UNIVERSITAT POLITÈCNICA DE VALÈNCIA

El estándar de compresión de imagen JPEG

Apellidos, nombre	Oliver Gil, José Salvador (joliver@disca.upv.es)
Departamento	Informática de Sistemas y Computadores
Centro	Escola Tècnica Superior d'Enginyeria Informàtica

1 Resumen de las ideas clave

En este artículo se describe el estándar de compresión JPEG, detallando el algoritmo que implementa el modo de funcionamiento secuencial, al ser este el más utilizado. En concreto veremos qué características tiene el estándar JPEG y qué modos de funcionamiento dispone. Posteriormente nos centraremos en el modo secuencial y describiremos cada uno de los pasos que lo forman.

2 Introducción

En compresión de imagen se busca reducir la redundancia espacial, que es aquella que tienen los píxeles vecinos en una imagen natural. Por ejemplo, en la fotografía de la Imagen 1, los píxeles del cielo son muy parecidos. Para eliminar esta redundancia emplearemos compresores basados en la fuente, normalmente con pérdidas, buscando un alto nivel de compresión con la mínima distorsión perceptible.

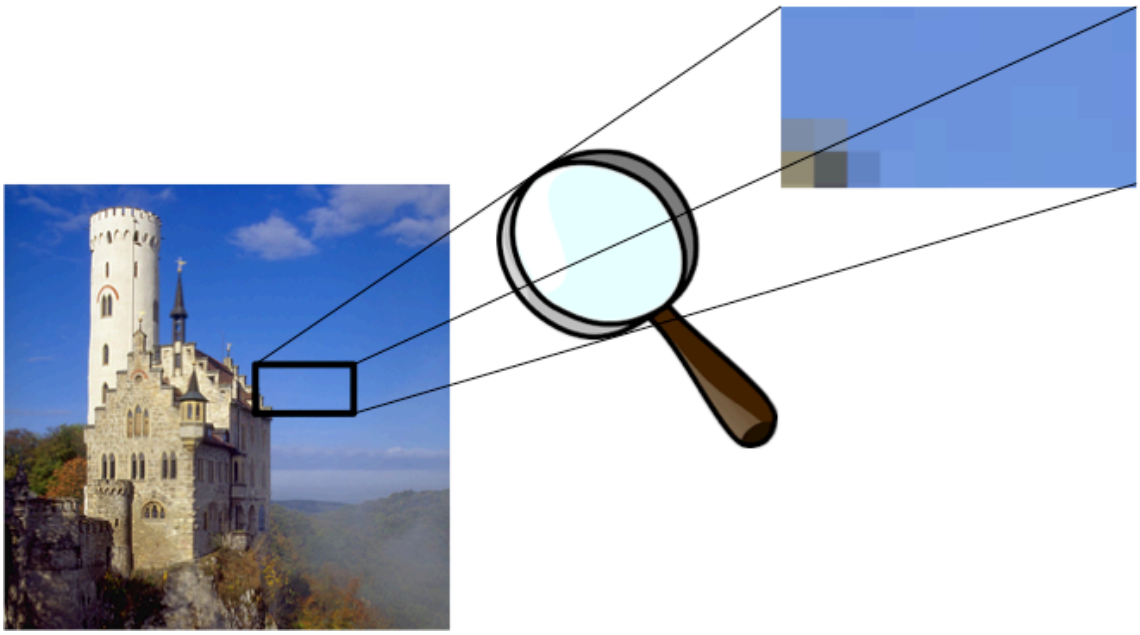

Imagen 1. Redundancia espacial en imágenes naturales

El compresor de imagen con pérdidas más empleado en la actualidad es JPEG. Dentro de los codificadores basados en la fuente, JPEG puede ser clasificado como codificador por transformación (*transform coding*), ya que emplea la transformada discreta del coseno (DCT) para representar la imagen en el domino de frecuencias, y así poder compactar más la energía en las componentes de baja frecuencia. Además, el uso de un análisis en frecuencia nos permitirá también reducir la precisión en las componentes de alta frecuencia, donde el ojo humano percibe menos los errores.

3 Objetivos

Una vez que el alumno lea con detenimiento este documento, será capaz de:

- Identificar los elementos clave del estándar de compresión JPEG.
- Enumerar y describir los distintos pasos que permiten la compresión de una imagen a partir de sus componentes de colores RGB.

4 Desarrollo

JPEG es un estándar para compresión de imagen desarrollado por la organización internacional de normalización (ISO) y publicado el año 1992. Recibe el nombre del comité que lo creó: el *Joint Photographic Expert Group*. Su objetivo es codificar imágenes de tono continuo, y dispone de cuatro modos de funcionamiento:

- Modo Secuencial: en el que las imágenes se decodifican por líneas, de arriba abajo.
- Modo Jerárquico: que proporciona escalabilidad espacial
- Modo Progresivo: que proporciona escalabilidad SNR o de la calidad.
- Modo Sin pérdidas (este último, lógicamente, comprime mucho menos)

En los tres primeros modos el nivel de calidad de la imagen lo determina un parámetro que llamamos factor de cuantización y que veremos más adelante. Evidentemente, a mayor compresión, mayor es la distorsión introducida sobre la imagen.

4.1 Modo secuencial del estándar JPEG

Veamos el algoritmo JPEG en modo secuencial, el más empleado, y cuyo diagrama de funcionamiento puedes ver en la Imagen 2, que muestra a modo de esquema los distintos pasos que se deben completar para codificar una imagen usando este modo.

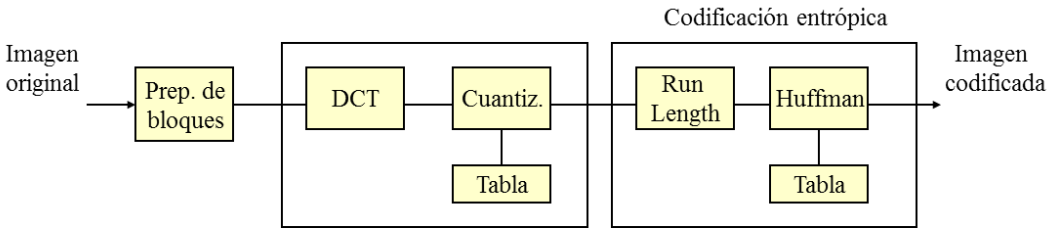

Imagen 2. JPEG modo secuencial

4.1.1 Preparación de bloques

En primer lugar tenemos la preparación de bloques. La imagen de entrada puede estar representada en cualquier espacio de colores (habitualmente es RGB), y la debemos recodificar al espacio de colores YCbCr con formato 4:2:0. A continuación cada uno de los tres planos lo dividimos en bloques contiguos de 8x8 elementos.

El resto de pasos se realizarán sobre cada uno de estos bloques individualmente.

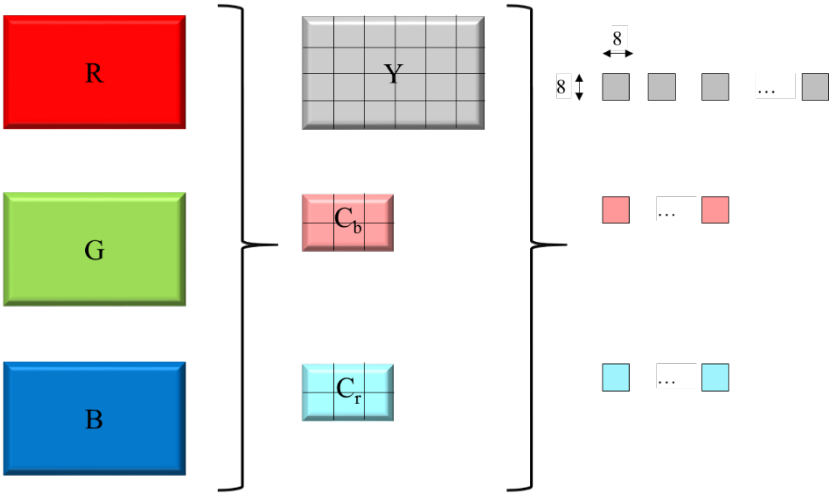

Imagen 3. Preparación de bloques

4.1.2 Transformada Discreta del Coseno (DCT)

El siguiente paso es aplicar a cada bloque la transformada discreta del coseno o DCT (de sus siglas en inglés, *Discrete Cosine Transform*).

Hemos visto que cada bloque representa la intensidad de luminancia o crominancia de un conjunto de píxeles. Por redundancia espacial, seguramente muchos de esos píxeles tendrán valores similares, con lo que el bloque contendrá 64 valores parecidos **pero no iguales**.

¿Crees que la representación actual de los bloques favorece el uso de codificadores basados en la entropía como Huffman? No, para poder usar este tipo de codificación nos **interesaría tener muchos valores iguales** y quizás unos pocos diferentes, de esta forma podríamos codificar con muy pocos bits los valores que más se repiten. Pero lo que sucede es que tenemos muchos valores parecidos, pero no iguales.

La DCT nos permite pasar de una representación espacial de los píxeles a una en frecuencia. Después de aplicarla, tal y como se muestra en la Imagen 4, en la esquina superior izquierda del bloque tenemos la componente de continua (llamada de forma abreviada DC). Esta componente refleja el valor medio del bloque. Entorno a la DC, tenemos las componentes de baja frecuencia, que indican en qué medida contribuyen estas frecuencias a la formación del bloque original.

Observa que cada componente del bloque transformado también se llama coeficiente, ya que pondera la importancia de un elemento de la base de la DCT.

Por otro lado, conforme más nos alejamos de la posición de la componente de continua, los coeficientes representan valores de frecuencias más altas.

Imagen 4. Ejemplo de aplicación de la DCT

Como los bloques de JPEG son matrices, la DCT que se aplica es bidimensional. Para su cálculo, basta con aplicar la DCT 1D a todas las filas y después a todas las columnas, almacenando cada resultado sobre el propio vector.

Pero, ¿*qué conseguimos con la representación en frecuencia del bloque*?

Básicamente, dos cosas:

- Por un lado, debido a que los elementos del bloque son parecidos, los coeficientes de alta frecuencia van a ser muy próximos a cero. Además, la componente de continua (DC) y las bajas frecuencias tendrán los valores más altos. Esto se denomina **concentración de energía** y, como hemos mencionado antes, nos será muy útil para aplicar posteriormente codificación basada en la entropía (como Huffman).
- Por otro lado, la representación en frecuencia nos permitirá perder precisión en las altas frecuencias sin que nuestro ojo apenas lo perciba.

4.1.3 Cuantización (*Quantization*)

La etapa en la que se introduce la pérdida de precisión se denomina cuantización (palabra que proviene del inglés *quantize* y **no** del latín *quantificare*). Gracias a la cuantización podremos alcanzar mayores niveles de compresión.

Básicamente la cuantización consiste en asignar valores discretos a intervalos continuos. Un ejemplo sencillo de cuantización es el truncado de una variable de tipo coma flotante a entero (o sea pasar de *float* a *integer*). Por ejemplo, al truncar una variable de tipo flotante, todos los valores que se encuentren en el intervalo [2 a 3[se cuantizan con el valor discreto 2.

Una forma muy sencilla de implementar la cuantización consiste en realizar una división entera del valor a cuantizar por el factor de cuantización. Observa que cuanto mayor es el denominador en la división, más valores se agrupan, y por tanto alcanzaremos un mayor nivel de compresión en posteriores etapas de codificación basada en la entropía. En este caso la distorsión introducida también será mayor, debido a la pérdida de precisión.

¿Cómo podemos determinar el factor de cuantización a usar (también llamado paso de cuantización o nivel de cuantización)? Este valor está asociado al nivel de compresión solicitado por el usuario.

Sin embargo, JPEG no aplica una cuantización uniforme, en la que todos los coeficientes se cuanticen por el mismo paso de cuantización. A los coeficientes que representan frecuencias mayores se les aplica una mayor

cuantización, aprovechando que el ojo humano no percibe tanto los errores en las frecuencias altas como en las bajas.

Para implementar la cuantización de esta forma, el estándar define una matriz de cuantización similar a la que se muestra en la Imagen 5, de manera que cada coeficiente se divide por el elemento de la matriz de cuantización que se ubica en su misma posición. El resultado de aplicar este proceso de cuantización a un bloque de ejemplo se muestra en la Imagen 5.

Bloque antes de cuantizar

150	7.7	3.8	8.1	4.1	0	1.2	0
8.8	5.6	9.2	9.3	2	0	0	0
2.3	3.2	3.4	4	0	0	0	0
4.0	6.1	3.6	7.2	0	1.3	0	0
1.1	0	5.2	0	2.8	0	0	0
0	1.7	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Matriz de cuantización

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Bloque cuantizado

150	7	2	2	0	0	0	0
8	5	4	2	0	0	0	0
1	2	2	1	0	0	0	0
1	1	0	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

Imagen 5. Ejemplo de cuantización de un bloque

A la hora de comprimir, el usuario escogerá un *nivel de calidad* entre 0 y 100. Este parámetro determinará los valores de la matriz de cuantización, siendo estos más altos, cuanto más bajo sea el nivel de calidad seleccionado.

4.1.4 Codificación diferencial de la DC

La siguiente etapa es la Codificación diferencial de la componente de continua (DC). La componente de continua representa la media de los valores en el bloque original y está situada en la posición (0,0) del bloque. Este valor es un valor especial, ya que normalmente es el que concentra la mayor parte de la energía (el de mayor magnitud). Además, en general existe una correlación con los bloques vecinos, ya que bloques vecinos representan zonas de la imagen similares. Por tanto, este valor se codifica diferencialmente con la componente de continua del bloque vecino previamente codificado.

4.1.5 *Run-length encoding* (RLE) en zig-zag

A continuación aplicamos *Run-length encoding* (RLE) en orden zig-zag. Como hemos visto, gracias a la DCT y a la posterior etapa de cuantización, se obtiene un gran número de ceros en las zonas del bloque que representan las altas frecuencias, por tanto, un codificador tipo *run-length* será idóneo para representar estos. Así, el estándar JPEG almacena los coeficientes de la DCT mediante una cuenta de ceros, recorriendo para ello el bloque en zig-zag (tal y como se aprecia en la Imagen 6), con el objetivo de recorrer los coeficientes de alta frecuencia en último lugar, y así concentrar el mayor número de ceros seguidos posible.

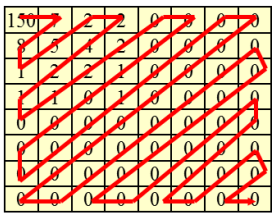

Imagen 6. Orden de recorrido en zig-zag

4.1.6 Codificación de longitud variable (VLC)

Finalmente el bitstream de cada bloque se genera mediante una última etapa de codificación Huffman, en la que se codifican los valores resultantes de la cuenta de ceros y los coeficientes distintos de cero. El estándar JPEG incluye unas tablas de Huffman pre-calculadas en base a las probabilidades de aparición de cada símbolo estimadas por el grupo creador del estándar. Este tipo de codificación Huffman con tablas pre-calculadas se denomina codificación de longitud variable (*variable length coding*, VLC).

5 Cierre

Hemos visto cómo funciona el estándar JPEG para comprimir una imagen usando el modo secuencial. Para la descompresión simplemente tendremos que revertir cada una de estas etapas. Así, el bitstream generado será decodificado por Huffman. Posteriormente expandiremos la cuenta de ceros y ubicaremos los coeficientes en las posiciones del bloque que les correspondan. Aplicaremos la cuantización inversa usando una multiplicación, y emplearemos la inversa de la DCT para devolver los coeficientes a una representación espacial.

Por último reflexiona sobre la siguiente cuestión, ¿crees que JPEG es un estándar simétrico o asimétrico? Para responderla, observa que la decodificación está formada por los mismos pasos que la codificación y fíjate si alguno de estos pasos es más complejo en el sentido directo que en el inverso.

6 Bibliografía

Ghanbari, M: "Standard Codecs: Image Compression to Advanced Video Coding", IEE Telecommunications Series 49, 2003, pág. 105-115.

Oliver, J; Pérez-Malumbres, M: "Compresión de imagen y vídeo: fundamentos teóricos y aspectos prácticos", Ed. Universidad Politécnica de Valencia, 2001, pág. 41-52.