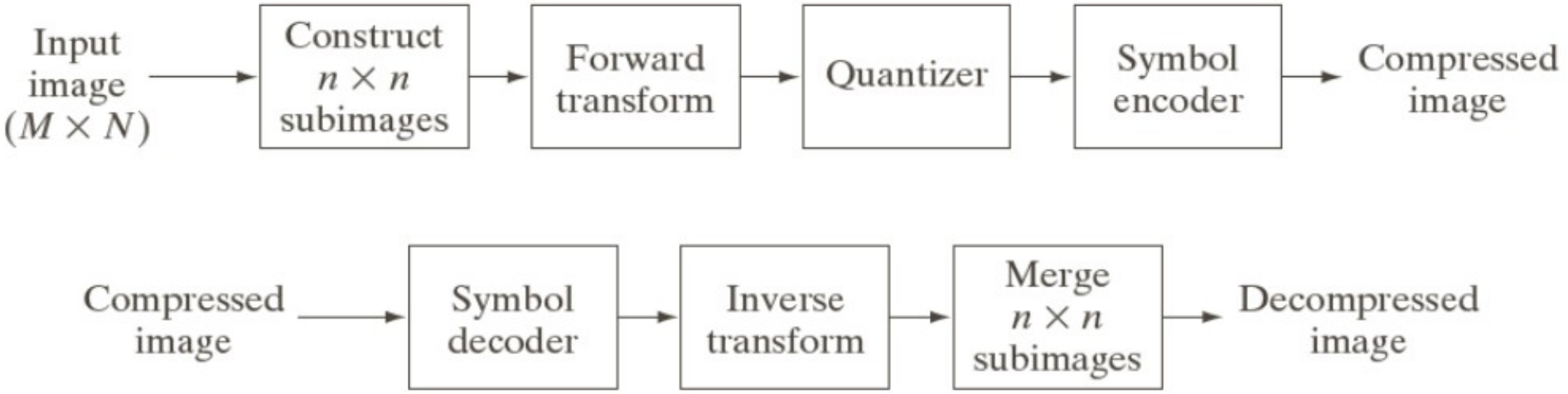
Tratamiento de Imágenes por Computadora

Transformada de Karhunen-Loeve (KLT) y
Transformada Discreta de Coseno (DCT)

Presentación basada en el libro Digital Image Processing, Gonzalez & Woods

Aplicación

- Aplicación a la compresión de imágenes por bloques



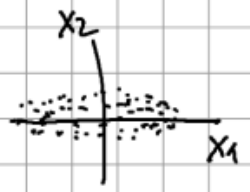
KLT

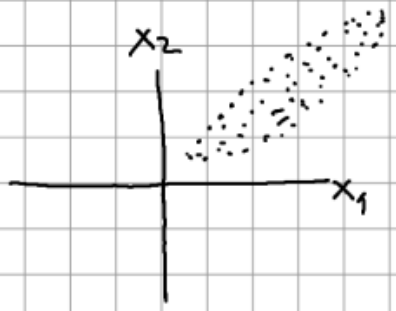
$\underline{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

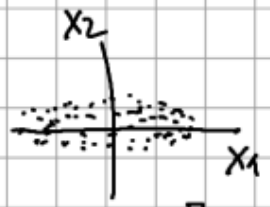
$\underline{m}_X = E[\underline{X}]$ 2x1 vector

$C_X = E\left[(\underline{X} - \underline{m}_X)(\underline{X} - \underline{m}_X)^t\right]$ 2x2 matrix

$C_X = \begin{bmatrix} \mathrm{var}(x_1) & \mathrm{cov}(x_1, x_2) \\ \mathrm{cov}(x_1, x_2) & \mathrm{var}(x_2) \end{bmatrix}$







$m_x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$var(x_1) > var(x_2)$

$cov(x_1, x_2) = 0$

$m_x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

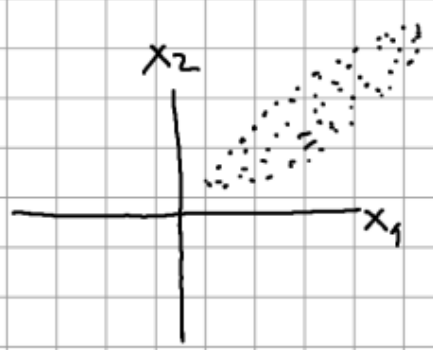
$var(x_1) = var(x_2)$

$cov(x_1, x_2) = 0$

$var(x_1) = var(x_2)$

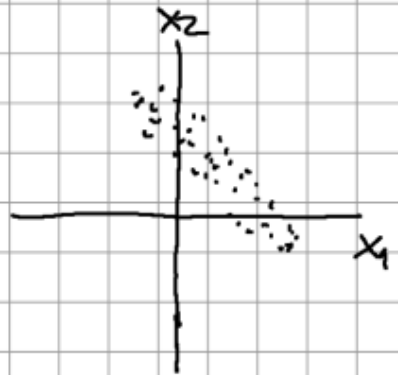
$cov(x_1, x_2) = 0$

x_1 y x_2 no correlacionadas



$cov(x_1, x_2) > 0$

correlación positiva



$cov(x_1, x_2) < 0$

correlación negativa

Estimadores

$$m_x = \frac{1}{K} \sum_{k=1}^{K} \underline{x}_k$$

$$C_x = \frac{1}{K} \sum_{k=1}^{K} \underline{x}_k \underline{x}_k^t - \underline{m}_x \underline{m}_x^t$$

C_x real y simétrica

⇒ se puede diagonalizar en una base de vectores propios.

$e_i, \lambda_i \quad i = 1, \dots, n \qquad \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$

$$A \triangleq \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

$$y = A(x - m_x)$$

* Transformada de
  - Hotelling
  - Karhunen-Loeve

* PCA Principal Component Analysis

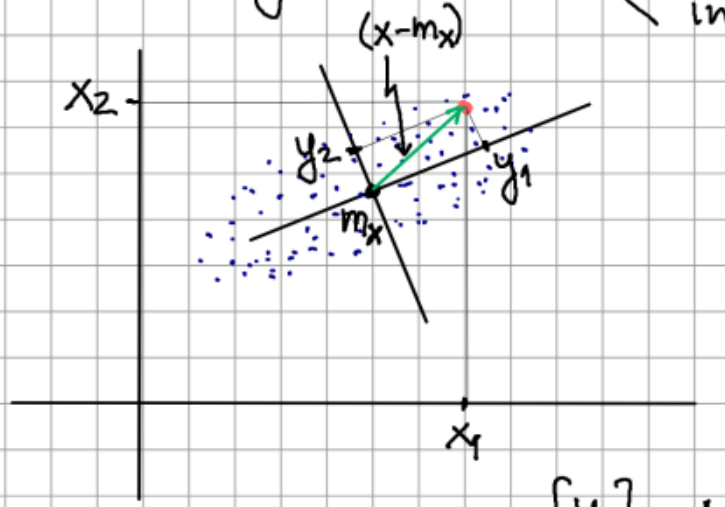
$$C_y = A C_x A^t \qquad C_y = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \\ & & & \lambda_n \end{bmatrix}$$

A tiene sus filas ortonormales (base de vectores propios)

$$A^{-1} = A^t$$

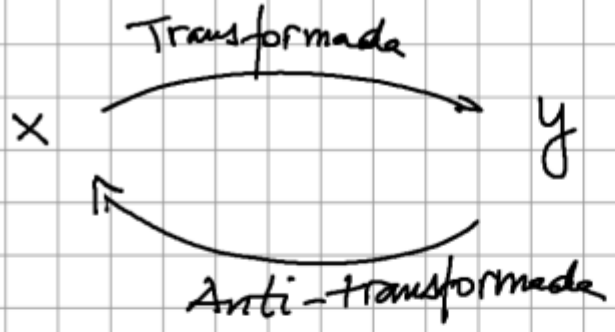
$\Rightarrow$ se puede recuperar x como

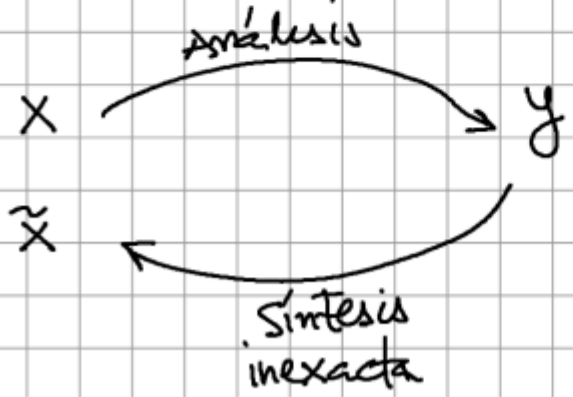
$$x = A^t y + m_x \qquad \text{(transformada inversa)}$$



$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = A(x - m_x)$$

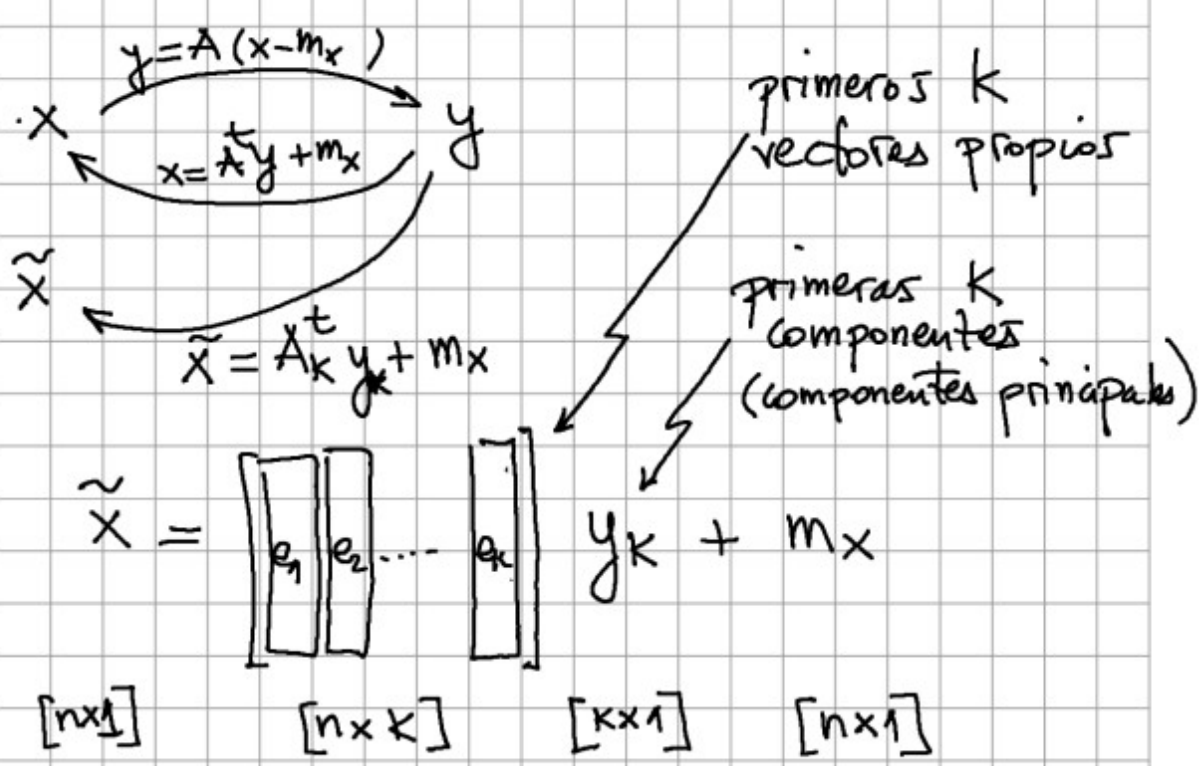
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = A^t y + m_x$$





Objetivos:

- filtrar
- comprimir
- eliminar ruido

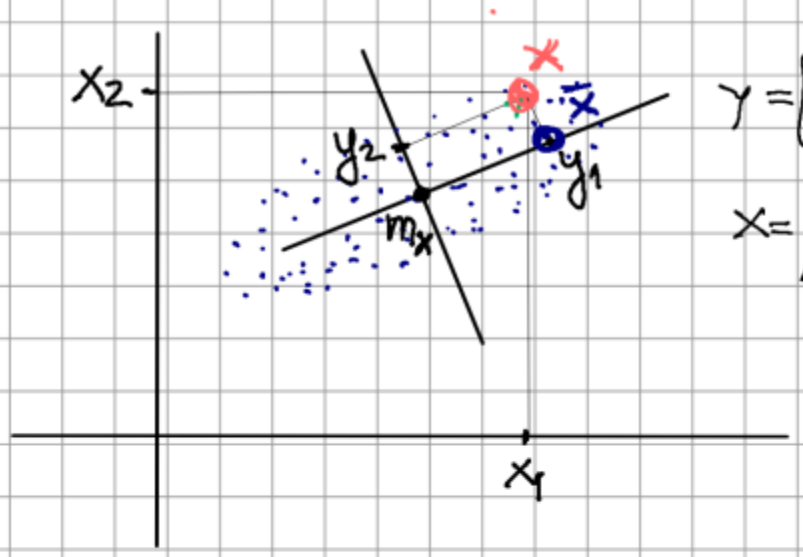


Error de truncamiento

$$MSE = E\left[(x-\tilde{x})^t (x-\tilde{x})\right]$$

⇑ La transformada KL minimiza el error de truncamiento.

⇓ La base depende de los datos.



$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = A(x - m_x)$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = A^t y + m_x$$

En este caso truncar sería quedarse con la componente principal y_1

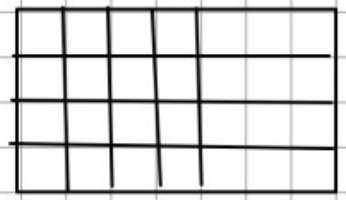
$$x = \begin{bmatrix} e_1 & e_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + m_x$$

síntesis exacta

$$\tilde{x} = \begin{bmatrix} e_1 \end{bmatrix} y_1 + m_x$$

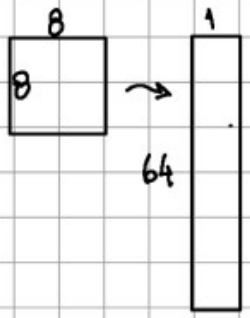
síntesis inexacta

Aplicación: compresión por bloques

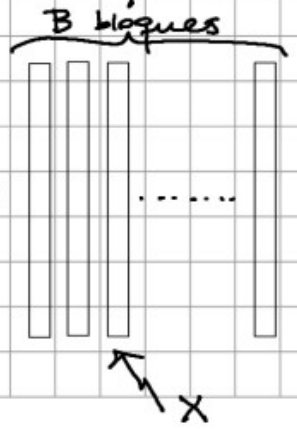


Partimos la imagen en bloques de 8×8 pixeles

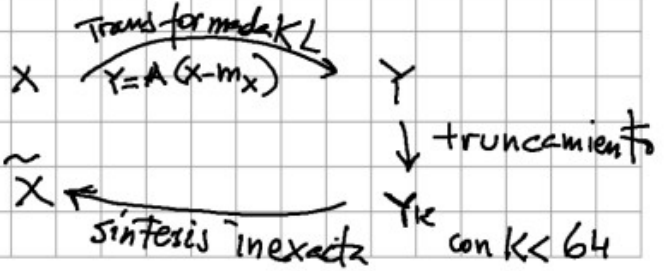
Cada bloque 8×8 lo podemos reordenar como un vector 64×1 recorriéndolo por filas

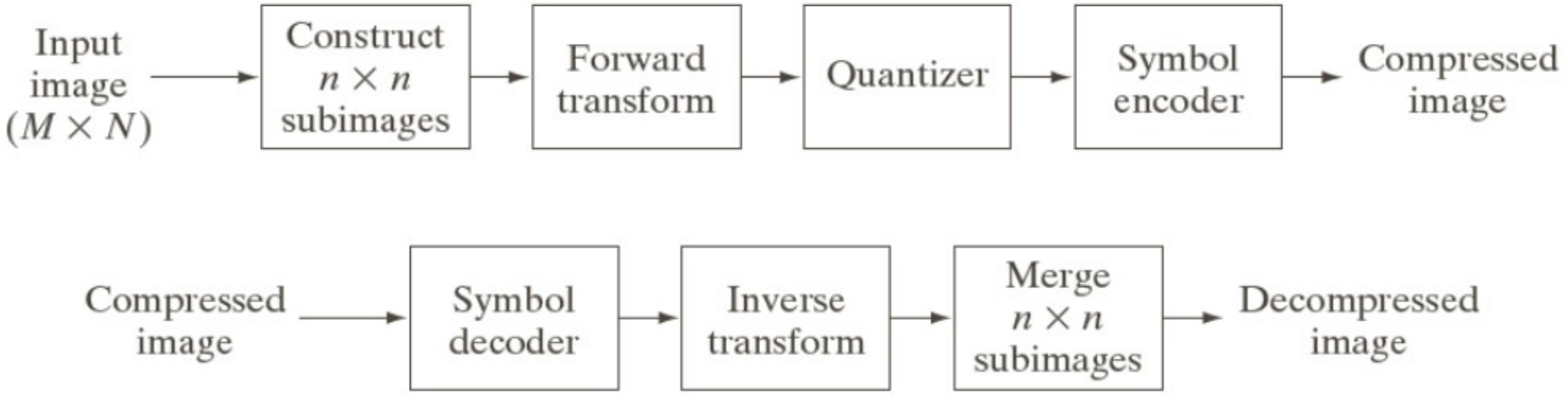


Todos los bloques los podemos pensar como realizaciones de un proceso estacionario (no es cierto para imágenes pero es una aproximación)

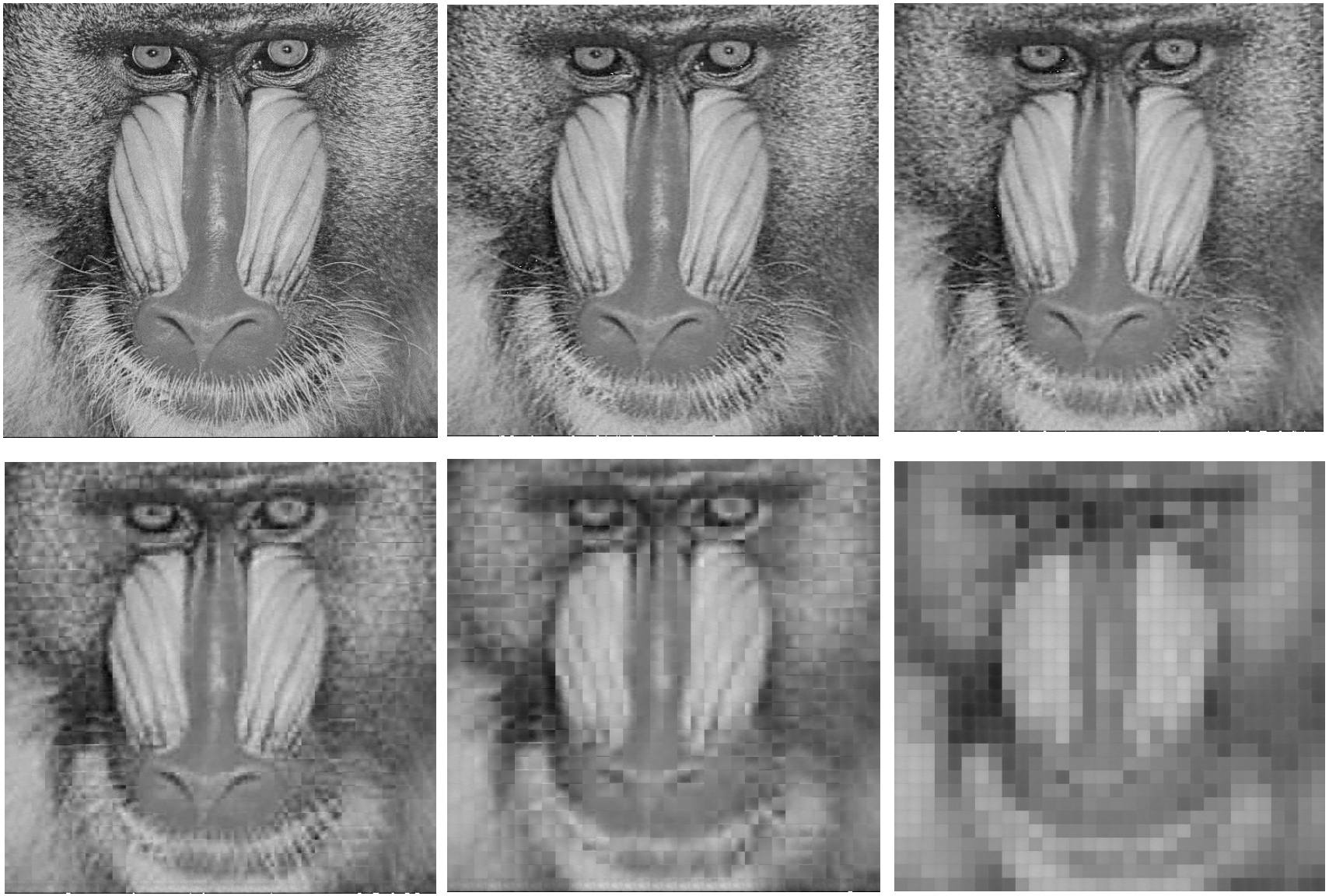


Ahora los vectores 64×1 son nuestras X



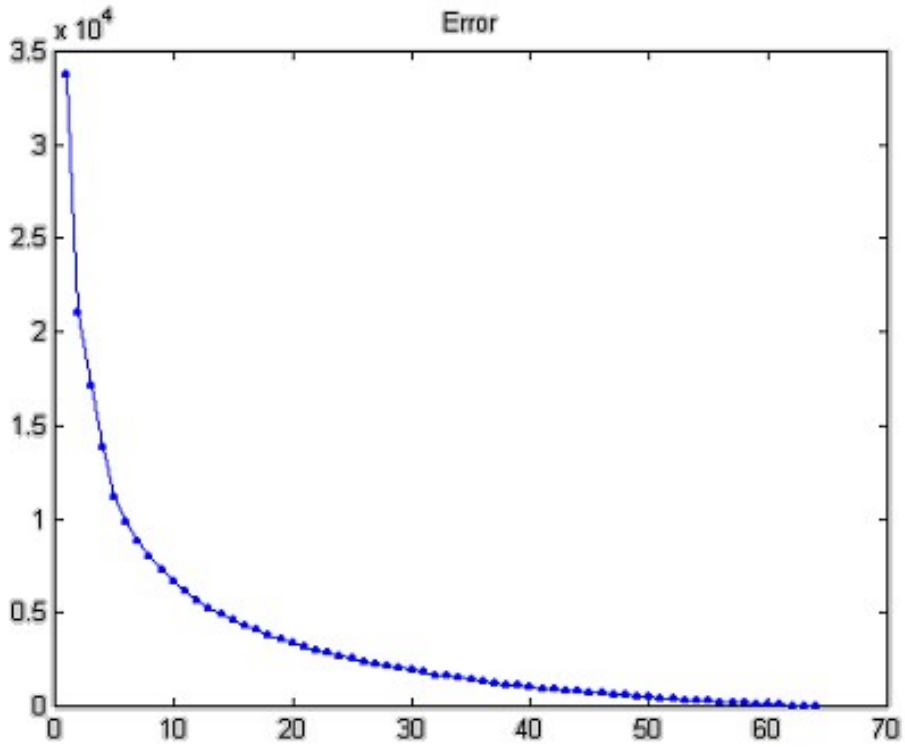

Input image ($M \times N$)
Construct $n \times n$ subimages
Forward transform
Quantizer
Symbol encoder
Compressed image
Compressed image
Symbol decoder
Inverse transform
Merge $n \times n$ subimages
Decompressed image

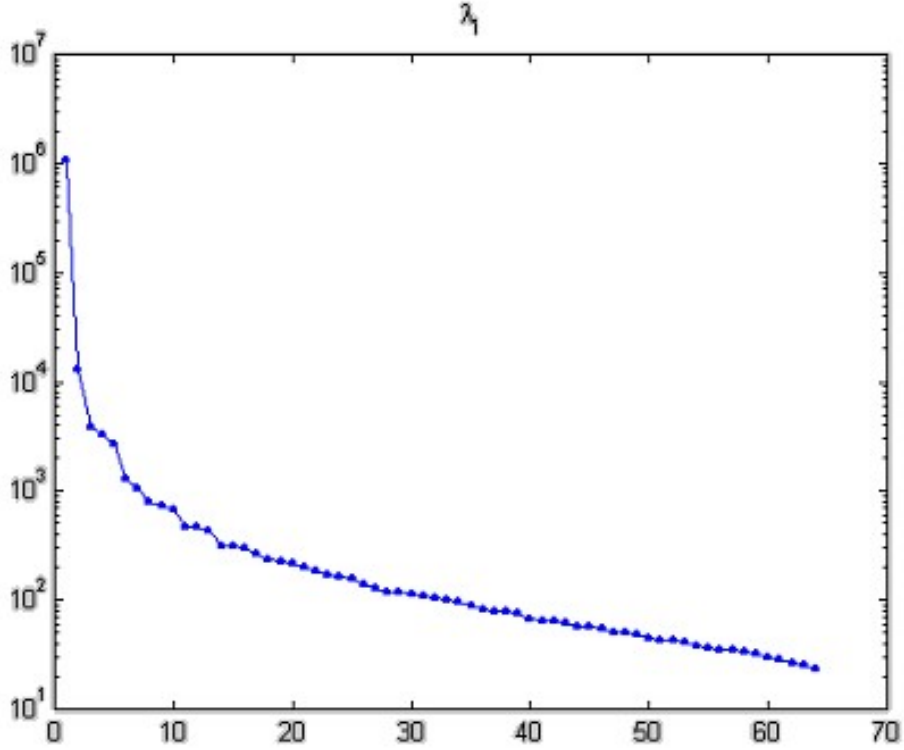
KLT: ejemplo



KLT. D = 64, 1, 4, 8, 16 y 32

KLT: aplicación a la compresión de imágenes





Izquierda: Disminución del error de proyección al ir agregando componentes.
Derecha: Magnitud de los valores propios.

Transformadas

- Pares transformados (análisis - síntesis)

$$T(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)g(x,y,u,v)$$

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)h(x,y,u,v)$$

Fourier (DFT)

$$g(x, y, u, v) = \frac{1}{N^2} e^{-j2\pi(ux+vy)/N}$$

$$h(x, y, u, v) = e^{j2\pi(ux+vy)/N}$$

Coseno discreto (DCT)

$$g(x, y, u, v) = h(x, y, u, v)$$

$$= \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

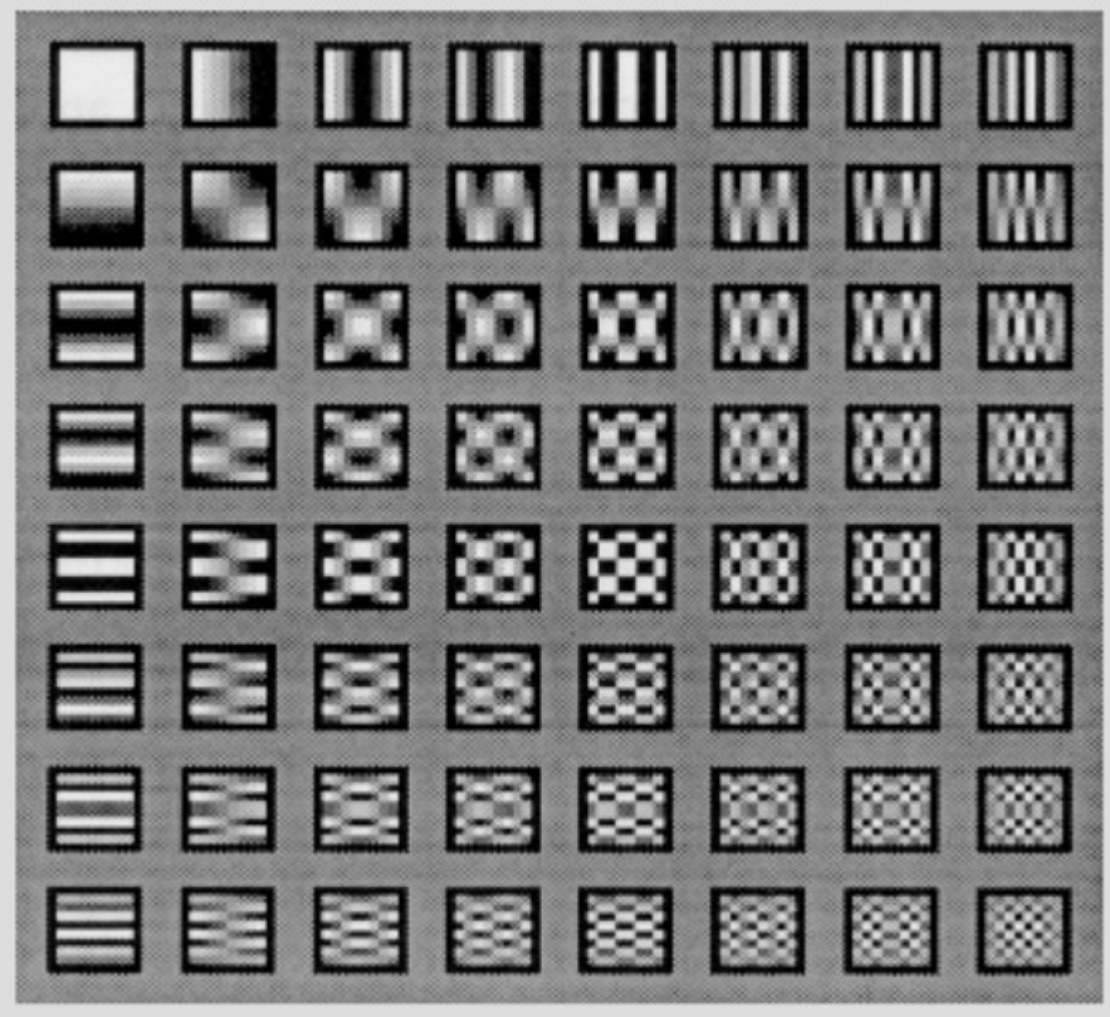
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \ldots, N-1 \end{cases}$$

Funciones base

$$\mathbf{F} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v)\mathbf{H}_{uv}$$

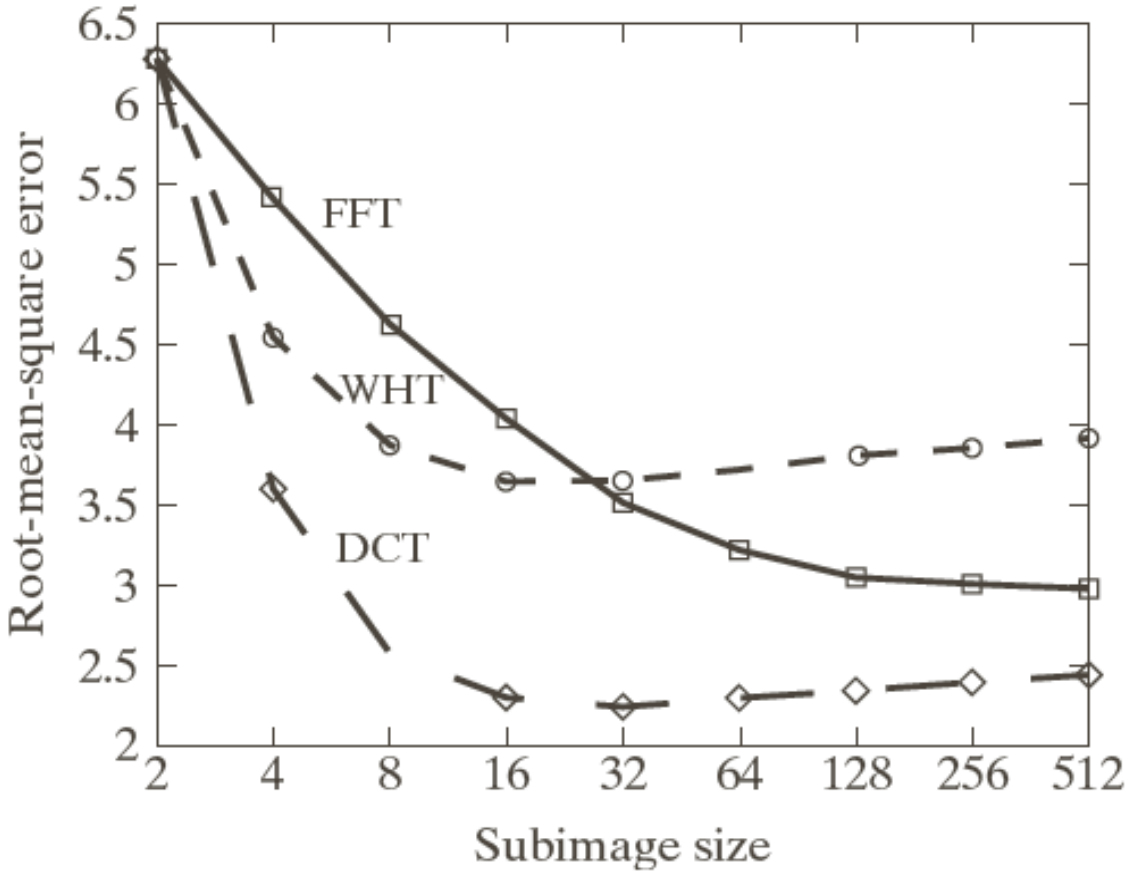
$$\mathbf{H}_{uv} = \begin{bmatrix} h(0,0,u,v) & h(0,1,u,v) & \cdots & h(0,n-1,u,v) \\ h(1,0,u,v) & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & & & \\ h(n-1,0,u,v) & h(n-1,1,u,v) & \cdots & h(n-1,n-1,u,v) \end{bmatrix}.$$

DCT 8x8

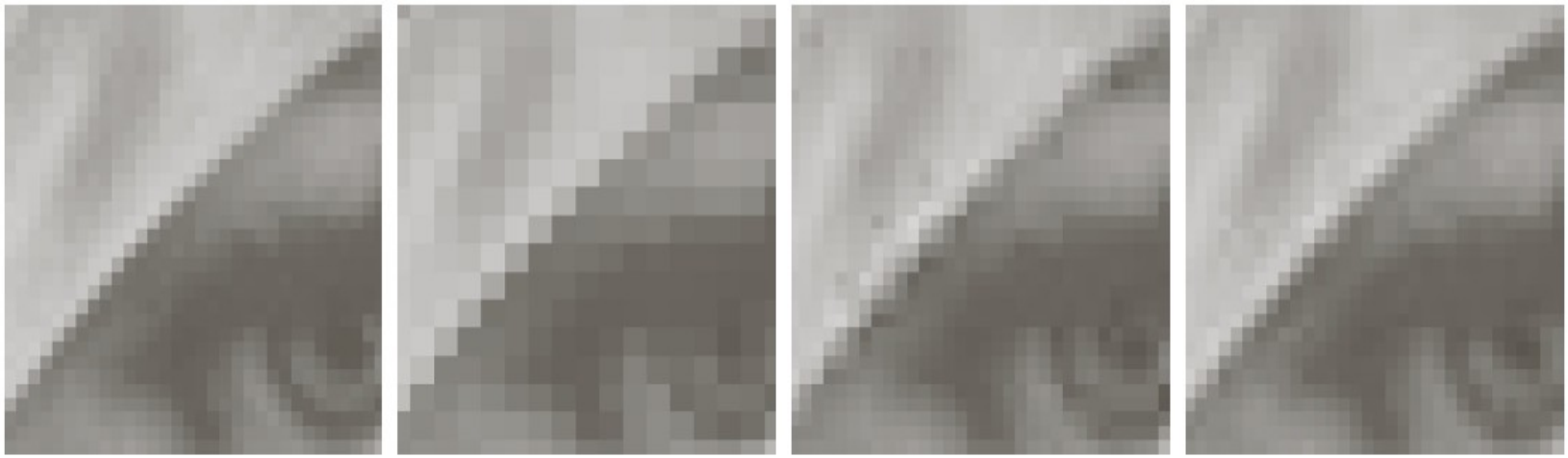


Tamaño de bloque

- Error de reconstrucción vs. Tamaño de bloque



Tamaño de bloque

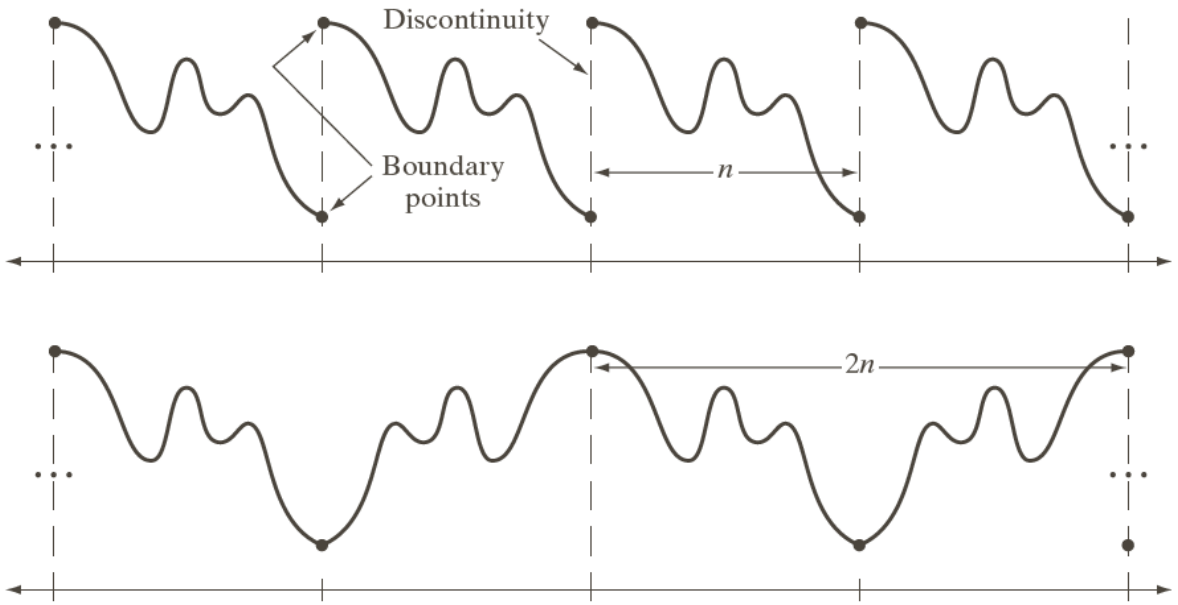


a b c d

FIGURE 8.27 Approximations of Fig. 8.27(a) using 25% of the DCT coefficients and (b) 2 × 2 subimages, (c) 4 × 4 subimages, and (d) 8 × 8 subimages. The original image in (a) is a zoomed section of Fig. 8.9(a).

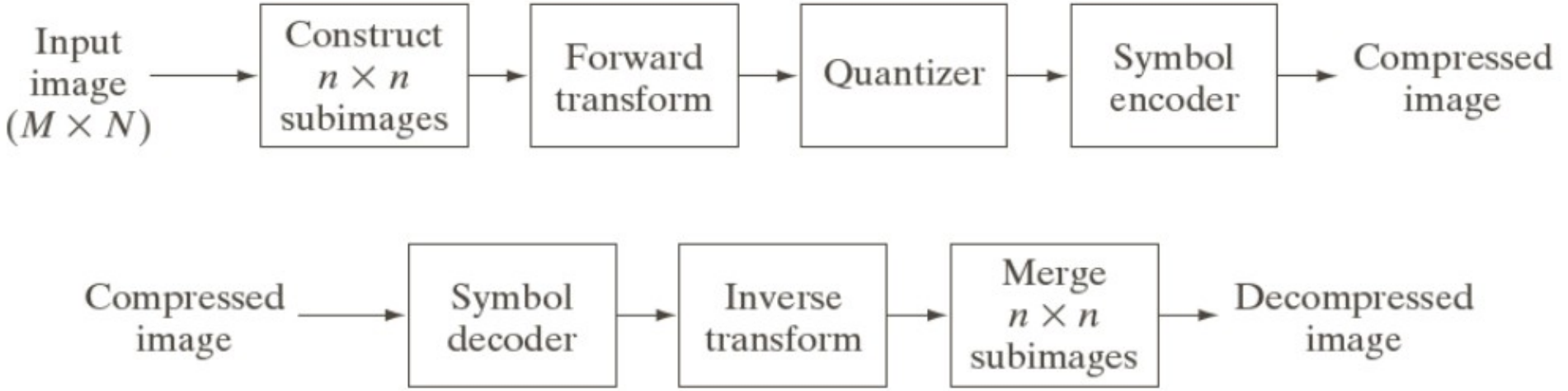
Error de truncamiento

- DCT preferida frente a DFT y WHT
  - Evita mejor efectos de bloque
  - Si bien no es óptima, aproxima bien a la KLT
  - Se pueden hacer implementaciones rápidas



Compresión por bloques

Opciones de cuantización



$$\hat{\mathbf{F}} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \gamma(u, v) T(u, v) \mathbf{H}_{uv}$$

Opciones de cuantización

- Zonal
  - Calcular coeficientes que tienen más info (con la mayor varianza)
    - Sobre la propia imagen
    - Con un modelo de tipo de imagen
- Por umbral
  - Quedarse con los coeficientes más grandes
  - Opciones para el umbral
    - Global para toda la imagen
    - Para cada subimagen
    - Dependiente de la ubicación del coeficiente

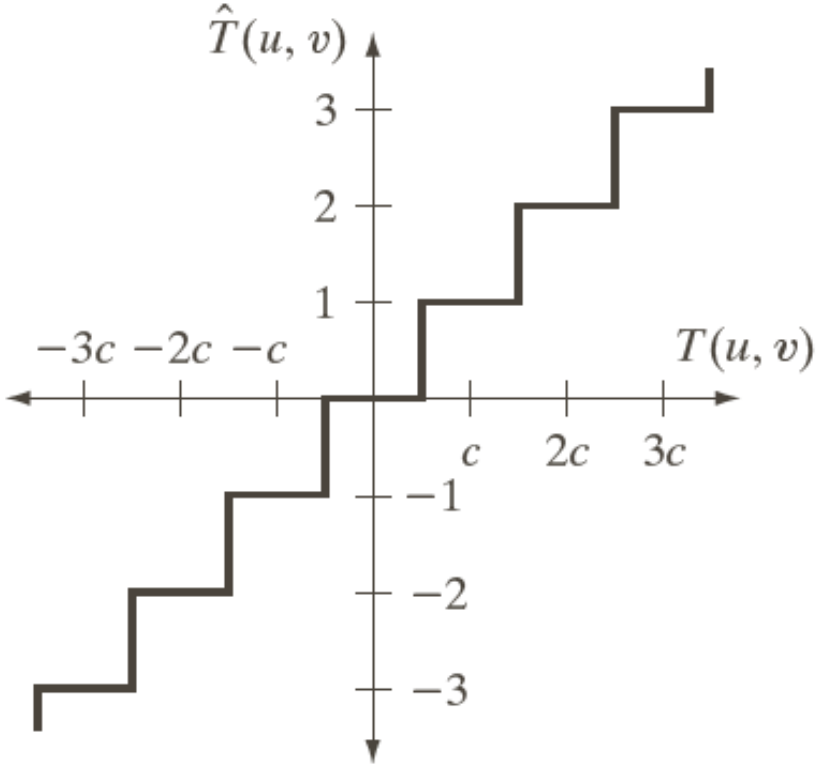
Cuantización

- Umbralización y cuantización
  - Z es la matriz de normalización (perceptual)
  - Variando valores de Z varía el nivel de compresión

$$\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$$

$$\mathbf{Z} = \begin{bmatrix} Z(0,0) & Z(0,1) & \cdots & Z(0,n-1) \\ Z(1,0) & & & \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & & \cdot \\ Z(n-1,0) & Z(n-1,1) & \cdots & Z(n-1,n-1) \end{bmatrix}.$$

Matriz Z



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

JPEG - Ejemplo

- Bloque original

52	55	61	66	70	61	64	73
63	59	66	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	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

JPEG - Ejemplo

- Centrado de niveles en 0 (-128 a 127)

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−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	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

JPEG - Ejemplo

- DCT

−415	−29	−62	25	55	−20	−1	3
7	−21	−62	9	11	−7	−6	6
−46	8	77	−25	−30	10	7	−5
−50	13	35	−15	−9	6	0	3
11	−8	−13	−2	−1	1	−4	1
−10	1	3	−3	−1	0	2	−1
−4	−1	2	−1	2	−3	1	−2
−1	−1	−1	−2	−1	−1	0	−1

JPEG - Ejemplo

- Cuantificación usando $\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	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

Ordenamiento de la subimagen transformada

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

JPEG - Ejemplo

- Ordenamiento

$$[-26\ -3\ 1\ -3\ -2\ -6\ 2\ -4\ 1\ -4\ 1\ 1\ 5\ 0\ 2\ 0\ 0\ -1\ 2\ 0\ 0\ 0\ 0\ 0\ -1\ -1\ \mathrm{EOB}]$$

JPEG - Ejemplo

- Codificación - decodificación

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	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

JPEG - Ejemplo

- Denormalización usando $\dot{T}(u, v) = \hat{T}(u, v)Z(u, v).$

−416	−33	−60	32	48	0	0	0
12	−24	−56	0	0	0	0	0
−42	13	80	−24	−40	0	0	0
−56	17	44	−29	0	0	0	0
18	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

JPEG - Ejemplo

- DCT inversa

−70	−64	−61	−64	−69	−66	−58	−50
−72	−73	−61	−39	−30	−40	−54	−59
−68	−78	−58	−9	13	−12	−48	−64
−59	−77	−57	0	22	−13	−51	−60
−54	−75	−64	−23	−13	−44	−63	−56
−52	−71	−72	−54	−54	−71	−71	−54
−45	−59	−70	−68	−67	−67	−61	−50
−35	−47	−61	−66	−60	−48	−44	−44

JPEG - Ejemplo

- De-centrado de niveles (llevar a 0-255)

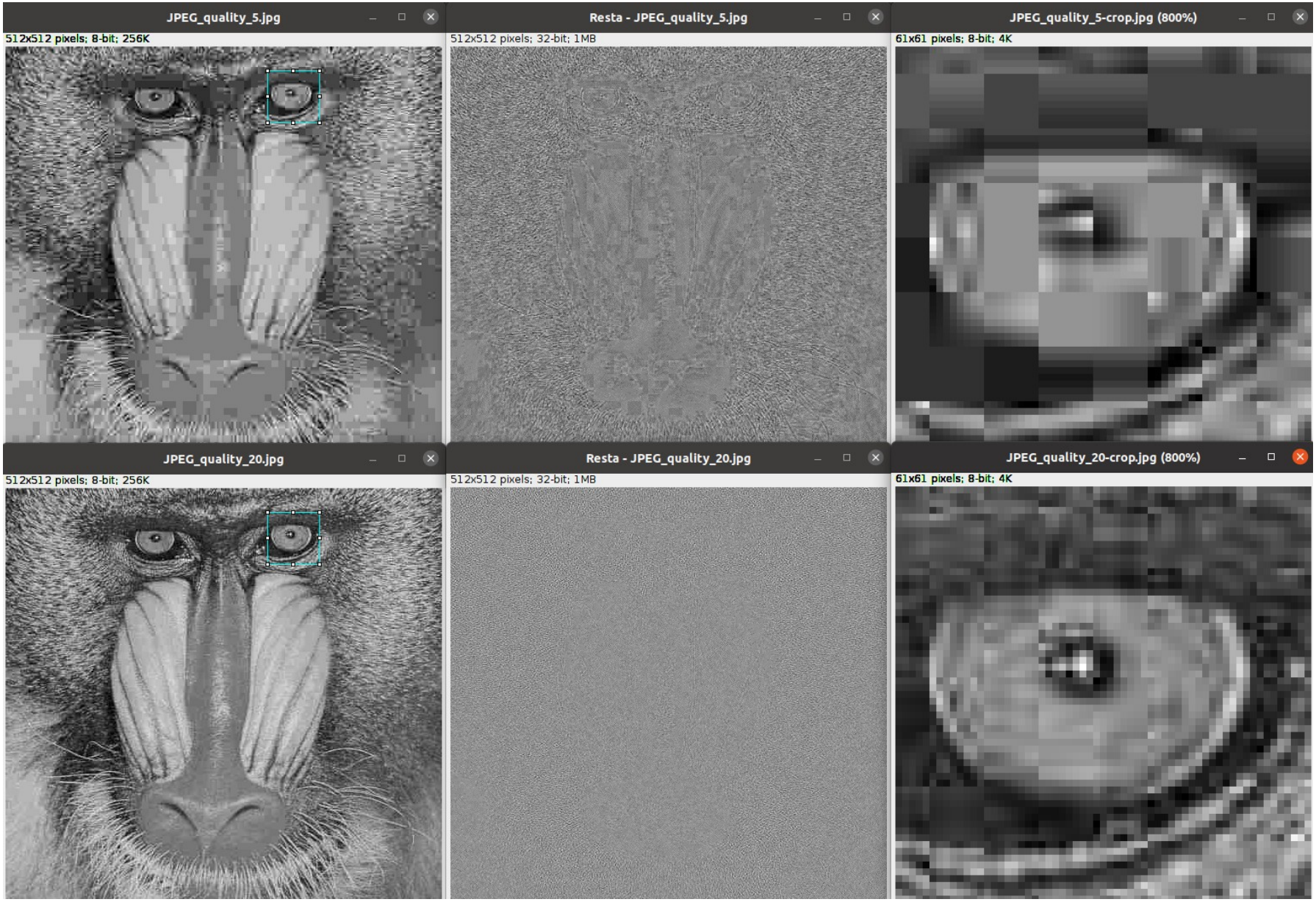
58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

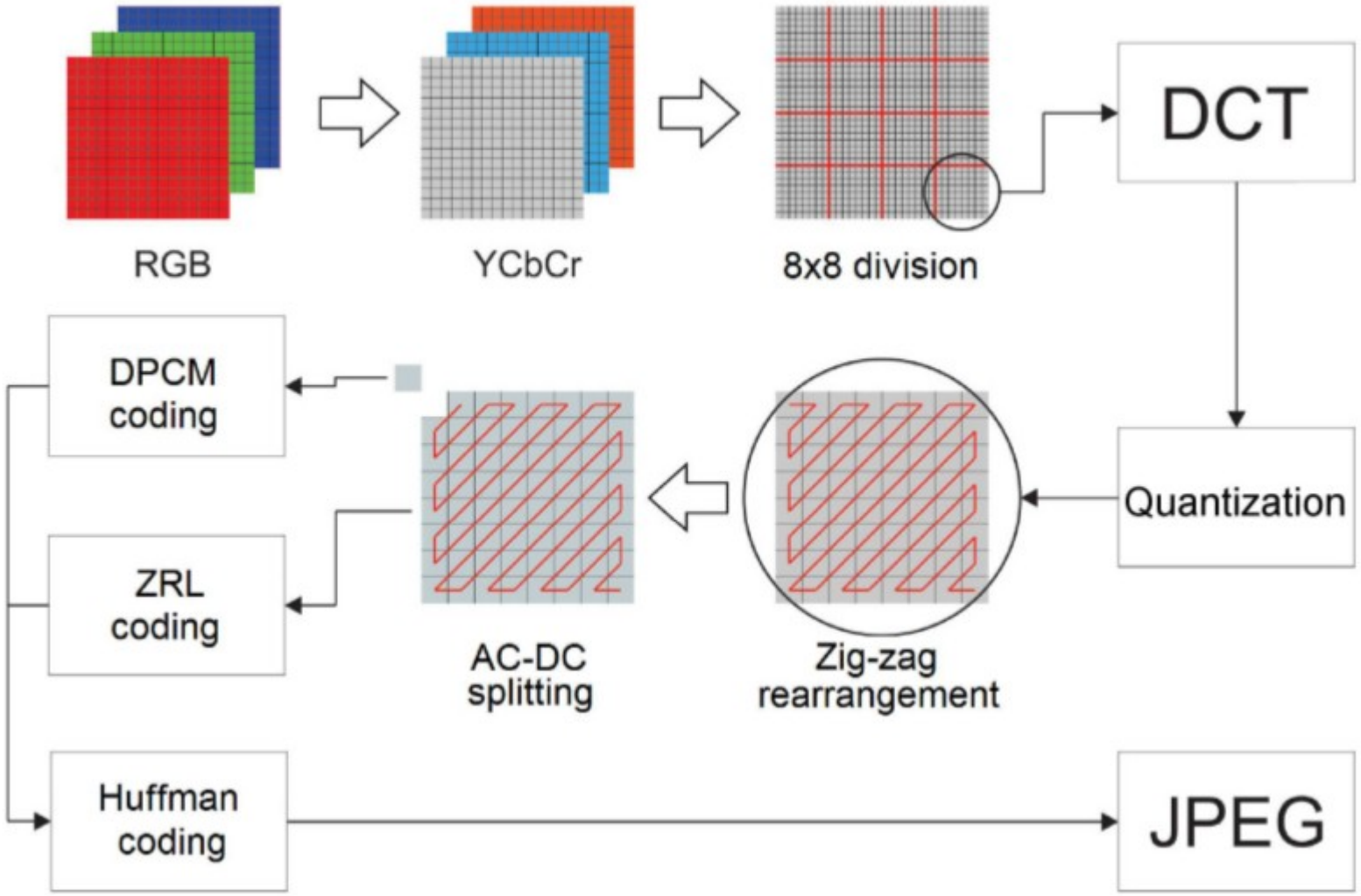
JPEG - Ejemplo

- Diferencia entrel bloque original y el reconstruído

−6	−9	−6	2	11	−1	−6	−5
7	4	−1	1	11	−3	−5	3
2	9	−2	−6	−3	−12	−14	9
−6	7	0	−4	−5	−9	−7	1
−7	8	4	−1	6	4	3	−2
3	8	4	−4	2	6	1	1
2	2	5	−1	−6	0	−2	5
−6	−2	2	6	−4	−4	−6	10

JPEG - Ejemplo




RGB
YCbCr
8x8 division
DCT
Quantization
Zig-zag
rearrangement
AC-DC
splitting
DPCM
coding
ZRL
coding
Huffman
coding
JPEG