Taller:
Codificación de imagen y vídeo

Miguel Onofre Martínez Rach
GATCOM-Grupo de Arquitectura y Tecnología de Computadores
Universidad Miguel Hernández
Elche - España

3-5 Octubre 2018 - Universidad Autónoma del Estado de Morelos - México

Bit de dato y bit de información

- Aunque se trata de conceptos profundamente relacionados, existe una sutil diferencia entre dato e información:

  Los datos son la representación de la información.

- Una consecuencia directa de este hecho es que la misma información puede ser representada de muchas formas, unas más compactas que otras. Esto explica la existencia de los compresores de datos.

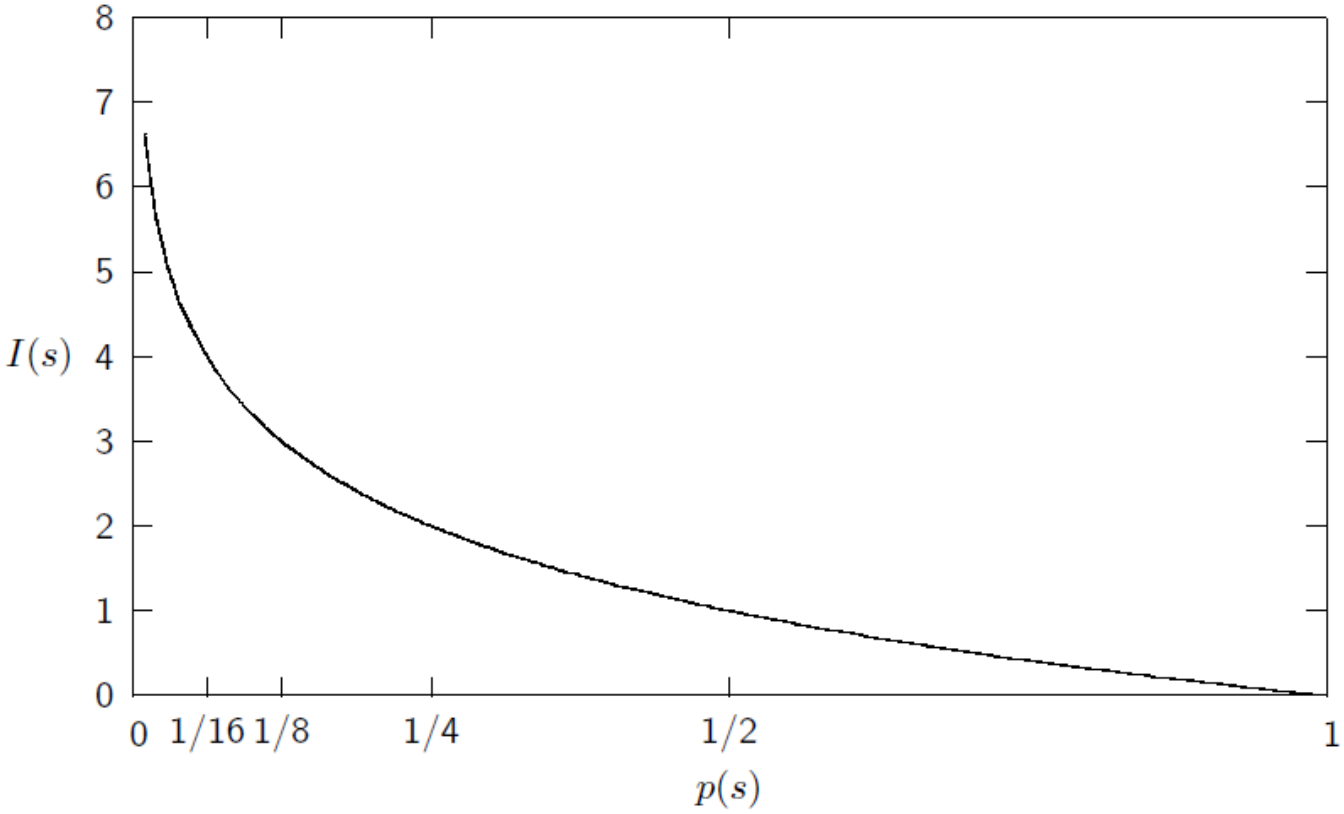
- Por definición, un bit de de datos transporta un bit de información si y sólo si representa la ocurrencia de un evento equiprobable, es decir, si la probabilidad de que dicho evento sea verdadero es igual a la de que sea falso. En cualquier otro caso, el bit de datos representará una cantidad diferente de bits de información. Por definición, esta cantidad para el s-ésimo símbolo del alfabeto es

$$I(s) = -\log_2 p(s)$$

  bits de información, donde $p(s)$ es la probabilidad de ocurrencia del símbolo s.

Bit de dato y Bit de Información

- Nótese que si la probabilidad de un símbolo es muy baja, entonces el número de bits de información representados es muy alto y viceversa. Gráficamente el número de bits de información asociados a un símbolo en función de su probabilidad es:



Entropía de una fuente de información

- La entropía es una medida de la cantidad de bits de información que una fuente de información proporciona en promedio, con cada símbolo generado. Por definición, la entropía $H(S)$ de una fuente S se calcula como

$$H(S) = \frac{1}{N} \sum_{s=1}^{N} p(s) \times I(s)$$

  donde N es el tamaño del alfabeto fuente (número de símbolos diferentes).

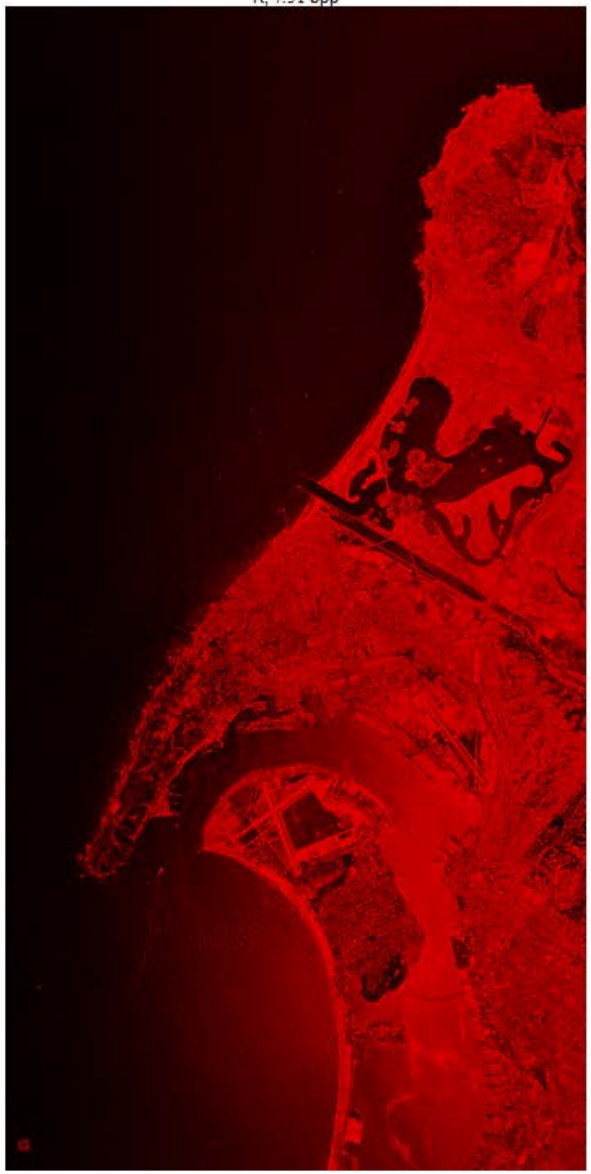
- La entropía se mide en bits de información por símbolo.
- La finalidad de la compresión estadística es la de encontrar una codificación tal que el número de bits de datos en promedio coincida con la entropía de la fuente.

Dominio RGB

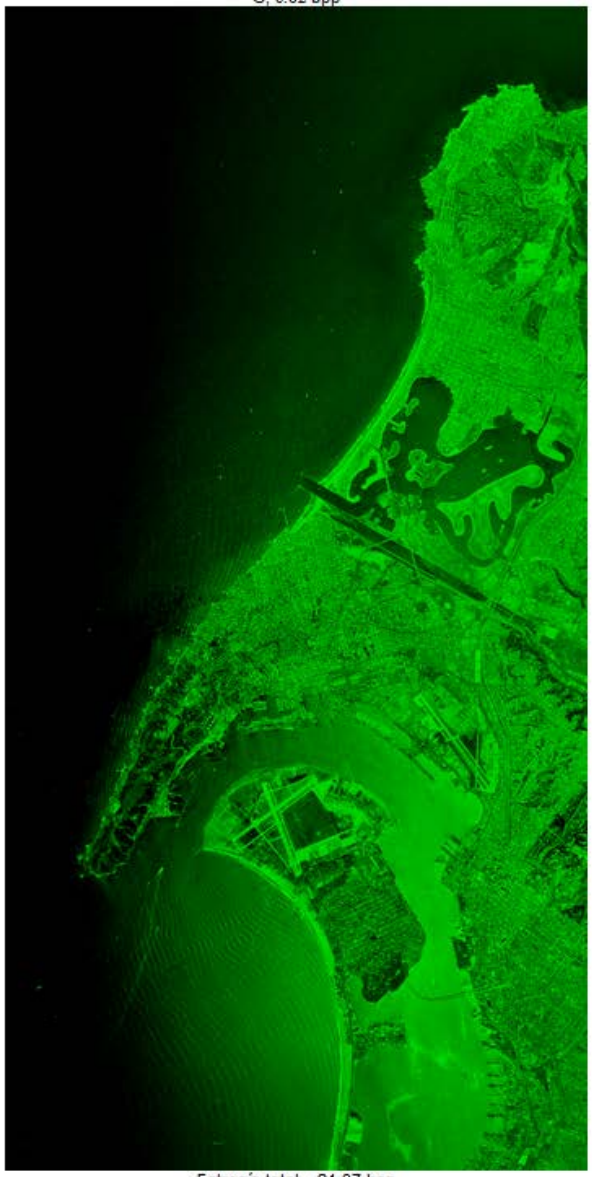
Ejemplo: San Diego (RGB)

Dominio RGB

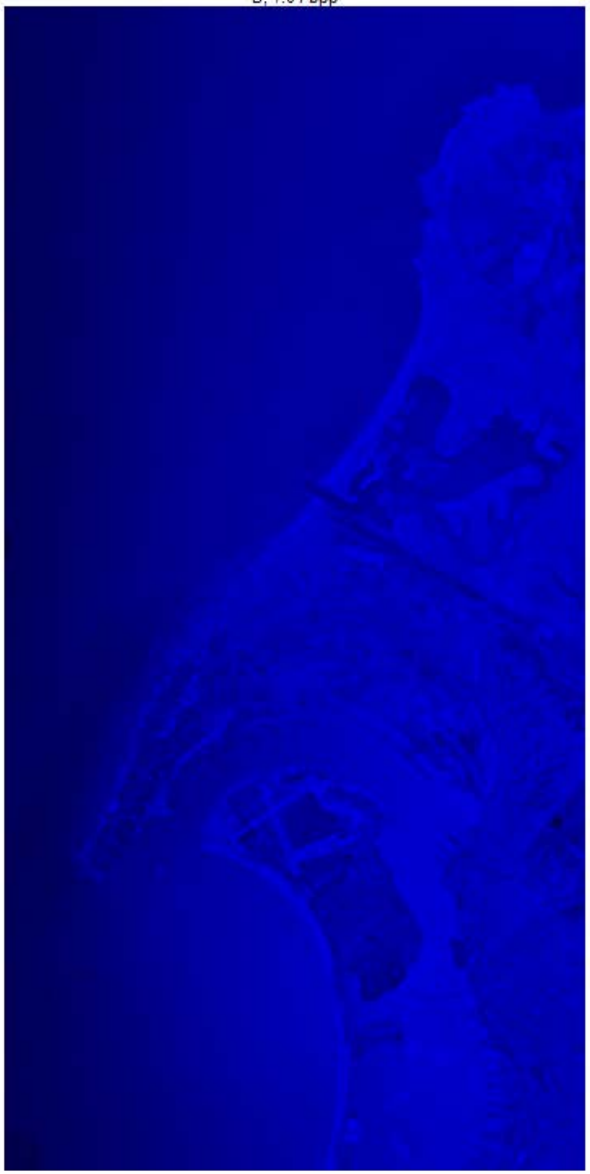
R, 7.51 bpp



G, 6.82 bpp



B, 7.04 bpp



Entropía total = 21,37 bpp

Dominio YCbCr

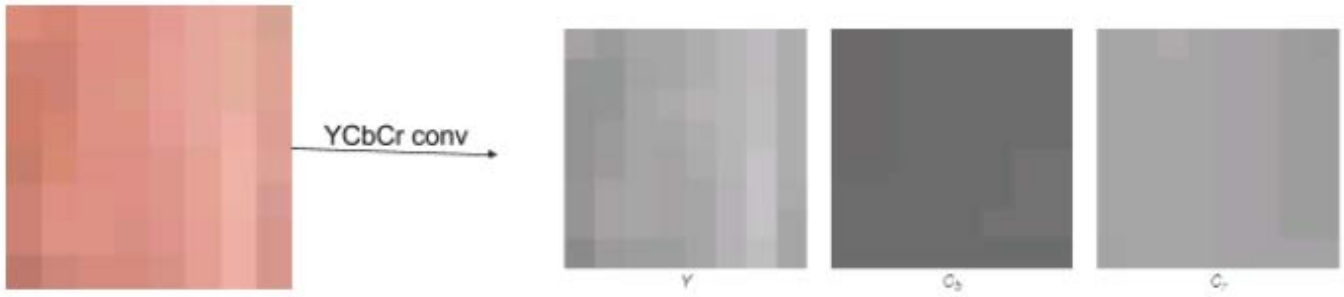
Los algoritmos de compresión rara vez se aplican directamente sobre el dominio RGB. Por tal motivo, primero se aplica una transformación que elimine correlación y por lo tanto decremente la entropía total de las 3 bandas.

Existen muchas transformaciones. Una de las más sencillas consiste en:

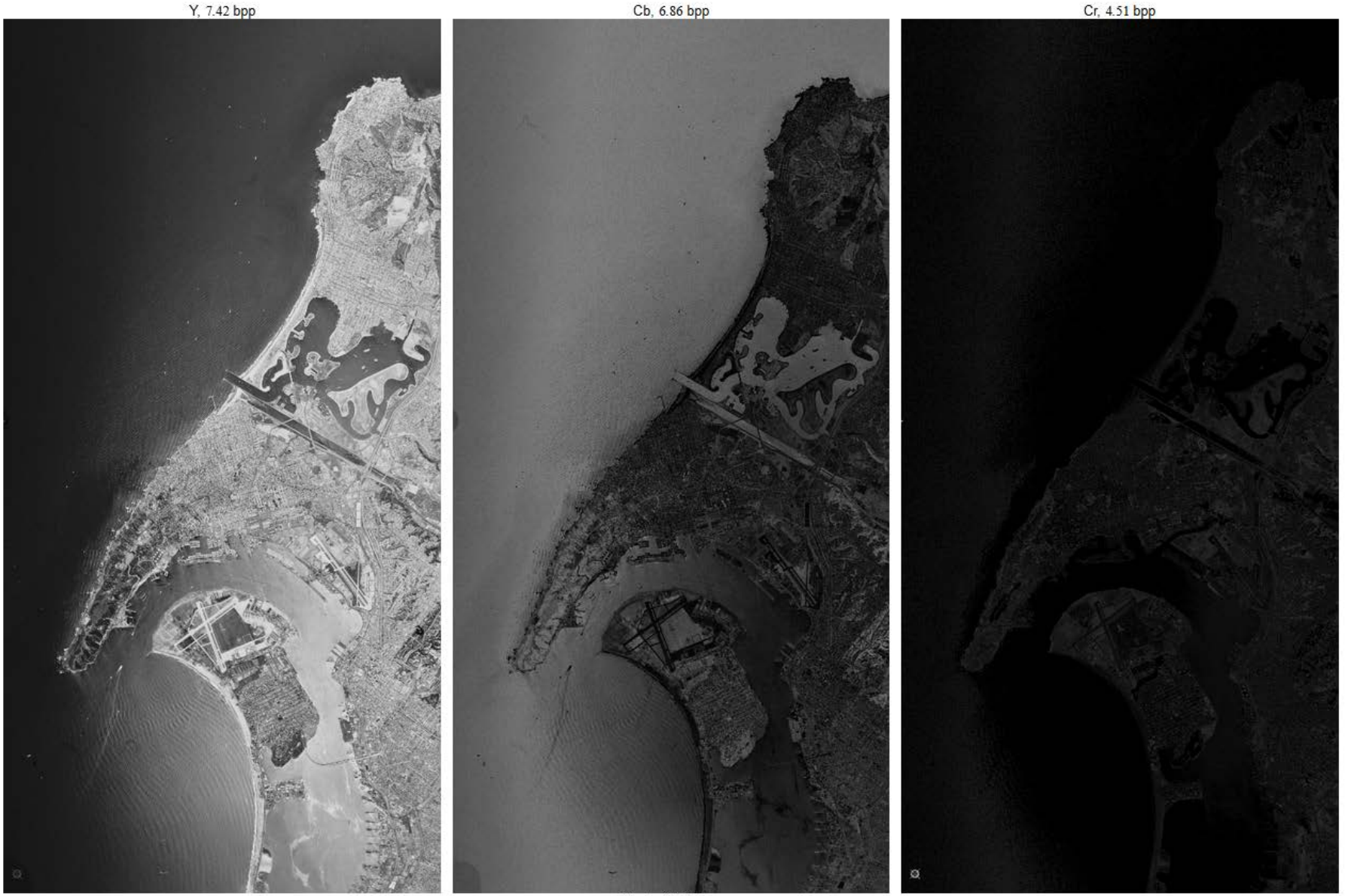
$$Y = \frac{R + 2 \times G + B}{4}$$
$$Cb = B - G$$
$$Cr = R - G$$

A la componente Y se le llama luminancia (o luma) y a las componentes Cb y Cr crominancia (o croma).

El sistema de visión humano es mucho menos sensible a la distorsión de la crominancia que a la de la luminancia. Esto es especialmente cierto en las altas frecuencias, donde la croma juega un papel secundario en la percepción de los detalles y las fronteras.



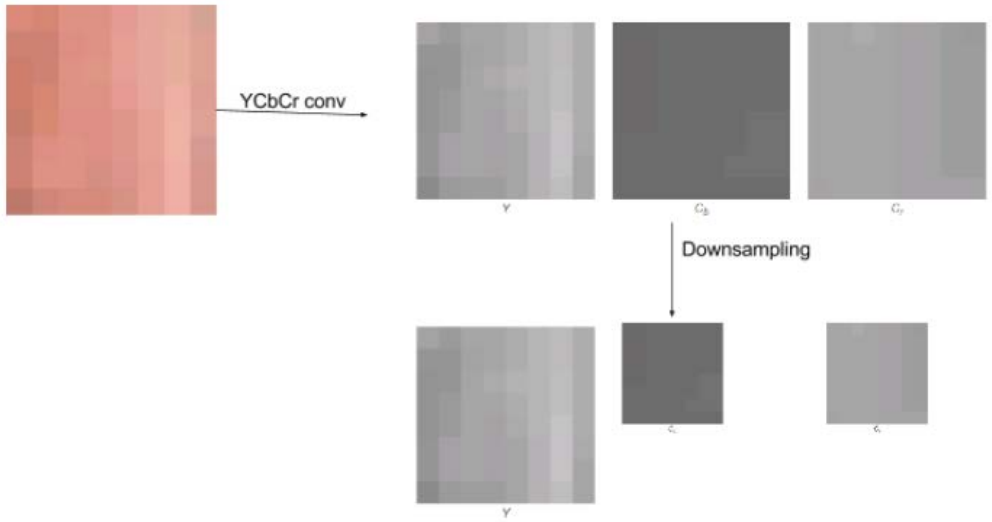
Dominio YCbCr

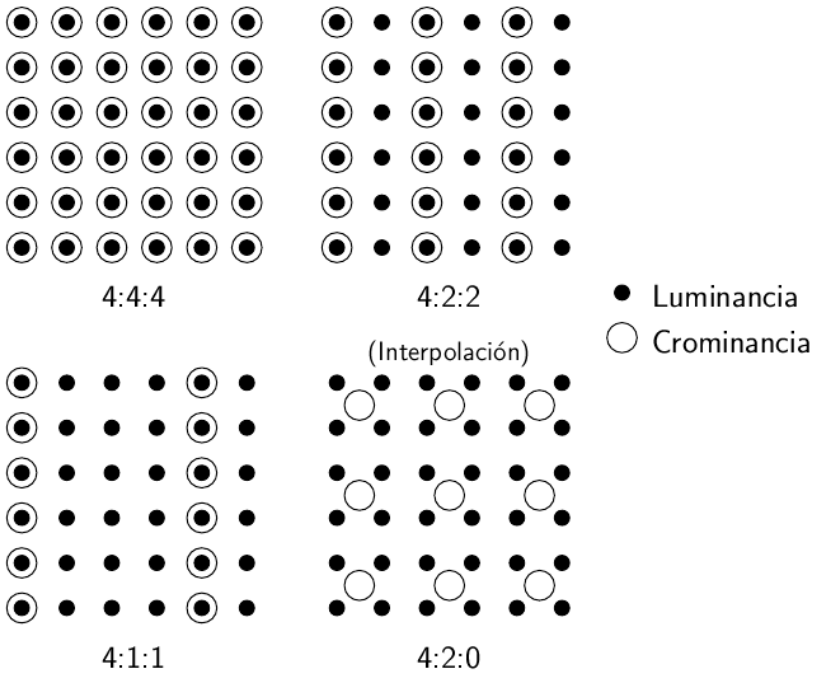


Submuestreo de Crominancia

El sistema visual humano es más sensible a la pérdida de información en la luma que en el croma. Por eso las bandas Cb y Cr se submuestrean en una de estas 4 formas:

- Formato 4:4:4. Las tres componentes se muestrean con la misma frecuencia.
  Es la que debería aplicarse en aplicaciones lossless.
- Formato 4:2:2. Cada dos muestras de Y (sólo) en horizontal se toma una para Cb y otra para Cr.
- Formato 4:1:1. Cada cuatro muestras de Y en horizontal se toma una para Cb y otra para Cr.
- Formato 4:2:0. Cada dos muestras de Y en horizontal y en vertical se toma una para Cb y otra para Cr, pero en los puntos intermedios.





RLE básico

- El tamaño del alfabeto fuente es 256 y la longitud mínima de la serie es 1.

Compresor

1. Mientras existan símbolos por codificar:
   a) Sea s el siguiente símbolo.
   b) Leer los siguientes n símbolos consecutivos iguales a s.
   c) Emitir un par ns.

Descompresor

1. Mientras existan pares ns que descodificar:
   a) Escribir n símbolos iguales a s.

RLE básico

- El tamaño del alfabeto fuente es 256 y la longitud mínima de la serie es 1.

Ejemplo de compresión

La secuencia de símbolos:

aaaabbbbbaaaaaabbbbbbbccccccc

se codificarían como:

4a 5b 6a 7b 6c

RLE binario

- Cuando sólo existen 2 símbolos diferentes no es necesario indicar el símbolo porque cuando una serie acaba es porque comienza otra con el símbolo alternativo.

Compresor

1. Sea $s \leftarrow 0$.
2. Mientras existan bits por codificar:
   a) Leer los siguientes n bits consecutivos iguales a s.
   b) Escribir n.
   c) $s \leftarrow (s+1)$ módulo 2.

Descompresor

1. Sea $s \leftarrow 0$.
2. Mientras existan items n que descodificar:
   a) Escribir n bits iguales a s.
   b) $s \leftarrow (s+1)$ módulo 2.

Ejemplo de compresión

La secuencia de símbolos:

000011111000000111111000000

se codificarían como:

4 5 6 7 6

Fixed and Variable Length Codes

Un flag con dos estados puede codificarse con 1 bit.

Cuando un elemento tiene más estados se puede especificar el número de bits necesarios para codificar los distintos estados.

Se puede usar la representación binaria del estado con N bits

Si hay diferentes probabilidades de aparición de los símbolos se puede optar por códigos de longitud variable.

Si se conoce la distribución de probabilidades de los símbolos se puede aplicar el VLC propuesto por Huffman. (ver cómo)

- Garantiza un prefijo (de bits) libre y único para cada símbolo.
- En número de bits utilizado en un símbolo decrece al aumentar su probabilidad.
- El límite superior está fijado en H+1 (Entropia +1)
- Se necesitan las tablas de códigos en codificador y decodificador

Compresión VLC

Variable Length Codes

Asigna a los símbolos de una fuente en códigos un número variable de bits.

Permiten una compresión sin perdidas.

A cada símbolos se le asigna un único código irrepetible.

Con una buena estrategia en la generación del código, una fuente puede llegar casi a su entropía.

Ejemplo Código único (non-singular)

$$M_2 = \{a \mapsto 1, b \mapsto 011, c \mapsto 01110, d \mapsto 1110, e \mapsto 10011, f \mapsto 0\}$$

Este ejemplo no es Unique-decodable,

011101110011 puede ser 01110 – 1110 – 011 o bien 011 – 1 – 011 – 10011

Ejemplo Unique-decodable

$$M_3 = \{a \mapsto 0, b \mapsto 01, c \mapsto 011\}$$

Tan pronto como llega un cero se sabe que ha terminado el código y se decodifica.

Hay que esperar a que llegue el cero para determinar que ha terminado el código anterior

Tras leer el cero no sabemos cuantos unos van a llegar

Variable Length Codes

Ejemplo Prefix Code

Pueden decodificarse conforme llega el último bit del código.

Symbol	Codeword
a	0
b	10
c	110
d	111

Example of encoding and decoding:

aabacdab → 00100110111010 → |0|0|10|0|110|111|0|10| → aabacdab

Ventajas sobre RLE

A los símbolos menos frecuentes se les pueden asignar códigos mas largos

A los símbolos muy frecuentes se les puede asignar códigos mas cortos.

probabilities of (a, b, c, d) were $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right)$

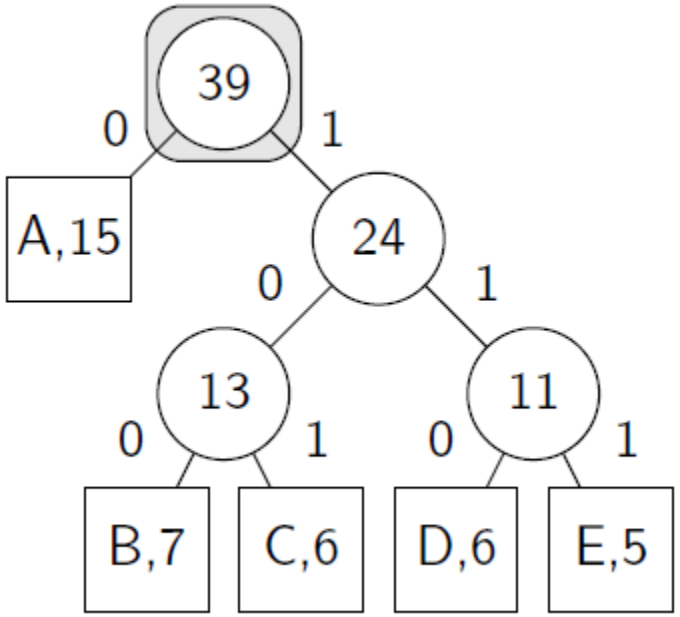
La entropía (esperanza del número de bits para un símbolo) sería 1.75 bits/simbolo

$$1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 3 \times \frac{1}{8} = \frac{7}{4}.$$

Algunos ejemplos son: **Códigos Huffman** y **Codificación Aritmética**

Compresión Huffman

- El codificador crea los códigos mediante un árbol binario en el cual los símbolos están en las hojas y las ramas son etiquetadas usando los dígitos binarios 0 y 1. La distancia de un símbolo a la raíz del árbol define la longitud del código de Huffman asignado a dicho símbolo, y esto depende en última instancia de la probabilidad del símbolo. En concreto, el código asignado a un símbolo es aquel número binario que resulta de viajar desde la raíz hasta dicho símbolo.



A -> 0
B -> 100
C -> 101
D -> 110
E -> 111

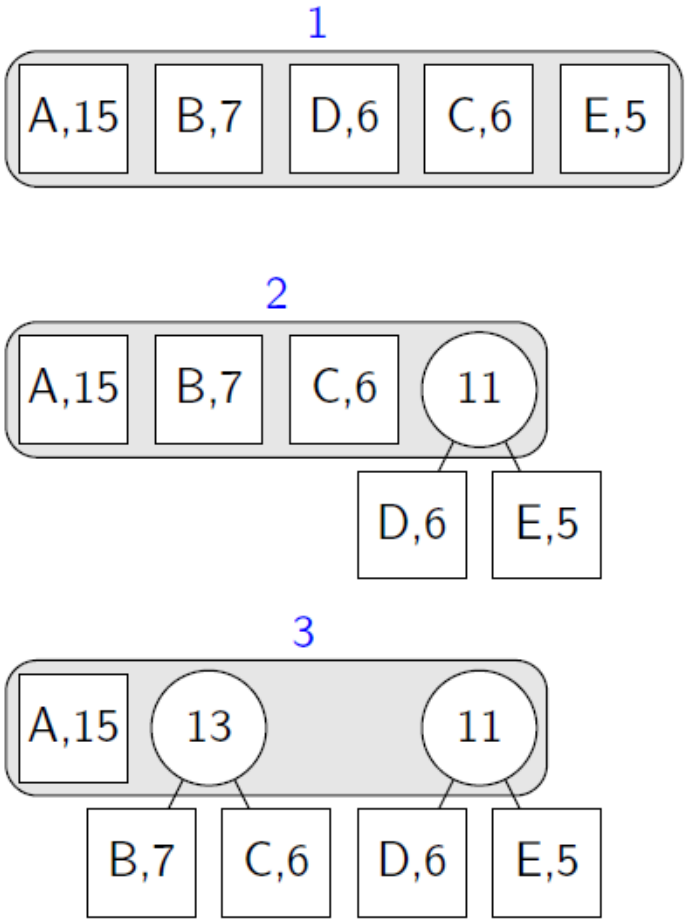
Generación del árbol de Huffman

1. Crear una lista de árboles binarios, en la que cada árbol está formado por un único nodo y cada nodo contiene un símbolo y su probabilidad.
2. Mientras existan al menos 2 árboles en la lista:
   a) Extraer de la lista los 2 árboles con menor probabilidad.
   b) Insertar en la lista un nuevo árbol binario cuyas hojas son los árboles extraídos y cuya raíz es la suma de las probabilidades de estos.

- Nótese que para diseñar el árbol de Huffman es necesario conocer la probabilidad de ocurrencia de todos los posibles símbolos de la secuencia. El modelo estadístico de la fuente es el elemento que proporciona dicha información.
- La descodificación se realiza conociendo el árbol de Huffman o lo que es lo mismo, el modelo estadístico utilizado por el codificador.

Compresión Huffman

Ejemplo



A -> 0
B -> 100
C -> 101
D -> 110
E -> 111

Limitaciones del código de Huffman

- Por desgracia, la codificación de Huffman asigna a cada símbolo un número entero de bits y por definición, el número de bits de información no tiene por qué serlo. En otras palabras, en la codificación de Huffman se cumple que

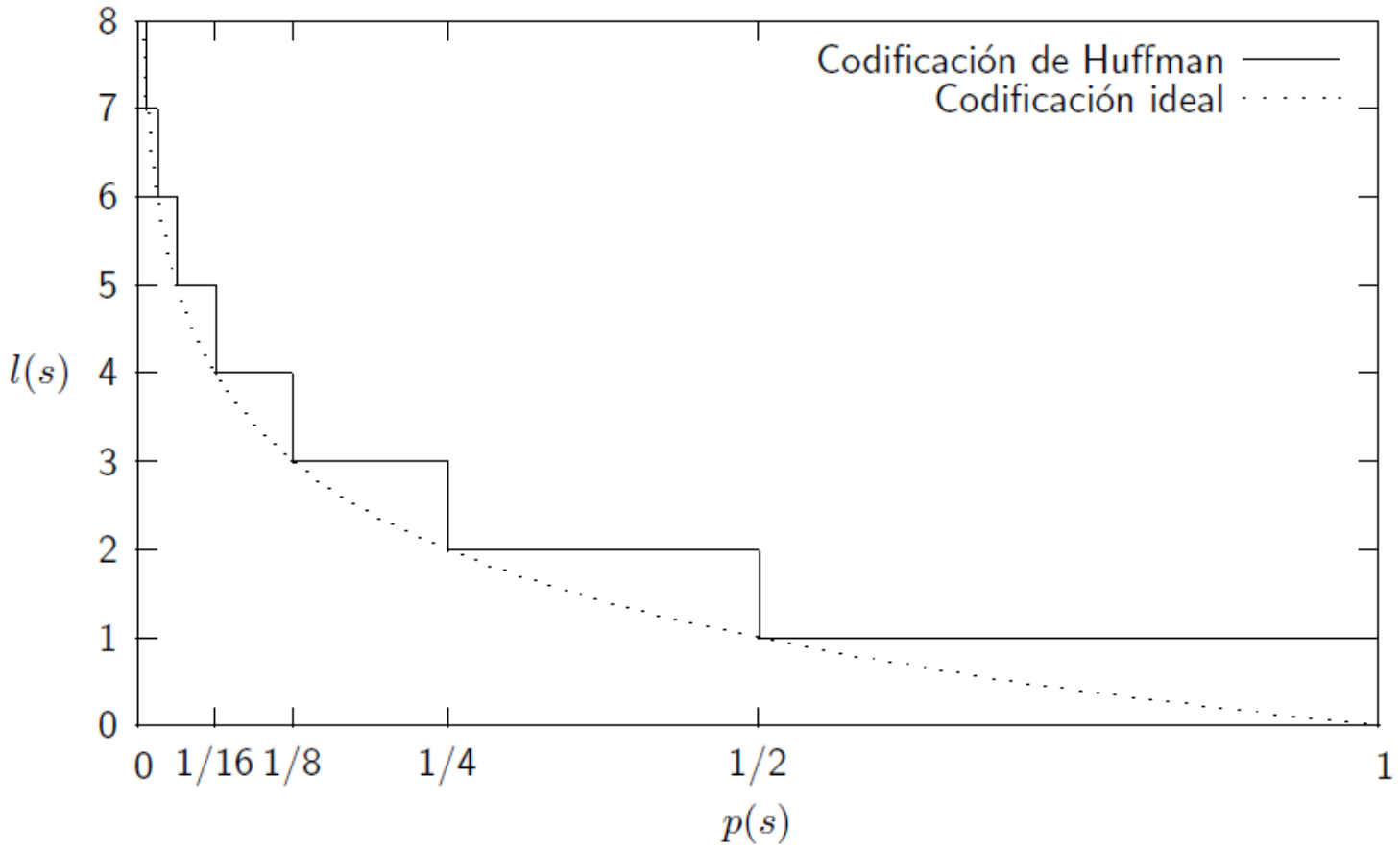
$$l\big(c(s)\big) = \lceil I(s) \rceil,$$

donde $l\big(c(s)\big)$ es la longitud del código de compresión asignado al símbolo s.

- Esto provoca que con cada codificación de un símbolo, hasta casi un bit de datos de redundancia podría ser introducido. Así, para una secuencia completa el número de bits de redundancia puede ser importante.

Compresión Huffman

- Este problema se agudiza cuando el número de símbolos es sólo 2. En este caso, la codificación de Huffman no cambia la representación original binaria (0 y 1), y la cantidad de redundancia introducida para codificar el símbolo más frecuente es importante. Gráficamente:



Codificador Aritmético

No existe una relación directa entre el símbolo y su representación en el bitstream (en los bits del bitstream), es decir, no se puede decir que “estos bits” corresponden a “este símbolo”.

Los símbolos se representan por la codificación de intervalos numéricos

Una secuencia de símbolos se representa por cualquier valor que esté en un intervalo numérico.

Permite codificar un símbolo con fracciones de bits. Muy útil para los muy probables. (En VLC deben tener al menos 1 bit)

Se usa en el H.264/AVC y HEVC a nivel de slice.

Las probabilidades de los símbolos pueden ser actualizadas dinámicamente lo que permite definir varios contextos (tablas de probabilidades distintas).

Codificador Aritmético - Concepto

El rango a usar para representar una secuencia es [0..1]

Supongamos un alfabeto con dos símbolos {A,B}

Debemos conocer sus probabilidades de aparición pA=0.7 pB=0.3 (la suma de las probabilidades del alfabeto debe ser 1)

A es el Most Probable Simbol (MPS) y B es el Least Probable Symbol (LPS)

Para codificar la secuencia el intervalo se divide proporcionalmente a las probabilidades de los símbolos conforme ocurren.

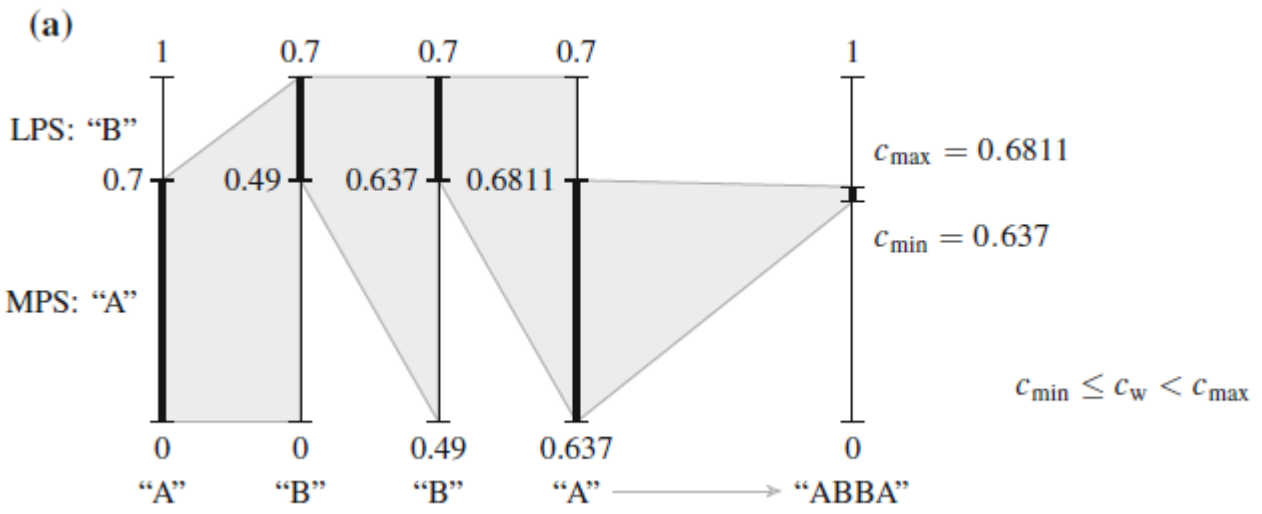
Una vez dividido el intervalo, la secuencia hasta el momento se puede representar por cualquier número dentro de su intervalo.

El símbolo anterior fija el intervalo disponible. Los símbolos (siguientes) se asignan a los intervalos del rango que va quedando disponible.

Con cada paso el intervalo se redefine (expande).

Si se alcanza un tamaño mínimo del intervalo se reescala el intervalo y el proceso continua.

En este punto se escriben los bits correspondientes al intervalo anterior en el bitstream.



Example for arithmetic coding of a sequence of binary symbols "A, B, B, A" with probabilities $p_A = 0.7$ and $p_B = 0.3$. **a** Arithmetic encoding corresponds to an interval subdivision. The sequence can be represented by a code word c_w from the last interval.

Codificador Aritmético - Concepto

El rango a usar para representar una secuencia es [0..1]

Supongamos un alfabeto con dos símbolos {A,B}

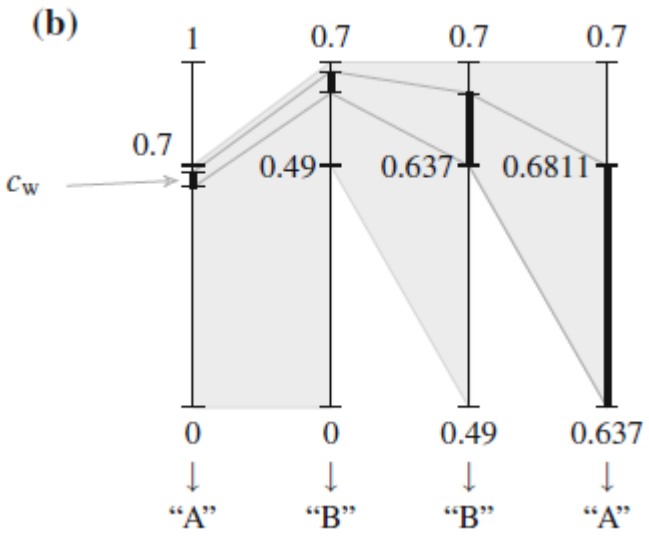
Debemos conocer sus probabilidades de aparición pA=0.7 pB=0.3 (la suma de las probabilidades del alfabeto debe ser 1)

A es el Most Probable Simbol (MPS) y B es el Least Probable Symbol (LPS)

Para codificar la secuencia el intervalo se divide proporcionalmente a las probabilidades de los símbolos conforme ocurren.

Una vez dividido el intervalo, la secuencia hasta el momento se puede representar por cualquier número dentro de su intervalo.

El símbolo anterior fija el intervalo disponible. Los símbolos (siguientes) se asignan a los intervalos del rango que va quedando disponible.



Para decodificar se determina en que parte del intervalo disponible cae el intervalo al que pertenece el codeword.

Esto determina el símbolo que corresponde.

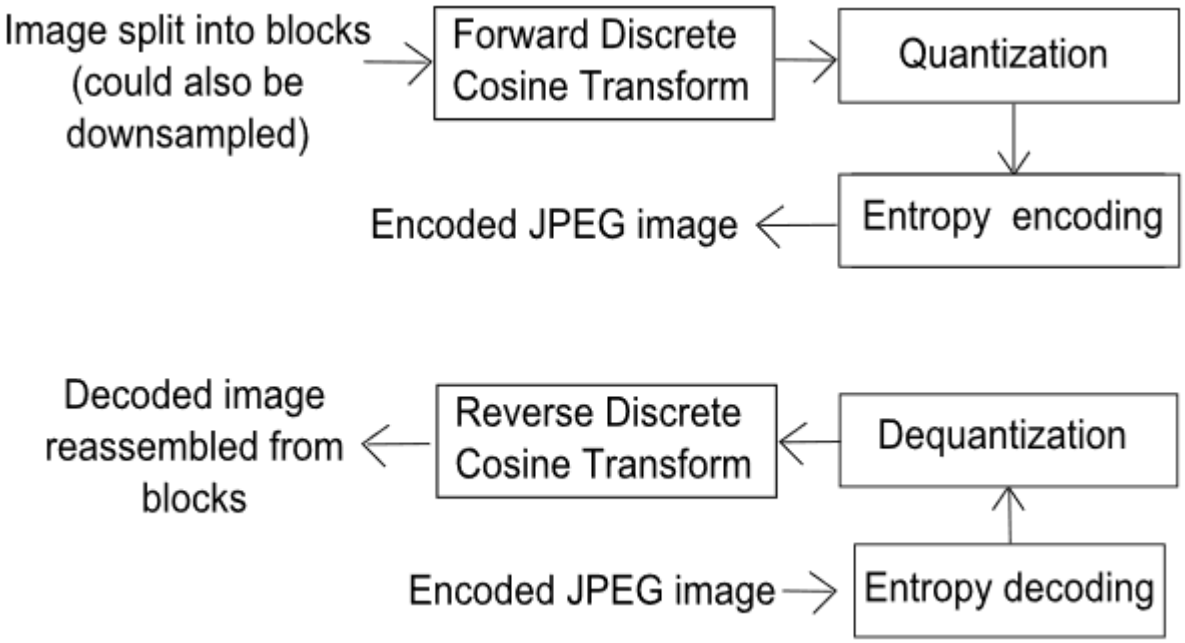
Se expande el intervalo y se vuelve a ver en que parte del nuevo intervalo cae el intervalo del codeword (expandido)

Hasta el final del bitstream

Cuando el tamaño del intervalo disponible es el determinado (el que fijó el punto de reescalado) se reescala el intervalo disponible y se lee un nuevo conjunto de datos del bitstream.

Arithmetic decoding compares the code word to the interval boundaries to successively reconstruct the sequence

Esquema JPEG

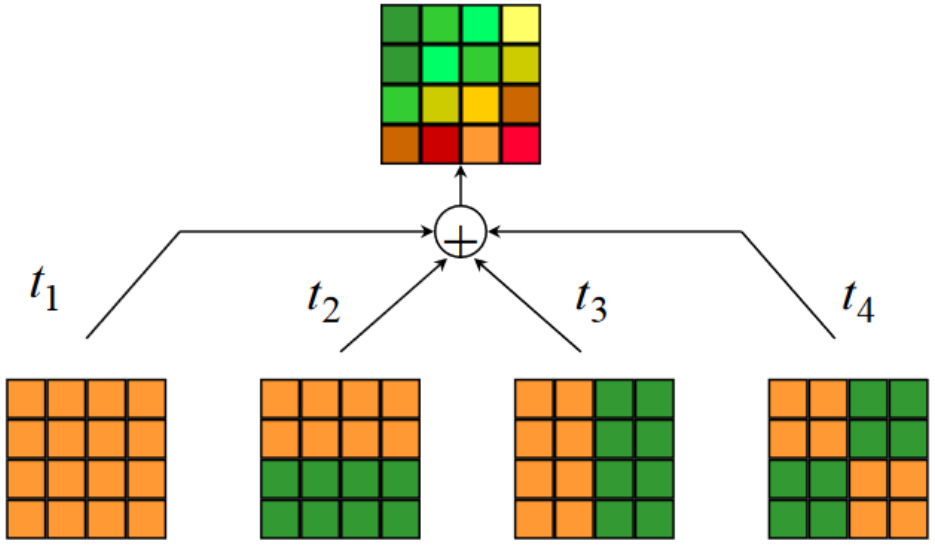


Transformada en Frecuencia

- Motivación para la transformación:

  Obtener una representación de las muestras más eficiente ya que los coeficientes transformados requieren menos bits para su codificación.

- Tipos de transformación:
  - Codificación del habla: prediction -> Code predictor and prediction error samples
  - Codificación de audio: subband decomposition -> Code subband samples
  - Codificación de Imagen: DCT and wavelet transforms -> Code DCT/wavelet coefficients
- La idea es representar una imagen como una combinación lineal de imágenes base especificando los coeficientes lineales de dicha combinación.



Transformada en Frecuencia

Construcción de las imágenes base (bases) guiadas por los siguientes criterios:

- **Compactación de la energía**: Con sólo unas bases es suficiente para representar la imagen. Acumula en pocos coeficientes la mayor parte de la energía de la señal.
- **Decorrelación**: Los coeficientes para bases separadas están uncorrelados.

- **Karhunen Loeve Transform (KLT)** : Es la transformada Optima. Complejidad N^2
- **Discrete Cosine Transform (DCT)** : Muy cercana a la KLT. Complejidad $N \log_2 N$

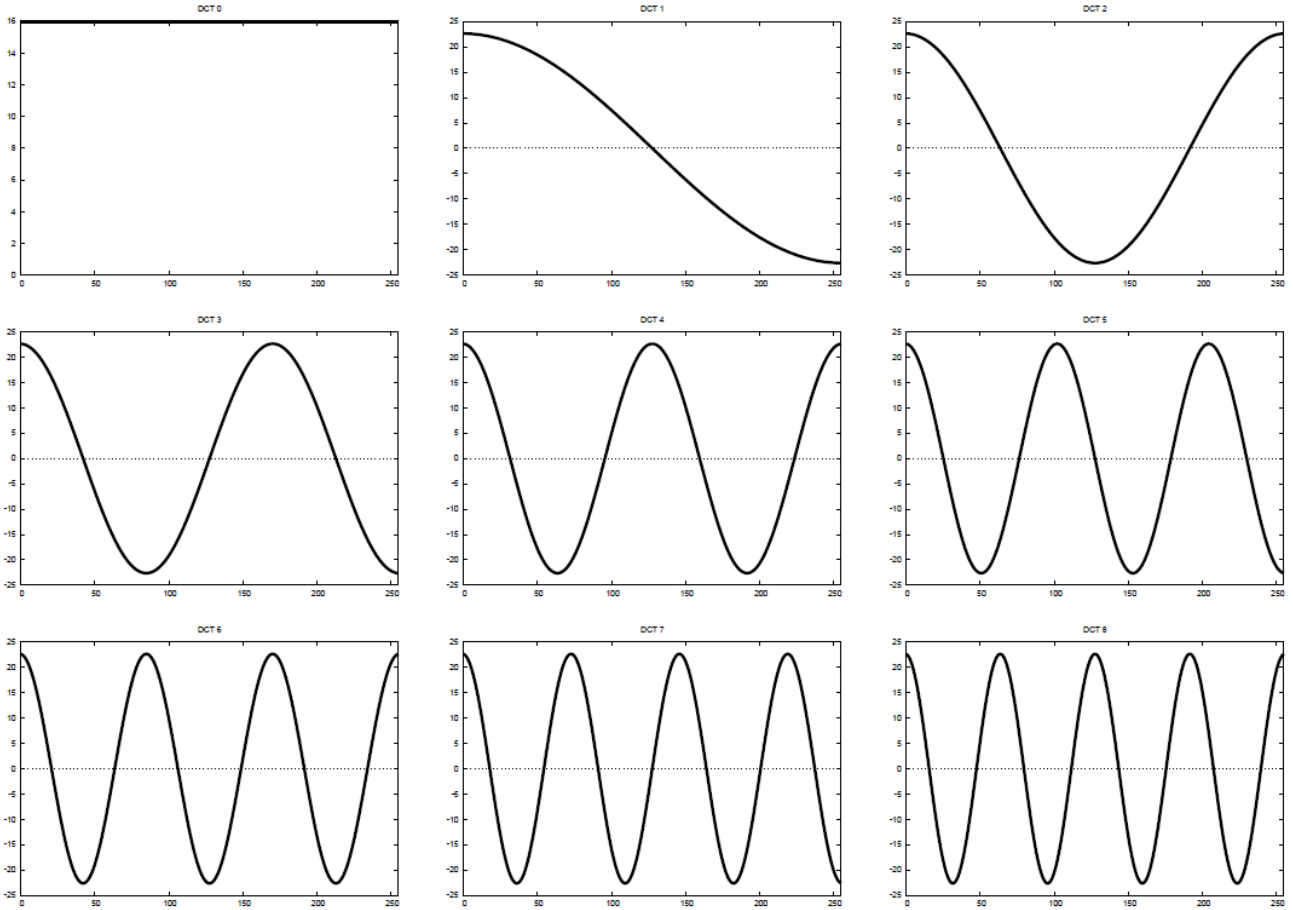
2D-DCT – Ventajas:

- **Rápida**: Existe un algoritmo rápido con complejidad $N \log_2 N$
- **Es idempotente**: La distancia euclidea es invariante en el dominio transformado.
- **Menor coste computacional por bloques**: Es más rápido calcular (N/8 x N/8) 2D-DCTs de 8x8 que una 2D-DCT de NxN
- **Operación in-line**: Este es un factor determinante cuando las imágenes son muy grandes. El compresor y el descompresor pueden trabajar por bloques de 8 x 8 puntos independientemente del tamaño de la imagen (y en paralelo).

Transformada en Frecuencia

Funciones base DCT

- Es interesante conocer la forma de las funciones base de la DCT porque así podemos hacernos una idea de la correlación espacial explotada y del aspecto de las reconstrucciones.
- Las primeras 9 funciones base de la DCT son:



La 2D-DCT por bloques de 8×8 puntos

(YCbCr)

- Cada componente se procesa por bloques de 8×8 puntos y se le calcula la 2D-DCT (Discrete Cosine Transform). Esta transformada puede calcularse a partir de la DCT gracias a que la 2D-DCT es separable. La DCT se calcula como

$$\text{DCT}[u] = \frac{\sqrt{2}}{\sqrt{N}} K(u) \sum_{n=0}^{N-1} x[n] \cos \frac{(2n+1)\pi u}{2n}$$

y su transformada inversa como

$$x[n] = \frac{\sqrt{2}}{\sqrt{N}} \sum_{u=0}^{N-1} K(u)\text{DCT}[u] \cos \frac{(2n+1)\pi u}{2n}$$

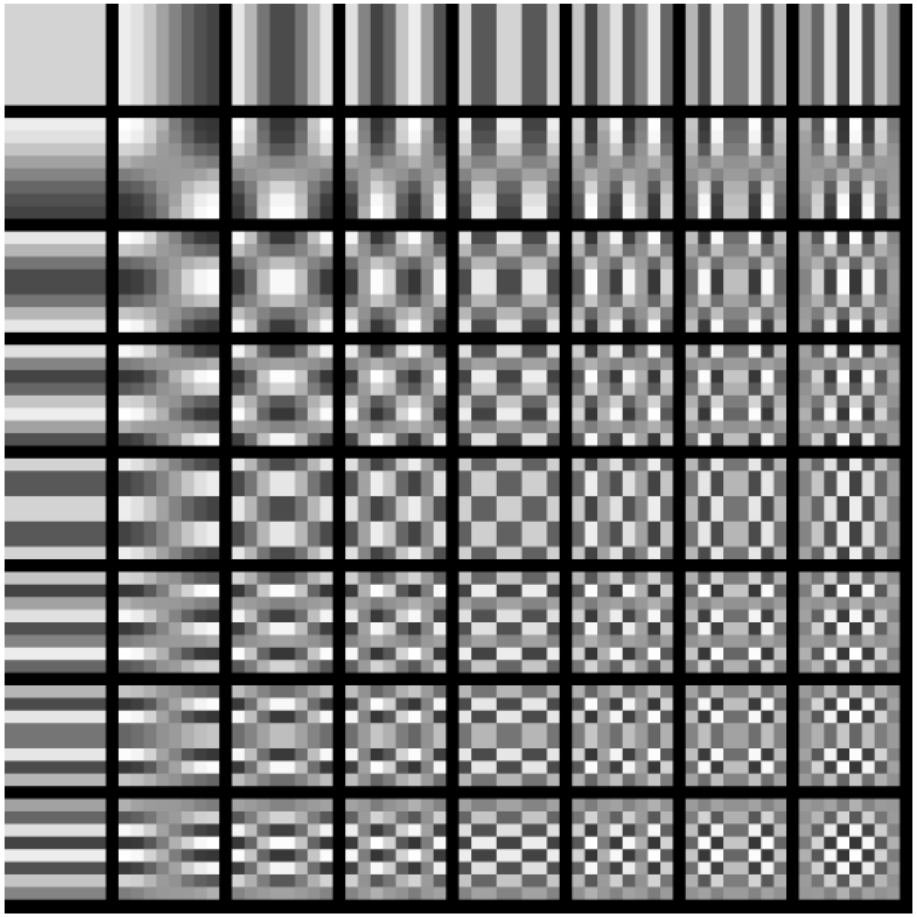
donde N es el número de puntos a transformar y

$$K(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0 \\ 1 & \text{si } u > 0. \end{cases}$$

Transformada en Frecuencia

Funciones base 2D-DCT de 8×8 puntos

- Cada coeficiente de la 2D-DCT de un bloque de 8×8 puntos representa el peso que tienen cada uno de estos 64 patrones a la reconstrucción del bloque.



Transformada en Frecuencia

Representación dominio espacial [0..255] (8bits)

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Shift

$$g = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

Shift Restamos 128

[0..255] -> [-128..127]

2D-DCT

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

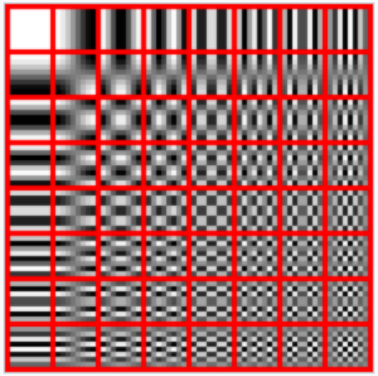
Representación dominio frecuencial

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

Coeficientes DCT

Correspondencia y peso

Bases DCT



Cuantización

Matriz de coeficientes DCT

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

($u \rightarrow$, $v \downarrow$)

Matriz de cuantización estándar

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Para Luminancia
Para Crominancia

Cuantización

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

$$Q_c = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

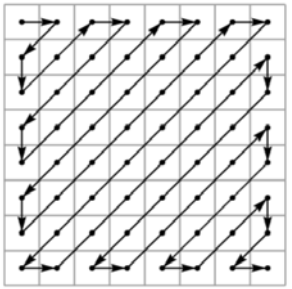
Codificador Entrópico

Matriz de coeficientes DCT cuantizados

Encoding Order in JPEG

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

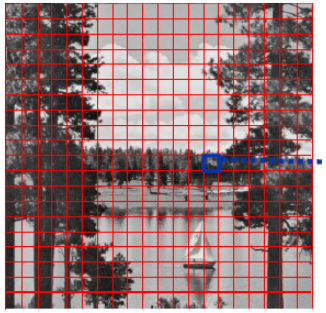
Recorremos **el bloque** de coeficientes cuantizados en ZIG-ZAG order



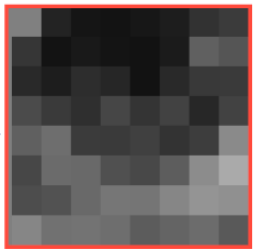
$-26,-3,0,-3,-2,-6,2,-4,1,-3,1,1,5,1,2,-1,1,-1,2,0,0,0,0,0,-$
$1,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$

Quantize

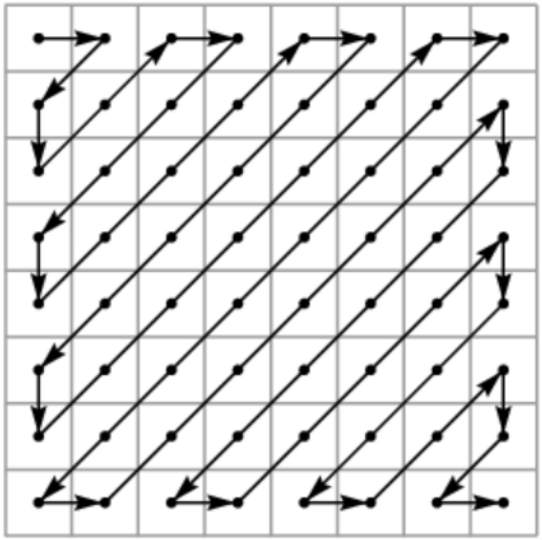
Transform



Input image



8x8 block



Input image

Recorremos los bloques de coeficientes cuantizados de **La imagen** en ZIG-ZAG order

Baseline Sequential Encodign

Progresive Encodign

Codificación de Vídeo

Un video no es mas que una secuencia de imágenes

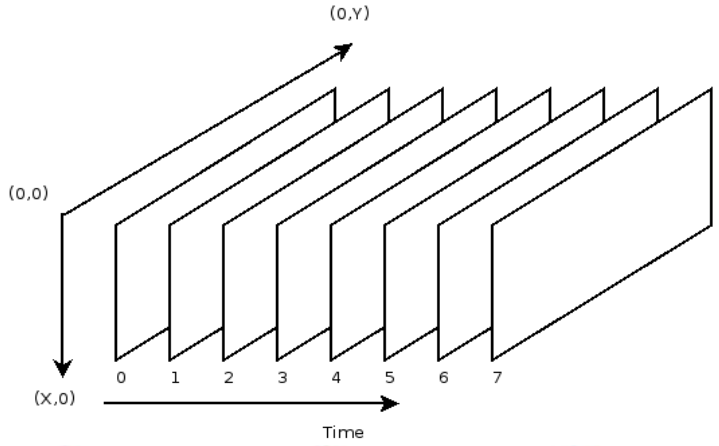
Atributos: Figura 1 representa los atributos del video

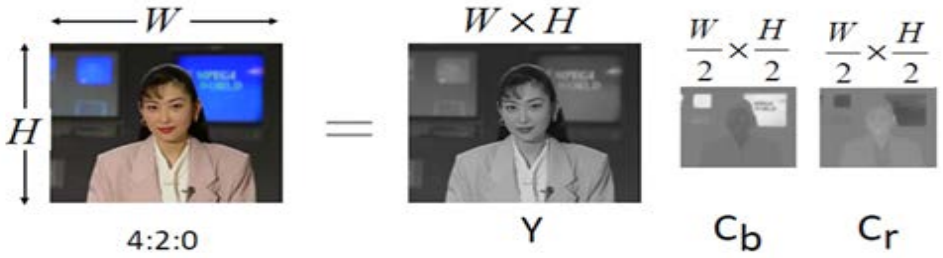
-Height

- Width

- Frame Rate

- Pixel Values





Fundamentos

- Vídeos = Secuencias de imágenes.
- Los compresores de secuencias de imágenes eliminan tanto la **redundancia espacial** como la **temporal**. Esta última es consecuencia directa de que en la mayoría de las secuencias, las imágenes adyacentes en el tiempo son muy parecidas y por lo tanto, para codificar la siguiente (por ejemplo), es muy eficiente indicar sólo las diferencias con respecto a la actual.
- Debido a la cantidad de datos que se generan cuando se comprime una señal de vídeo, la inmensa mayoría de los **compresores** de secuencias de imágenes (también llamados compresores de vídeo) son **lossy**, porque las tasas de compresión son mucho mayores.

Motivación

- Las secuencias de vídeo digital en formato PCM ocupan mucha memoria. Un segundo de vídeo en color, a una resolución de 640×480 puntos/imagen y 25 imágenes/segundo necesita:

$$25\frac{\text{imágenes}}{\text{segundo}} \times 640 \cdot 480\frac{\text{puntos}}{\text{imagen}} \times 24\frac{\text{bits}}{\text{punto}} = 184.320.000\frac{\text{bits}}{\text{segundo}}$$

Esto significa que, por ejemplo, una hora de dicho vídeo ocupa:

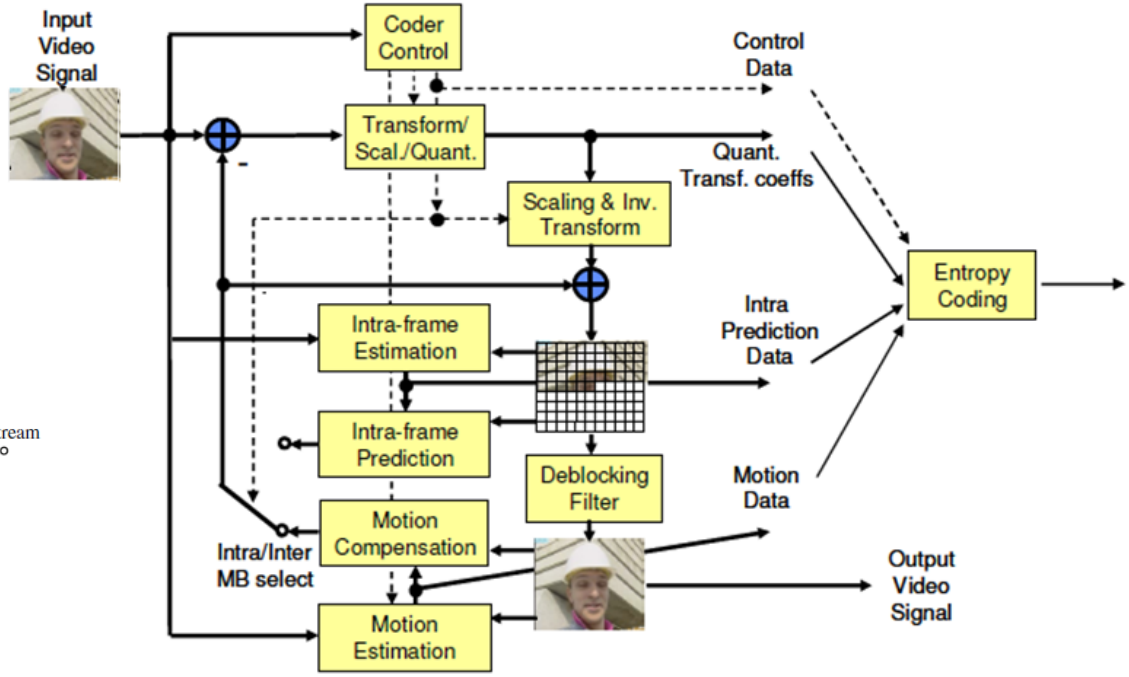
$$184.320.000\frac{\text{bits}}{\text{segundo}} \times 3.600\frac{\text{segundos}}{\text{hora}} \times \frac{1\text{ G}}{1.024^3} \times \frac{1\text{ byte}}{8\text{ bits}} \approx 77\text{ Gbytes}$$

Las etapas DCT, Q, BC y VLC

- MPEG-1 utiliza básicamente las mismas técnicas que JPEG para comprimir las imágenes residuo:
  - **DCT (Discrete Cosine Transform):** es la 2D-DCT y elimina la correlación espacial por bloques de 8×8 puntos.
  - **Q (Quantization):** básicamente elimina los planos de bits menos significativos de los coeficientes DCT. En ellos tiende a acumularse el ruido de las imágenes y además contienen información visualmente poco relevante.
  - **VLC (Variable Length Coding):** al igual que en JPEG, los coeficientes DCT cuantificados se recorren en zig-zag y se comprimen usando un código estático de Huffman.
- La cantidad de información eliminada por Q depende del bit-rate de salida y del deseado por el usuario. La etapa **BC (Bit-rate Control)** se encarga de que a la salida no se generen más datos de los permitidos.

Modelo hibrido de codificación

- Block-based
- Variable block sizes
- Block motion compensation
- Fractional-pelmotion vectors
- Spatial intra prediction
- Spatial transform of residual difference
- Integer-based transform designs
- Arithmetic or VLC-based entropy coding
- In-loop filtering to form final decoded picture



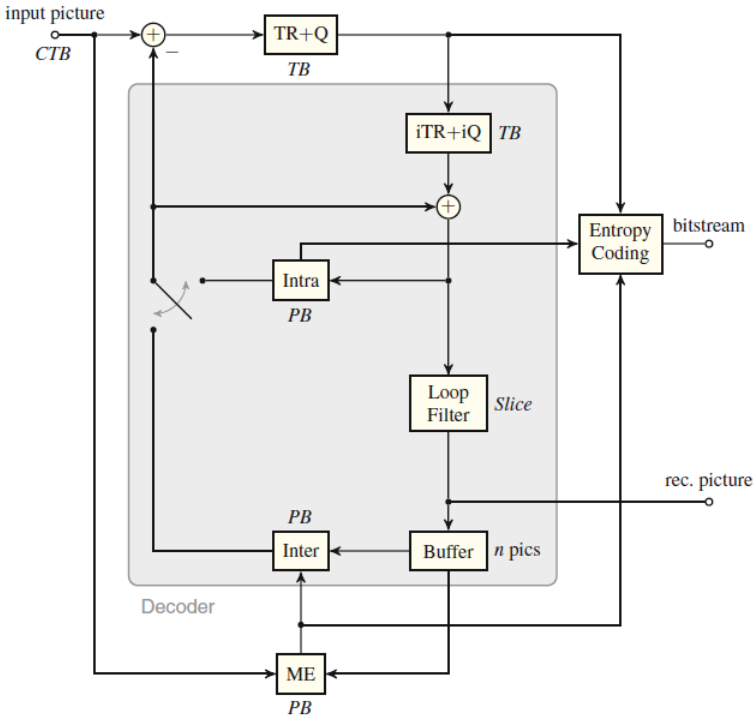


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Transformación y Cuantización

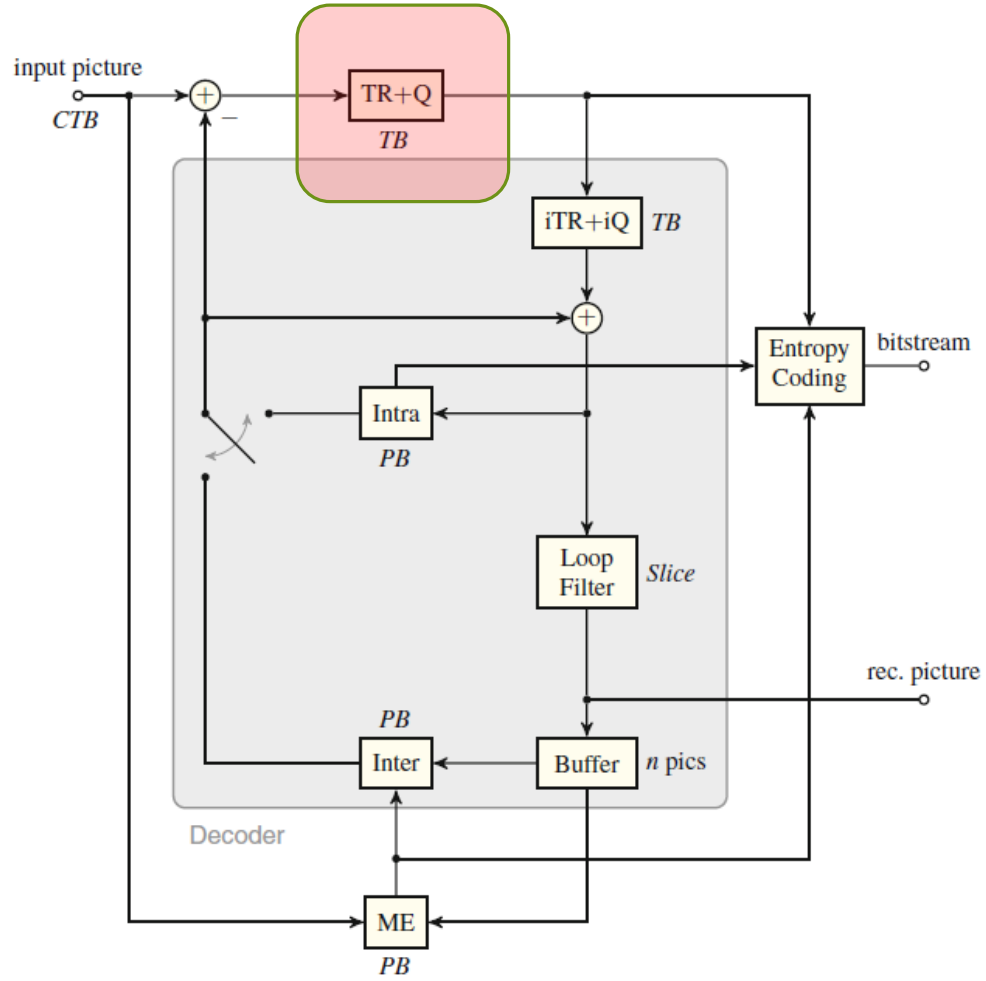


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Transformada y Quantización

- Se encarga de realizar la transformada DCT/DST para pasar al dominio de la frecuencia. Se aplica por bloques y obtenemos bloques de coeficientes.
- La Quantización es la reducción del peso de los coeficientes. Aquellos coeficientes que están por debajo de un umbral desaparecen. Es el único punto donde se pierde información.

Codificación Entrópica

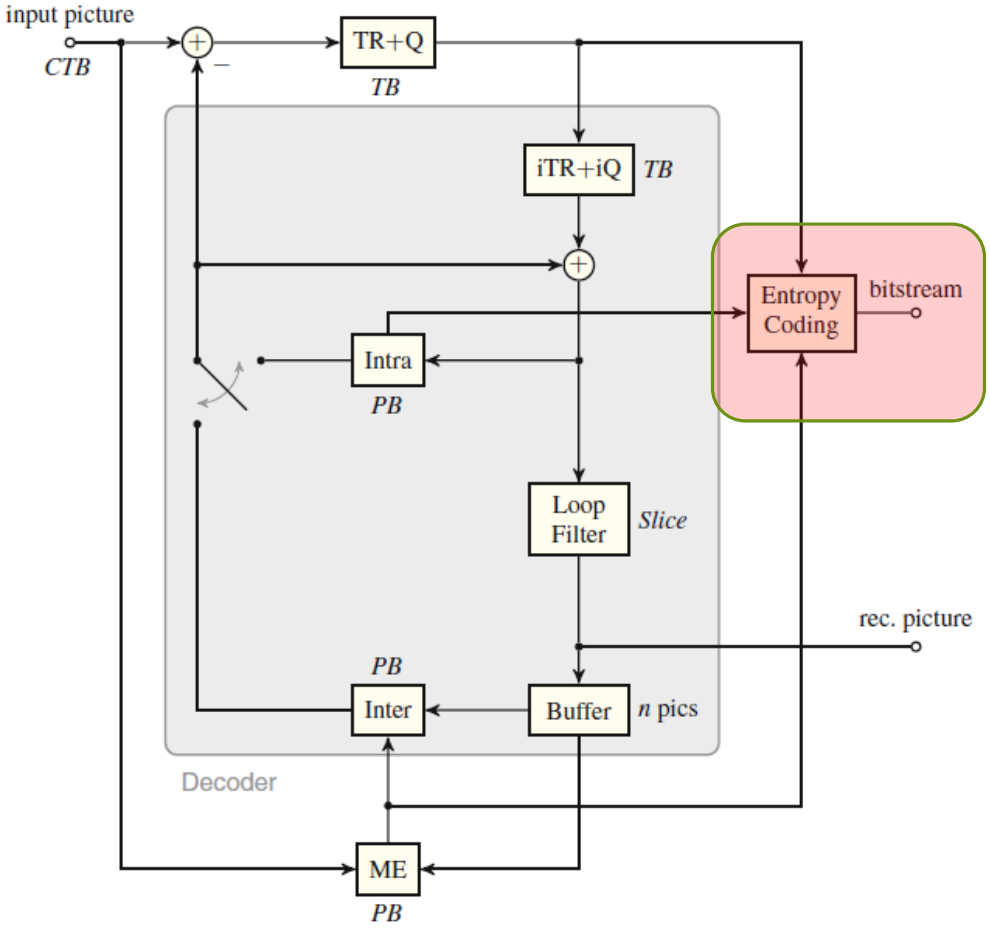


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Codificación entrópica

- Se encarga transformar toda la información del video, sus cabeceras, ajustes, flags, etc, en un flujo de bits llamado bitstream.
- El bitstream tiene una estructura bien definida por el estándar.
- Esta etapa codifica el bitstream en el orden definido.

Transformación y Cuantización Inversas

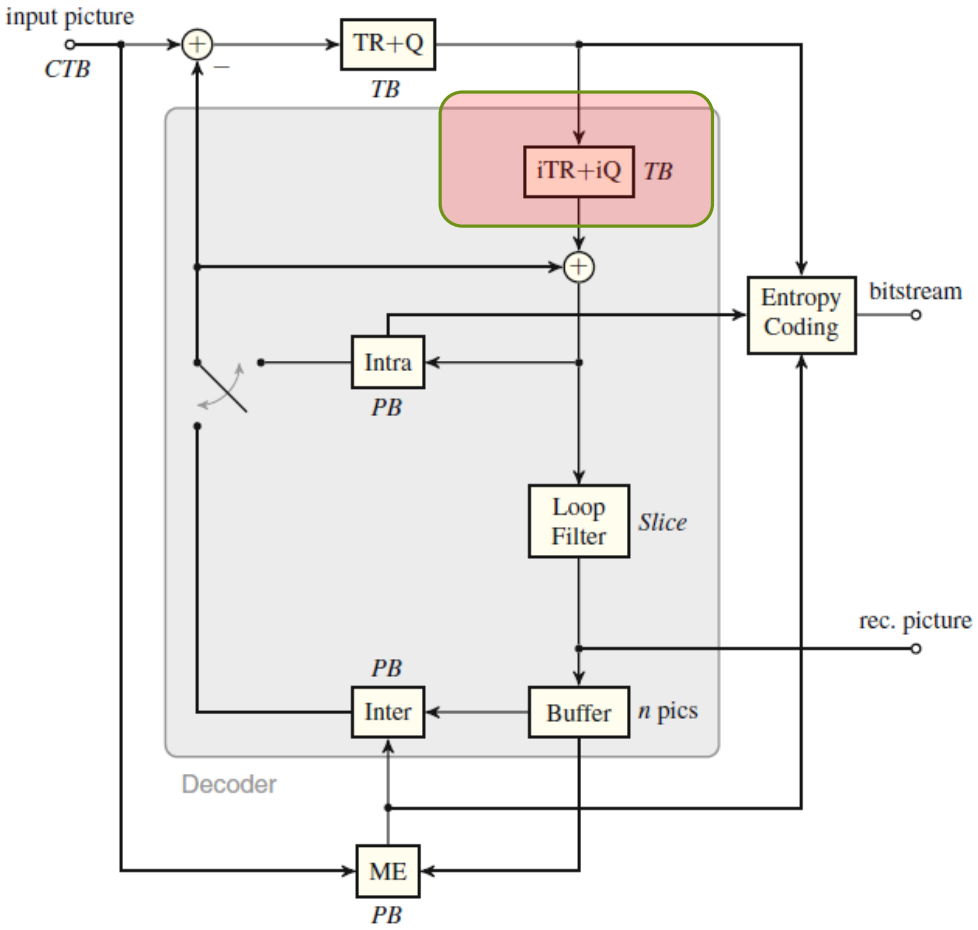


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Cuantización y Transformación Inversas

- Se encarga de recuperar los coeficientes cuantizados a su valor original
- Se transforma de nuevo al dominio del espacio.
- Aquellos coeficientes que se eliminaron no se pueden recuperar.
- La imagen no se recupera exactamente igual.

Estimación Intra

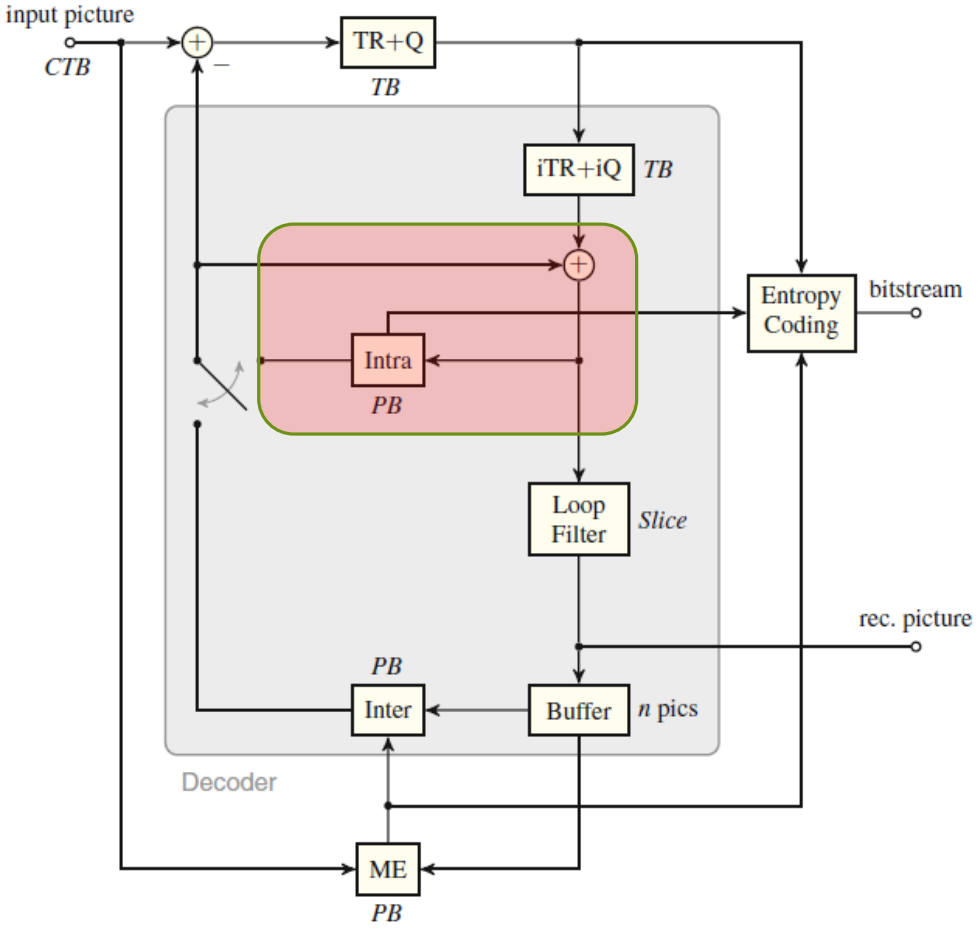


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Intra Prediction

- En el decodificador recibiremos las diferencias del bloque (transformado y cuantizado) con su predicción.
- Se encargan de tratar de predecir cómo es el bloque actual. Etapa muy costosa computacionalmente.
- Se basa en la información de los bloques previamente decodificados
- Suma la predicción a lo que recibe.

Loop Filtering

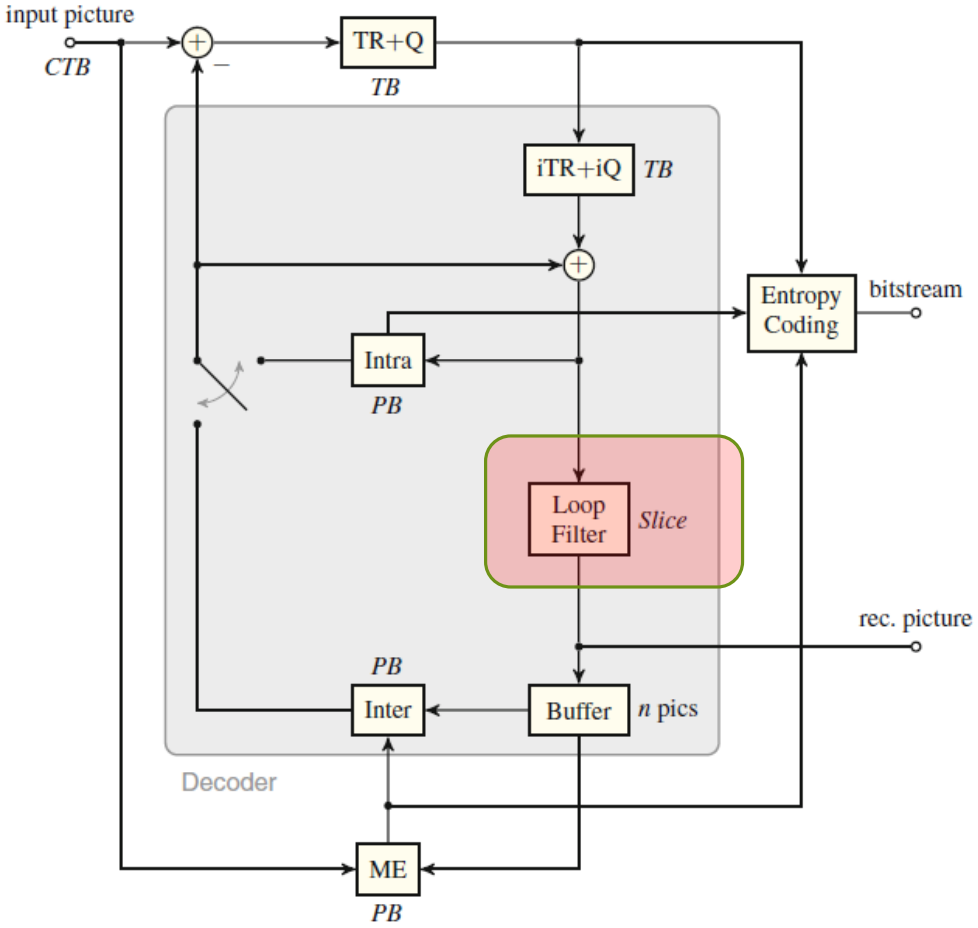


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Loop Filter

- Una vez reconstruida la imagen se le aplican una serie de filtros para mejorar su aspecto
- Se aplican los filtros a toda la imagen.

Buffer de frames

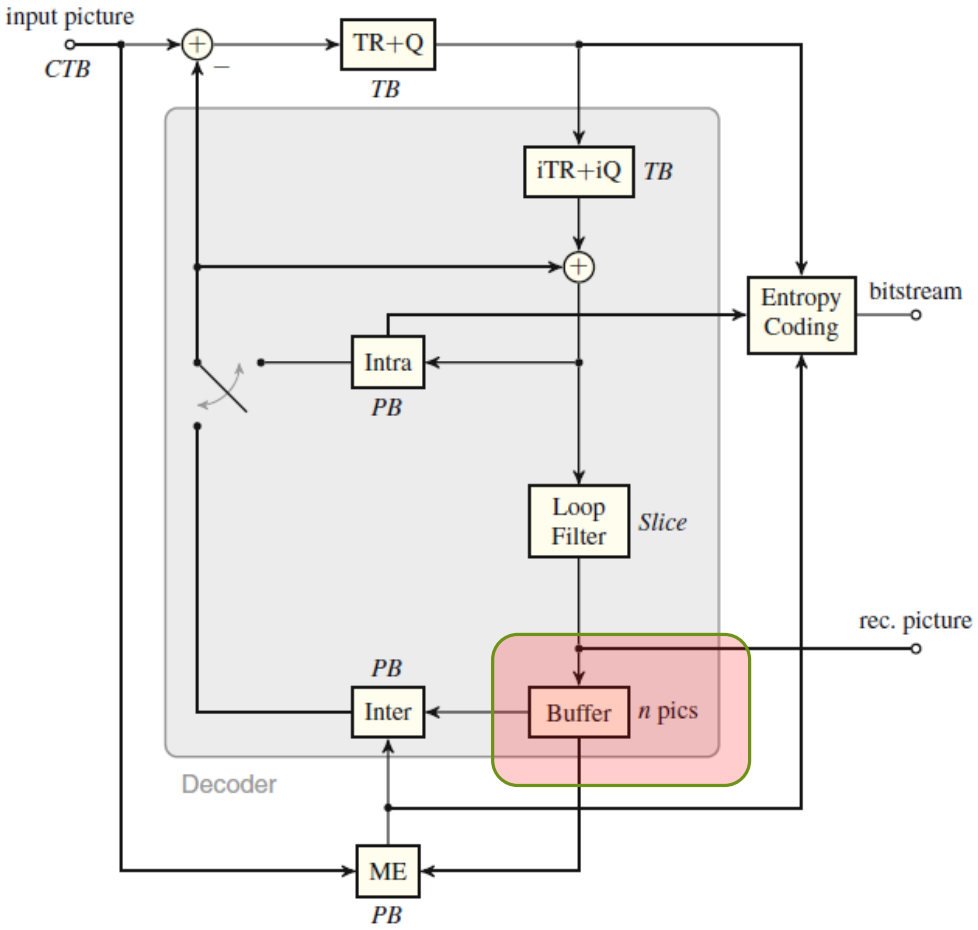


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Previous Frames Buffers

- Un conjunto de Frames ya decodificados se guardan en un buffer
- La idea es que sirvan para predecir el movimiento del frame actual basándose en donde estaban/estarán los objetos en frames anteriores/posteriores

Estimación de Movimiento

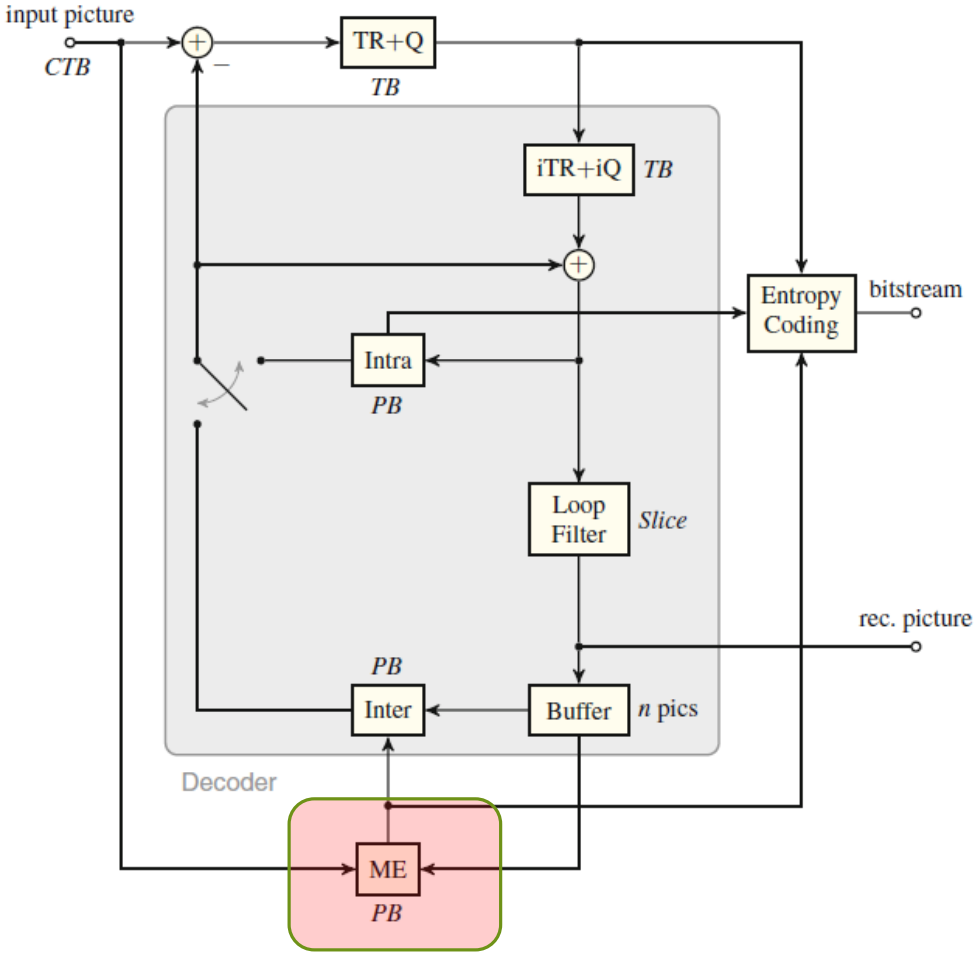


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Motion Estimation

- Se encarga de realizar la estimación de movimiento
- Se aplica por bloques.
- Se busca el contenido del bloque en frames que están en el buffer, anteriores o posteriores
- Hay varios patrones/algoritmos de búsqueda.
- Es la etapa mas costosa computacionalmente

Compensación de Movimiento

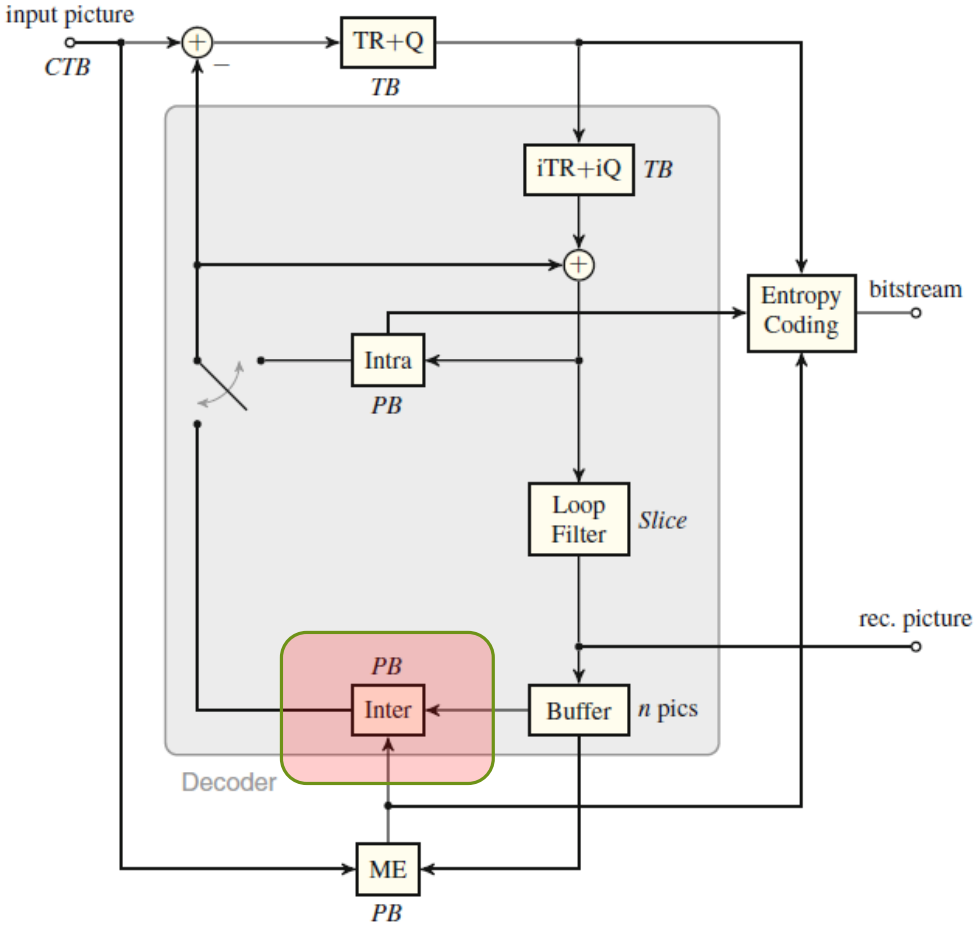


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Estimación Inter - Compensación

- Se encarga de realizar predecir cómo será el bloque actual teniendo en cuenta la predicción de movimiento.
- Construye la predicción de un bloque realizando una compensación de movimiento.

Decisión Intra/Inter

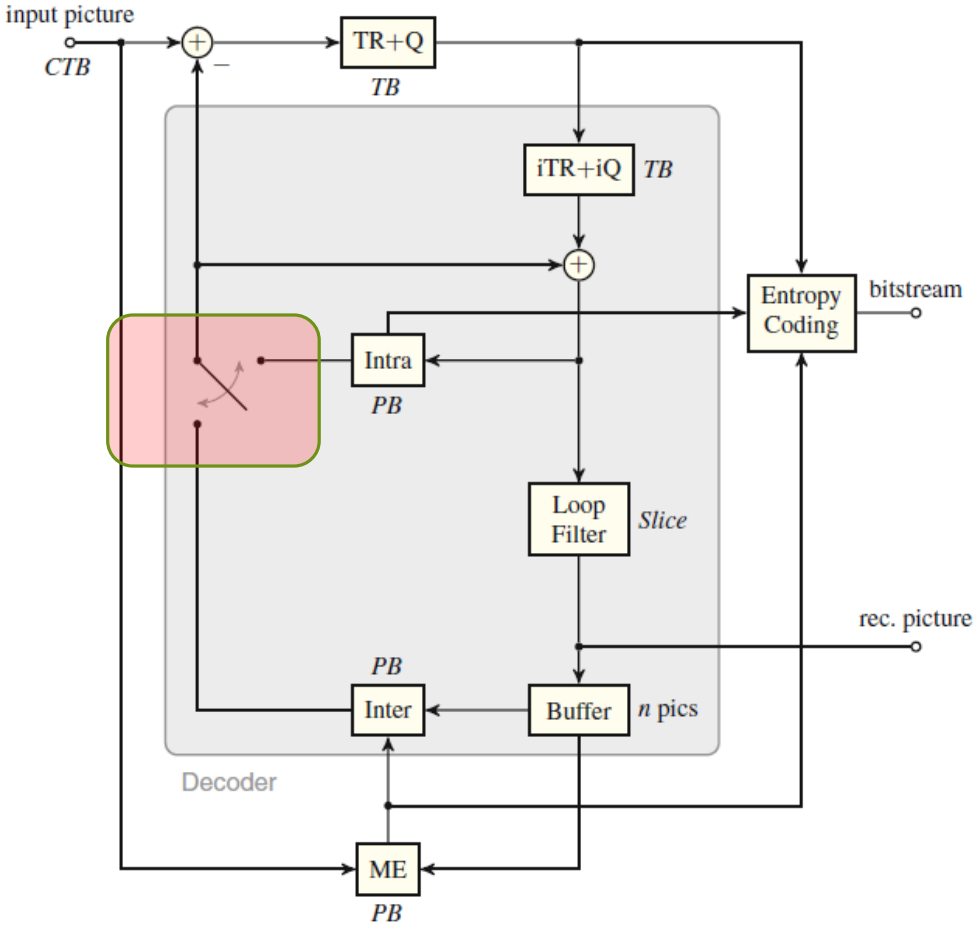


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Intra / Inter Selector

- Se encarga de decidir que predicción utilizar, la Intra o la Inter.
- La decisión puede ser aplicada a todos los frames de una secuencia (Intra mode), solo a algunos (Random Access mode), e incluso sólo a ciertos bloques de algunos frames.

Residual calculation

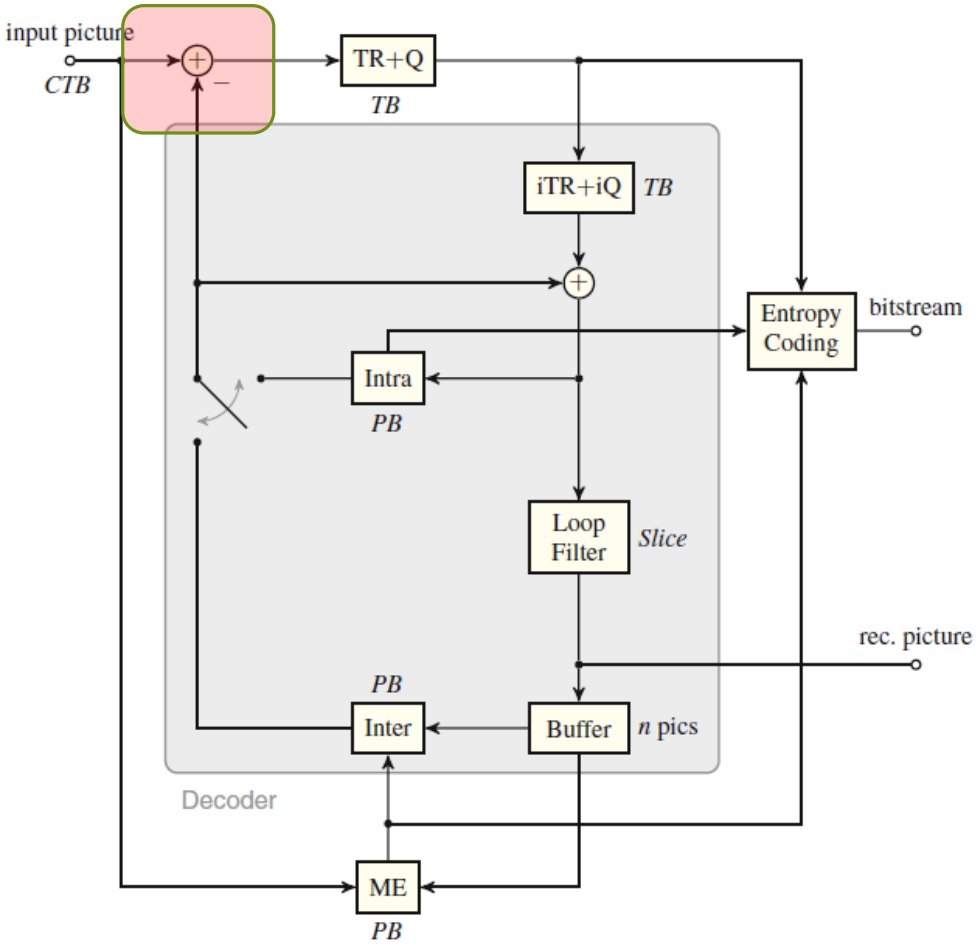


Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

Prediction differences

- Se encarga de obtener las diferencias del bloque actual con la predicción del mismo que va a tener el decodificador.
- Sólo se codifican las diferencias con la predicción lo que reduce el bitrate a enviar.
- El primer frame va completo.

Codificación Entrópica

En función del tipo de información a codificar, el criterio de diseño de la codificación entrópica varía.

- Elementos de sintaxis que indican características de **alto nivel** del bitstream se suelen codificar con una representación en **Fixed Length Codes (FLC)** que son fáciles de acceder por cualquier software para adquirir información general del bitstream. Normalmente al comienzo del bitstream y alineados al byte en posiciones relativas al inicio.
- Elementos de sintaxis que llevan información a nivel de Frame, Slice o bloque se suelen codificar con **Variable Length Codes (VLC)**.

  La mayor parte del bitstream está codificado de esta forma.

  Se busca la eficiencia por lo que se opta por esquemas adaptativos basados en contextos **Context-Dependent Adaptive Models**.

  En H.264/AVC y HEVC se utilizan FLC, CAVLC (Context Adaptive VLC) y CABAC (Context Adaptative Binary Arithmetic Coder) en función del element a codificar.

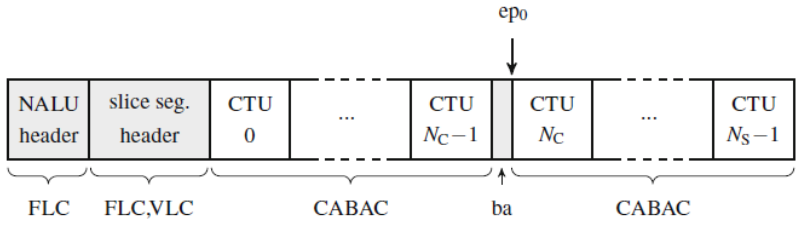


Fig. 10.1 Illustration of the application of fixed-length coding (FLC), variable length coding (VLC), and context-based adaptive binary arithmetic coding (CABAC) in a coded slice segment NAL unit with N_S CTUs and a tile boundary before CTU N_C. Byte alignment (ba) is required to allow to start decoding at the entry point ep_0

Motion Estimation

La etapa ME

- Debido a su sencillez y eficiencia, la *estimación de movimiento* o ME (*Motion Estimation*) es la técnica más utilizada (no sólo en MPEG-1) para reducir la redundancia temporal en las secuencias de vídeo digitales.

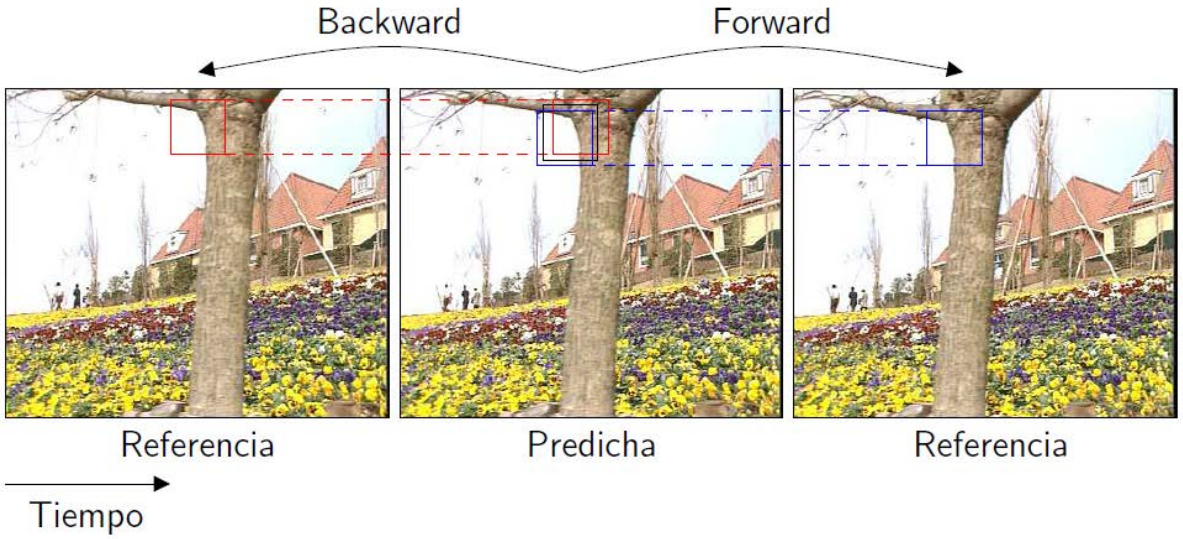
- Sólo se realiza en el compresor porque, como veremos, es un proceso costoso en términos de operaciones artiméticas. Esto responde a la necesidad de "comprimir una vez, descomprimir muchas".

- ME se basa en que una imagen de la secuencia (*imagen predicha*) puede codificarse a partir de otra (*imagen de referencia*) con muy poco error si somos capaces de encontrar una proyección (*imagen predicción*) de la imagen de referencia que se asemeje mucho a la imagen predicha.

Estimación hacia delante, hacia detrás y bi-direccional

- Dependiendo de la posición relativa (en el tiempo) de la(s) imágen(es) de referencia y de la imagen predicha, hablaremos de:

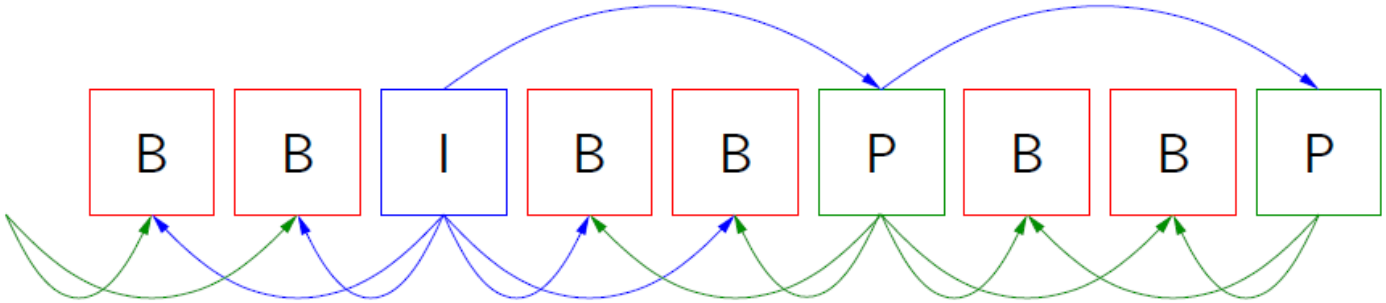
1. **Predicción hacia delante (forward):** cuando la imagen de referencia es posterior a la imagen predicha.
2. **Predicción hacia detrás (backward):** cuando la imagen de referencia es anterior a la imagen predicha.
3. **Predicción bi-direccional (bi-directional):** cuando la imagen predicha está entre dos imágenes de referencia.



Imágenes I, P y B

- Dependiendo de la forma en que se estima una imagen, MPEG-1 distingue entre:

  1. **I-pictures (Intra-coded):** si la imagen no es estimada a partir de ninguna otra.
  2. **P-pictures (Predictive-coded):** si la imagen es estimada a partir de UNA imagen de predicción I o P, anteriores en el tiempo.
  3. **B-pictures (Bidirectionally predictive-coded):** cuando la imagen predicción se genera a partir de dos imágenes de referencia I o P, necesariamente una anterior y otra posterior. Las imágenes B nunca se toman como imágenes de referencia.



El GOP (Group Of Pictures)

- En MPEG-1, un GOP es un conjunto de imágenes que son continuas en el tiempo. Se cumple que:

  1. Todo GOP debe poseer al menos una imagen de tipo I. Estas imágenes suelen colocarse periódicamente en el tiempo (aproximadamente cada $0{,}5$ segundos), y generalmente es la primera del GOP.
  2. Su longitud no está limitado (excepto por el número de imágenes de la secuencia). Sin embargo, tamaños superiores a 16 no son frecuentes para facilitar el acceso aleatorio y el control de errores en el bit-stream.
  3. Su composición, en términos de imágenes I, P y B, puede cambiar a lo largo de la secuencia comprimida. El MPEG aconseja que ocurran 1 I-picture por cada 3 P-pictures y de 2 a 5 B-pictures por cada P-picture.

El GOP (Group Of Pictures)

4. Puede existir más de una imagen I por GOP.
5. Algunos ejemplos de GOP*:

```
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
---------------------------> time
I
I P P
I B P B P
B B I B P B P
B B I B B P B B P B B P
B I B B B B P B I B B I I
I B B P B B P B B P B B P
B B I B B P B B P B B P B B P
```

Display (time) and bit-stream orders

- Cuando existen imágenes de tipo B en un GOP, en MPEG-1 se habla de dos tipos de ordenación de imágenes:

  1. **Display order:** es aquel en el que las imágenes necesitan ser mostradas en el visualizador. Ningún GOP en este orden puede comenzar en un P-picture o acabar en un B-picture.
  2. **Bit-stream order:** es aquel en el que las imágenes necesitan ser colocadas en el bit-stream. En este se cumple que todo GOP comienza por un I-picture.

- Estas ordenaciones son diferentes sólo cuando aparecen imágenes B. Esto es así porque como estas imágenes dependen de la siguiente I o P (en display order), dicha imágen I o P debe ser transmitida antes que la B.

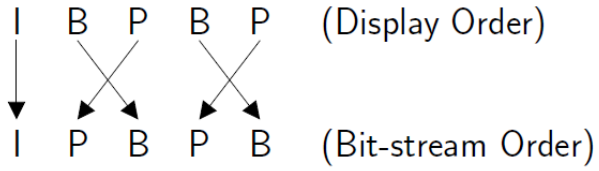
Motion Estimation

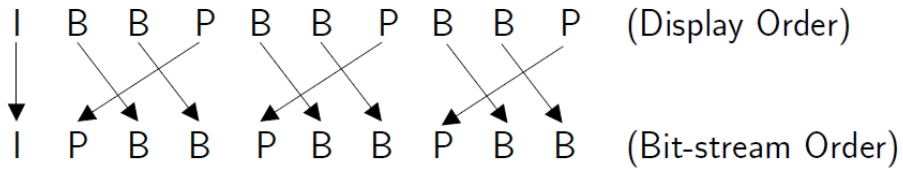
Display (time) and bit-stream orders

- Cuando existen imágenes de tipo B en un GOP, en MPEG-1 se habla de dos tipos de ordenación de imágenes:

  1. **Display order:** es aquel en el que las imágenes necesitan ser mostradas en el visualizador. Ningún GOP en este orden puede comenzar en un P-picture o acabar en un B-picture.
  2. **Bit-stream order:** es aquel en el que las imágenes necesitan ser colocadas en el bit-stream. En este se cumple que todo GOP comienza por un I-picture.

- Estas ordenaciones son diferentes sólo cuando aparecen imágenes B. Esto es así porque como estas imágenes dependen de la siguiente I o P (en display order), dicha imágen I o P debe ser transmitida antes que la B.





GOP's abiertos y cerrados

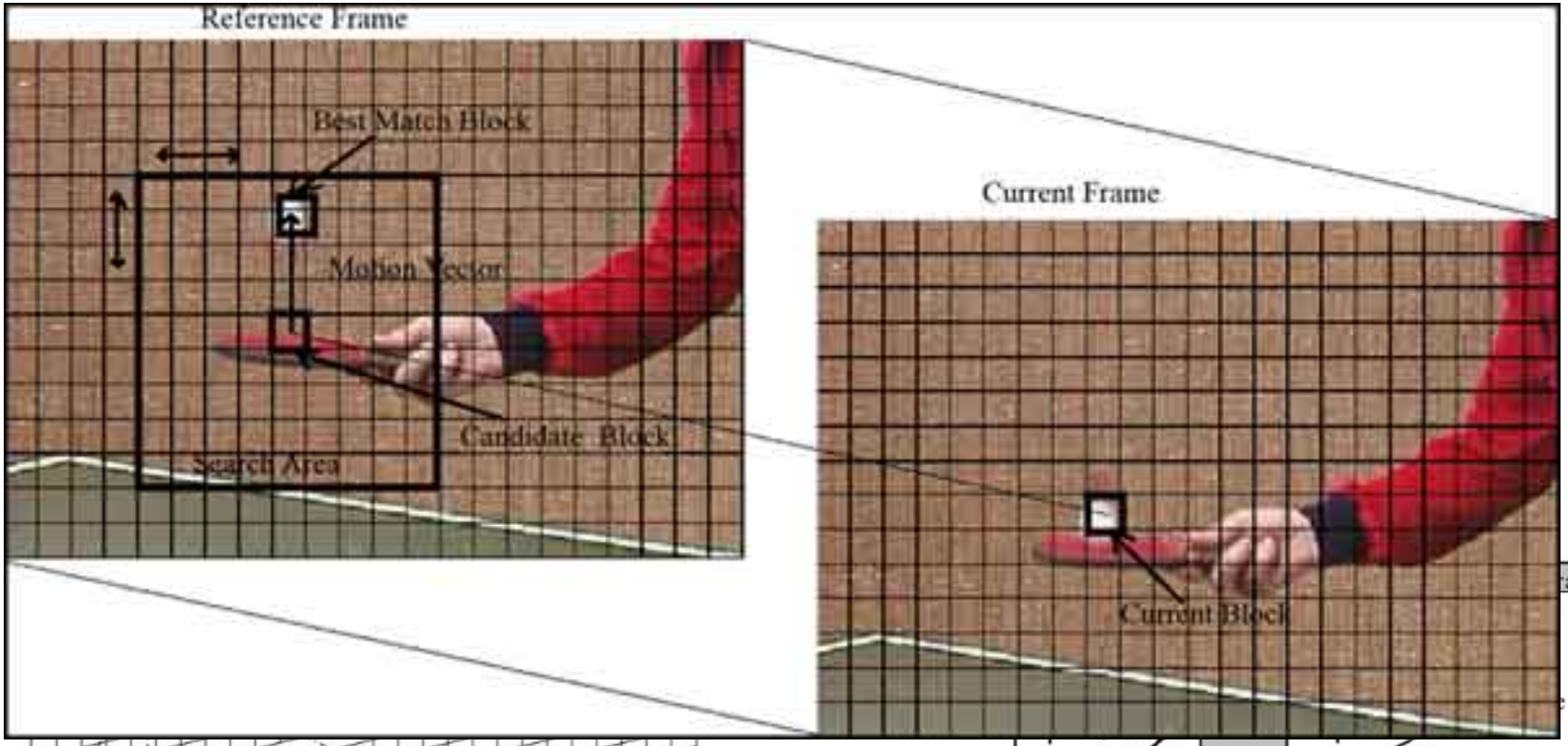
- En MPEG-1 se habla de:

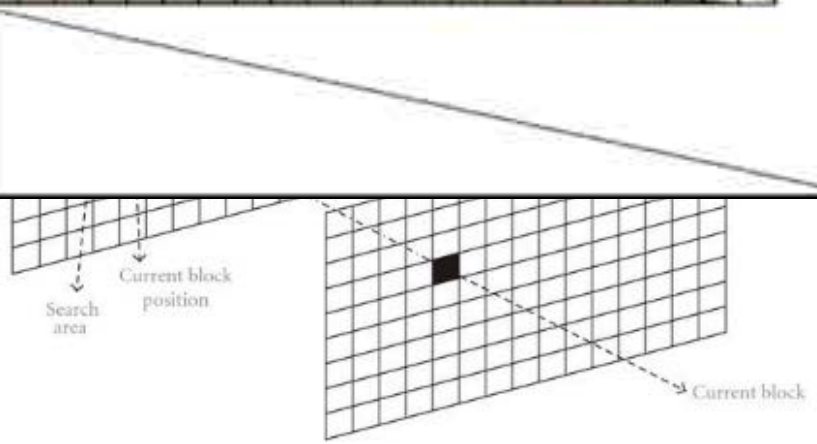
1. **GOP cerrado (closed)** cuando puede descodificarse (y visualizarse) de forma independiente a cualquier otro GOP. En display order, los GOP's cerrados comienzan necesariamente por un I-picture.
2. **GOP abierto (open)** cuando necesita el GOP anterior para descodificarse. Los GOP's abiertos comienzan siempre en un B-picture, en display order. Estos GOP's se utilizan rara vez porque dificultan el acceso aleatorio a las imágenes del vídeo.

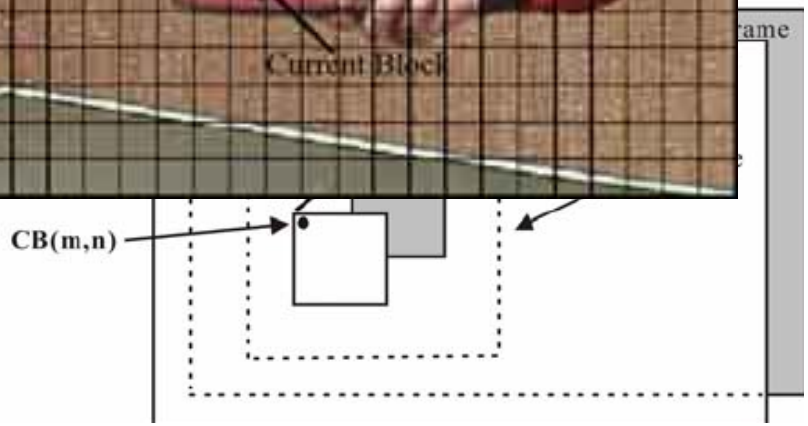
```
I  B  B  P  B  B  P  B  B  P  B  B  P        (closed GOP)
B  B  I  B  B  P  B  B  P  B  B  P  B  B  P   (open GOP)
```

Motion Estimation

Estimación de movimiento basada en la búsqueda de bloques







Estimación de movimiento

La técnica de Block Matching se basa en dos asunciones básicas.

1. Los movimientos translacionales son constants para bloques pequeños (8x8 o 16x16) de la imagen.

   Esto es lo mismo que decir que existe un tamaño mínimo de objeto que es mas grande que el tamaño de bloque.

2. Hay un rango máximo (predeterminado) para las componentes vertical y horizontal del vector de movimiento en cada pixel.

   Esto es lo mismo que asumir una velocidad máxima para el movimiento de los objetos de la escena.

   Esto restring el rango de los vectores a ser considerados por lo que se reduce el coste del algoritmo (se reduce el área de búsqueda)

Estimación de movimiento basada en la búsqueda de bloques

- En la estimación de movimiento se puede utilizar una precisión de 1 punto o de $1/2$ de punto. En este caso, tanto el macrobloque de referencia como el predicho deben interpolarse según:

A	X_1	B
X_2	X_3	X_4
C	X_5	D

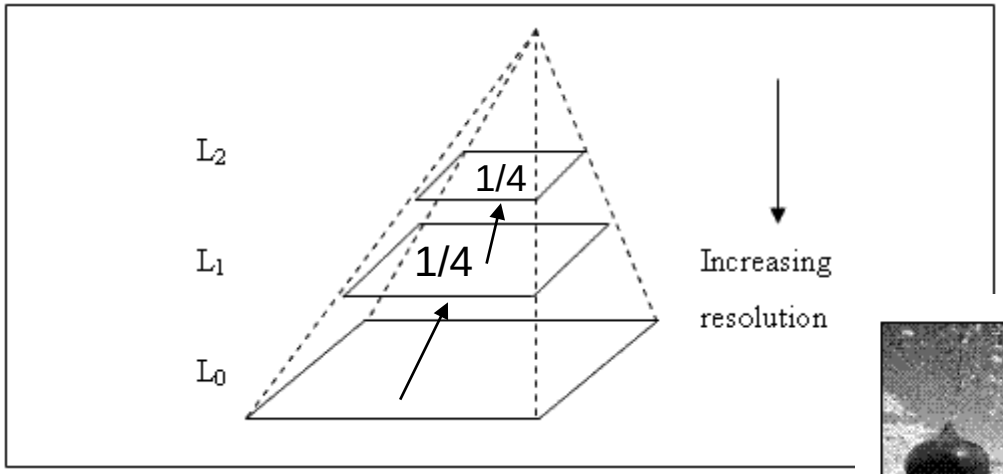
$$X_1 = (\mathrm{A} + \mathrm{B})/2$$

$$X_2 = (\mathrm{A} + \mathrm{C})/2$$

$$X_3 = (\mathrm{A} + \mathrm{B} + \mathrm{C} + \mathrm{D})/4$$

Motion Estimation

Hierarchical Estimation of the motion vector field

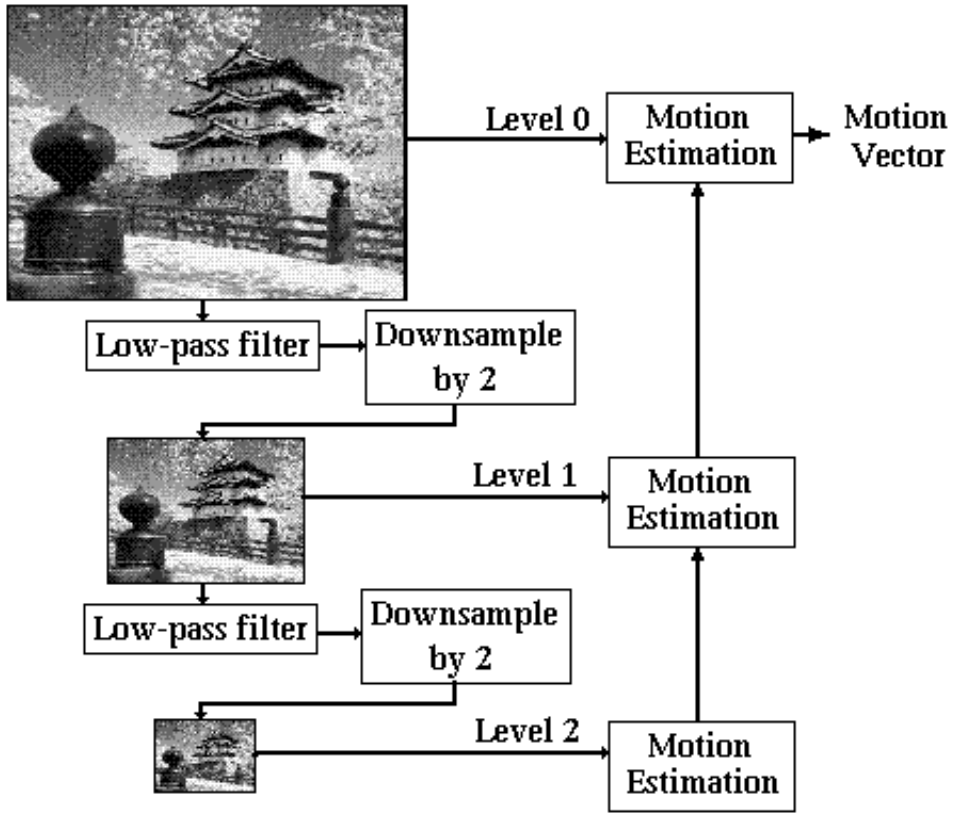


Common Algorithms

A Comparion of Block-Matching Motion Estimation Algoritms

Dos técnicas de reducción de resolución (Downsampling):

- Subsampling
- Mean Intensity



Matching criteria

- MPEG-1 propone para saber cómo de parecidos son dos macrobloques a y b, las siguientes medidas:

  1. **El error cuadrático medio:**

$$\frac{1}{16 \times 16} \sum_{i=1}^{16} \sum_{j=1}^{16} (a_{ij} - b_{ij})^2$$

  2. **El error absoluto medio:**

$$\frac{1}{16 \times 16} \sum_{i=1}^{16} \sum_{j=1}^{16} |a_{ij} - b_{ij}|$$

- Sin embargo, debe tenerse en cuenta que esto es únicamente usado por el compresor y éste no está estandarizado. Por tanto, cualquier otra medida (como la varianza o la entropía del macrobloque residuo) podría se utilizada.

Motion Vectors used for Motion Compensation

Residual Error

No motion compensation

With motion compensation