## Tema IV: Compresión de imágenes

Esquema del tema

- Introducción. Necesidad de la compresión.
- Tipos de compresión: con y sin pérdidas.
- Compresión sin pérdidas: códigos de reducción de bits. Ej.: códigos de Huffmann
- Compresión sin pérdidas: compresión predictiva
- Compresión con pérdidas de imágenes estáticas. El estándard JPEG.
- Compresión con pérdidas de secuencias de imágenes. El estándard MPEG.



Dado el tamaño de las imágenes, y especialmente de las secuencias de éstas (video) es muy conveniente comprimirlas para su almacenamiento, y absolutamente necesario para su transmisión.

Como en toda compresión, existe la posibilidad de realizarla con o sin pérdidas. En el segundo caso, el archivo resultante de un proceso de compresión seguido de uno de decompresión es idéntico bit a bit al original. En el segundo caso no, pero esperamos que las diferencias no sean significativas.

Existen varios estándares usuales para compresión de imágenes sin pérdidas, que dan lugar a varios formatos (.png,.pcx etc.) y otros que permiten elegir compresión con o sin pérdidas (.tiff,.gif,.jpg etc.). En contrapartida a las pérdidas se obtienen por supuesto tasas de compresión mayores. Entre los métodos con pérdidas más usados destacan el JPEG para imágenes estáticas y el MPEG para secuencias de imágenes.

## Compresión sin pérdidas

Esencialmente se trata de métodos de codificación, es decir, métodos que tratan de asignar un código binario a cada uno de los valores de gris de la imagen de modo que el tamaño de los códigos asignados sea el justo para representar la información de la imagen, y no más.

En la codificación uniforme simplemente se dispone de $n$ símbolos de longitud fija (digamos, cada una de las 255 palabras de 8 bits) y se asigna uno de estos símbolos (un código) a cada valor de gris, sin tener en cuenta el número de veces que el valor aparece.

En la codificación no uniforme se tiene en cuenta la distribución de probabilidad de los valores, y lo que se hace es usar $m$ símbolos de longitudes diversas, tratando de asignar los símbolos (códigos) de menor longitud a los valores que aparecen más frecuentemente.



## Codificación de Huffmann

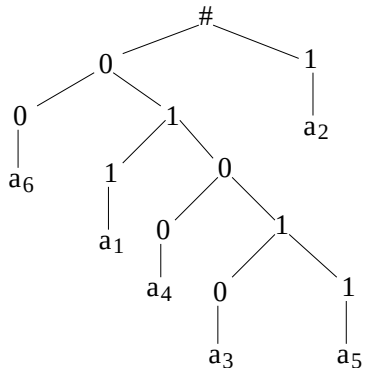
Es el método más usado de codificación no uniforme. Se basa en ordenar los símbolos por su probabilidad de aparición, y proceder por pasos, agrupando en cada paso los dos símbolos menos probables. Hecho esto, se vuelve atrás asignando a cada grupo de símbolos un bit distinto. Como ejemplo:

| Fuente original | | Reduccion de la fuente | | | |
|---|---|---|---|---|---|
| Simbolo | Probabilidad | 1 | 2 | 3 | 4 |
| $a_2$ | 0.4 | 0.4 | 0.4 | 0.4 | 0.6 |
| $a_6$ | 0.3 | 0.3 | 0.3 | 0.3 | 0.4 |
| $a_1$ | 0.1 | 0.1 | 0.2 | 0.3 | |
| $a_4$ | 0.1 | 0.1 | 0.1 | | |
| $a_3$ | 0.06 | 0.1 | | | |
| $a_5$ | 0.04 | | | | |

| Fuente original | | | Reduccion de la fuente | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Simbolo | Probabilidad | Codigo | 1 | | 2 | | 3 | | 4 | |
| $a_2$ | 0.4 | 1 | 0.4 | 1 | 0.4 | 1 | 0.4 | 1 | 0.6 | 0 |
| $a_6$ | 0.3 | 00 | 0.3 | 00 | 0.3 | 00 | 0.3 | 00 | 0.4 | 1 |
| $a_1$ | 0.1 | 011 | 0.1 | 011 | 0.2 | 010 | 0.3 | 01 | | |
| $a_4$ | 0.1 | 0100 | 0.1 | 0100 | 0.1 | 011 | | | | |
| $a_3$ | 0.06 | 01010 | 0.1 | 0101 | | | | | | |
| $a_5$ | 0.04 | 01011 | | | | | | | | |

Nótese que en una codificación uniforme el número de bits a usar sería de 3 (hay 6 símbolos) con lo que la longitud media es obviamente 3; en cambio en ésta la longitud media sería $\bar{l} = \sum_{i=1}^{6} p_i l_i = 0{,}4 + 0{,}3 * 2 + 0{,}1 * 3 + 0{,}1 * 4 + 0{,}06 * 5 + 0{,}04 * 5 = 2{,}20$. La reducción en este caso sería pues de un 25.6 %.

Una interesante ventaja adicional de la codificación de Huffmann es que los símbolos obtenidos se pueden concatenar sin ninguna marca de división entre ellos, y sin que por eso la interpretación resulte ambigüa, pues la decodificación se puede realizar siguiendo un árbol, donde sabemos que el símbolo acaba al llegar a un nodo terminal.




De acuerdo al mecanismo de asignación de símbolos visto antes, cuanto mayor sea el desequilibrio en el número de apariciones de símbolos entre unos y otros (o sea, cuantos más valores en el histograma sean 0) mayor será la compresión. Ello hace que sean más compresibles imágenes con sus histogramas fuertemente picados.



**Codificación predictiva**

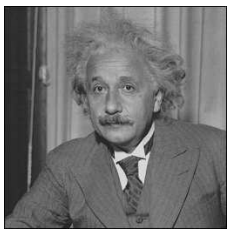
Se supondrá que se da un orden prefijado para el recorrido de la imagen, de tal modo que se podrá hablar de pixels anteriores y posteriores en la ordenación. Generalmente se supone la ordenación usual de lectura occidental.

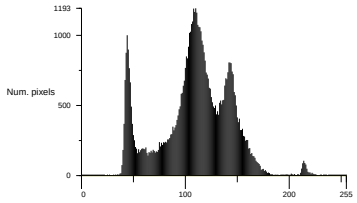
Hecho esto, se intentará predecir el valor de un pixel a partir de los anteriores usando alguna fórmula simple dada, y lo que se guardará no es el valor del pixel, sino la diferencia entre éste y su predicción. Si la predicción es aceptable esta diferencia será pequeña, y el pequeño número de valores que toma hará que se repita muy frecuentemente. Como ejemplo, se puede predecir el valor de X a partir del de sus vecinos A,B y C según alguna de las fórmulas 0 a 7 de la tabla dada:

| | | | | |
|---|---|---|---|---|
| | | | | |
| | C | B | | |
| | A | X | | |
| | | | | |

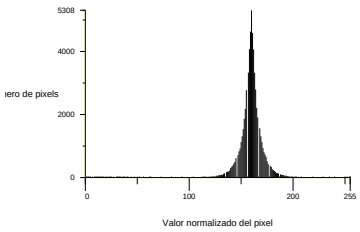
| Núm. de fórm. | Fórmula | Núm. de fórm. | Fórmula |
|---|---|---|---|
| 0 | Sin predicción | 4 | A+B-C |
| 1 | A | 5 | A+((B-C)/2) |
| 2 | B | 6 | B+((A-C)/2) |
| 3 | C | 7 | (A+B)/2 |

Como ejemplo de codificación predictiva se muestra una imagen, su histograma, la imagen que se almacena usando la fórmula número 4 y su histograma. Esta última ha sido normalizada por propósitos de visualización, con lo el pico del histograma no aparece en 0, como resulta directamente de la imagen diferencia de predicción sin normalizar.






$$H = -\sum_{i=1}^{256} p_i \log(p_i) = 6{,}884 \text{ bits}$$




$$H = -\sum_{i=1}^{256} p_i \log(p_i) = 5{,}163 \text{ bits}$$

Como se ve, el histograma se estrecha. Es por tanto especialmente apropiado para aplicar ahora una codificación de Huffmann. Con ambos pasos se consiguen tasas de compresión en torno al 50 %.



**El estándard JPEG** (**J**oint **P**hotographic **E**xperts **G**roup)

— Adoptado en 1993 a propuesta del comité JPEG del ISO.

— Pretende ser genérico (no orientado a una clase particular de imágenes).

— Existen dos subestándares, uno basado en codificación predictiva sin pérdidas, y otro mucho más usado basado en la DCT, con perdidas.

El compresor consta de tres pasos:

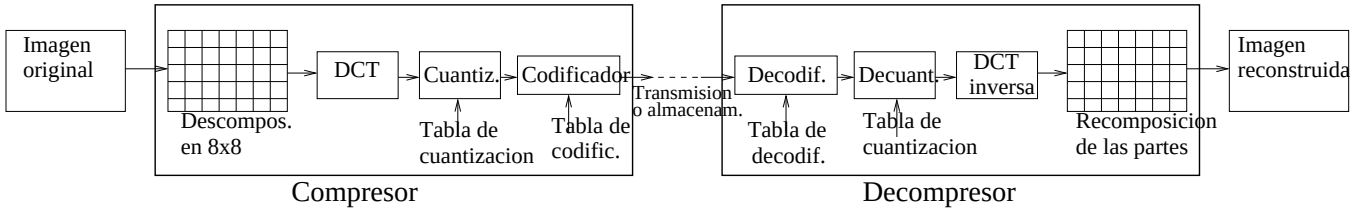
— La división de la imagen en bloques de $8 \times 8$ pixels, y la ejecución de la DCT sobre cada uno.

— La cuantización de los valores resultantes de la DCT a un número menor de niveles.

— La codificación de los valores cuantizados para convertirlos en cadenas de bits de la menor longitud posible.

El decompresor opera invirtiendo cada paso, en orden también inverso.

El esquema del compresor/decompresor (llamado a veces codificador/decodificador o CODEC) es:




Después del primer paso, cada bloque de $8 \times 8$ ha sido sustituído por su DCT. Obsérvese que las funciones trigonométricas que aparecen en la DCT sólo se calculan en un número reducido de valores ($\frac{2(n+1)\pi u}{16}$ con $n = 0 \ldots 8$ y $u = 0 \ldots 8$), con lo cual se pueden tabular.

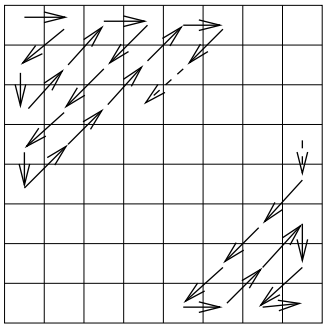
Al final de este paso cada subimagen del plano DCT suele tener valores altos cerca de la esquina superior derecha y bajos en el resto. Ello es porque las imágenes usuales suelen contener pocas frecuencias espaciales altas. Un ejemplo típico podría ser:



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 100.0 | 80.3 | -16.4 | 3 | 0 | 0 | 0 | 0 |
| 78.0 | 60.0 | 10.0 | 0 | 0 | 0 | 0 | 0 |
| 27.0 | 12.0 | 5.0 | 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 | 1.7 | 1 | 2.2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

En el siguiente paso (cuantización) cada elemento de la tabla se divide por un coeficiente específico de su posición en la subimagen $8 \times 8$ (el mismo para todos los bloques de la imagen). Estos coeficientes (de 1 a 255) se han determinado experimentalmente y tienen que ver con la mínima discriminación en frecuencia espacial que el ojo es capaz de percibir a esa cierta frecuencia espacial. Esto genera una tabla con casi todos sus elementos nulos, que se recorre en el llamado " recorrido de Hilbert":

| | | | | |
|---|---|---|---|---|
| 15 | 0 | −1 | ··· | 0 |
| −2 | −1 | 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 |



15, 0, −2, −1, −1, −1, 0, 0, −1, 0, ...

La lista de 64 elementos resultante se codifica como un alfabeto de símbolos de la forma:
- El primero (el término de contínua, DC) como (tamaño)(diferencia con el DC anterior), excepto para el primer DC, que es (tamaño)(valor)
- Los demás como (long. de la cadena de ceros, tamaño)(valor)



Como ejemplo, supongamos que el término DC anterior hubiera sido 12. Entonces la cadena anterior queda: (*)(3) (1,*)(-2) (0,*)(-1) (0,*)(-1) (0,*)(-1) (2,*)(-1) (0,0) donde la * será el tamaño en bits asignado al símbolo por un codigo de Huffmann, p. ej.:

| | | | |
|---|---|---|---|
| (2) | 011 | (0,0) | 1010 |
| (3) | 11 | (0,1) | 00 |
| (-2) | 01 | (1,2) | 11011 |
| (-1) | 0 | (2,1) | 11100 |

con lo que la cadena queda: (2)(3) (1,2)(-2) (0,1)(-1) (0,1)(-1) (0,1)(-1) (2,1)(-1) (0,0)
y la codificación final: 011,11;11011,01;00,0;00,0;00,0;11100,0;1010
(signos de puntuación añadidos por legibilidad, en realidad es 01111110110100000000011100001010).

El total en este caso es de 31 bits sobre los 64 bytes iniciales, es decir una razón de compresión de 16:1 (0.5 bits/pixel).
Las pérdidas se producen sólo en el paso de cuantización (aunque la representación decimal de la DCT puede perder decimales, pero es despreciable).

**El estándard MPEG** (**M**oving **P**icture **E**xperts **G**roup)

— Adoptado a partir de 1998 a propuesta del comité MPEG del ISO.

— Pretende ser genérico (no orientado a una clase particular de imágenes) aunque funciona mejor con imágenes que contengan pocos objetos en movimiento, o en el que éste sea lento.

— Existen 4 subestándares:
  — MPEG1: pensado para el DVD con tasas de 1.5 Mb/s.
  — MPEG2 (H.262): un estándard generalizado, incluyendo la TV de alta definición (HDTV) y otros en que puede configurarse a voluntad la tasa de transmisión binaria.
  — MPEG4: estándard con el objetivo de conseguir tasas binarias muy bajas para transmitir la señal por RDSI a 32 ó 64 Kb/s.
  — MPEG7: estándard de almacenamiento de imágenes en bases de datos, como un SQL para imágenes. No trata la compresión.

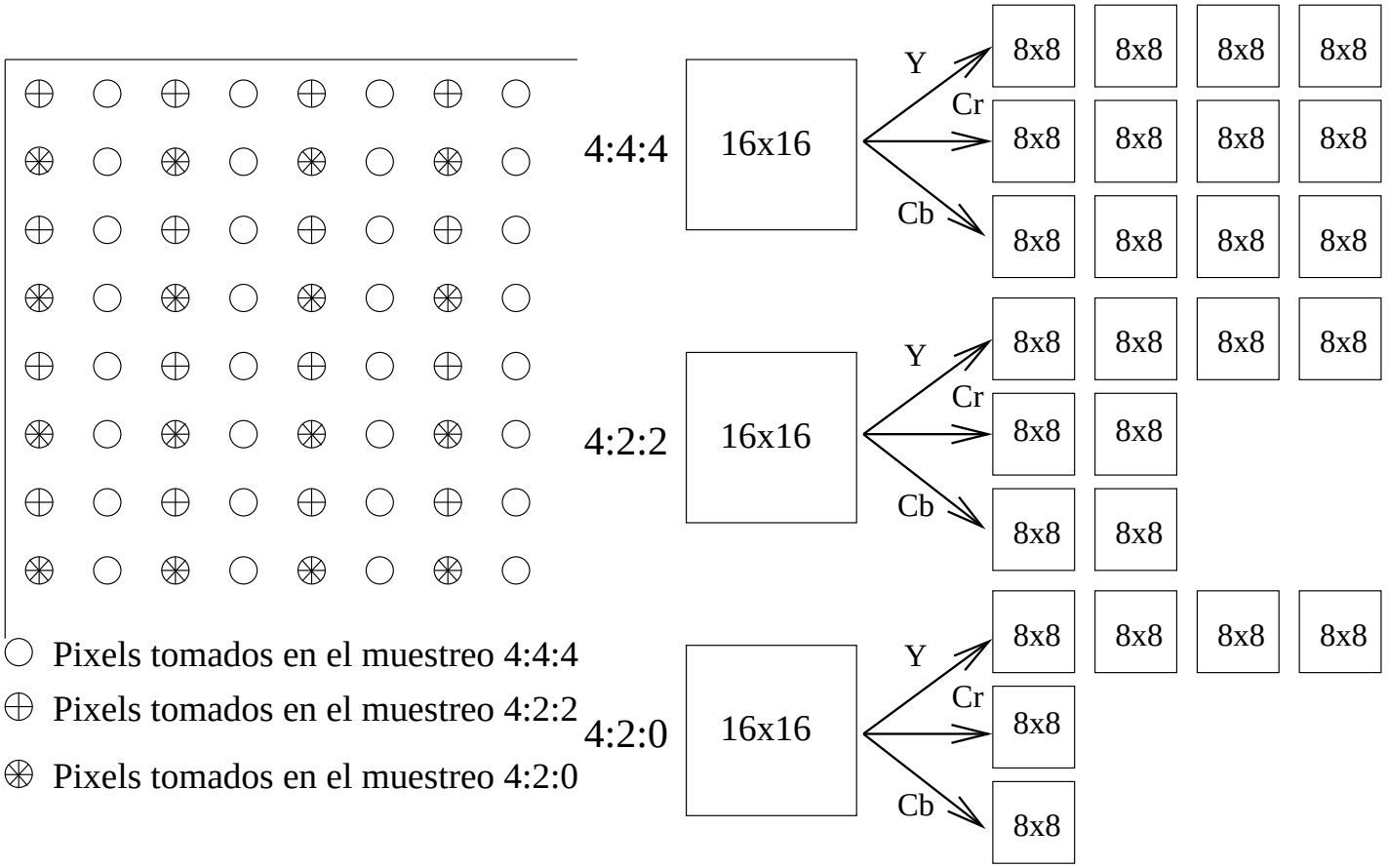


**MPEG2**

Se supone que se parte de una secuencia de imágenes en color, cada una de las cuales contiene la banda de intensidad ($Y$) y las dos bandas de crominancia ($C_r$ y $Cb$) que son imágenes $M \times N$.

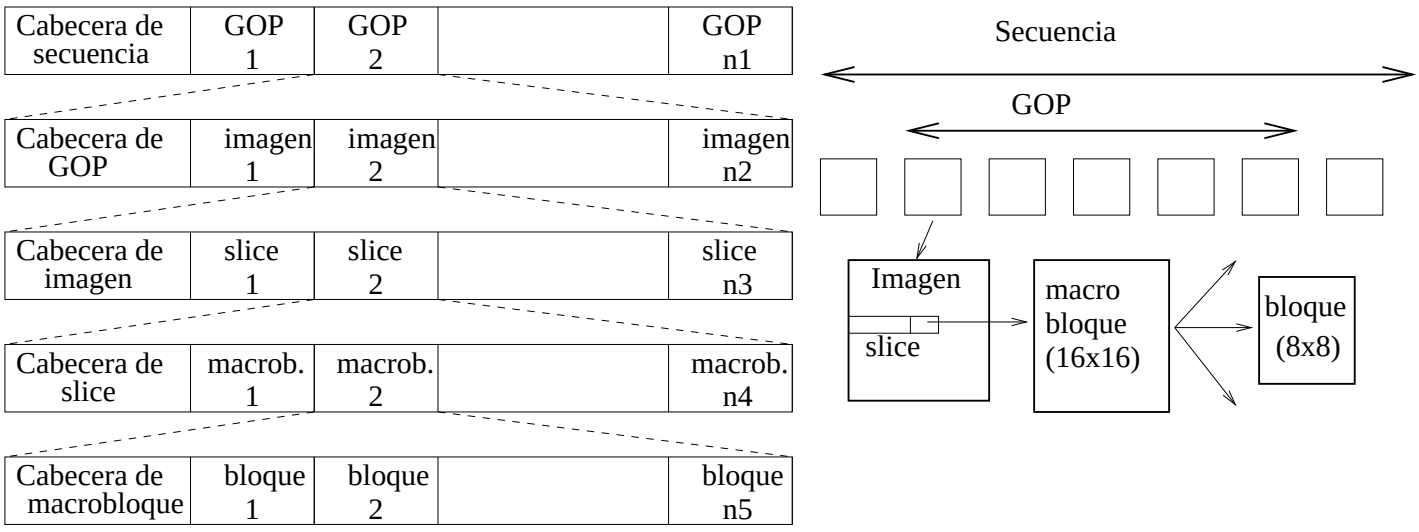
Se comienza dividiendo cada banda en bloques de $16 \times 16$ pixels, y submuestreando las bandas $C_r$ y $C_b$ de acuerdo con uno de los esquemas mostrados en la siguiente página: 4:4:4, 4:2:2 ó 4:2:0.

Los procesos de compresión posteriores se aplicarán a cada una de las matrices $8 \times 8$ resultantes, que llamaremos **bloque**. Obviamente, el esquema 4:4:4 es el que da mejor calidad, especialmente cromática, pero también mayor tasa binaria.






La secuencia de imágenes se dividirá jerárquicamente en varias estructuras, como muestra el siguiente diagrama:




Cada grupo de imágenes (GOP) se comprimirá de forma independiente de los otros. Dentro de él MPEG2 establece que puede haber tres tipos de cuadros (imágenes): el I, el P y el B. La cabecera del GOP indica cuántas imágenes siguen, y de qué tipo es cada una.

Es posible elegir la composición de los GOPs, pero en general se suelen usar GOPs de 15 imágenes, de la forma IBBPBBPBBPBBPBB (codificación llamada 15/3) o GOPs de 12, de la forma IBBPBBPBBPBB (llamada 12/3).

Los tipos de imágenes I, P y B se diferencian por su codificación como sigue:

— Las imágenes de tipo I no requieren información de ninguna otra imagen para decodificarse. En concreto, se pueden considerar como imágenes estáticas compridas con el estándard JPEG, aunque con una matriz de cuantización distinta.

— Las imágenes de tipo P contienen las "diferencias"entre la imagen de ese instante y la imagen de referencia más cercana. Dicha imagen de referencia sólo puede ser o bien una imagen tipo I o bien otra de tipo P, pero siempre anteriores a la dada.

— Las imágenes de tipo B contienen las "diferencias"entre la imagen de ese instante y la predicha a partir de las imágenes de tipo I y de tipo P más cercanas, incluso en el futuro. Esto significa que las imágenes no se pueden transmitir en el orden en que se toman, p. ej. un GOP tipo 12/3 debería transmitirse como:
Original : I1 B2 B3 P4 B5 B6 P7 B8 B9 P10 B11 B12 I13 ...
Transmitido: I1 P4 B2 B3 P7 B5 B6 P10 B8 B9 I13 B11 B12 ...
Por tanto, las imágenes se mostrarán con un retraso de un GOP.



Expliquemos qué se entiende en la página anterior por "diferencias".
Respecto a las imágenes de tipo P, sea R su imagen de referencia (la I ó P anterior más cercana). Se buscan macrobloques en P que:

— Sean idénticos al correspondiente macrobloque de R. Esto se indica con un código especial y el macrobloque se omite. Son las áreas de la imagen sin cambios.

— Sean idénticos a otro macrobloque de R. En ese caso sólo se envía el vector de movimiento (número de macrobloques del desplazamiento en x e y). Esto corresponde a objetos que se han movido de un lugar a otro.

— Sean similares al correspondiente, o a otro, macrobloque de R. En ese caso, se usa un codificador predictivo que genera a partir de R el bloque predicho, sea $P_c$. Entonces se envía el vector de movimiento y las diferencias entre P y $P_c$ comprimidas en JPEG. Esto corresponde a objetos que se han movido una distancia no múltiplo del tamaño del macrobloque, o que han cambiado ligeramente.

— No sea similar a ningún macrobloque de R. Debe enviarse el contenido del macrobloque de P, comprimido en JPEG. Esto ocurre cuando aparecen objetos nuevos en una imagen.

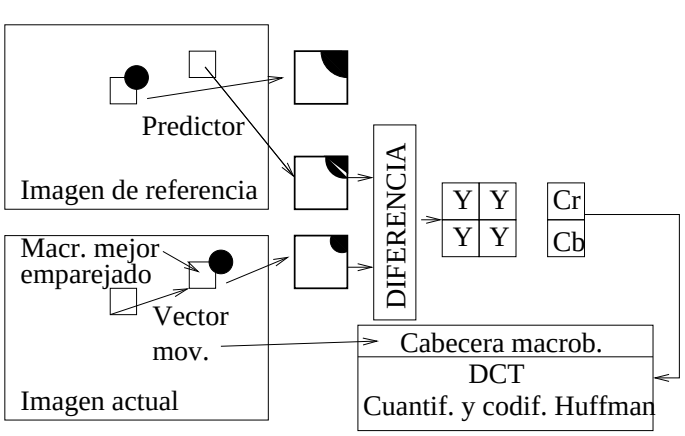
La tasa de compresión varía con el macrobloque, pero puede estar en el orden de 7:1.

Respecto a las imágenes de tipo B, se procede igual que en el caso de las P para macrobloques idénticos en valor y posición, o idénticos desplazados. La diferencia es que aquí es más probable encontrar tal identidad, dado que buscamos no en una, sino en dos imágenes de referencia.

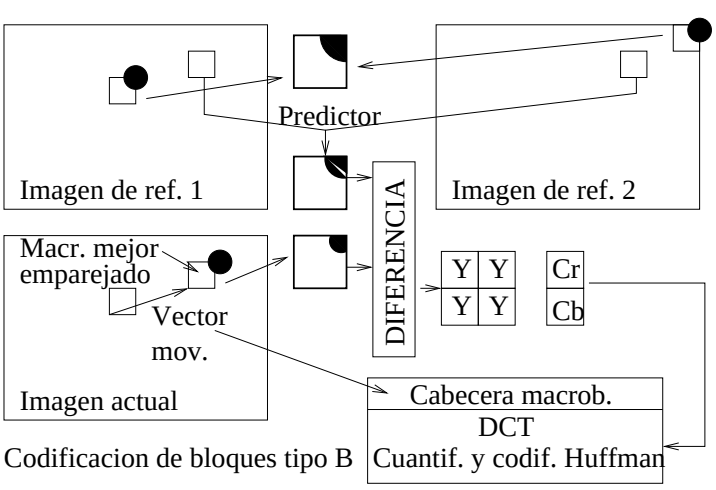
Respecto a los macrobloques similares, se usa igualmente un algoritmo predictivo con fórmula simple, pero esta vez usando dos imágenes de referencia. La predicción, sea $B_c$, suele ser por tanto bastante correcta y por tanto se comprime mucho más, dado que los valores diferencia $B - B_c$ suelen ser muy pequeños. La tasa de compresión para macrobloques B puede llegar hasta 100:1.

Finalmente, hacer notar que todo el proceso puede hacerse sobre imágenes que han sido generadas de modo entrelazado, o no entrelazado. En las primeras, se pueden crear los macrobloques ordenando (entrelazando posteriormente) las líneas, y codificar esto, o crear en cada macrobloque los dos microbloques superiores con el campo impar y los dos inferiores con el campo par. Esto último es lo más apropiado para comprimir bien el movimiento.






Codificacion de bloques tipo P




Codificacion de bloques tipo B

| niveles/perfiles | SIMPLE<br>4:2:0<br>Sin cuadros B | PRINCIPAL<br>4:2:0<br>Cuadros I,P,B | ESCALABLE SNR<br>4:2:0<br>Cuadros I,P,B | ESCALABLE ESPACIAL<br>4:2:0<br>Cuadros I,P,B | ALTO<br>4:2:2<br>Cuadros I,P,B |
|---|---|---|---|---|---|
| ALTO<br>1920x1152 | | 80 Mb/s | | | 100 Mb/s<br>25 en la capa base |
| ALTO-1440<br>1440x1152 | | 60 Mb/s | | 60 Mb/s<br>15 en la capa base | 80 Mb/s<br>20 en la capa base |
| PRINCIPAL<br>720x576 | 15 Mb/s | 15 Mb/s | 15 Mb/s<br>10 en la capa base | | 20 Mb/s<br>4 en la capa base |
| BAJO<br>352x288 | | 4 Mb/s | 4 Mb/s<br>3 en la capa base | | |