Video

Agenda

- Flujo óptico
- Compresión de video
- Background subtraction

Video

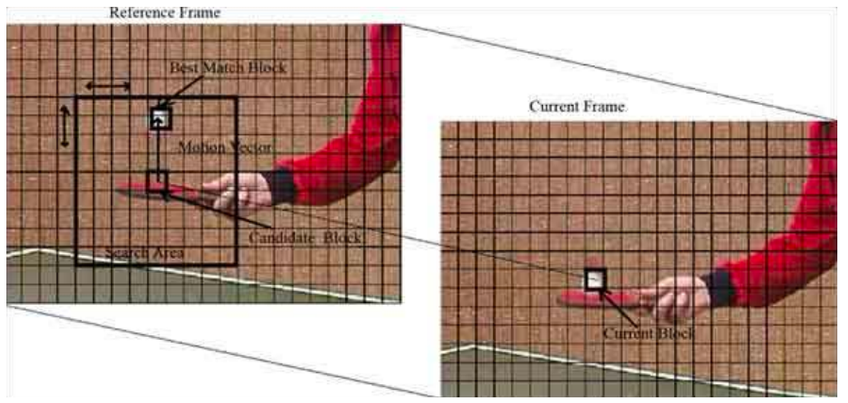
- Secuencia d
- Muestreo te
  - Ancho de b
  - Redundanc
  - **Movimien**

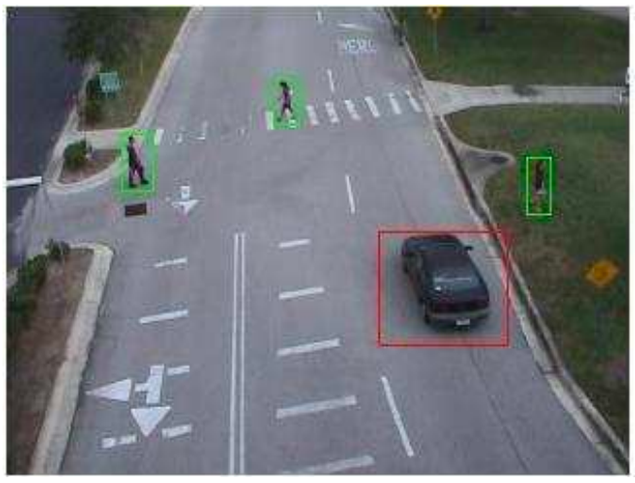
Diferencia entre dos cuadros consecutivos

Vectores de movimiento entre cuadros

Estimación de movimiento

- Una de las aplicaciones más estudiadas en computer vision.
- Aplicaciones:
  - Registrado (alineación) de imágenes
  - Estabilización de imágenes en cámaras (IS)
  - Compresión de video
  - Detección de objetos
  - *Structure from motion*



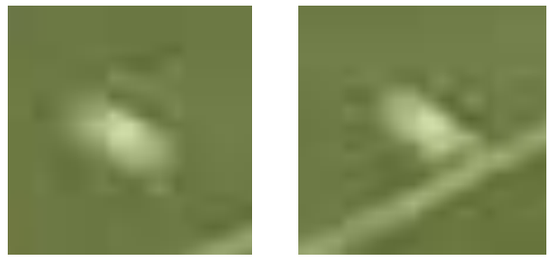


Flujo óptico

Estimación de movimiento

- Movimiento aparente de los pixeles/objetos en una secuencia de imágenes

- Se **asume** que localmente la intensidad se conserva
  - *Brightness constancy constrain* $I_1(\mathbf{x}+\mathbf{u}) = I_0(\mathbf{x})$
  - Buscamos el desplazamiento $\mathbf{u}=(u,v)$



- El error cometido es

$$E = \sum_i d(I_1(\mathbf{x}+\mathbf{u}), I_0(\mathbf{x}))$$

con alguna métrica d.

Estimación de movimiento

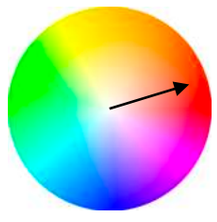
$$E = \sum_i d(I_1(\mathbf{x} + \mathbf{u}), I_0(\mathbf{x}))$$

- Algunas opciones para d
  - Distancia euclídea (LSE, SSD), l_2: $E_{\text{SSD}} = \sum_i (I_1(\mathbf{x} + \mathbf{u}) - I_0(\mathbf{x}))^2$
  - Sum absolute differences (SAD), l1: $E_{\text{SAD}} = \sum_i |I_1(\mathbf{x} + \mathbf{u}) - I_0(\mathbf{x})|$
  - *Median absolute differences* (MAD): $E_{\text{MAD}} = \text{mediana}_i(|I_1(\mathbf{x} + \mathbf{u}) - I_0(\mathbf{x})|)$
  - Distancias robustas: $\rho_{GM}(x) = \dfrac{x^2}{1 + \frac{x^2}{a^2}}$
  - Con pesos: $E_{\text{WSSD}} = \sum_i w_0(\mathbf{x}) w_1(\mathbf{x} + \mathbf{u}) (I_1(\mathbf{x} + \mathbf{u}) - I_0(\mathbf{x}))^2$
  - Correlación: $E_{\text{NCC}} = \sum_i I_1(\mathbf{x} + \mathbf{u}) I_0(\mathbf{x})$

Estimación de movimiento

- También se **asume** que las variaciones locales son suaves
  - Pixeles vecinos tiene un desplazamiento similar

$$\mathbf{u}_p = \mathbf{u}_n \quad n \in G(p)$$



código de color para el vector de movimiento

Flujo óptico

- *Brightness constancy constrain* $I(x+dx, y+dy, t+dt) = I(x,y,t)$
- Con la siguiente aproximación (Taylor)

$$I(x+dx, y+dy, t+1) \approx I(x,y,t) + dx\frac{\partial}{\partial x}I(x,y,t) + dy\frac{\partial}{\partial y}I(x,y,t) + dt\frac{\partial}{\partial t}I(x,y,t)$$

tenemos que

$$dx\frac{\partial}{\partial x}I(x,y,t) + dy\frac{\partial}{\partial y}I(x,y,t) = -dt\frac{\partial}{\partial t}I(x,y,t)$$

y llamando

$$u = dx, I_x = \frac{\partial}{\partial x}I(x,y,t), v = dy, I_y = \frac{\partial}{\partial y}I(x,y,t), dt = 1, I_t = \frac{\partial}{\partial t}I(x,y,t)$$

llegamos a la *ecuación de restricción de flujo óptico*

$$uI_x + vI_y = -I_t$$

- Se **asumió** que: se conserva la intensidad (con errores gaussianos), el flujo es pequeño y suave, la imagen es diferenciable y Taylor es una buena aproximación.

Flujo óptico de Horn y Schunck

- Introduce una restricción de **suavidad** del FO sobre las variaciones de $\mathbf{u}(x, y) = (u, v)$

$$||\nabla u||^2 + ||\nabla v||^2$$

- Plantea la minimización de un funcional

$$E = \iint (I_x u + I_y v + I_t)^2 + \alpha^2 (||\nabla u||^2 + ||\nabla v||^2) \, \mathrm{d}x\mathrm{d}y$$

- Define un sistema de ecuaciones con (u, v) incógnitas

$$(I_x u + I_y v + I_t) I_x + \alpha^2 \Delta u = 0$$

$$(I_x u + I_y v + I_t) I_y + \alpha^2 \Delta v = 0$$

- Usando una aproximación discreta del Laplaciano permite escribir un sistema matricial en *cada pixel* $\Delta u = \kappa(u - \bar{u})$

$$\begin{bmatrix} I_x^2 + \lambda & I_x I_y \\ I_x I_y & I_y^2 + \lambda \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \lambda \bar{u} - I_t I_x \\ \lambda \bar{v} - I_t I_y \end{bmatrix}$$

$$A\mathbf{u} = b$$

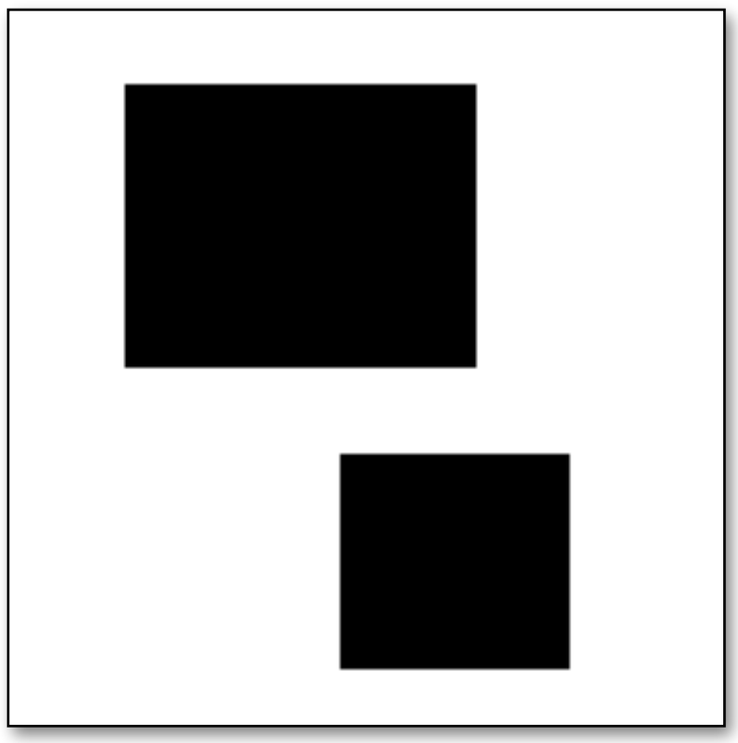
Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. Artificial intelligence, 17(1-3), 185-203.

Flujo óptico de Horn y Schunck

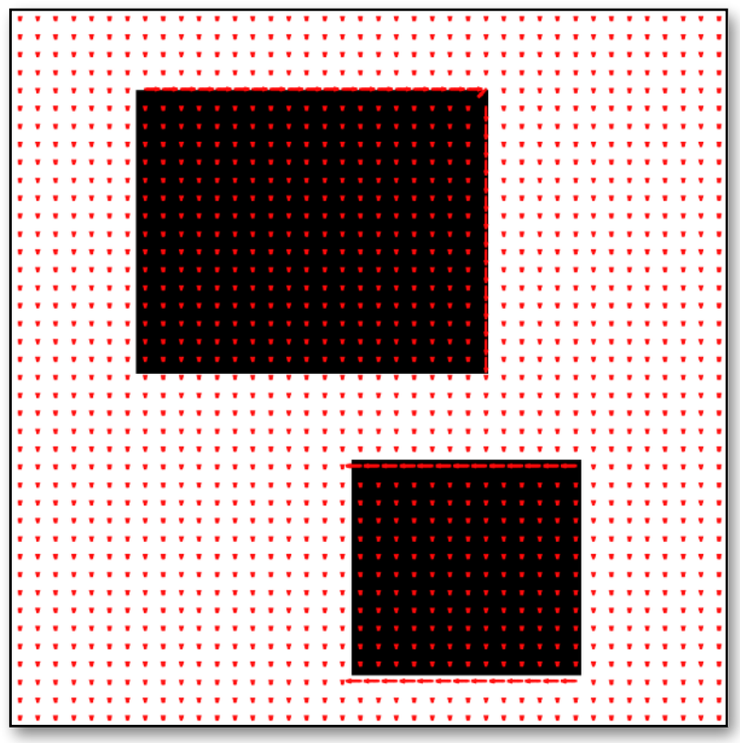
- El cálculo depende del vecindario para suavizar la solución (mediante $(\bar{u}, \bar{v})$)
  - Se resuelve de forma iterativa, iniciando con $\mathbf{u}^{(0)} = \mathbf{0}$

$$u^{(k+1)} = \bar{u}^{(k)} - I_x \frac{I_x \bar{u}^{(k)} + I_y \bar{v}^{(k)} + I_t}{\lambda + I_x^2 + I_y^2}$$

$$v^{(k+1)} = \bar{v}^{(k)} - I_y \frac{I_x \bar{u}^{(k)} + I_y \bar{v}^{(k)} + I_t}{\lambda + I_x^2 + I_y^2}$$



Secuencia de dos imágenes



Iteraciones 1, 10 100 y 500 de Horn-Schunck

Flujo óptico de Lucas y Kanade

- Plantea el problema de FO dividiendo la imagen en *patches* y considerando el FO constante cada *patch*
  - En cada patch vale

$$\begin{aligned} I_x(\mathbf{x}_1)u + I_y(\mathbf{x}_1)v &= -I_t(\mathbf{x}_1) \\ I_x(\mathbf{x}_2)u + I_y(\mathbf{x}_2)v &= -I_t(\mathbf{x}_2) \\ &\vdots \\ I_x(\mathbf{x}_n)u + I_y(\mathbf{x}_n)v &= -I_t(\mathbf{x}_n) \end{aligned}$$

  - Podemos escribirlo matricialmente como $A\mathbf{u} = b$

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(\mathbf{x}_1) \\ -I_t(\mathbf{x}_2) \\ \vdots \\ -I_t(\mathbf{x}_n) \end{bmatrix}$$

- Puede resolverse usando la pseudo inversa

$$\mathbf{u} = (A^T A)^{-1} A^T b$$

- Puede incluir una matriz W de *pesos*

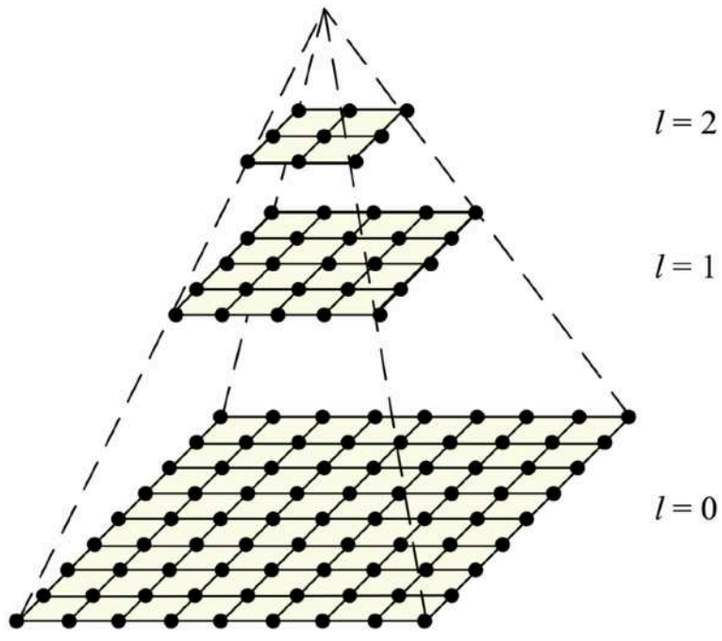
$$\mathbf{u} = (A^T W A)^{-1} A^T W b$$

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision.

Flujo óptico de Lucas y Kanade

- Combinado con algún selector de puntos de interés permite seguirlos.

- Seguimiento de movimientos grandes creando una pirámide y buscando primero en niveles gruesos
  - Submuestreo: $I_k^{(l)}(\mathbf{x}_j) \leftarrow \tilde{I}_k^{(l-1)}(2\mathbf{x}_j)$
  - Búsqueda: $\mathbf{u}^{(l)}$
  - Predicción: $\hat{\mathbf{u}}^{(l-1)} \leftarrow 2\mathbf{u}^{(l)}$



Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision.

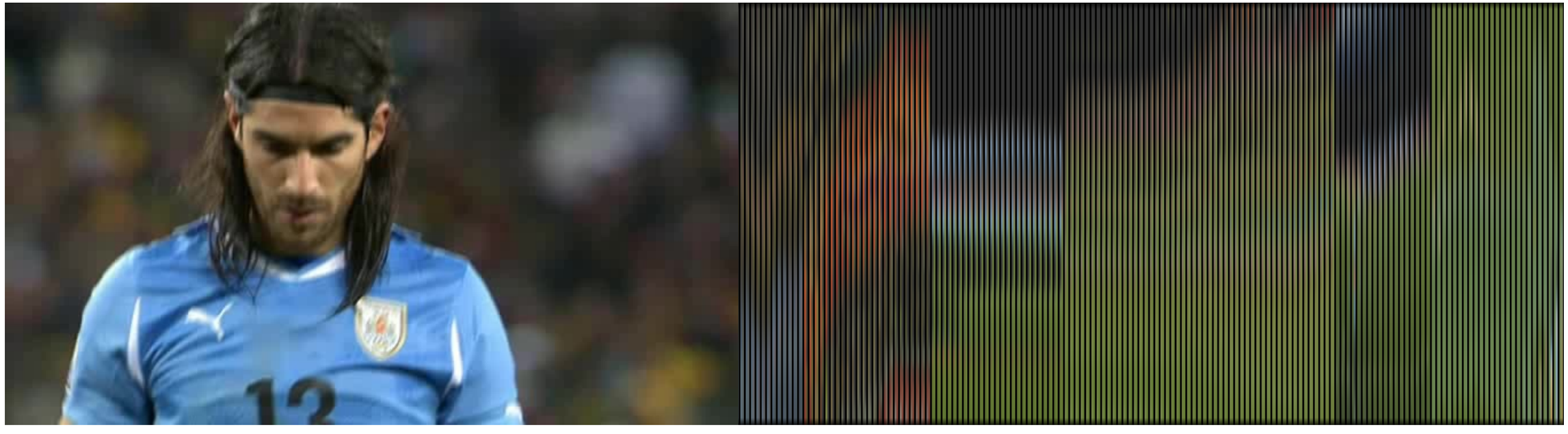
40 años de flujo óptico

- Uno de los temas de mayor investigación en video
- Variantes: robusto, 3D, múltiples cuadros, alto/bajo frame-rate, gradient-based, iterativo, grueso-fino, modelos de movimiento, en capas, probabilísticos, ...
- Benchmarks de comparación
  - Middlebury Computer Vision benchmark (http://vision.middlebury.edu/flow/)
  - MPI Sintel benchmark (http://sintel.is.tue.mpg.de/)
  - KITTI Flow 2015 evaluation (http://www.cvlibs.net/datasets/kitti/)
  - Heidelberg HD1K Flow benchmark (http://hci-benchmark.org/flow/)
  - Robust Vision Challenge 2018 (http://www.robustvision.net/)

	Stereo	MVS	Flow	Depth	Semantic	Instance
Middlebury	✘	✘	✘			
KITTI	✘		✘	✘	✘	✘
MPI Sintel			✘			
ETH3D	✘	✘				
HD1K			✘			
ScanNet				✘	✘	✘
Cityscapes					✘	✘
WildDash					✘	✘

Codificación de video

Codificación de video



765 cuadros individuales: 288.543.232 bytes (288.064.673 bytes *zipeados*)

Video MPEG-4: 3.139.697 bytes

Video MPEG-4: 887.947 bytes

Codificación de video

- Reescribir
- Codificación **sin pérdidas** (*lossless*) o **con pérdidas** (*lossy*)
  - ¿Qué descartar? Sistema visual **humano**
  - ¿Cómo hacerlo? Descripción del **decodificador**.
- Adaptación a un cierto **estándar** (ISO/IEC: Moving Picture Experts Group (MPEG1, MPEG2, MPEG4, ...), ITU: H.26x, VPx (google), ...)
- Redundancia **espacial** y **temporal**
- Movimiento de objetos/cámara
- Proyección 3D a 2D: movimiento aparente
- Estimación y compensación de movimiento
- Predicción y codificación de diferencias

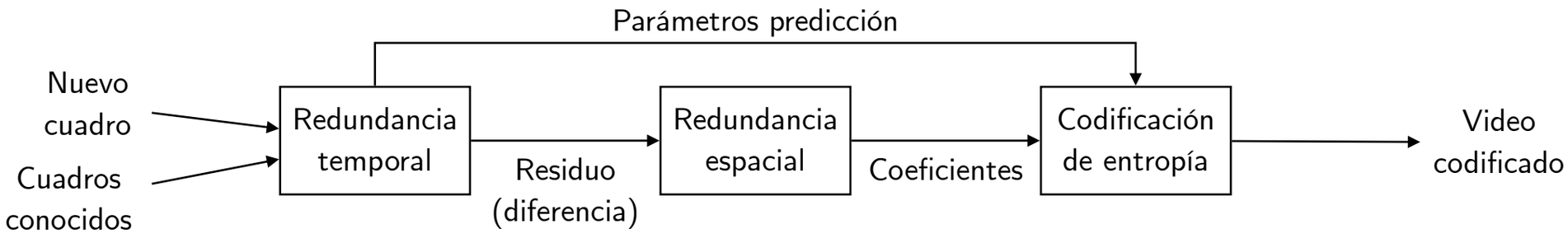
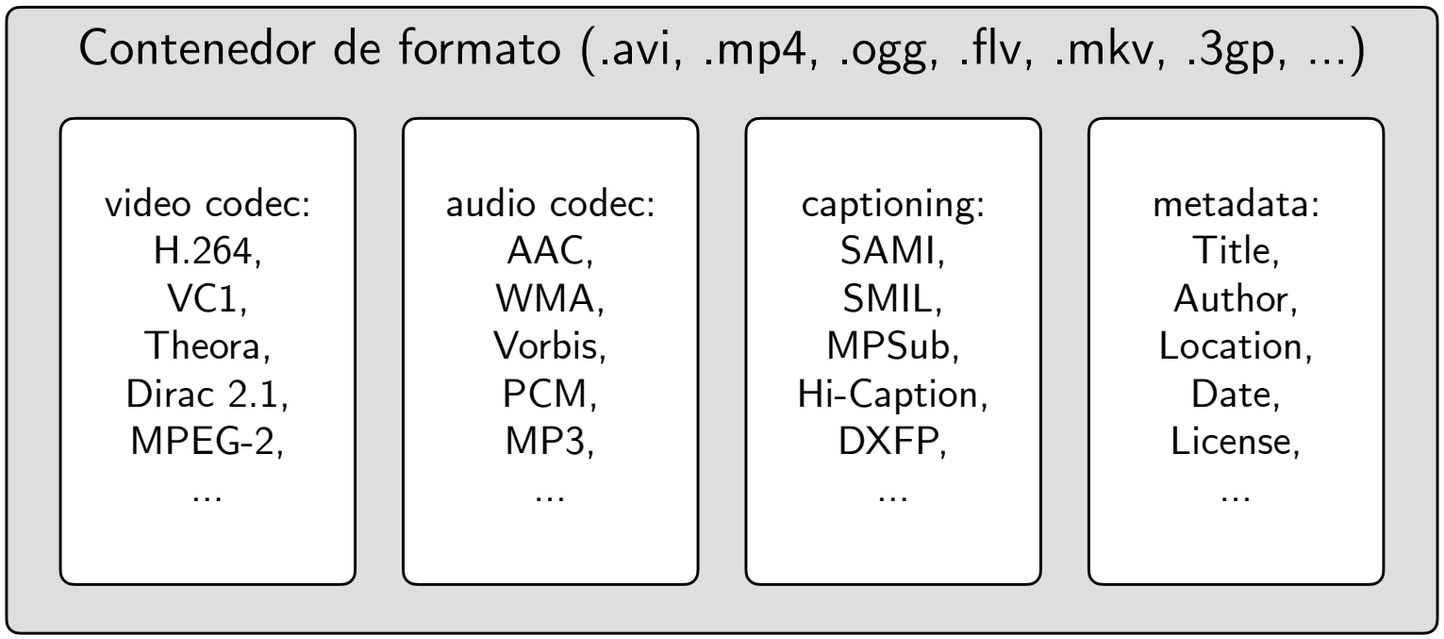


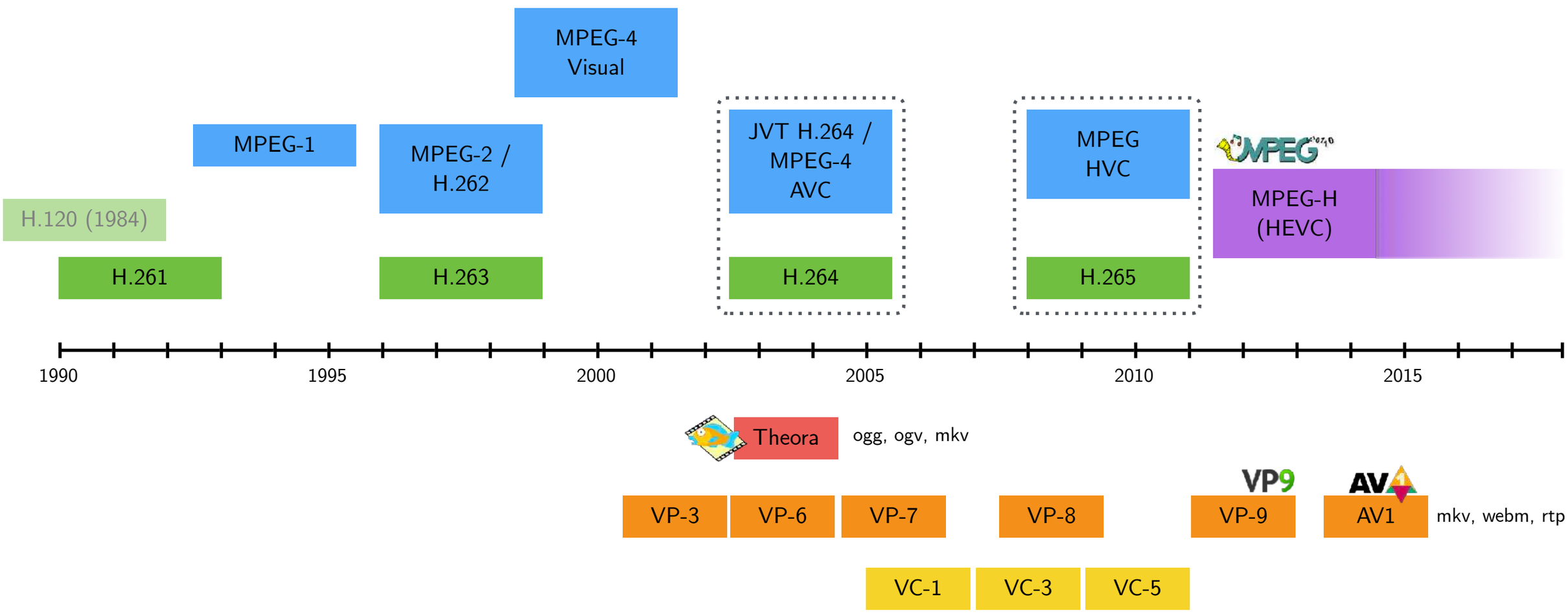
Diagrama conceptual de un codificador de video.

Archivos (contenedores) de video



AVI: Windows Professional
MOV: Mac
MKV: Open Source
MPEG or MPG: MPEG group
FLV: Flash video
MP4: MPEG group
MTS or TS: Transport Stream
MXF: Material Exchange Format
WMV: Windows Consumer
VOB: DVDs
R3D: Redcode RAW
DNG: Digital Negative, by Adobe
TIFF: Tagged Image File Format, by Adobe
TARGA: non-proprietary
...

(algunos) estándares de codificación de video



... y muchos más ...

Familias y *vecindarios* de estándares de codificación de video

MPEG-1 part-2 [1]	• Standardised in 1993 • Developed for video and audio storage on CD-ROMs • Has support for progressive video • Supports YUV 4:2:0 at common intermediate format (352x288) resolution • Group of pictures include I-picture, P-picture and B-picture • Motion vectors are coded losslessly
MPEG-2 part-2 [2]	• Standardised in 1995 • Supports video on DVDs, standard definition TVs and high definition TVs • Can handle YUV 4:2:0 and YUV 4:2:2 formats • Supports interlaced and progressively scanned pictures • Introduction of *profiles* and *levels* to define the various capabilities of the standard as sub-sets. For example, MPEG-2 *Main profile / high level* is suitable for high definition television (HDTV) • Scalable extensions which permit the division of a continuous video signal into two or more coded bit streams representing the video at different resolutions, picture quality (i.e. SNR), or picture rates • Other features include data partitioning, non-linear quantization, VLC tables and improved mismatch control
MPEG-4 part-2 (Visual) [3]	• Standardised in 1999 • Supports video on low-bit rate multimedia applications on mobile platforms and the Internet. • Supports object-based or content-based coding where a video scene is coded as a set of foreground and background objects. • Supports coding of synthetic video and audio including animation • Shares subset with H.263 ("short header mode")
MPEG-4 Part 10 [4] (Advanced video coding)	• Standardised in 2003 • Co-published as H.264 AVC, see Table 2 for details

H.261 [5]	• Standardised in 1988 [6] • Developed for video conferencing over ISDN • Support for CIF and QCIF resolutions in YUV 4:2:0 format • Uses block-based hybrid coding with integer pixel motion compensation.
H.262 [2]	• Standardised as MPEG-2 part-2 (Video) in 1995 • See Table 1 for details
H.263 / H.263+ [7]	• H.263 was standardised in 1996. H.263+ was standardised in 1998 • Improved quality compared to H.261 at lower bit rate to enable video conferencing/telephony • Sub-pixel motion vectors up to 1/8th pixel accuracy (H.263+) for improved compression • Shares subset with MPEG-4 part 2
H.264 Advanced Video Coding [4]	• Standardised in 2003 • Supports video on the Internet, computers, mobile and HDTVs • Significantly improved picture quality compared to H.263, at low bit rates but at the cost of increased computational complexity • Improved motion compensation with variable block-size, multiple reference frames and weighted prediction • In-loop deblocking filter to reduce block discontinuities
H.265 / HEVC [8]	Standardised in 2013, HEVC's basic structure is similar to H.264/AVC with improvements such as: • Support for ultra HD video up to 8K resolutions with frame rates up to 120 fps • Greater flexibility in prediction modes and transform block sizes • More sophisticated interpolation and deblocking filters • Features to support parallel processing • More efficient compared to H.264 in terms of bitrate savings for the same picture quality [9,10]

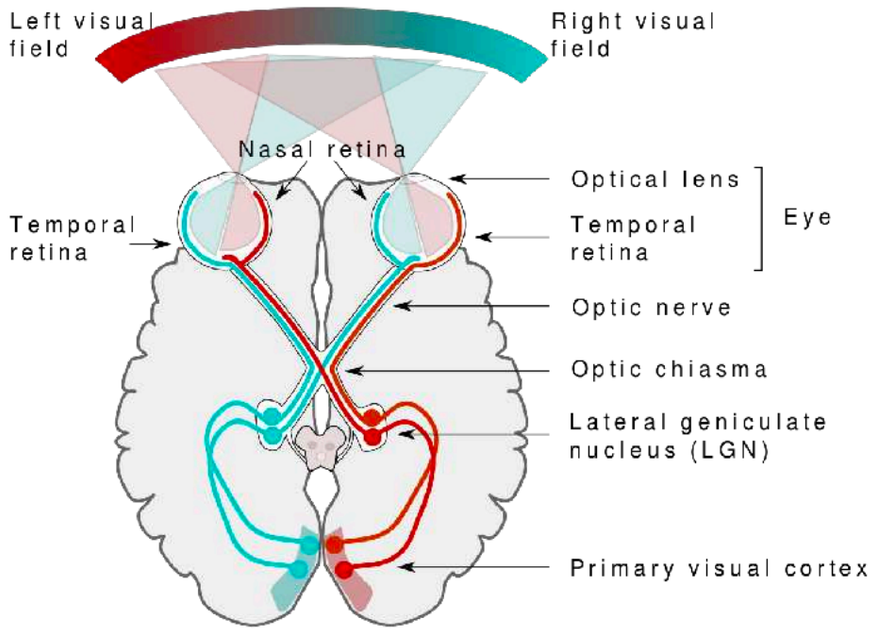
Theora [13]	• Based on On2's VP3.2 format • Support for video on the Internet • Bitstream format was frozen in 2004 • Support for more pixel formats - YUV444, YUV422 and YUV420 • No B-pictures • No global motion compensation • Only half-pixel precision motion compensation • In-loop deblocking filter to reduce block discontinuities
VP8 [14]	• Designed to operate in low bandwidth environment such as web video • Support for Web video format - YUV420, 8-bit colour depth, progressive scan and resolution up to 4K • Flexible reference frames with buffer size limited to three frames • Efficient intra and inter prediction with motion vector accuracy of 1/4th pixel for luma and 1/8th pixel for chroma • Adaptive in-loop deblocking filter to reduce block discontinuities • Features such as data partitioning enable parallel processing

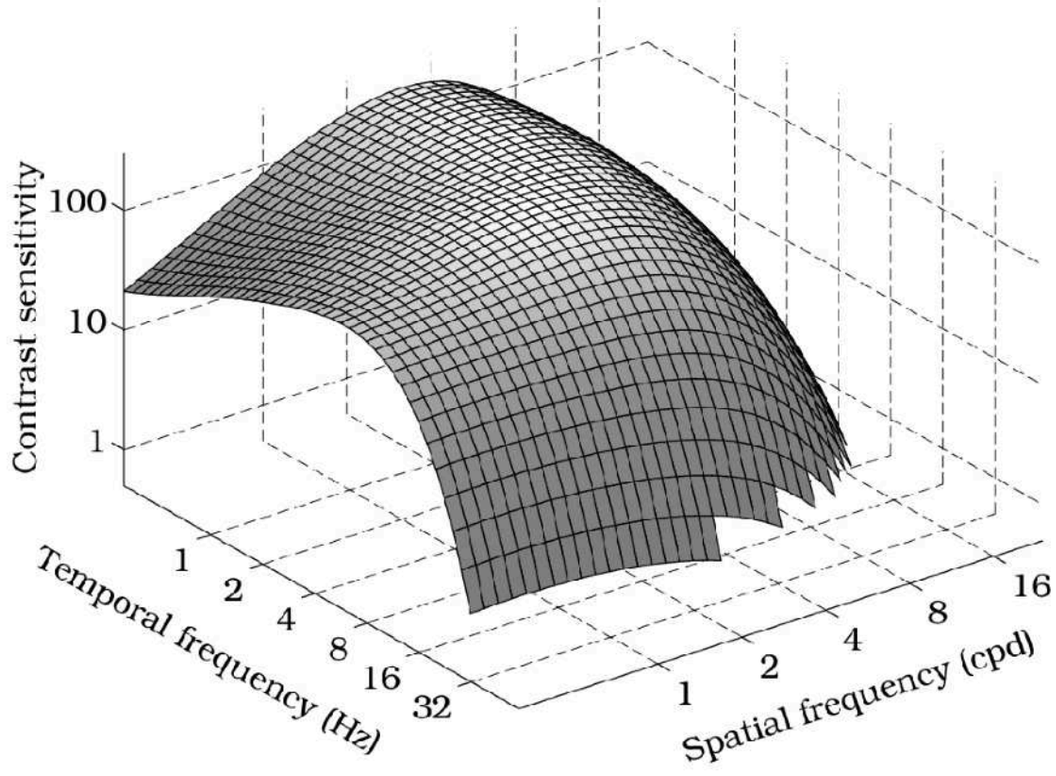
VP9 [15]	Has basic structure similar to VP8 with improvements such as: • Support for resolution up to 8k and frame rate up to 120fps • Use of superblocks (32x32 and 64x64) to exploit high correlation over larger areas in HD video • Enhanced interpolation for motion compensation with 8-tap filters to achieve 1/8th pixel accuracy in motion vectors. • Reference frame scaling for improved bitrate adjustment, where the reference frame can be scaled up or down during inter prediction.
Daala [16]	Daala is being designed to differ from common video compression techniques. Some distinguishing features include: • Lapped transforms instead of block-based DCT • Lifting pre- and post-filtering instead of deblocking filter • Frequency domain intra prediction instead of pixel domain prediction • Time/Frequency Resolution Switching where image blocks can be split apart or merged together in the frequency domain.

Fuente: https://www.vcodex.com/historical-timeline-of-video-coding-standards-and-formats/

Sistema visual humano

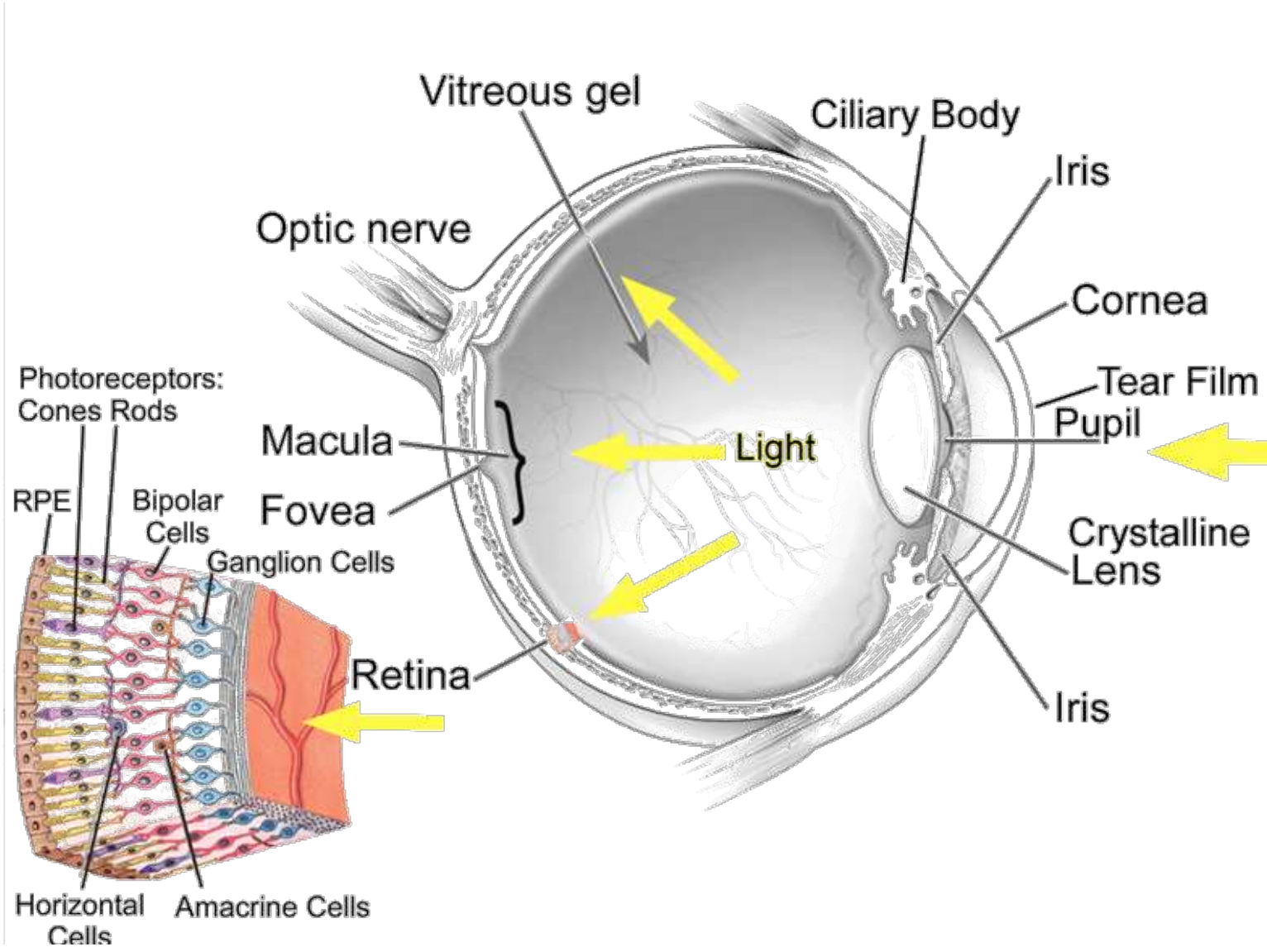
- Diferentes tipos de neuronas (fotorreceptores)
- Diferentes respuestas a:
  - frecuencias espaciales
  - estímulos temporales
- Respuesta al contraste (contrast sensitivity function)
  - umbral de contraste
  - la respuesta depende de la frecuencia del estímulo
  - depende de la distancia al objeto
- Movimiento
  - la frecuencia temporal afecta el estímulo
- Es un *pasabajos* espacio-temporal

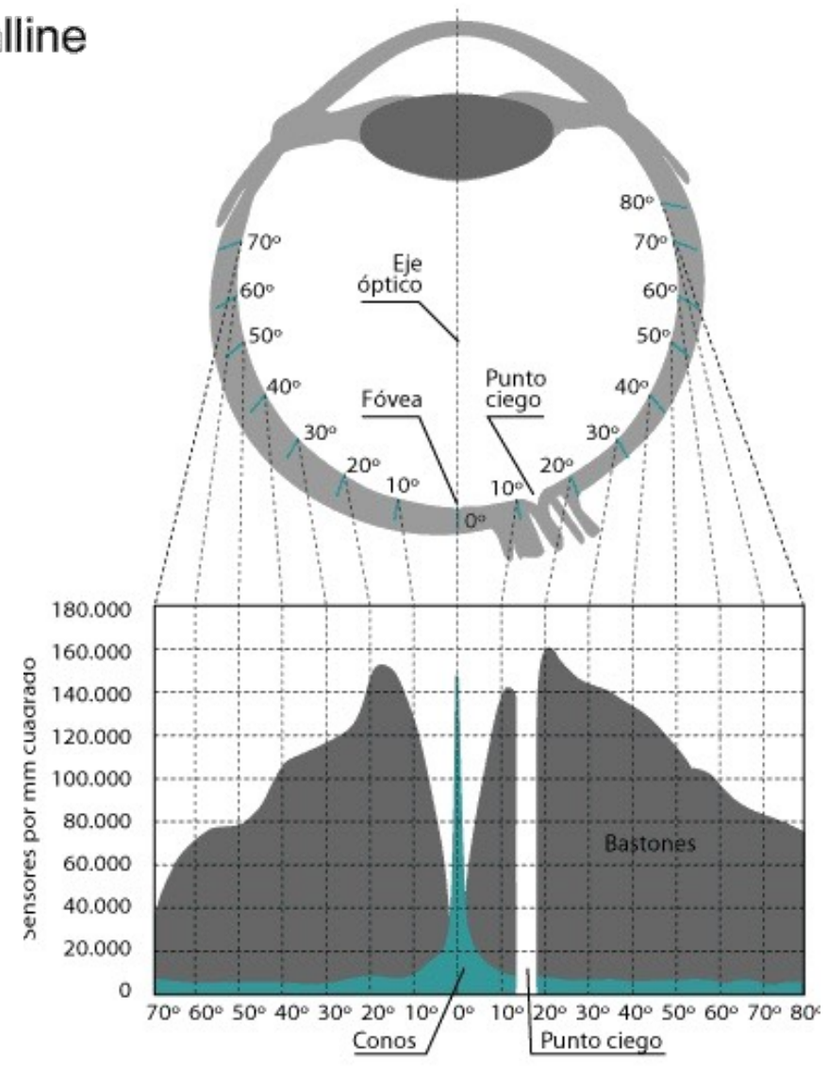




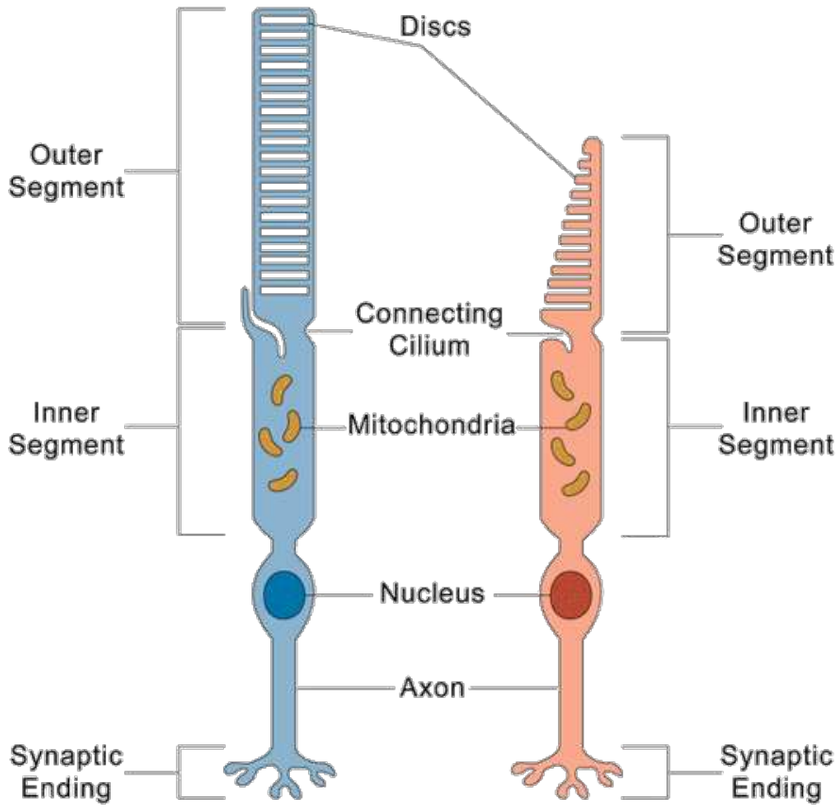
Wandell, B. A. (1995). *Foundations of vision* (Vol. 8). Sunderland, MA: sinauer Associates.

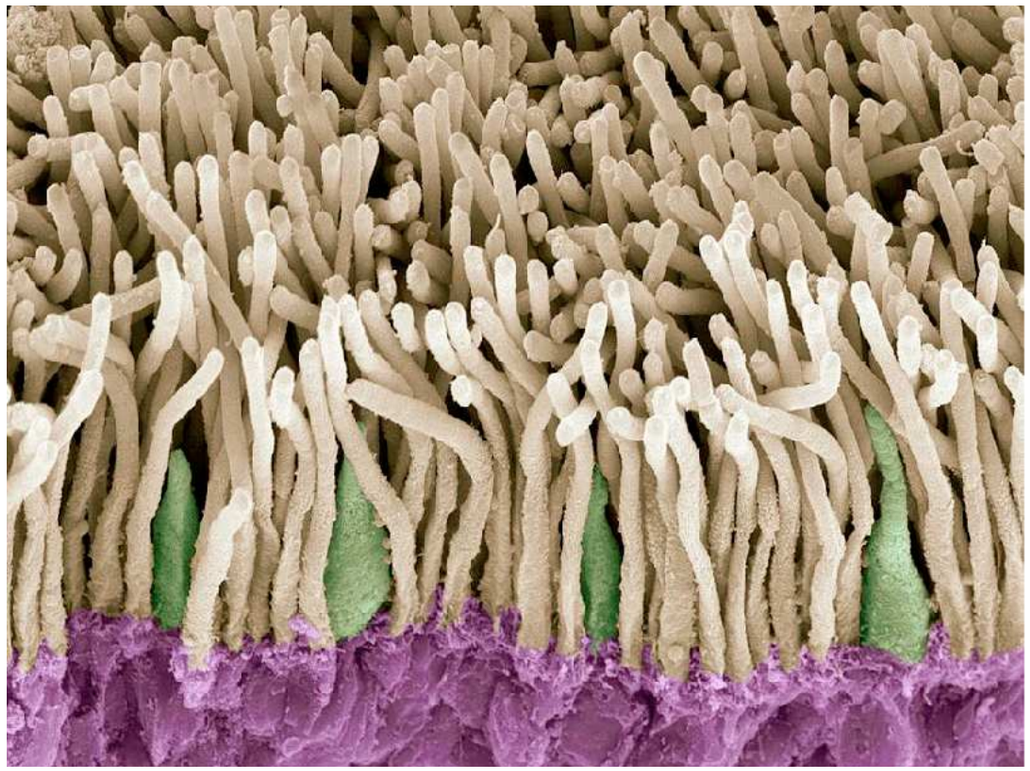
El ojo

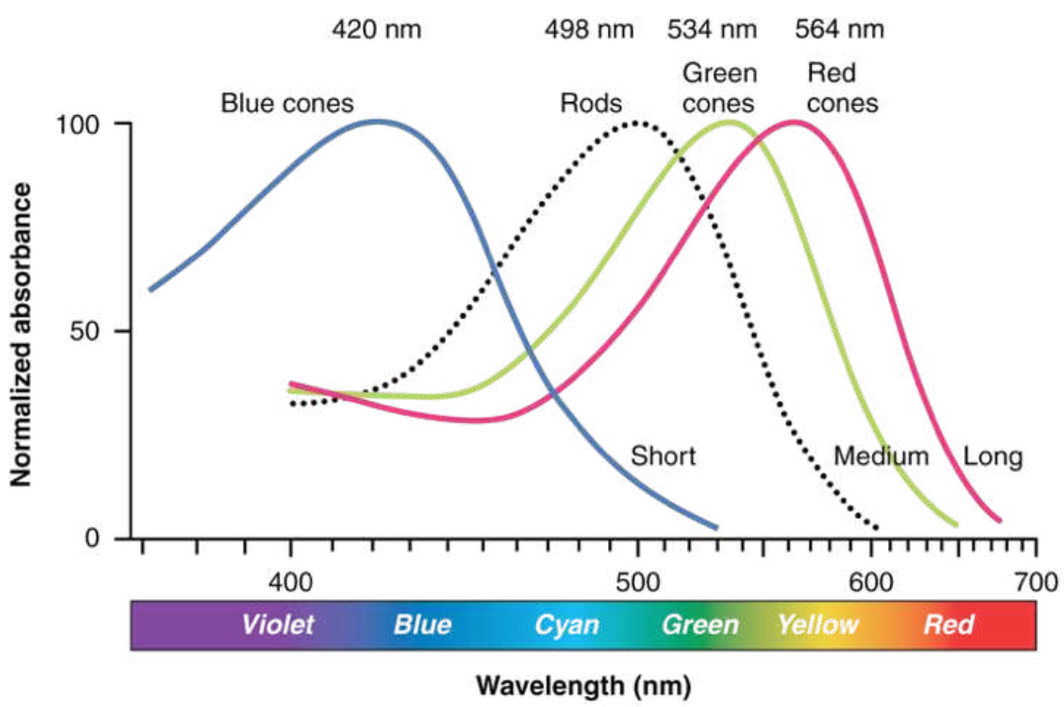

Vitreous gel
Ciliary Body
Iris
Optic nerve
Cornea
Tear Film
Pupil
Macula
Light
Fovea
Crystalline Lens
Photoreceptors: Cones Rods
RPE
Bipolar Cells
Ganglion Cells
Retina
Iris
Horizontal Cells
Amacrine Cells


80°
70°
60°
50°
40°
30°
20°
10°
0°
Eje óptico
Fóvea
Punto ciego
180.000
160.000
140.000
120.000
100.000
80.000
60.000
40.000
20.000
0
Sensores por mm cuadrado
Bastones
70° 60° 50° 40° 30° 20° 10° 0° 10° 20° 30° 40° 50° 60° 70° 80°
Conos
Punto ciego

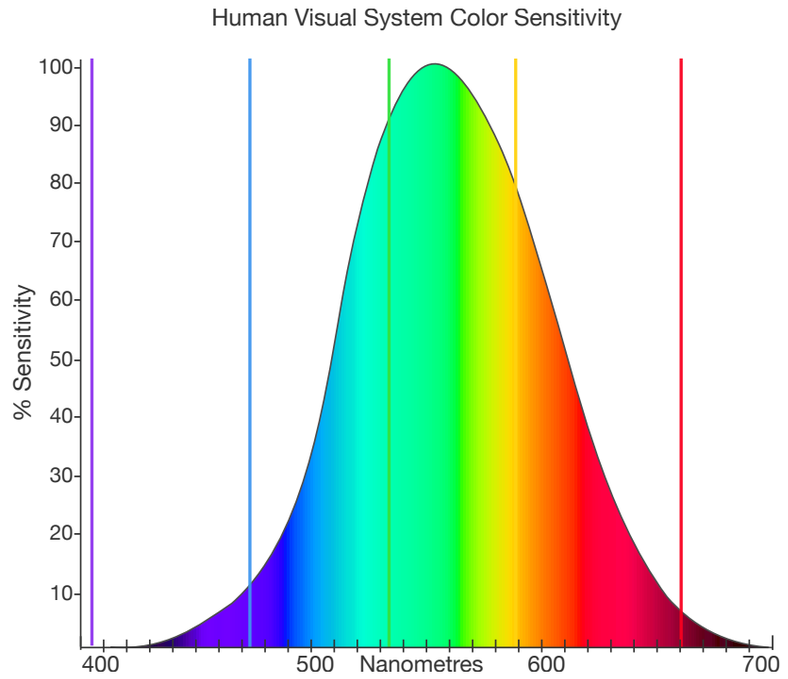
Fotorreceptores







https://upload.wikimedia.org/wikipedia/commons/9/94/1416_Color_Sensitivity.jpg



[Adaptado de https://upload.wikimedia.org/wikipedia/commons/c/c0/Eyesensitivity.svg]

Sistema visual humano

- Resumen
  - Bajas frecuencias tienen más importancia que altas frecuencias
  - Menor resolución espacial para las cromas, contra la luminancia
  - Cromas son menos relevantes para captar movimiento
  - Altas frecuencias temporales (variaciones rápidas) son menos relevantes

Codificación imágenes JPEG

DCT

8×8

Cuantificar

matriz de cuantificación psicovisual Q

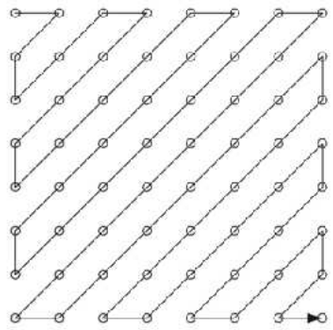
codificación de longitud variable (VLC)

DC: codificación diferencial

Codificación de Entropía

AC: recorrido zig-zag

Codificación de Entropía



codificación sin pérdidas basada en probabilidad de aparición de símbolos

Archivo JPEG

encabezado, datos y metadatos

Discrete Cosine Transform (DCT)

- Secuencia de largo N, $x[n]$:
- DCT

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

$$c[k] = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{2}{N}} & k \neq 0 \end{cases}$$

- IDCT

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

- Propiedades
  - Fourier real, ortogonal y separable
  - Compacta energía
  - Base fija
  - Algoritmos rápidos

Discrete Cosine Transform (DCT)

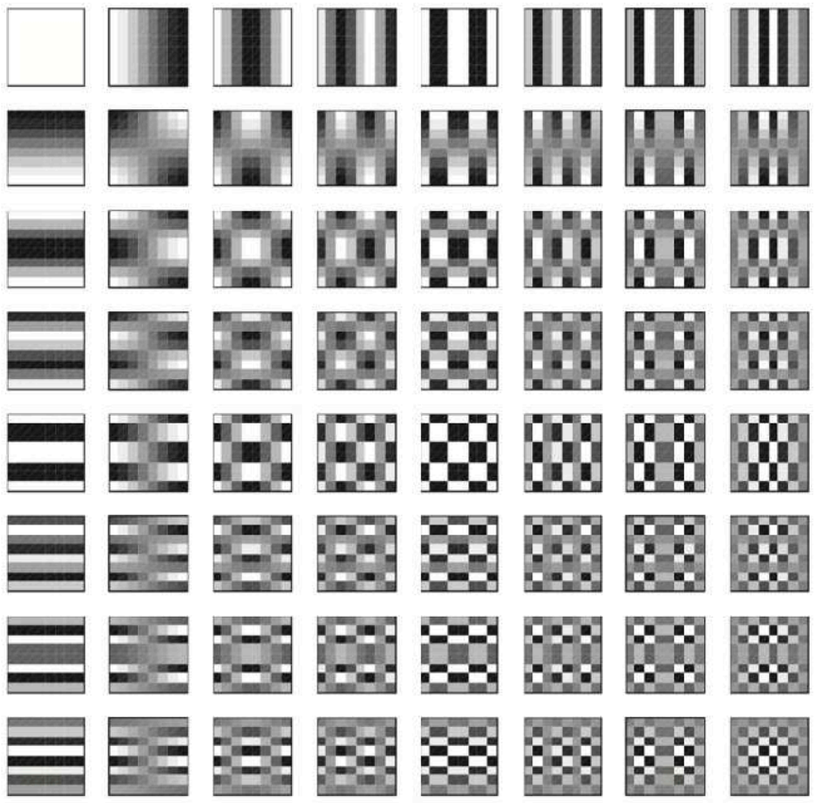
- Una imagen $N \times M$, $i[n,m]$:
- 2D-DCT

$$I[k,j] = c[k]c[j] \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} i[n,m] \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \cos\left[\frac{\pi}{M}\left(m+\frac{1}{2}\right)j\right]$$

- 2D-IDCT

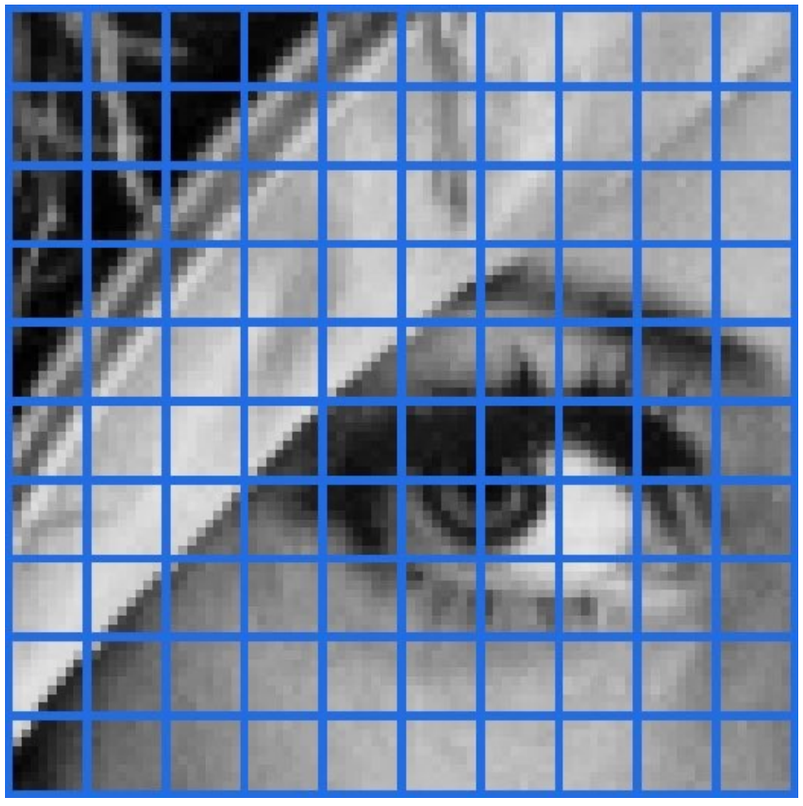
$$i[n,m] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} c[k]c[j]I[k,j] \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \cos\left[\frac{\pi}{M}\left(m+\frac{1}{2}\right)j\right]$$

- Es una base del espacio de bloques imágenes de 8×8 pixeles

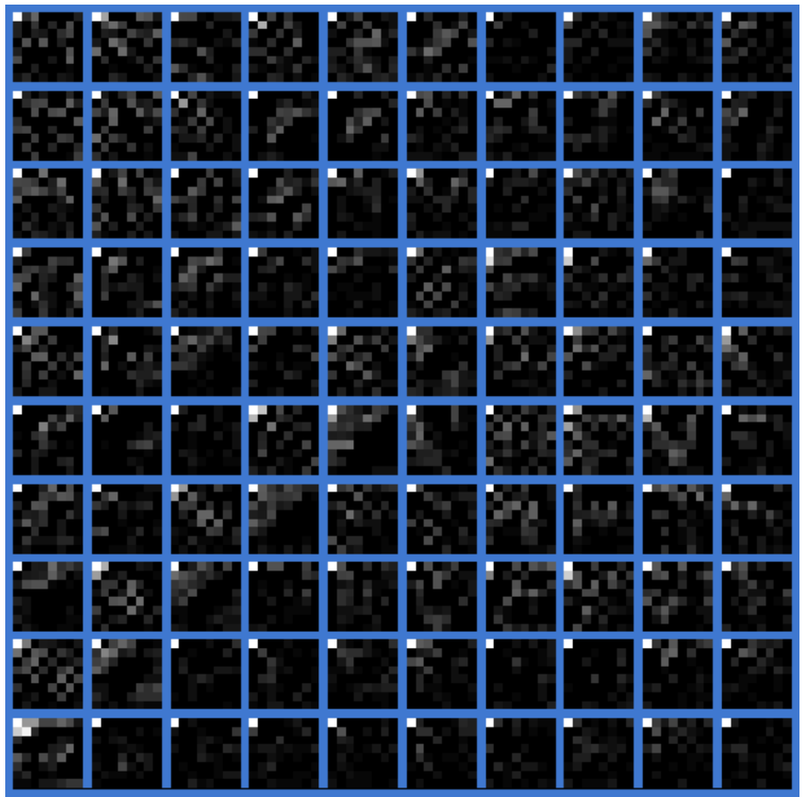


Vectores 8×8 de la base

Discrete Cosine Transform (DCT)



División en bloques de 8×8 pixeles



DCT de cada bloque

Discrete Cosine Transform (DCT)

Original

Reconstrucción con 1 coeficiente

Discrete Cosine Transform (DCT)

Original

Reconstrucción con 3 coeficientes

Discrete Cosine Transform (DCT)

Original

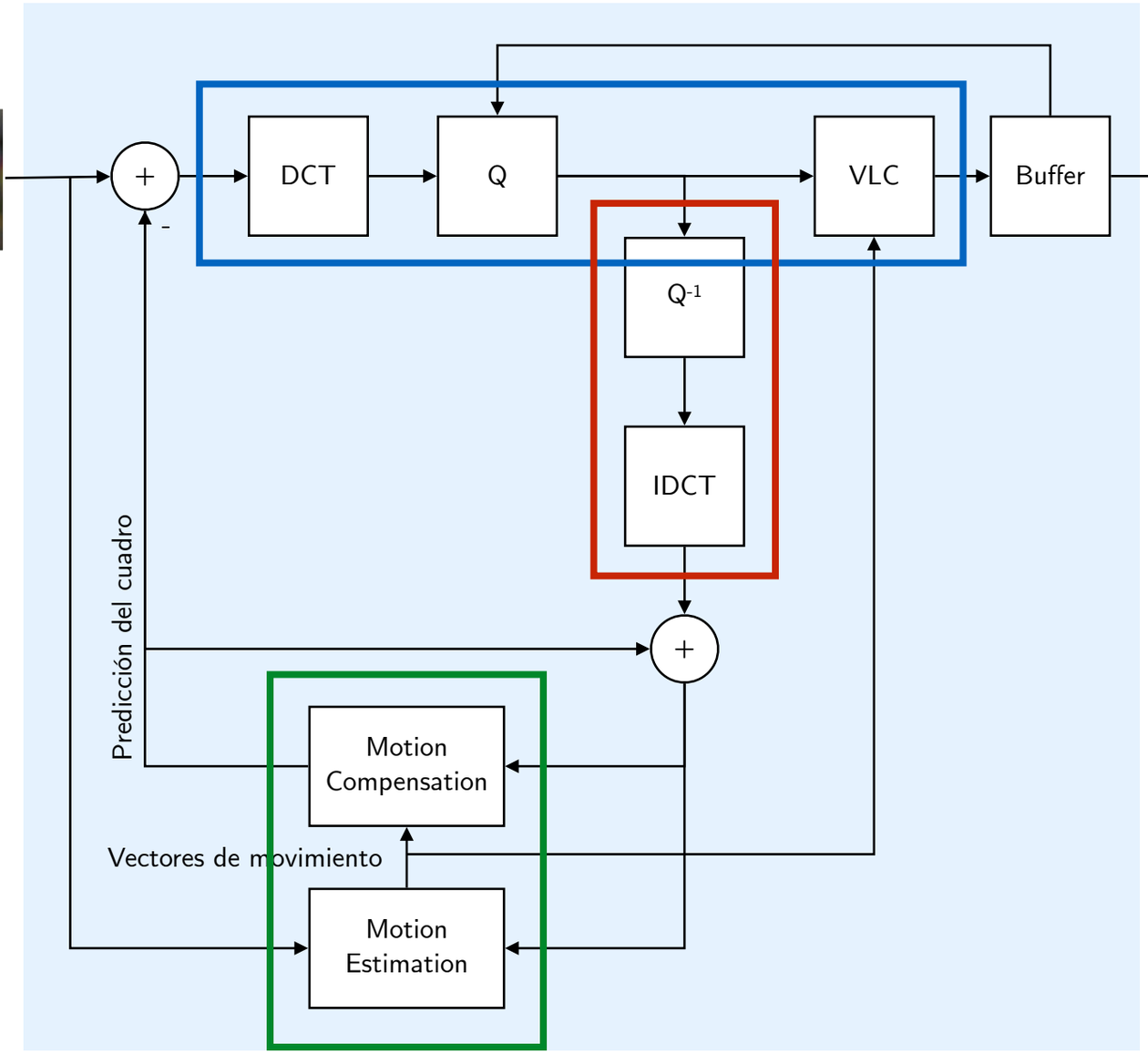
Reconstrucción con 6 coeficientes

Discrete Cosine Transform (DCT)

Original

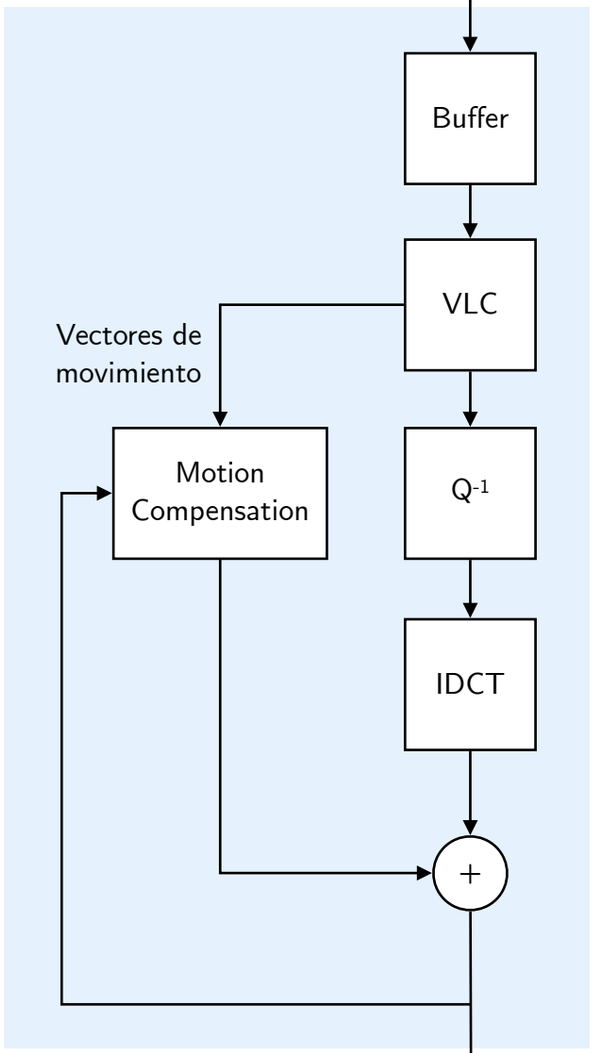
Reconstrucción con 10 coeficientes

Codificador - Decodificador MPEG



Codificador

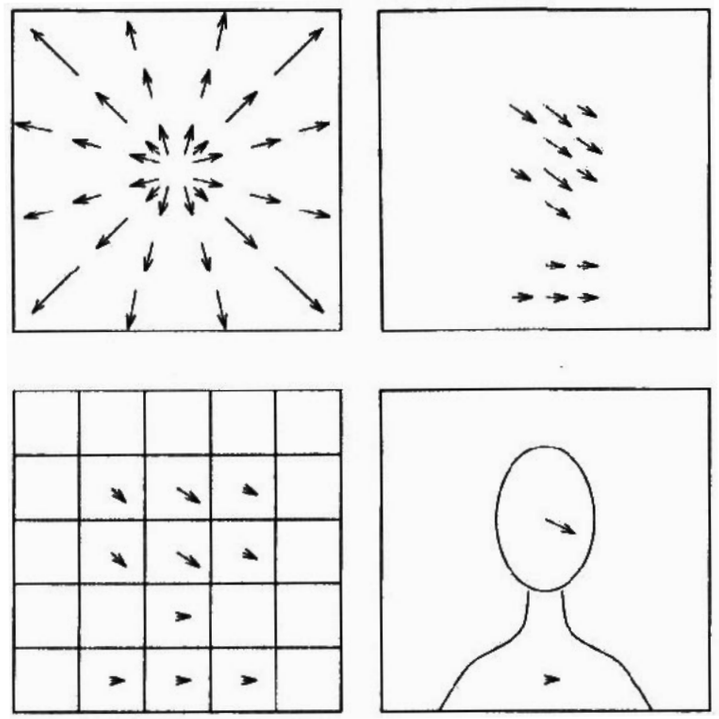
Archivo o *stream* de video

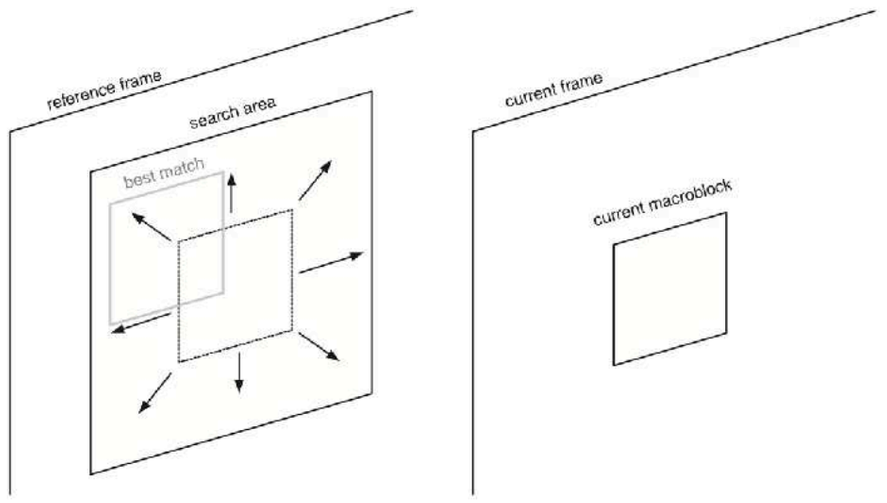


Decodificador

Estimación y compensación de movimiento

- Modelo del movimiento (global, pixel, bloque, regiones).
  - Representado por un conjunto de parámetros.
  - Es una de las etapas con mayor requerimiento de cómputo.
- MPEG-1/2 utilizan bloques (*macrobloques*).
  - Se asumen traslaciones (rotaciones pequeñas).
  - Búsqueda exhaustiva o rápida (iterativa).
  - Impreciso para representar bordes de objetos.
- MPEG-4/H.264/VP9 define regiones (objetos)
  - Necesaria una segmentación de la imagen.
  - Agrupar los objetos con el mismo movimiento.
  - Mejor estimación vs. mayor complejidad.

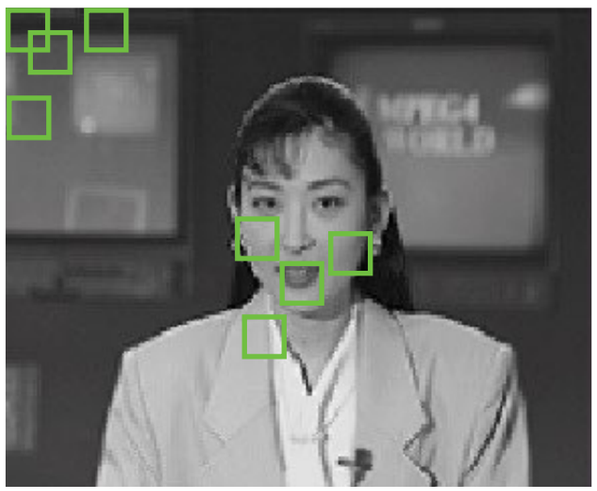




Estimación y compensación de movimiento

- Se crea una predicción del cuadro a partir de cuadros conocidos.
  - Se busca minimizar el error de predicción (usualmente con SAD)

$$e(\mathbf{x}) = d(I(\mathbf{x}, t) - \hat{I}(\mathbf{x} + \mathbf{u}_{t,\tau}, \tau))$$

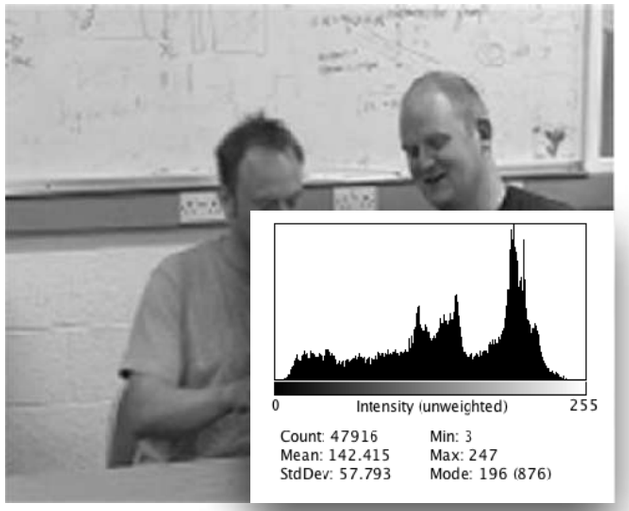


Cuadro t
(conocido)

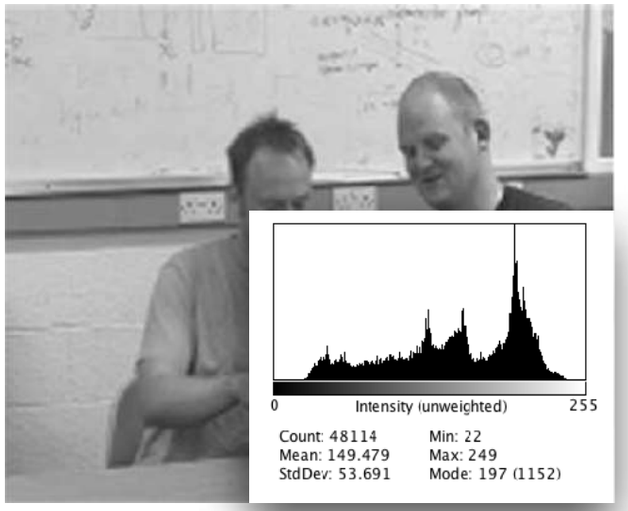
Cuadro t+1
(a codificar)

Cuadro t+1 estimado mediante
compensación de movimiento

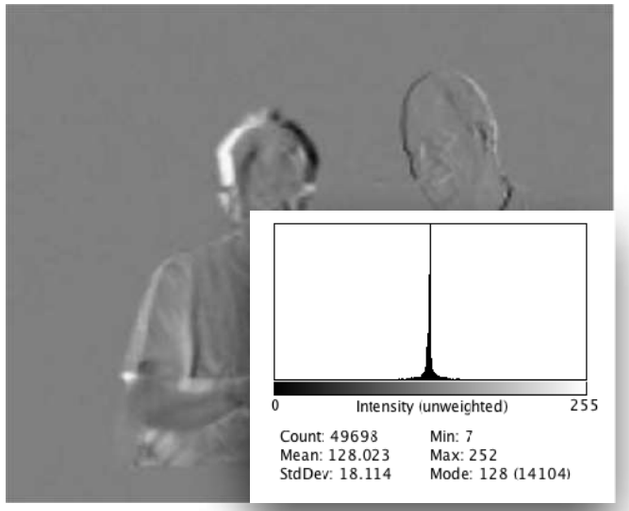
Residuo de la estimación de movimiento



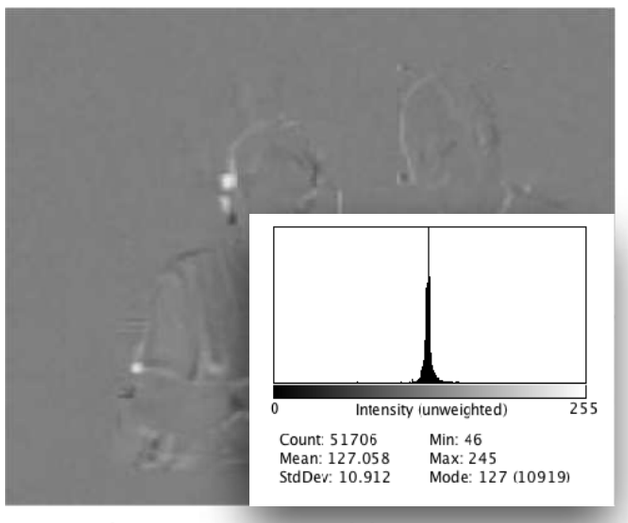
Cuadro t



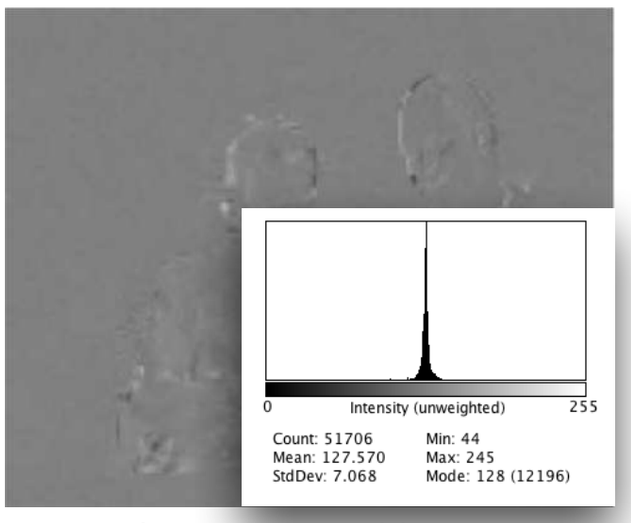
Cuadro t+1



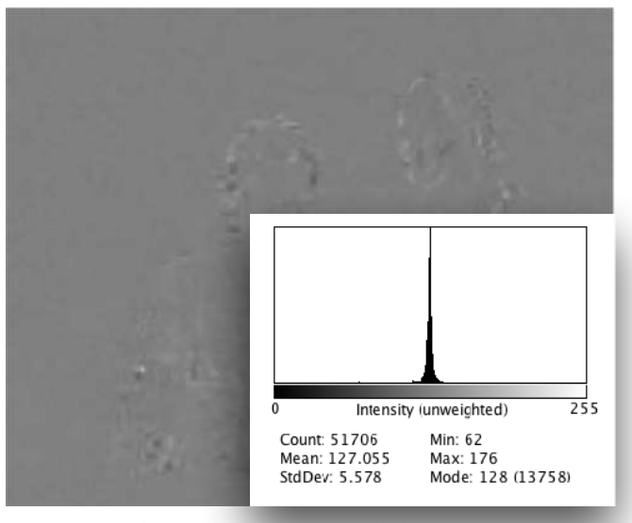
Diferencia



Residuo con macrobloque 16x16



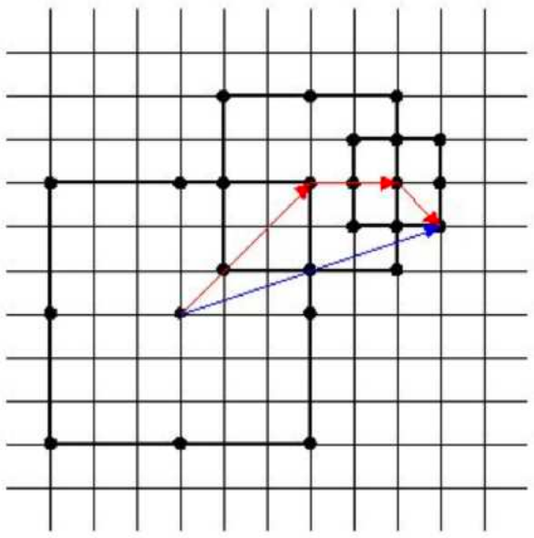
Residuo con macrobloque 8x8



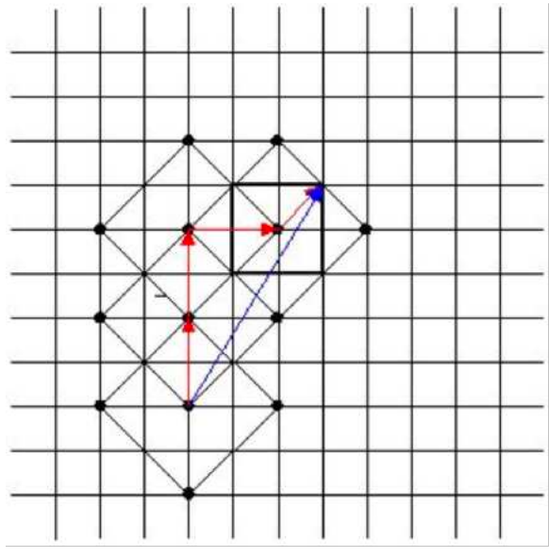
Residuo con macrobloque 4x4

Estrategias de búsqueda de movimiento

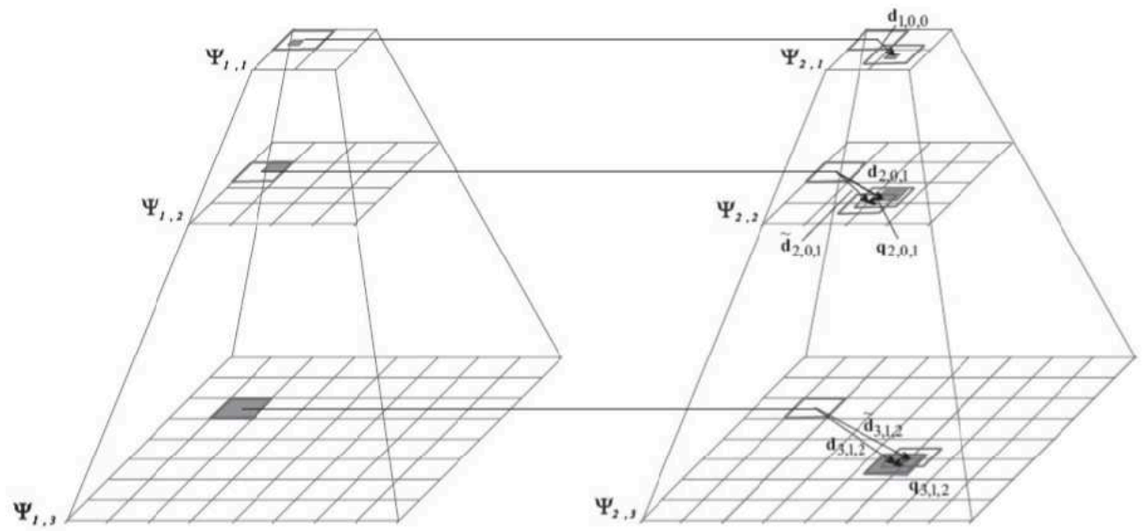
- ¿Dónde buscar?
  - Búsqueda exhaustiva (costosa)
  - Búsquedas rápidas (compromiso)
    - Asumiendo pequeños movimientos de traslación



Búsqueda en tres pasos



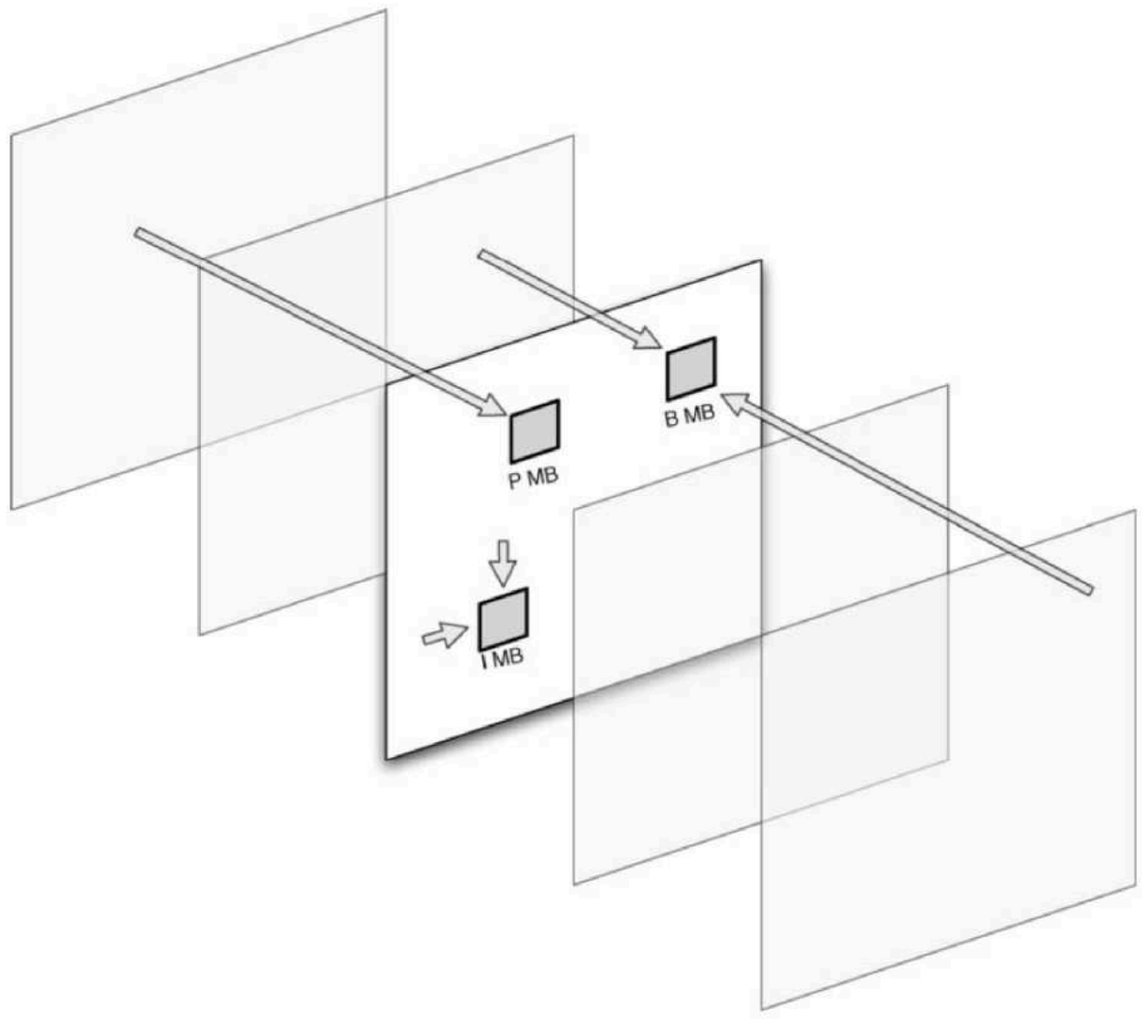
Búsqueda logarítmica



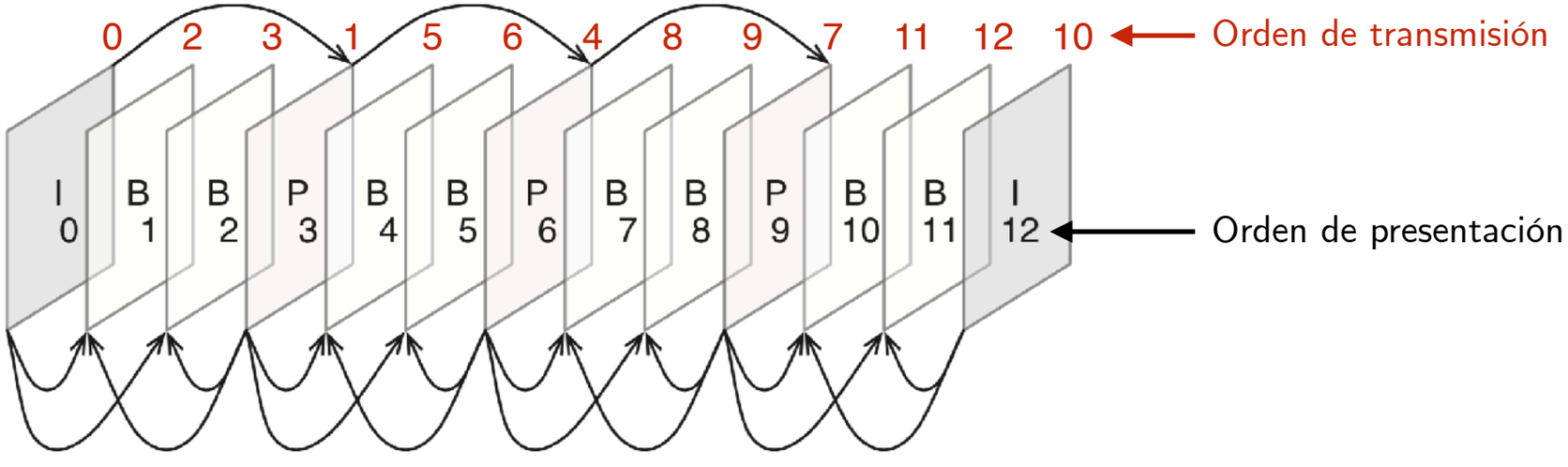
Búsqueda jerárquica o multiresolución

Predicción de cuadros

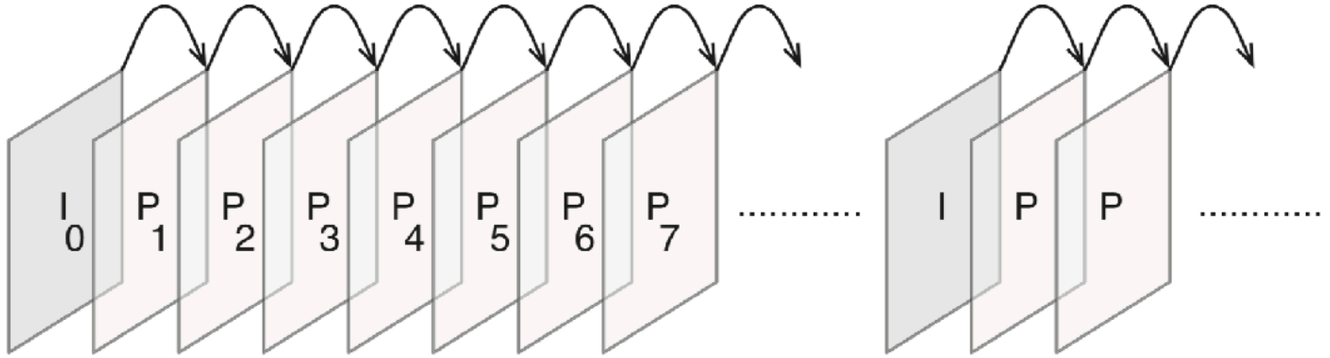
- Tres tipos de macrobloques/cuadros:
  - I (Intra): no se usa información temporal para codificarlo
  - P (Predicted): se predice a partir de un solo cuadro anterior (I o P)
  - B (Bidirectional): se predice a partir de más de un cuadro anterior o posterior (I o P)



Group of Pictures



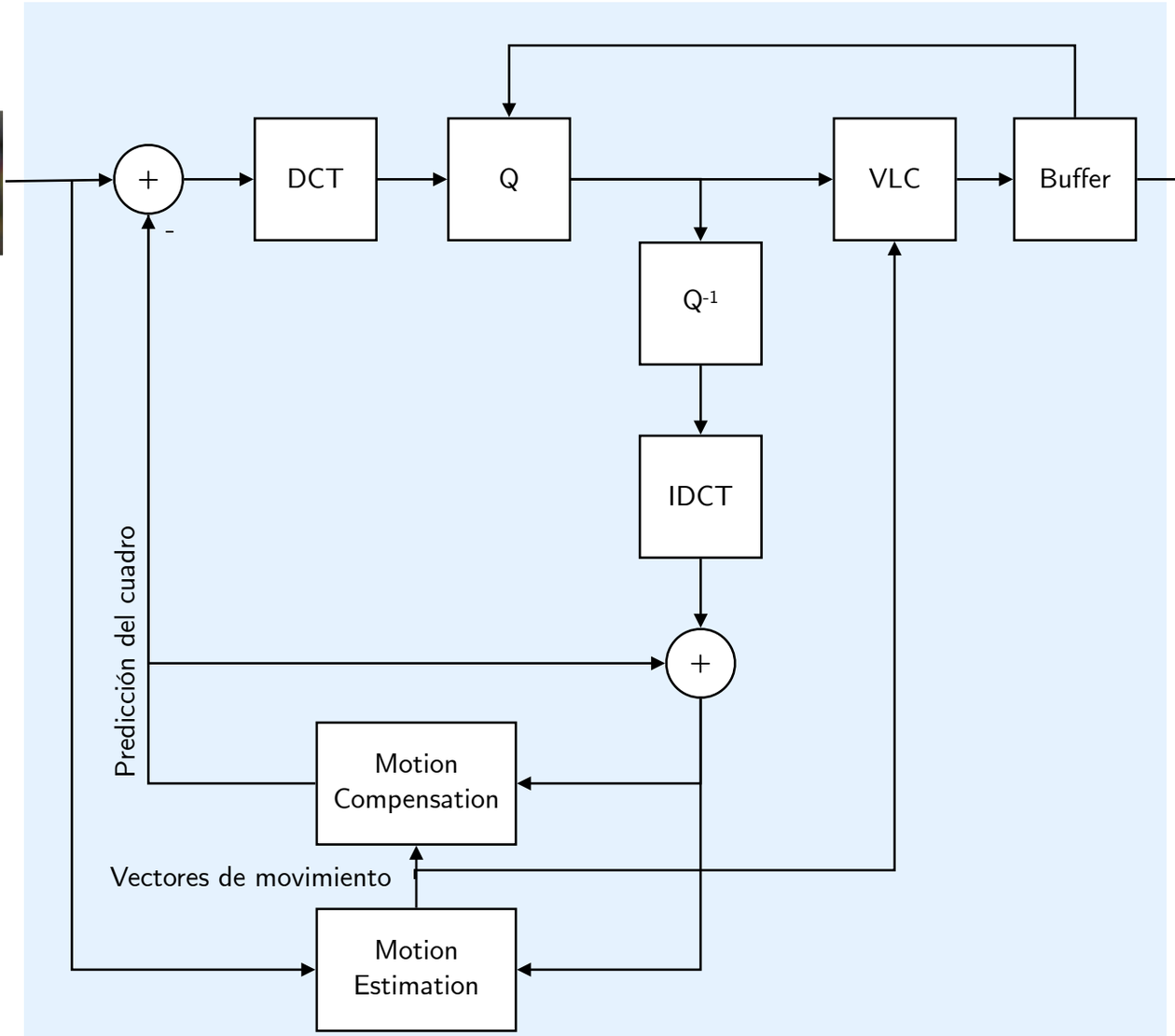
GOP clásico (MPEG-1 o MPEG-2)



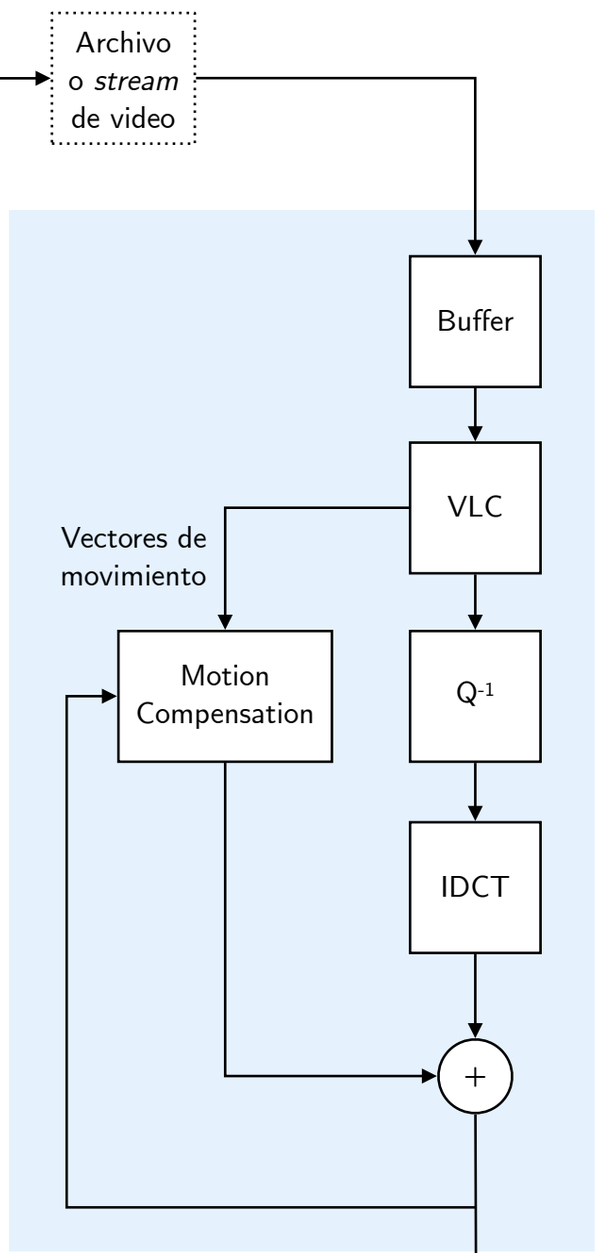
GOP de bajo retardo

- El GOP clásico usualmente comprime más al usar dos imágenes de referencia.
- El GOP de bajo retardo requiere menor cantidad de memoria intermedia.

Codificador - Decodificador MPEG



Codificador

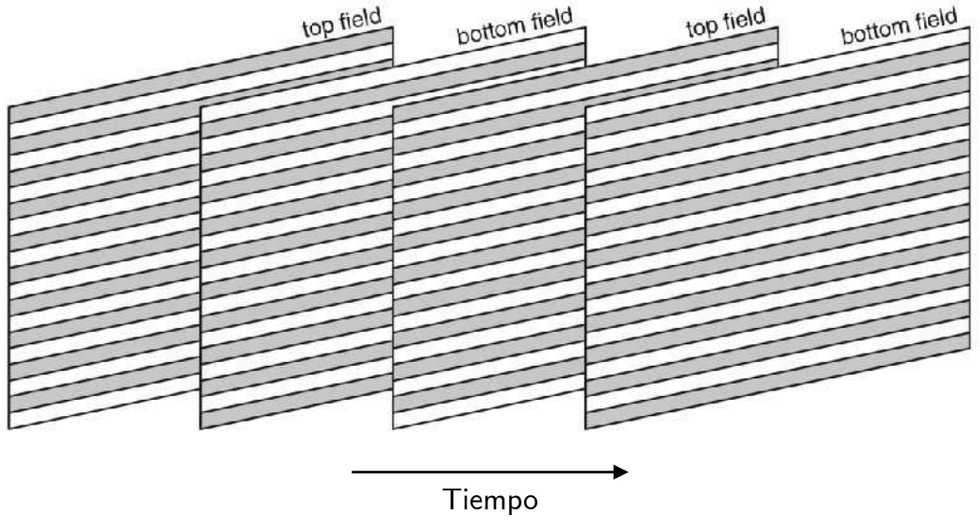


Decodificador

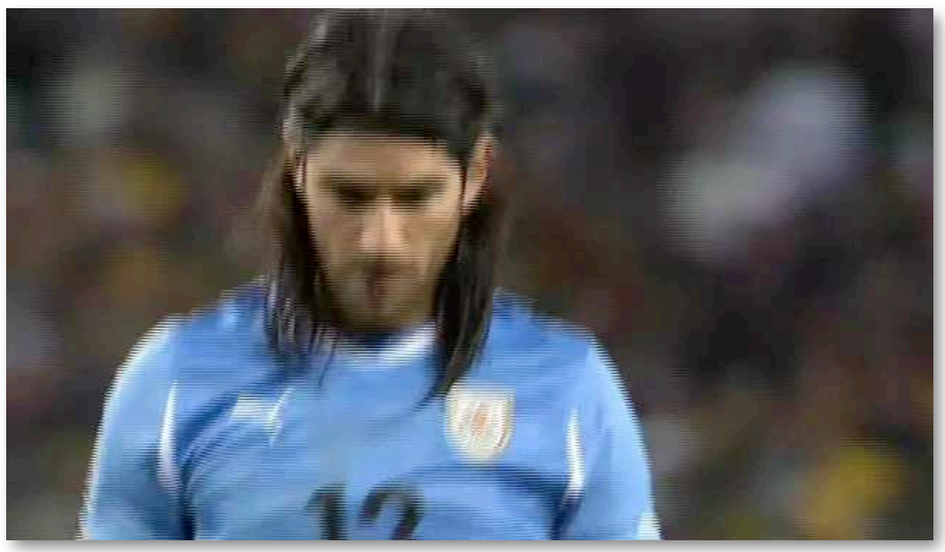
- División del cuadro en macrobloques (y bloques)
- Reducción de la redundancia temporal usando compresión inter-frame (estimación y compensación de movimiento)
- Reducción de la redundancia espacial usando compresión intra-frame (transformación y cuantificación)
- Reducción de la redundancia final (residuos y vectores de movimiento) mediante codificación de entropía

Video interlaceado (y progresivo)

- Interlaceado (*interlaced*): Aumento del frame rate percibido sin aumento del ancho de banda.
- Se utilizan las líneas pares/impares de cuadros consecutivos.
- ¿Reminiscencia de la TV analógica?



- 1080i@60fps vs. 1080p@30fps
- Artefactos típicos en bordes de movimientos *rápidos*
- E os para d *deinterl* re den *mu* se



Codificación y muestreo del color

- Adquisición en RGB, procesamiento luma o luminancia (Y) y croma o crominancia (CrCb, PrPb, UV, IQ).
  - Herencia de la TV monocromática.
- Es posible submuestrear las cromas (C_r, C_b) debido a la respuesta del SVH a los colores manteniendo similar percepción.
- Esquemas de muestreo 4:4:4, 4:2:2, 4:2:0

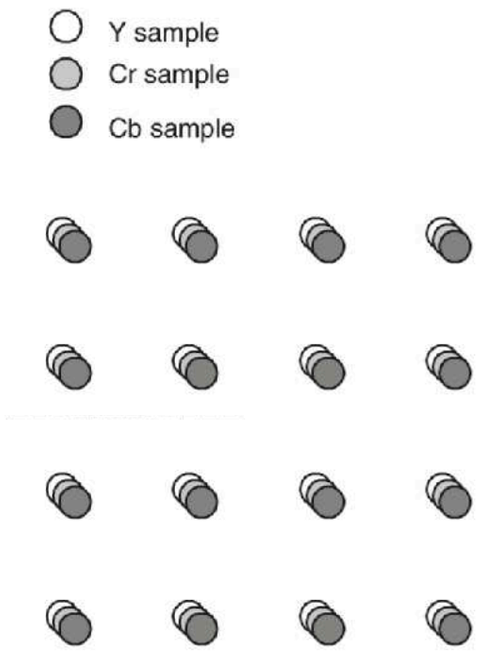
$$Y = k_r R + k_g G + k_b B$$

$$R = Y + 1.402 C_r$$
$$G = Y - 0.344 C_b - 0.714 C_r$$
$$B = Y + 1.772 C_b$$

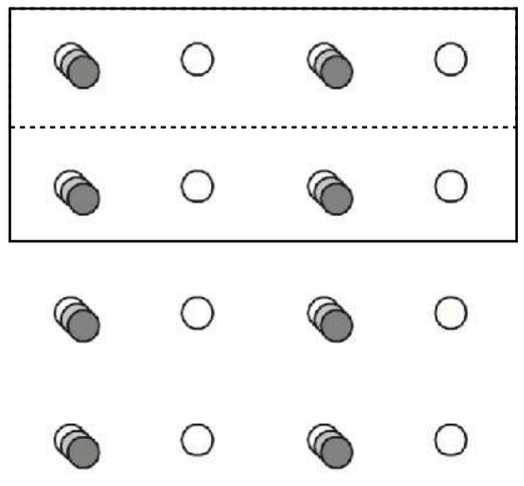
$$Y = 0.229R + 0.587G + 0.114B$$
$$C_b = 0.564(B - Y)$$
$$C_r = 0.713(R - Y)$$

$$R = Y + 1.408(V - 128)$$
$$G = Y - 0.346(U - 128) - 0.717(V - 128)$$
$$B = Y + 1.779(U - 128)$$

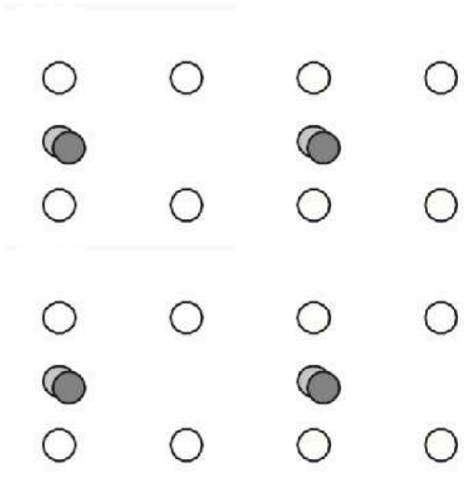
$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.169R + -0.331G + 0.500B + 128$$
$$V = 0.500R + -0.419G + -0.0813B + 128$$



4:4:4



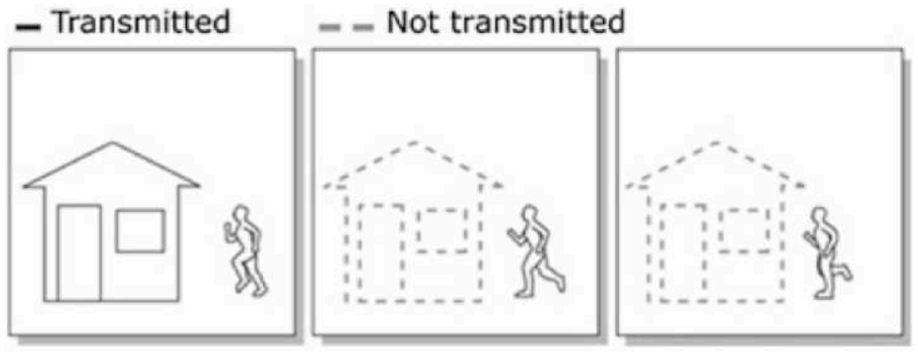
4:2:2

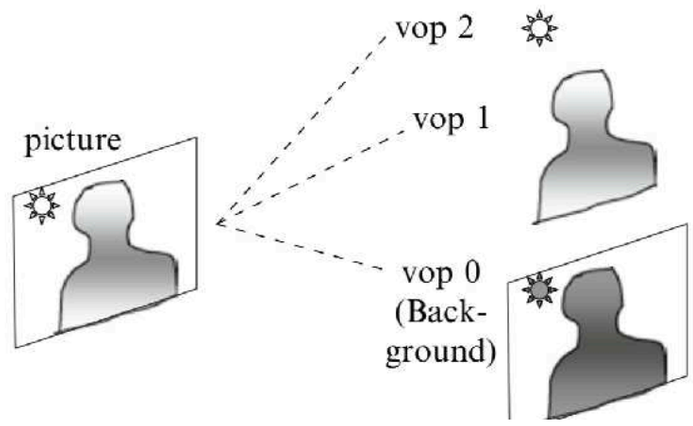


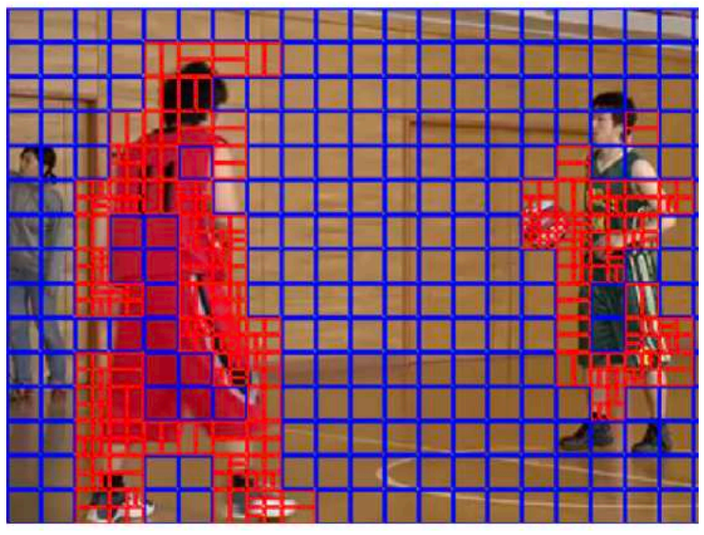
4:2:0

Características *avanzadas* de los codificadores

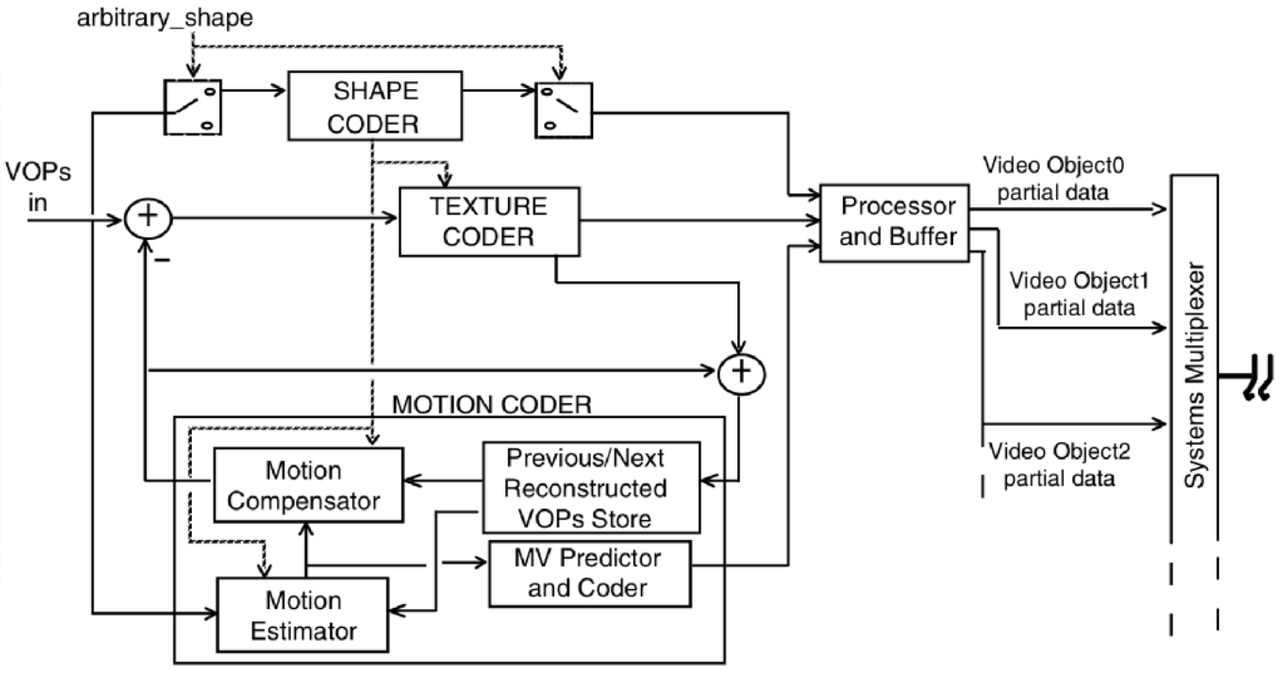
- Detección y codificación por objetos: *video object plane* (VOP), *sprite*.
- Macrobloques de tamaño variable.
- Otras transformaciones para redundancia espacial (wavelet).
- Cálculo paralelizable.
- Codificación escalable: adaptar la resolución/calidad con el decodificador.





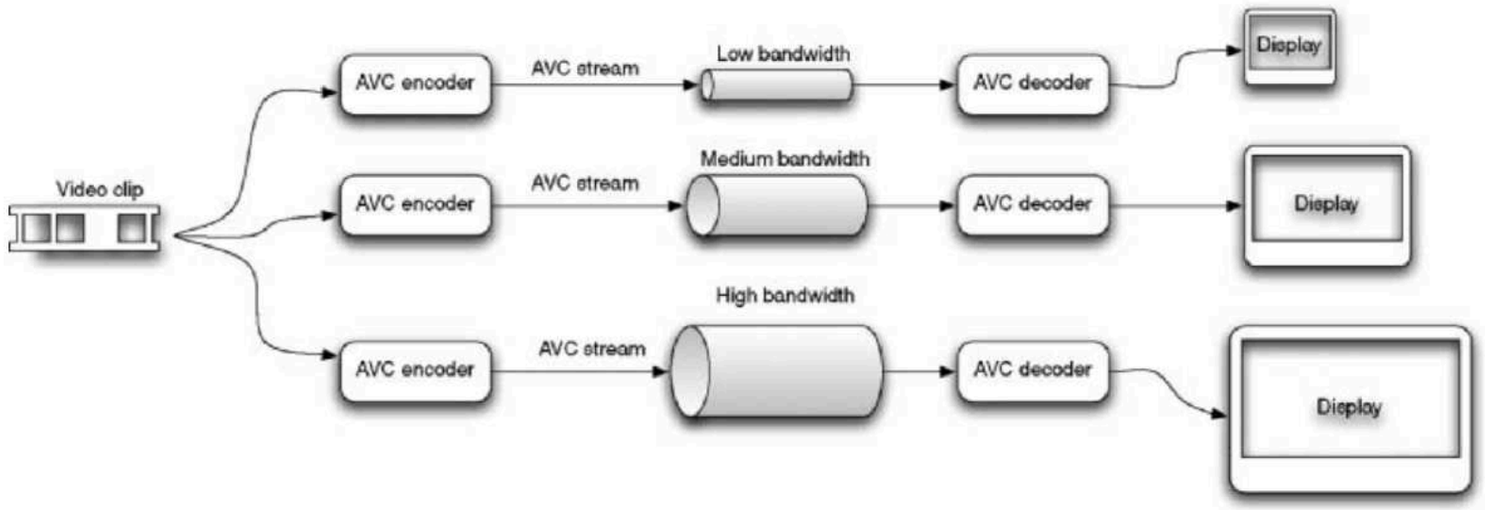


H.264/AVC

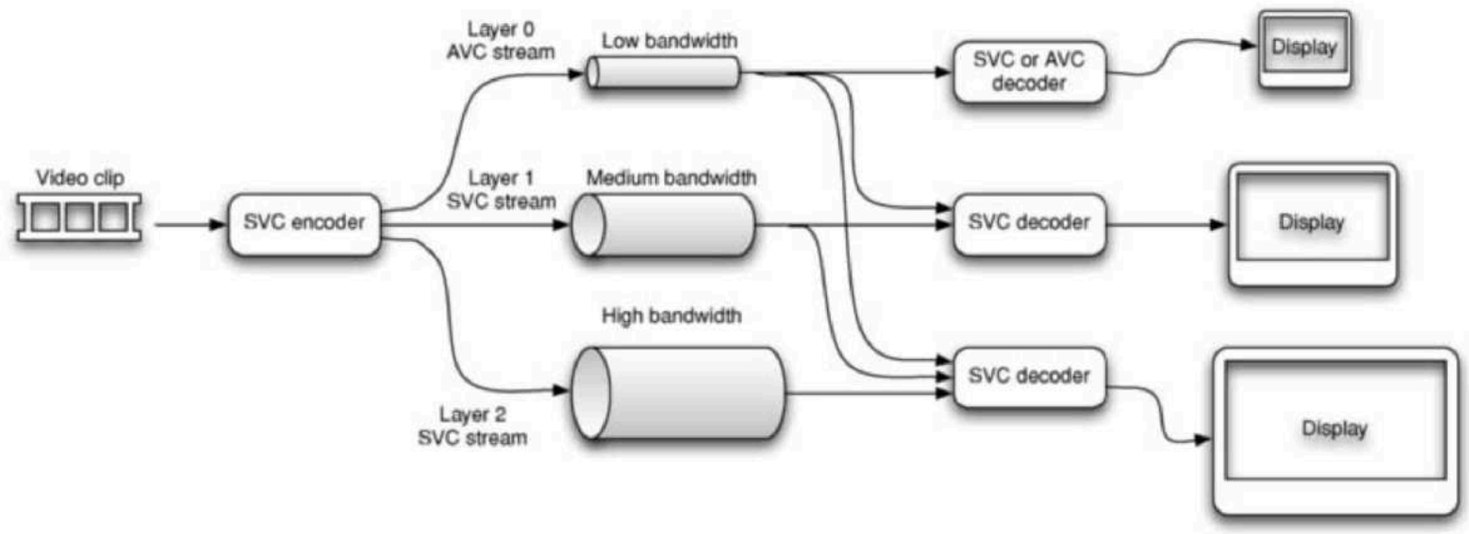


Codificador MPEG-4

Características *avanzadas* de los codificadores



Simulcast (AVC)



Escalable (SVC)

Medidas de calidad de imagen

- Peak Signal-to-Noise Ratio (PSNR) y error cuadrático medio (MSE)

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{2^B-1}{\mathrm{MSE}}\right) \qquad \mathrm{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(I_1(i,j)-I_0(i,j))^2$$

- *Structural Similarity* (SSIM): método (índice) para predecir la calidad percibida
  - Varía en [-1, 1], vale 1 cuando son idénticos.
  - Evalúan las *variaciones* que relacionan con la luma (l), el contraste (c) y la estructura (s).
  - Se evalúa con ventanas deslizantes basada en media (μ) y covarianza (σ).

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2+\mu_y^2+c_1} \qquad c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2+\sigma_y^2+c_2} \qquad s(x,y) = \frac{\sigma_{xy}+c_3}{\sigma_x\sigma_y+c_3}$$

$$c_1 = (0.01L)^2, c_2 = (0.03L)^2 = 2c_3, L = 2^B-1$$

$$\mathrm{SSIM}(x,y) = l(x,y)^\alpha c(x,y)^\beta s(x,y)^\gamma \qquad \alpha>0, \beta>0, \gamma>0$$

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \qquad \alpha=\beta=\gamma=1$$

$$\mathrm{MSSIM} = \frac{1}{M}\sum_{j=1}^{M}\mathrm{SSIM}(x_j,y_j)$$

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, *13*(4), 600-612.

Background subtraction

Background subtraction

- Background subtraction o foreground detection.
- Técnica para la detección de objetos o estructuras que se mueven sobre un fondo estático o que varía *más lentamente*.
- Se crea un **modelo** del fondo y los objetos son **anomalías** (outliers) de ese modelo.
- Varias aplicaciones: video-vigilancia, interacción hombre-computadora, codificación de video con objetos (MPEG-4/H.264), monitoreo de tráfico, ...

Secuencia original ("hall monitor")

Diferencia entre cuadros

Diferencia entre cuadros (MOG2)

Diferencia entre cuadros (GMG)

MOG2: Zivkovic, Z., & Van Der Heijden, F. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern recognition letters, 27(7), 773-780.
GMG: Godbehere, A. B., Matsukawa, A., & Goldberg, K. (2012, June). Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *2012 American Control Conference (ACC)* (pp. 4305-4312). IEEE.

Background subtraction

- Usualmente se toma la diferencia con el modelo de fondo B, ¿cómo aprender B?
  - Promedio de los cuadros anteriores (pixel): $B(x,y,t) = \frac{1}{N}\sum_{i=1}^{N} I(x,y,t-i)$
  - Con un factor de olvido o *running average*:

$$B(x,y,t) = \alpha\, B(x,y,t-2) + (1-\alpha)\, I(x,y,t-1) \quad \alpha \in (0,1)$$

  - Aprender una distribución de probabilidad (modelo):
    - *running average* en una ventana:

$$\mu_t = \alpha I_t + (1-\alpha)\mu_{t-1}, \sigma_t^2 \alpha |I_t - \mu_t|^2 + (1-\alpha)\sigma_{t-1}^2 \Rightarrow |I_t - \mu_t| \gtrless k\sigma_t$$

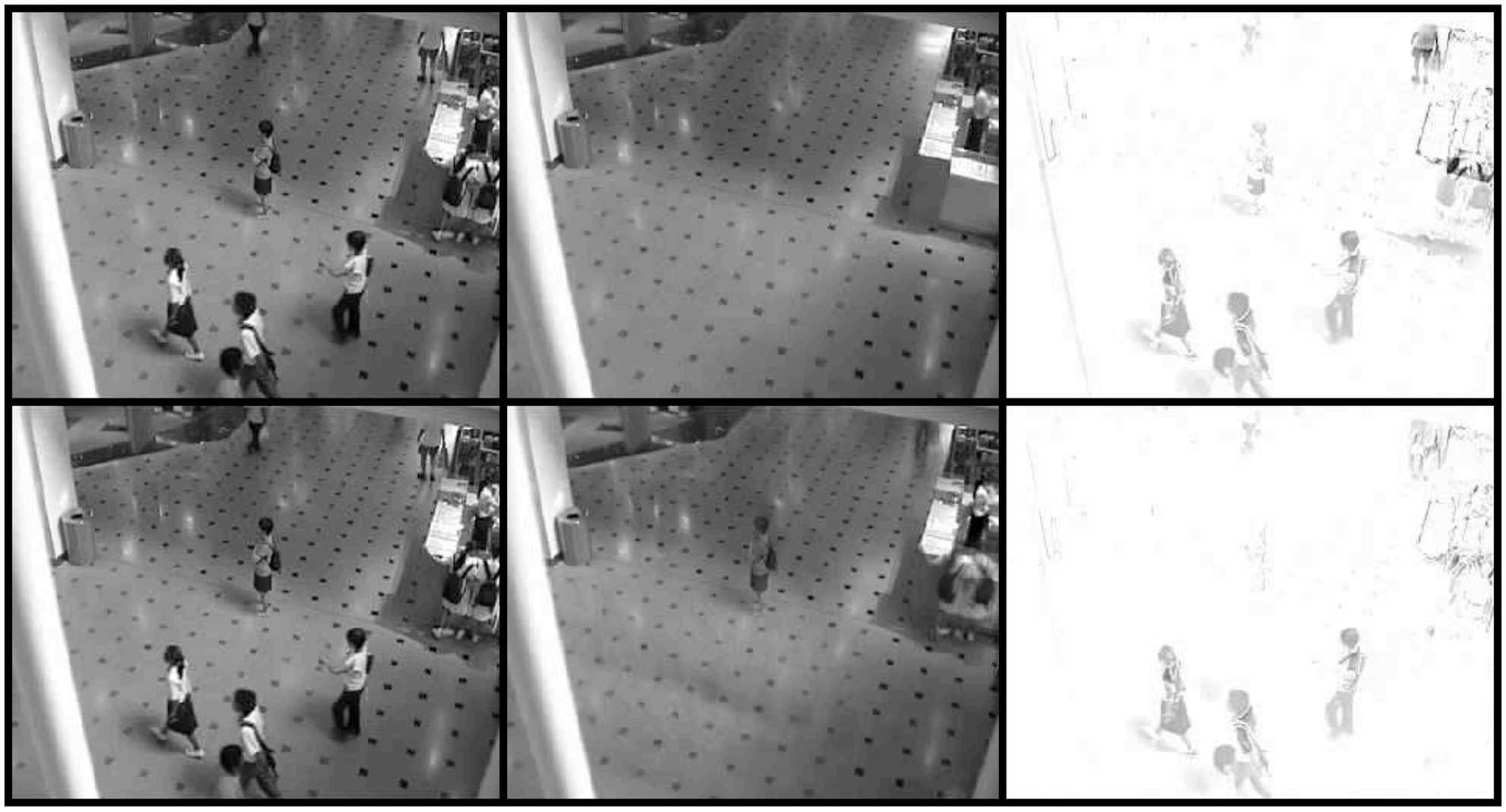
    - ajuste de un mezcla de Gaussianas (MOG)

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} N\left(X_t \mid \mu_{i,t}, \Sigma_{i,t}\right) \qquad K?$$

    - ...

- Otros enfoques pueden ser no causales trabajando con todo el video
  - Principal Component Analysis
  - Ajuste modelo de bajo rango
  - Aprendizaje: diccionarios en modelos ralos

Background subtraction



Video de entrada

Fondo detectado y aprendido

Objetos detectados

Herramientas para video

- OpenCV: algoritmos de *mayor nivel*
- FFmpeg (https://www.ffmpeg.org/)
  - (casi) todo lo que se pueda hacer de *bajo nivel*
  - *Core* de muchas aplicaciones (VLC, MPlayer, HandBrake, Blender, Google Chrome, Videomorph, QuickTime, ...)
  - Línea de comando

```
$ ffmpeg [global_options] {[input_file_options] -i input_url} ... {[output_file_options] output_url}
```

```
$ ffmpeg -i input.mov -o output.mp4
```

```
$ ffmpeg -i input.ts -c:a mp2 -c:v mpeg2video output.mp4
```

  - *Familia* de programas: ffmpeg , ffserver (streaming server), ffplay (player), libavcodec (códecs), libavformat (mux/demux), libavutil (rutinas comunes FFmpeg), libpostproc (postproc.), libswscale (escalado)
- ffmpy: python FFmpeg command line wrapper (documentación)

```
>>> import ffmpy
>>> ff = FFmpeg(
...     inputs={'input.ts': None},
...     outputs={'output.mp4': '-c:a mp2 -c:v mpeg2video'}
... )
>>> ff.cmd # Ver la 'línea de comando' que se  ejecuta
'ffmpeg -i input.ts -c:a mp2 -c:v mpeg2video output.mp4'
>>> ff.run()
```

Bibliografía

- Richardson, I.E., 2010. *The H.264 Advanced Video Compression Standard*, Wiley.
- Szeliski, R., 2010. *Computer Vision: Algorithms and Applications*, Springer Science & Business Media. (http://szeliski.org/Book)
- Fleet, D., & Weiss, Y. (2006). Optical flow estimation. In Handbook of mathematical models in computer vision (pp. 237-257). Springer, Boston, MA.
- Xu, Y., Dong, J., Zhang, B., & Xu, D. (2016). Background modeling methods in video analysis: A review and comparative evaluation. CAAI Transactions on Intelligence Technology, 1(1), 43-60.