaciones hardware de alta velocidad.

7.3.6 Imágenes con varias componentes

Hay que destacar que todo el proceso de codificación descrito anteriormente se realiza sobre una única componente de la imagen. En el caso de imágenes en blanco y negro la imagen solo necesita de la componente luminancia para representarse, pero en el caso de imágenes en color, que es el caso que nos ocupa, las imágenes constan de luminancia y crominancia.

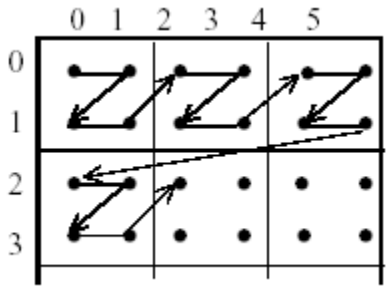
Realmente, para representar una imagen en color se podría utilizar el espacio de componentes denominado RGB, en el que cada punto está representado por tres valores del 0 al 255, cada uno definiendo la componente rojo (Red), verde (Green), azul (Blue). En el caso que nos ocupa, el formato JPEG utiliza el espacio de de colores YCbCr en el que se representan por un lado la luminancia Y y por otro las crominancias de azul y verde (Cb y Cr) únicamente. Ambos espacios son equivalentes y se puede pasar de uno a otro mediante las siguientes relaciones:

Y = 0.299R + 0.587G + 0.114B
Cb = -0.1687R - 0.3313G + 0.5B + 128
Cr = 0.5R – 0.4187G – 0.0813B + 128

R = Y + 1.402 (Cr – 128)
G = Y – 0.34414 (Cb – 128) – 0.71414 (Cr – 128)
B = Y + 1.772 (Cb – 128)

Por tanto, para codificar la imagen hay que codificar entrelazadas cada una de las tres componentes que forman cada punto. El formato da la posibilidad de utilizar mayor espacio para unas componentes que para otras. El caso más habitual es utilizar el doble de información para la luminancia que para las crominancias, ya que se demuestra empíricamente que la componente de luminancia posee mayor información que las otras dos en una imagen a color.

Así pues, en nuestro caso particular, para un bloque de 2x2 bloques de 8x8, cada bloque posee su propia información de luminancia pero sin embargo la información de crominancia es compartida entre los 4 bloques como se aprecia en la figura, donde también se especifica el orden que sigue cada bloque en la codificación.



7.3.7 Ejemplo para un bloque 8x8

A continuación se muestra un ejemplo del paso de un bloque de 8x8 muestras a través del codificador y por el decodificador para reconstruir el bloque original:

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(1) bloque original

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

(2) coeficientes DCT

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

(3) tabla cuantización

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(4) coeficientes DCT cuantizados

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(5) coeficientes DCT descuantizados

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

(6) bloque reconstruido