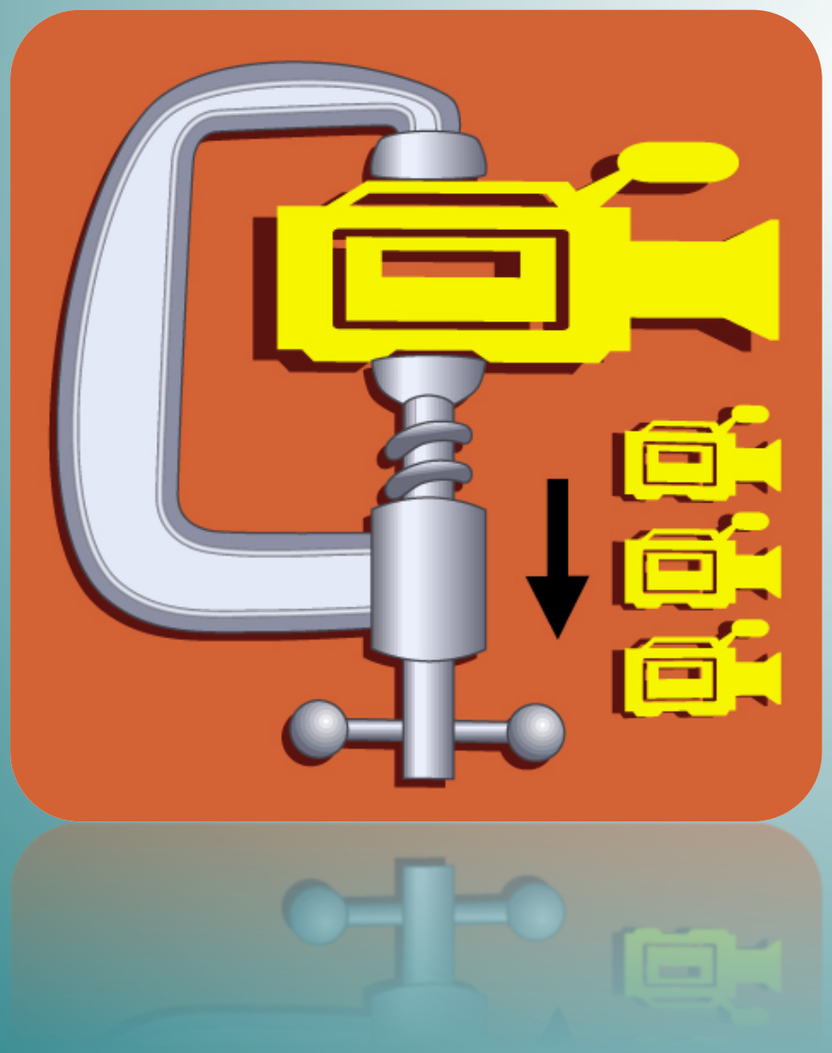
Compresión de Vídeo

Tema 1.4. Codificación por transformada: JPEG



Juan A. Michell Martín
Gustavo A. Ruiz Robredo

Departamento de Electrónica y Computadores

Este tema se publica bajo Licencia:
Creative Commons BY-NC-SA 4.0

TRANSFORMADA DISCRETA DEL COSENO (DCT)

- **Propuesta por Ahmed, Natarajan y Rao en 1974, resulta idónea para codificación de imagen y video.**
- **Es eficaz transformando los datos de imagen en una clase de datos que resulta fácil de comprimir.**
- **Se aplica generalmente a pequeños bloques de muestras de imagen (por ejemplo 8×8).**
- **Puede implementarse eficientemente tanto en *software* y como en *hardware*.**
- **Es el núcleo de estándares actuales de compresión de imagen y vídeo, tales como JPEG, MPEG-2 Vídeo y MPEG-4 Visual.**

TRANSFORMADA DISCRETA DEL COSENO EN DOS DIMENSIONES (2D-DCT)

2D-DCT de un bloque $f(x,y)$ de M×N muestras:

$$F(u,v) = \frac{2 \cdot \alpha(u) \cdot \alpha(v)}{\sqrt{M \cdot N}} \cdot \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot C_{M,N}(u,v,x,y)$$

$$\wedge \quad C_{M,N}(u,v,x,y) = \cos\left[\frac{(2x+1)u\pi}{2M}\right] \cdot \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$\wedge \quad u = 0,1,\ldots,M-1 \quad , \quad v = 0,1,\ldots,N-1$$

Siendo:

$$\alpha(u) = \begin{cases} 1/\sqrt{2}\,, u = 0 \\ 1\,, u > 0 \end{cases} \quad , \quad \alpha(v) = \begin{cases} 1/\sqrt{2}\,, v = 0 \\ 1\,, v > 0 \end{cases}$$

TRANSFORMADA DISCRETA DEL COSENO INVERSA EN DOS DIMENSIONES (2D-IDCT)

2D-IDCT de un bloque $F(u,v)$ de M×N coeficientes:

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\frac{2\cdot\alpha(u)\cdot\alpha(v)}{\sqrt{M\cdot N}}\cdot F(u,v)\cdot C_{M,N}(u,v,x,y)$$

$$\wedge \quad C_{M,N}(u,v,x,y)=\cos\left[\frac{(2x+1)u\pi}{2M}\right]\cdot\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$\wedge \quad x=0,1,\ldots,M-1 \quad , \quad y=0,1,\ldots,N-1$$

Siendo:

$$\alpha(u)=\begin{cases}1/\sqrt{2}\,,\,u=0\\1\,,\,u>0\end{cases} \quad , \quad \alpha(v)=\begin{cases}1/\sqrt{2}\,,\,v=0\\1\,,\,v>0\end{cases}$$

2D-DCT en MATLAB

`X=dctmtx(N)`

X es la matriz del coseno de dimensión NxN.

`F=dct2(f)`

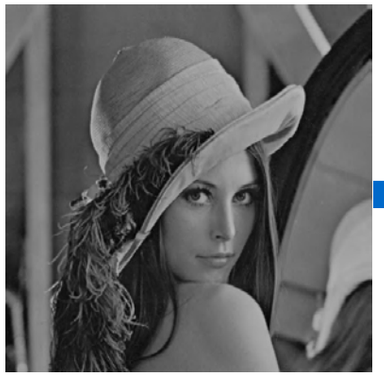
2D-DCT of *f*; *F* es una matriz real del mismo tamaño que *f*.

`f=idct2(F)`

2D-IDCT of *F*; *f* es una matriz real del mismo tamaño que *F*.

Transformada DCT reversible

Imagen Original



f

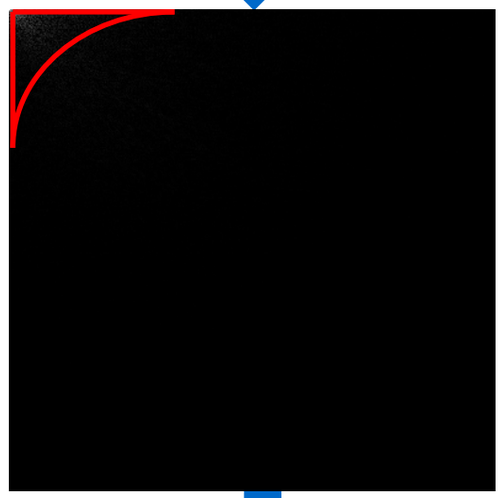
uint8

dct2(*f*)

F

double

Concentración de la energía



IDÉNTICAS

Imagen recuperada

f

uint8

round(f')

f'

double

idct2(*F*)

F

double

2D-DCT PARA BLOQUES 8X8

2D-DCT de un bloque de 8×8 muestras:

$$F(u,v)=\frac{\alpha(u)\cdot\alpha(v)}{4}\cdot\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cdot C_{M,N}(u,v,x,y)$$

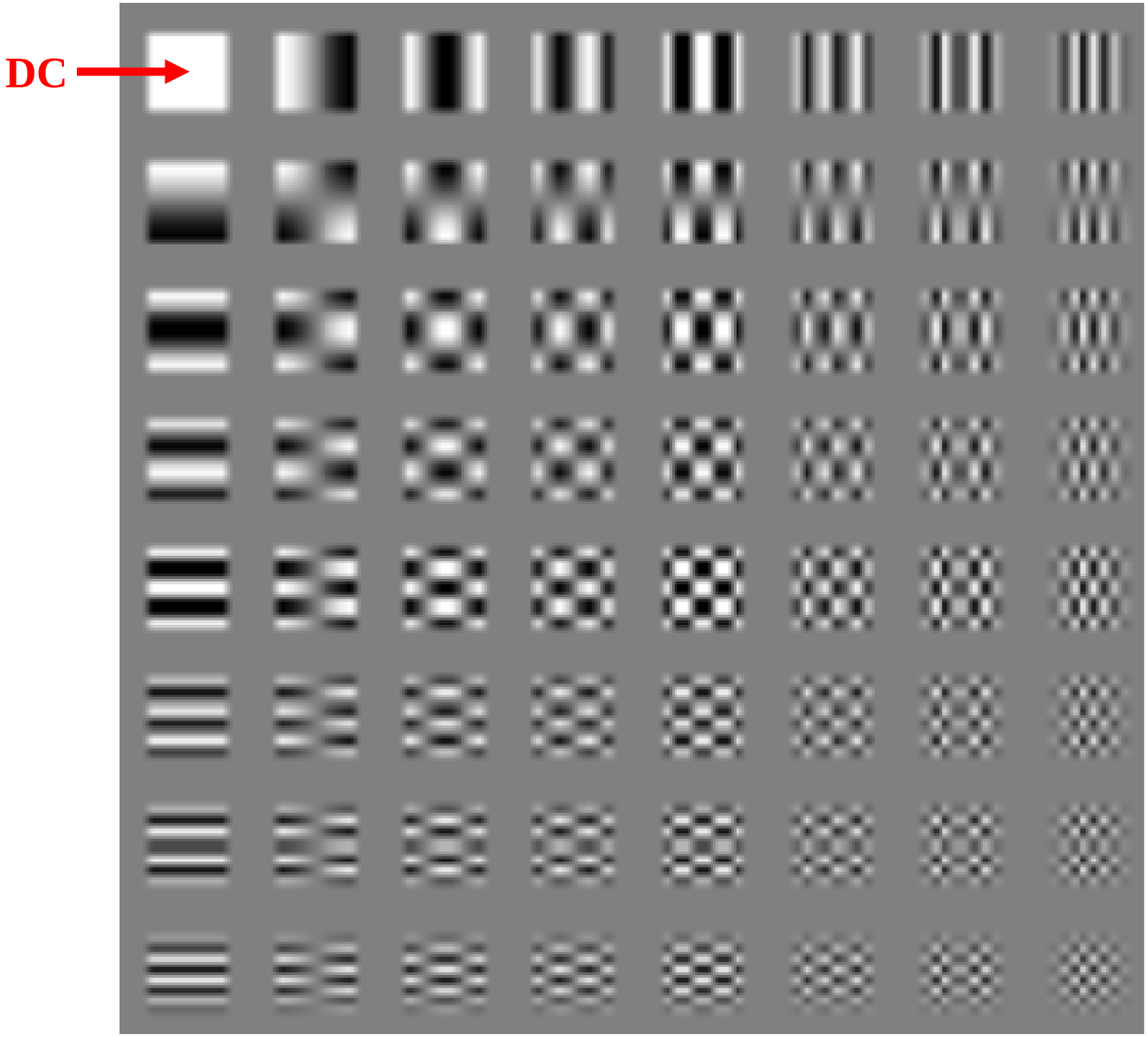
$$\wedge \quad C_{M,N}(u,v,x,y)=\cos\left[\frac{(2x+1)u\pi}{16}\right]\cdot\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

$$\wedge \quad u,v=0,1,\ldots,7$$

Siendo:

$$\alpha(u)=\begin{cases}1/\sqrt{2}\,,\,u=0\\1\,,\,u>0\end{cases}\quad,\quad\alpha(v)=\begin{cases}1/\sqrt{2}\,,\,v=0\\1\,,\,v>0\end{cases}$$

Bases 2D-DCT 8x8



2D-IDCT PARA BLQOUES 8X8

2D-IDCT de un bloque de 8×8 coeficientes:

$$f(x,y)=\sum_{u=0}^{7}\sum_{v=0}^{7}\frac{\alpha(u)\cdot\alpha(v)}{4}\cdot F(u,v)\cdot C_{M,N}(u,v,x,y)$$

$$\wedge \quad C_{M,N}(u,v,x,y)=\cos\left[\frac{(2x+1)u\pi}{16}\right]\cdot\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

$$\wedge \quad x,y=0,1,\ldots,7$$

Siendo:

$$\alpha(u)=\begin{cases}1/\sqrt{2}\,,\,u=0\\1\,,\,u>0\end{cases}\quad,\quad\alpha(v)=\begin{cases}1/\sqrt{2}\,,\,v=0\\1\,,\,v>0\end{cases}$$

FORMULACIÓN MATRICIAL DE LA 2D-DCT

2D-DCT de un bloque f de N×N muestras:

$$F = T_N \cdot f \cdot T_N^{-1} \quad \Leftrightarrow \quad f = T_N^{-1} \cdot F \cdot T_N$$

$$T_N = \left[t_{k,n} \right] \quad \wedge \quad t_{k,n} = \frac{\sqrt{2} \cdot \alpha(k)}{\sqrt{N}} \cdot \cos\left[\frac{(2n+1)k\pi}{2N} \right]$$

$$\wedge \quad \begin{cases} n = 0,1,\ldots,N-1 \\ k = 0,1,\ldots,N-1 \end{cases}, \quad \alpha(k) = \begin{cases} 1/\sqrt{2}, k = 0 \\ 1, k > 0 \end{cases}$$

$$T \text{ es ortogonal} \quad \Rightarrow \quad T_N^{-1} = T_N^t$$

$$F = T_N \cdot f \cdot T_N^t \quad \Leftrightarrow \quad f = T_N^t \cdot F \cdot T_N$$

Formulación matricial de la 2D-DCT 8x8

dctmtx(8)

$$T_8 = \begin{bmatrix} g & g & g & g & g & g & g & g \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ b & -d & -a & -c & c & a & d & -b \\ h & -h & -h & h & h & -h & -h & h \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{bmatrix}$$

$$g = \frac{1}{2\sqrt{2}}, h = \frac{\cos(\pi/4)}{2}, e = \frac{\cos(\pi/8)}{2}, f = \frac{\cos(3\pi/8)}{2}$$

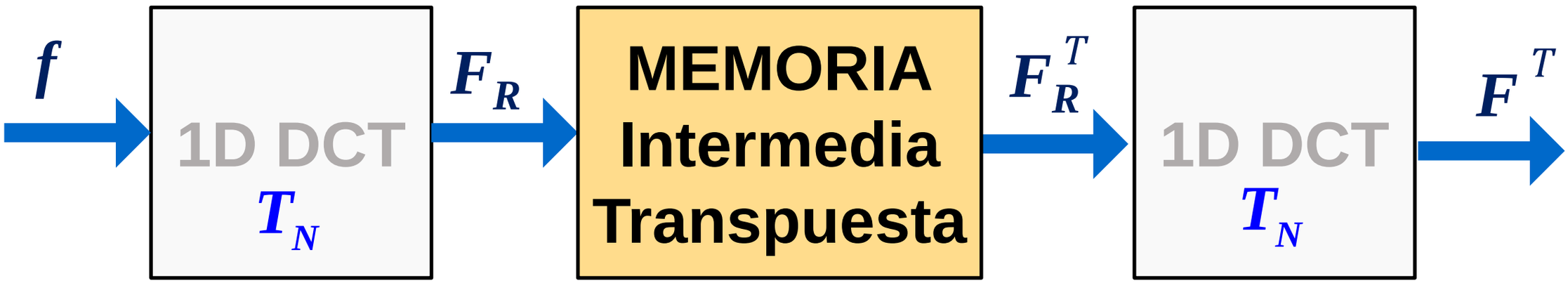
$$a = \frac{\cos(\pi/16)}{2}, b = \frac{\cos(3\pi/16)}{2}, c = \frac{\cos(5\pi/16)}{2}, d = \frac{\cos(7\pi/16)}{2}$$

Implementación Hardware

$$F = T_N \cdot F \cdot T_N^{-1}$$

$$F_R = T_N \cdot F$$

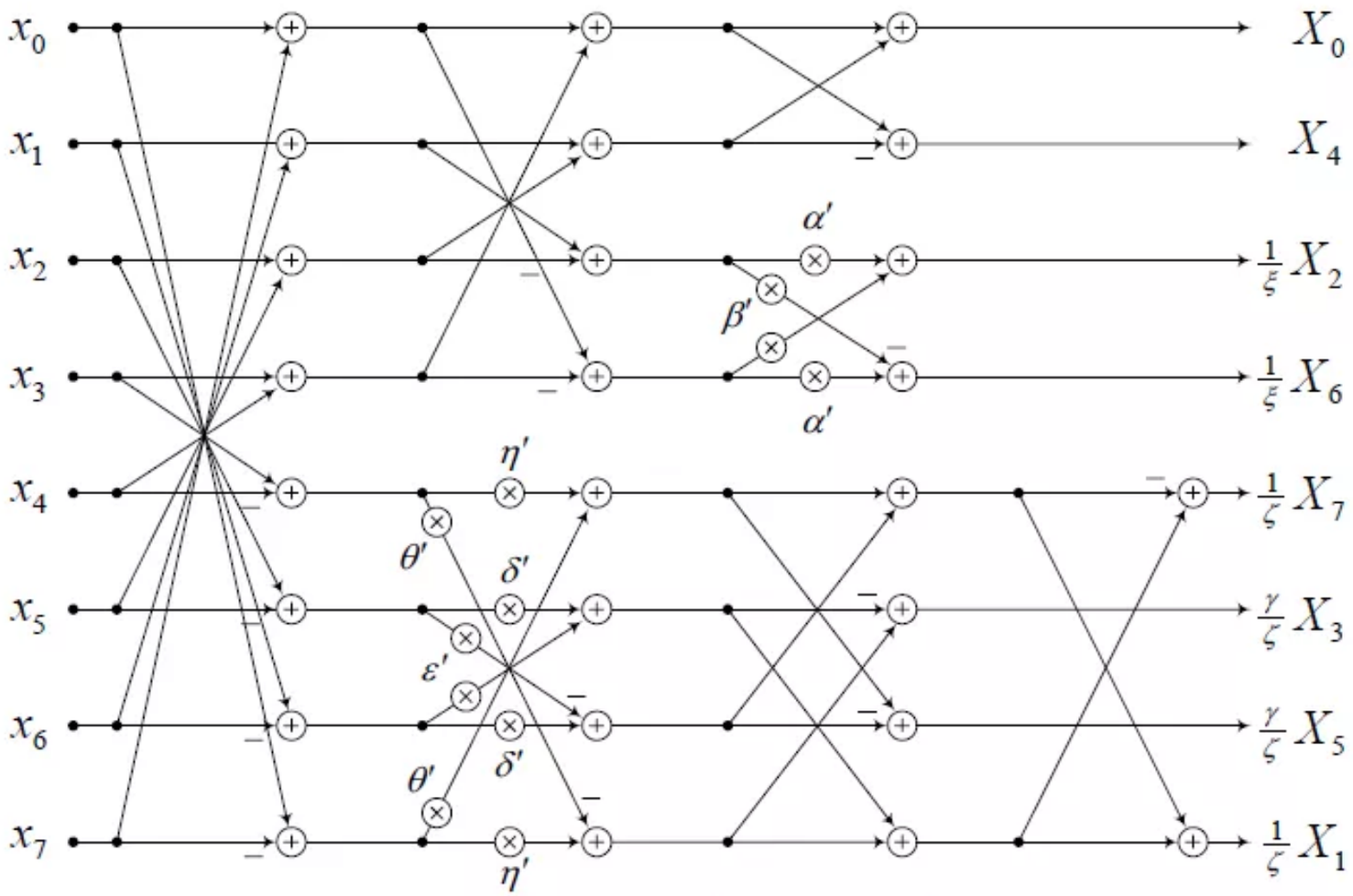
$$F = F_R T_N^{-1} \Rightarrow F^T = \left(T_N F_R^T \right)$$



FAST 1D DCT butterfly

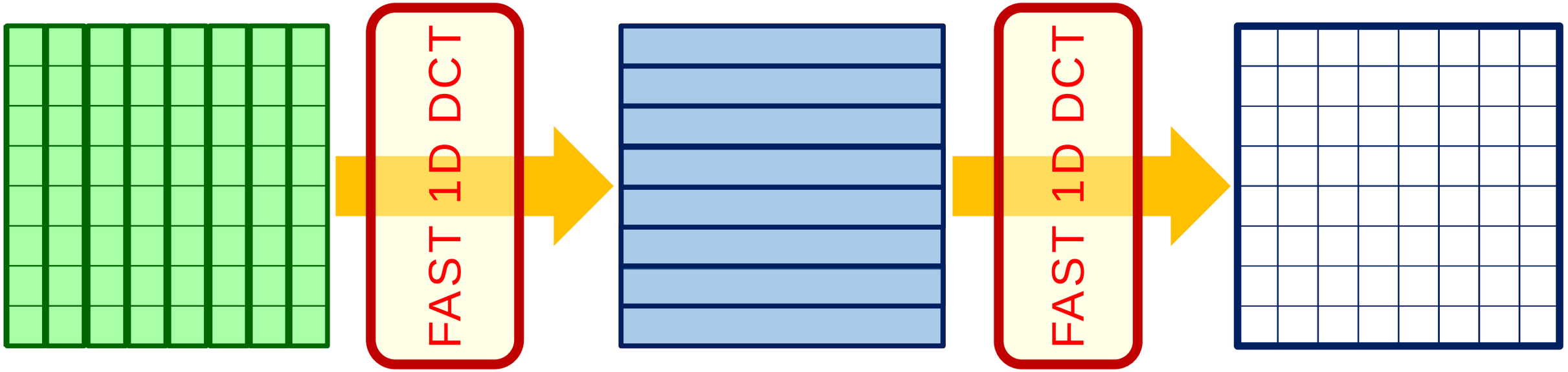
Algoritmo matricial: 64 multiplicaciones + 64 sumas.

Algoritmo rápido: 18 multiplicaciones +24 sumas. Solo válido para 2^n.



1D DCT algoritmo rápido (butterfly)

Cómputo de la 2D-DCT vía FAST 1D-DCT



COMPRESIÓN DE IMAGEN: CONCEPTOS BÁSICOS

Las técnicas de compresión surgen por la necesidad de reducción de espacio de almacenamiento y acortar el proceso de transmisión. Su objetivo es eliminar redundancias de información.

Métodos de compresión :

- **Con pérdidas (lossy): Cuando la degradación de la calidad de la imagen es aceptable.**
- **Sin pérdidas (lossless): La imagen es codificada sin pérdida de calidad.**

Técnicas de compresión sin perdidas (Lossless)

- **VCL (Error-Free Compression using Variable Length Coding).**

  Utiliza símbolos de longitud variable. Por ejemplo en esta tabla se asigna un conjunto de bits dependiendo de la frecuencia de aparición.

Símbolo	Código
a	0
b	10
c	110
d	111

aabacdab → 00100110111010

- **RLE (Run-Length Encoding).**

  Consiste en separar una secuencia por una pareja que representa un símbolo y su repetición.

BBBBBBBBBBBBNBBBBBBBBBBBBBNNNBBBBBBBBBBB
BBBBBBBBBBBBBNBBBBBBBBBBBBBB

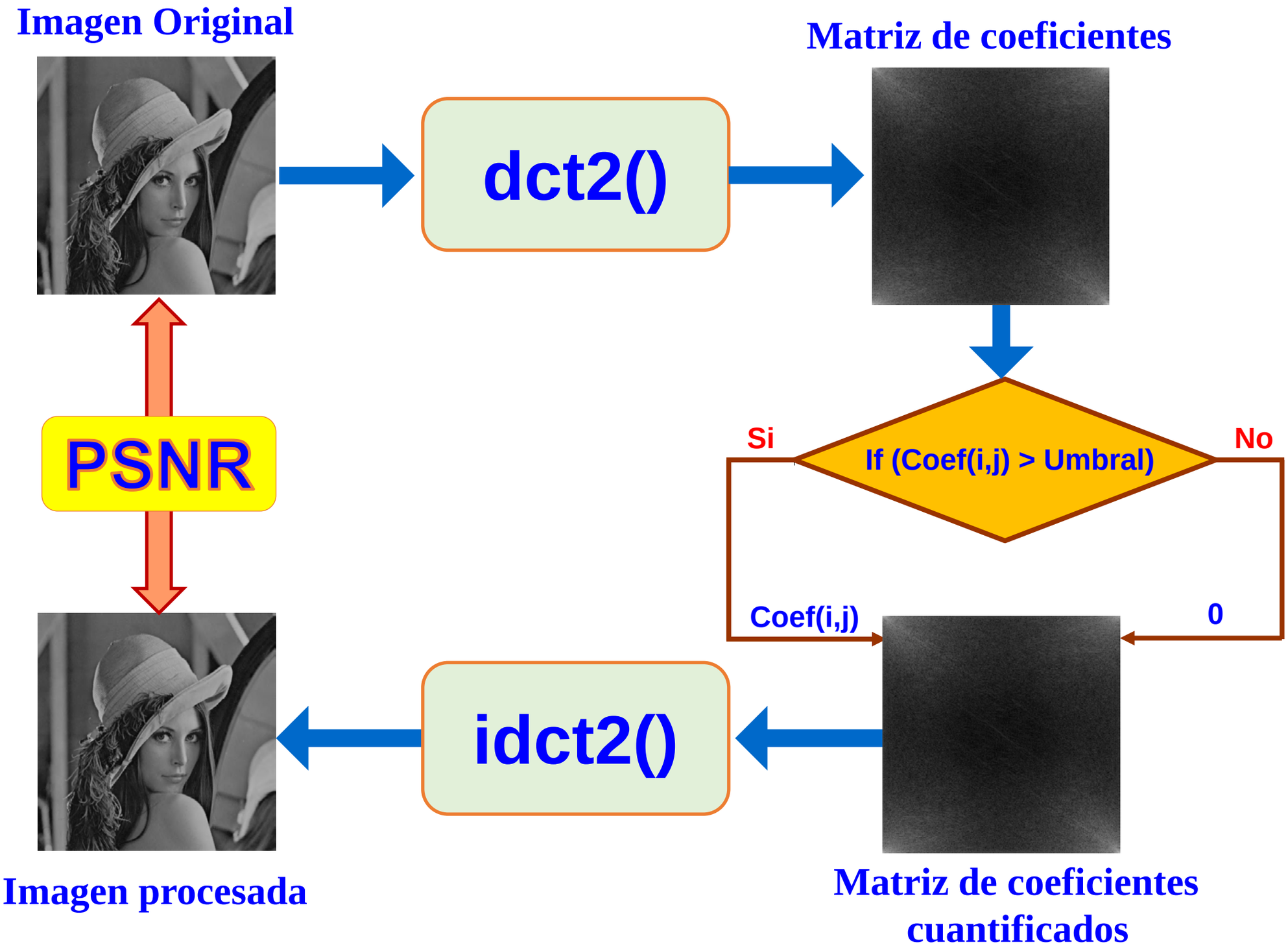
12B1N12B3N24B1N14B

Técnicas de compresión con pérdidas (Lossy)

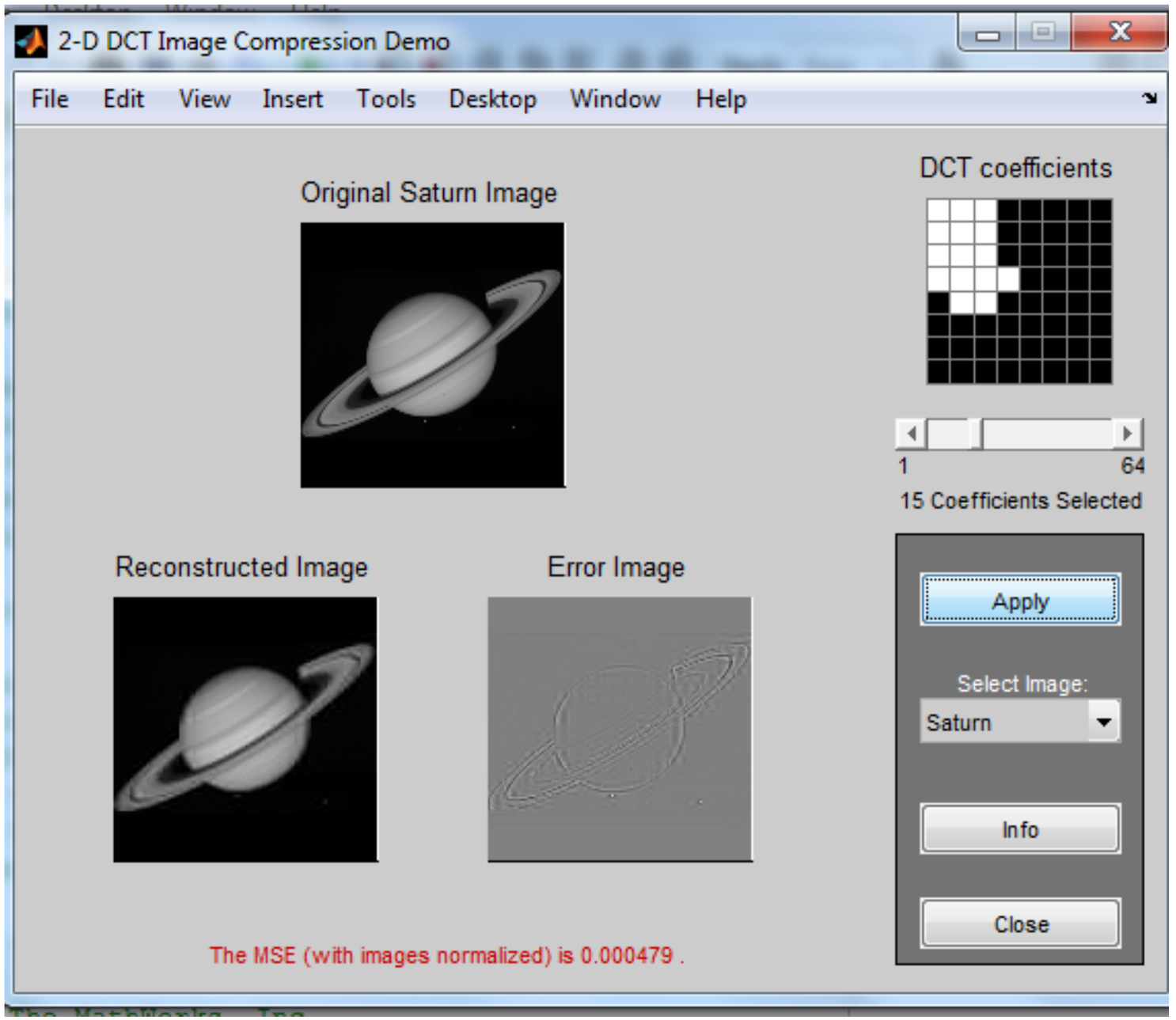
- **Codificación por transformada: Son técnicas que usan una transformada reversible (DCT, FFT, wavelet) que convierten la imagen en un conjunto de coeficientes que son cuantificados y codificados. Este proceso se suele realizar sobre la imagen subdividida en bloques de menor tamaño, típicamente, 8x8, 16x16, 32x32, ...**

- **Cuantificación: Asigna a cada coeficiente de la transforma un valor de peso específico.**

Proceso de cuantificación simple

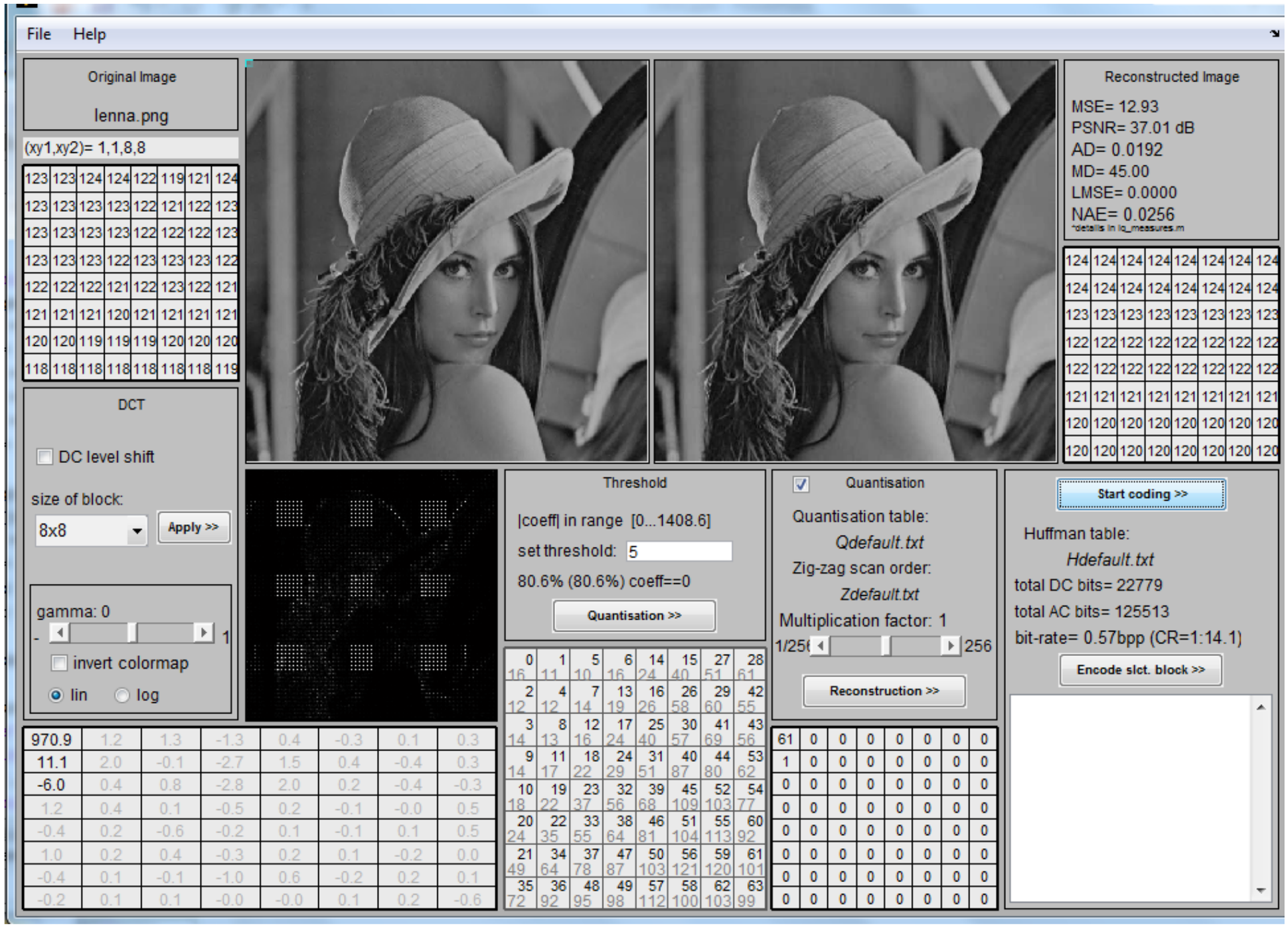


Aplicación dctdemo.m



Copyright 1993-2002 The MathWorks

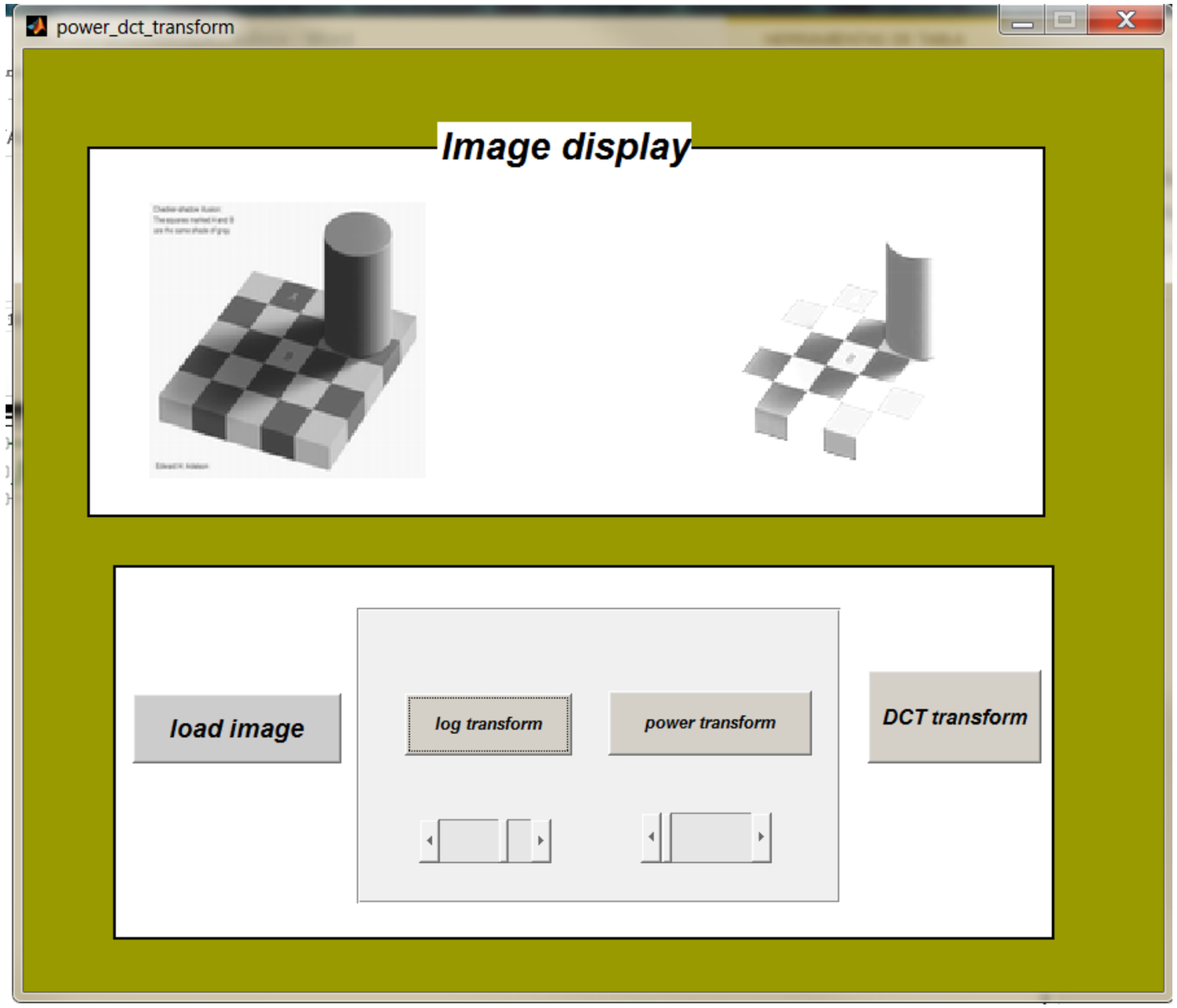
Aplicación DCTlab.m



https://es.mathworks.com/matlabcentral/fileexchange/36421-dctlab-gui?focused=5229681&tab=function

Aplicación power_dct_transform.m

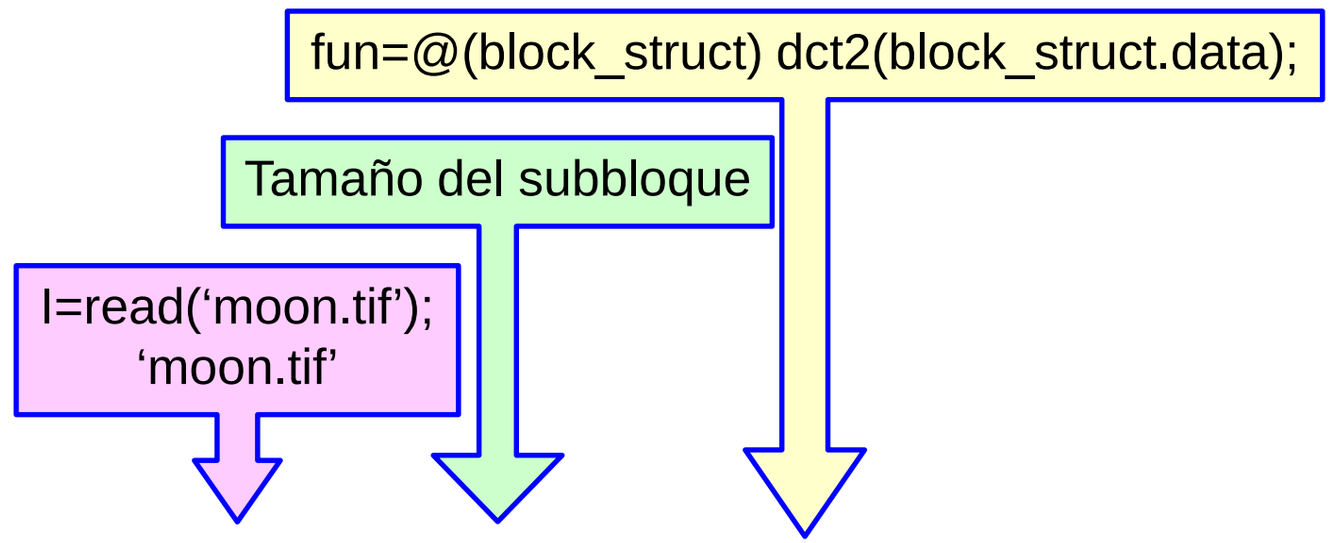
Esta aplicación permite cambar las escala de grises de una imagen aumentando (*log transform*) o disminuyendo (*power transform*) su luminosidad.



https://es.mathworks.com/matlabcentral/fileexchange/49095-a-gui-based-log--power-and-dct-tranformer?focused=3859477&tab=function

MATLAB: blockproc

blockproc(I, [M N], fun): Procesa cada uno de los subbloques de dimensión MxN de una imagen I aplicando una función fun previamente definida.

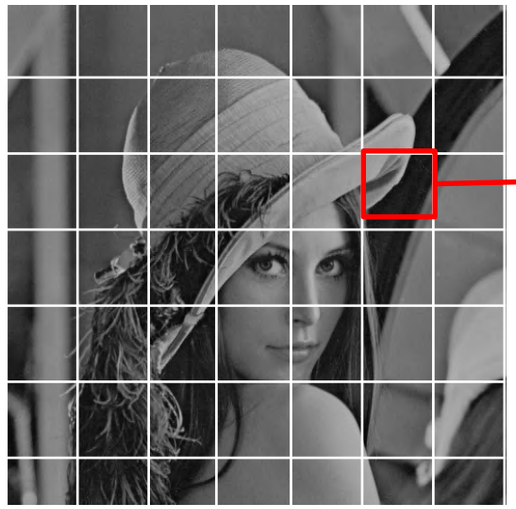


Ejemplo de blockproc

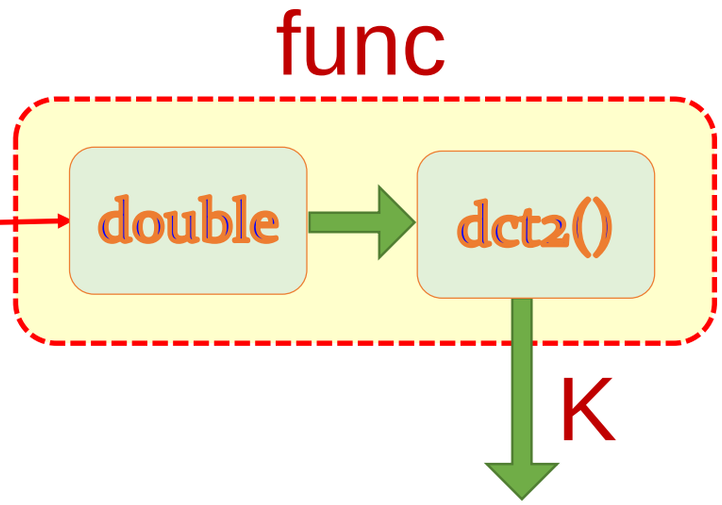
```
func=@(block_struct) dct2(double(block_struct.data));
K=blockproc( ‘lenna.tif’ , [8 8] , func);
```

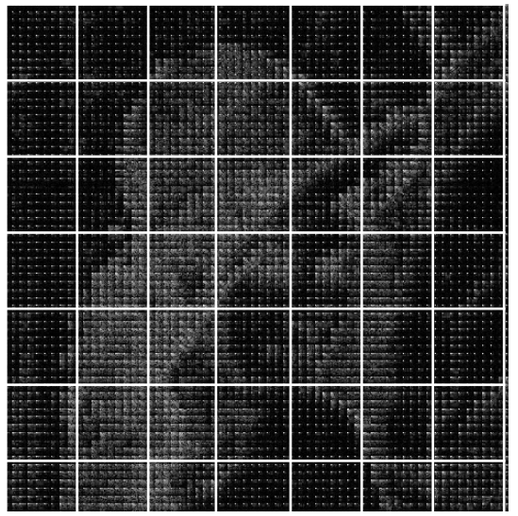
‘lenna.tif’

Imagen Original (uint8)



Partición en bloques 8x8 (blockproc)





Coeficientes 2D DCT

Otros ejemplos

```
I = imread('lenna.tif');

fun = @(block_struct) (imresize(block_struct.data,0.15));
K = blockproc(I,[100 100],fun);

figure, imshow(I),title('IMAGEN ORIGINAL');
figure, imshow(K), title('IMAGEN PROCESADA');
```

```
I = imread('peppers.png');

fun = @(block_struct) (block_struct.data(:,:,[2 1 3]));
blockproc(I, [200 200], fun,'Destination','grb_peppers.tif');

subplot(1,2,1), imshow(I), title('IMAGEN ORIGINAL');
subplot(1,2,2), imshow(imread('grb_peppers.tif')), title('IMAGEN PROCESADA');
```

```
I = imread('lenna.tif');

fun = @(block_struct) (transpose(block_struct.data)) ;
K=blockproc(I, [32 32], fun);

subplot(1,2,1), imshow(I), title('IMAGEN ORIGINAL');
subplot(1,2,2), imshow(K), title('IMAGEN PROCESADA');
```

COMPRESIÓN JPEG

- El formato JPEG fue originalmente publicado como standard ISO IS 10918-1 por el Joint Photographic Experts Group (JPEG Group) en 1984.

- El codificador baseline JPEG incluye básicamente los siguientes pasos:

  1. Conversión RGB a YCbCr 4:2:0. Las cromas son submustreadas.
  2. Imágenes divididas en bloques de 8x8.
  3. Cada bloque es procesado con la 2D DCT.
  4. Cada coeficiente es cuantificado de acuerdo a unas tablas para limitar aceptables niveles de pérdida de calidad.
  5. Los coeficientes son escaneados en orden zig-zag y codificados en run-length.
  6. El código run-length es convertido a un código binario de longitud variable usando la codificación de Huffman.

- El proceso de decodificación es reversible y realiza los pasos anteriores en orden inverso.

CODEC base

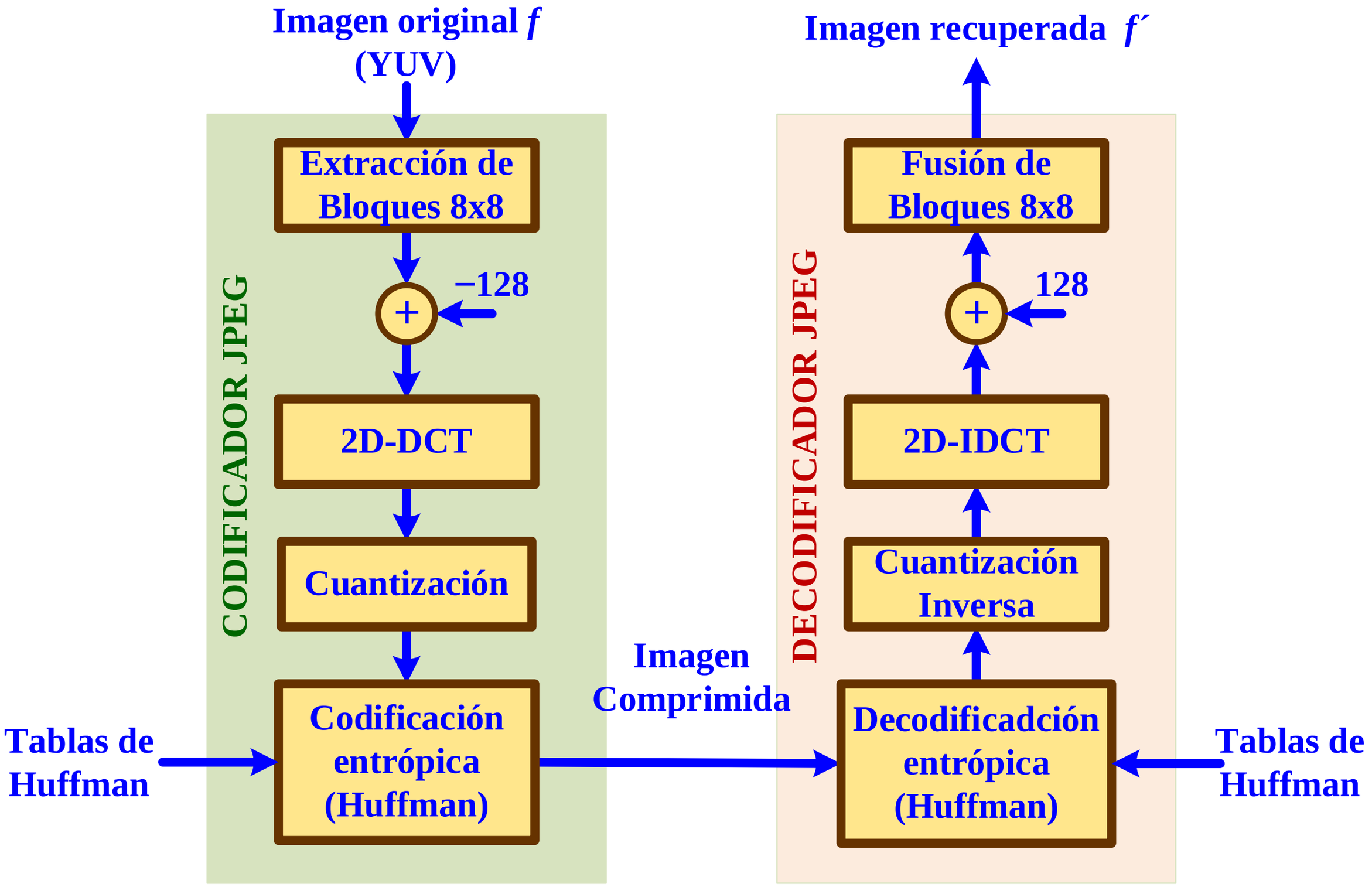
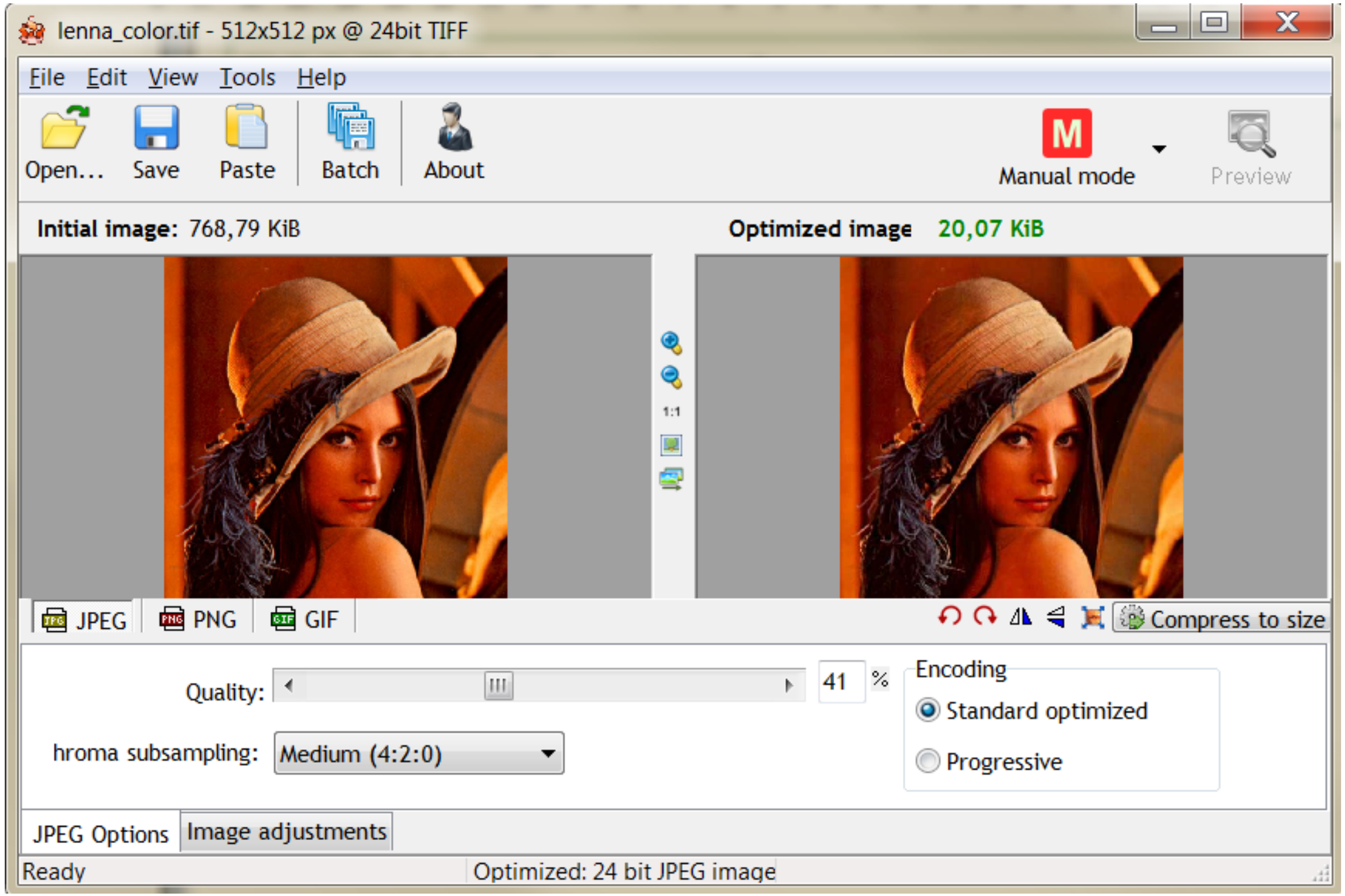


Imagen original f
(YUV)
CODIFICADOR JPEG
Extracción de Bloques 8x8
−128
2D-DCT
Cuantización
Codificación entrópica (Huffman)
Tablas de Huffman
Imagen Comprimida
DECODIFICADOR JPEG
Fusión de Bloques 8x8
128
2D-IDCT
Cuantización Inversa
Decodificadción entrópica (Huffman)
Tablas de Huffman
Imagen recuperada f´

RIOT Windows tool



http://luci.criosweb.ro/riot/

JPEG 2000

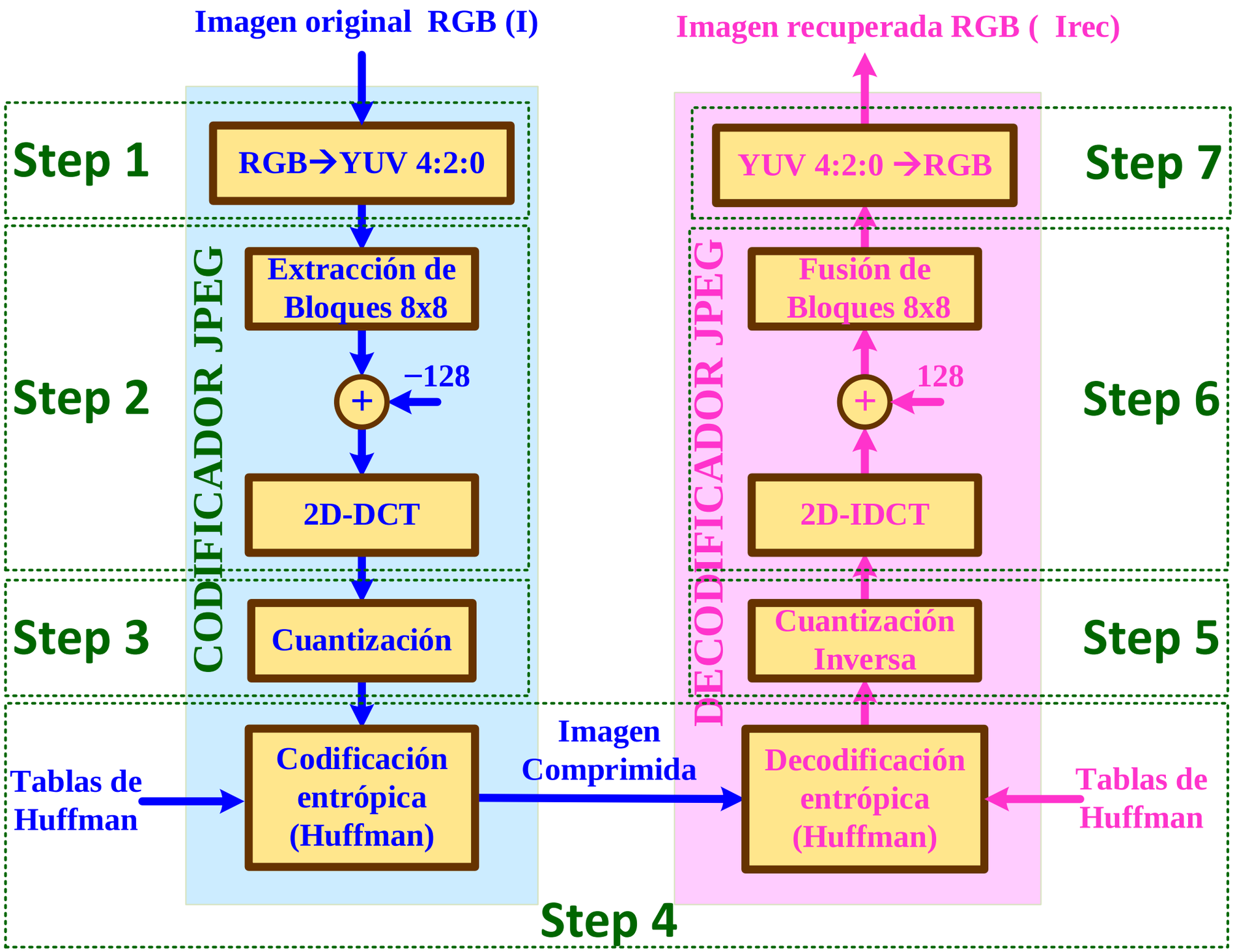
- **El formato JPEG 2000 es un compresor basado en la transformada wavelet (ISO/IEC 15444-1:2000) creado por el JPEG Group. Se identifica en los ficheros por la extensión .jpg2.**

- **Ventajas:**
  1. **Compresión a bajo bit-rate.**
  2. **Superior lossless/lossy tasas de compresión.**
  3. **Capacidad para manejar imágenes de gran tamaño.**
  4. **Arquitectura de descompresión simple.**
  5. **Resistente a errores de transmisión.**
  6. **Codificación de región de interés (ROI).**

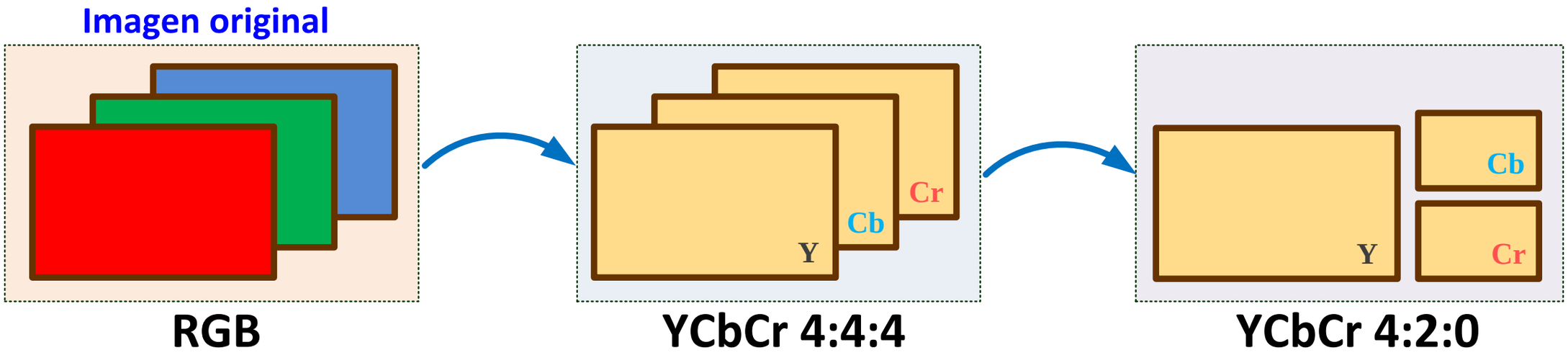
JPEG-LS

- **Compresor (ISO/IEC-14495 1/ITU-T) concebido para lossless/near lossless aplicaciones.**

DESCRIPCIÓN DE UN JPEG BÁSICO



Step 1. Conversión RGB → YCbCr 4:2:0



RGB → YCbCr 4:4:4

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

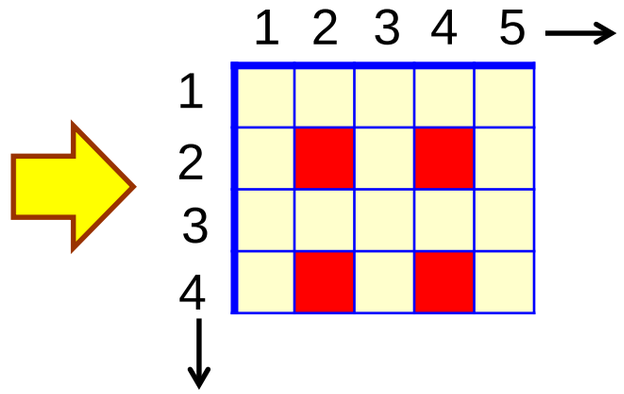
RGB→ YCbCr 4:4:4

```
RGB=[ 0.299 0.587 0.114; -0.1687 -0.3313 0.5; 0.5 -0.4187 -0.0813];
for x=1:size(I,1)
    for   y=1:size(I,2)
        Ycbcr(x,y,1:3)=RGB*double( [I(x,y,1); I(x,y,2); I(x,y,3)])+ [0; 128; 128];
    end
end
Y0=Ycbcr(:,:,1); %luma
Cb=Ycbcr(:,:,2); % Croma Cb
Cr=Ycbcr(:,:,3); % Croma Cr
```

YCbCr 4:4:4 → YCbCr 4:2:0

```
% Definición filtro 'average' 2x2
h = fspecial('average',[2 2]);

% Filtrado de las cromas
Cbh = imfilter(Cb,h,'replicate'); % Filtrado Cb
Crh = imfilter(Cr,h,'replicate'); % Filtrado Cr

% Subsampling 4:2:0
Cb0=cbh(2:2:size(Cbh,1),2:2:size(Cbh,2));
Cr0=crh(2:2:size(Crh,1),2:2:size(Crh,2));
```

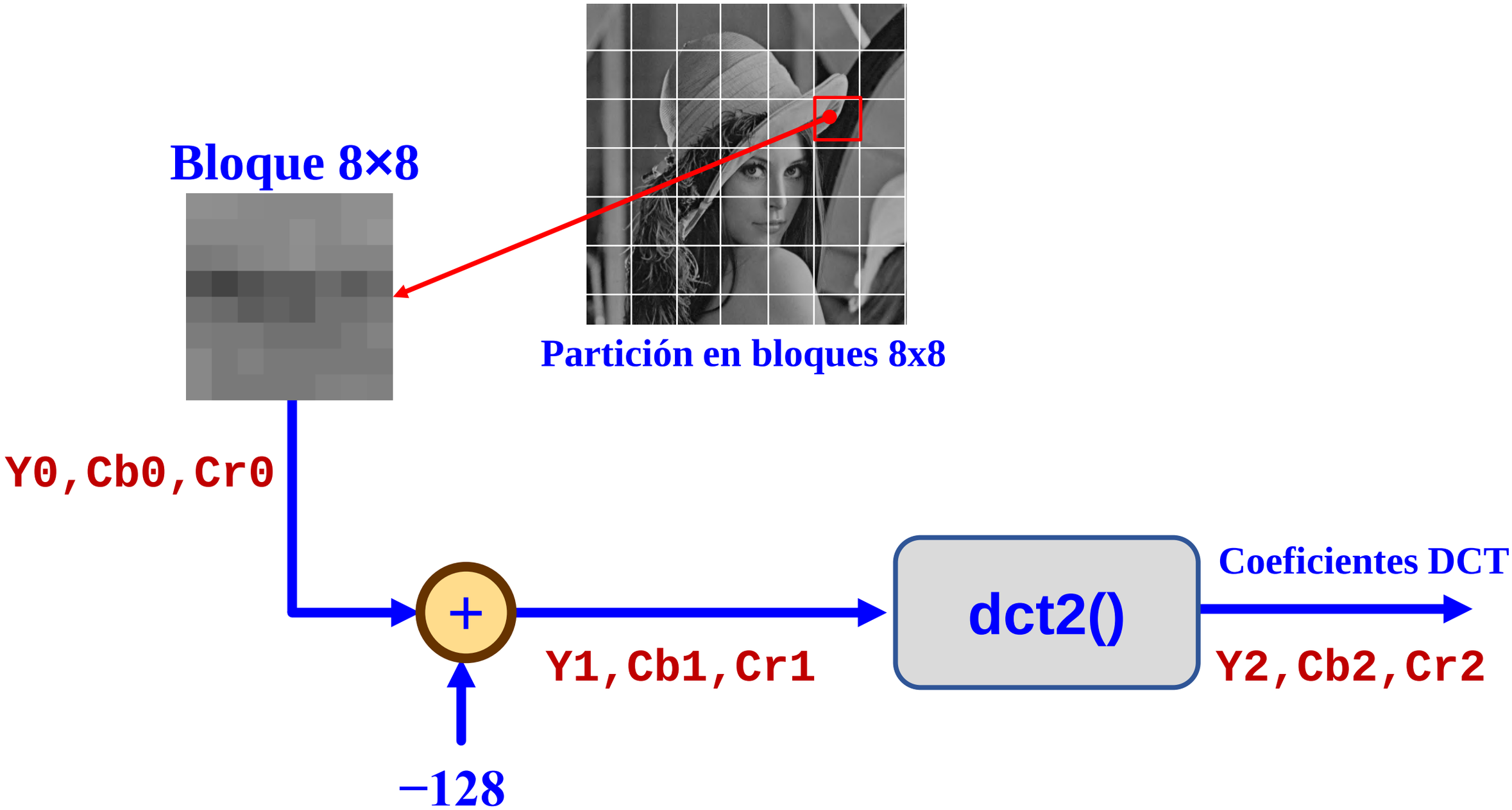
h= 1/4 1/4 1/4 1/4 Filtro ‘average’



Cbh,Crh

Cb0,Cr0

Step 2. 2D DCT


Bloque 8×8
Partición en bloques 8x8
Y0,Cb0,Cr0
+
−128
Y1,Cb1,Cr1
dct2()
Coeficientes DCT
Y2,Cb2,Cr2

```
%% 2D DCT para bloques 8x8

% Offset -128
Y1=int16(Y0)-128;
Cb1=int16(Cb0)-128;
Cr1=int16(Cr0)-128;

% 2D DCT
dct=@(block_struct) dct2( block_struct.data);
Y2=blockproc(Y1,   [8 8], dct);
Cb2=blockproc(Cb1, [8 8], dct);
Cr2=blockproc(Cr1, [8 8], dct);
```

Ejemplo de 2D DCT

Bloque 8×8

143	142	137	136	136	136	143	143
136	136	136	136	143	136	143	149
121	124	132	136	142	132	132	132
82	67	82	92	92	106	92	106
113	106	92	98	92	113	113	121
124	121	121	113	113	113	121	130
136	121	128	121	121	121	121	121
136	121	121	121	121	128	130	128

−128

15	14	9	8	8	8	15	15
8	8	8	8	15	8	15	21
-7	-4	4	8	14	4	4	4
-46	-61	-46	-36	-36	-22	-36	-22
-15	-22	-36	-30	-36	-15	-15	-7
-4	-7	-7	-15	-15	-15	-7	2
8	-7	0	-7	-7	-7	-7	-7
8	-7	-7	-7	-7	0	2	0

dct2()

-48,88	-18,92	17,57	3,16	8,38	3,78	7,47	0,18
42,33	-11,23	-11,75	-5,24	1,24	-5,98	-5,99	0,92
97,27	19,81	1,59	0,27	-2,70	1,57	-6,59	8,14
8,06	11,42	20,33	-4,41	-4,72	-1,54	-3,80	-1,93
-52,88	-11,36	9,73	9,19	-0,63	4,31	4,22	-10,65
-26,55	2,10	-13,56	0,19	2,46	2,99	7,12	2,49
21,54	3,14	-3,84	-4,53	-2,27	-1,66	-4,59	2,48
27,85	10,15	1,88	3,41	-4,00	0,59	-5,46	-1,85

Coeficientes 2D DCT

Step 3. Cuantificación

Tablas de cuantificación tabuladas

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

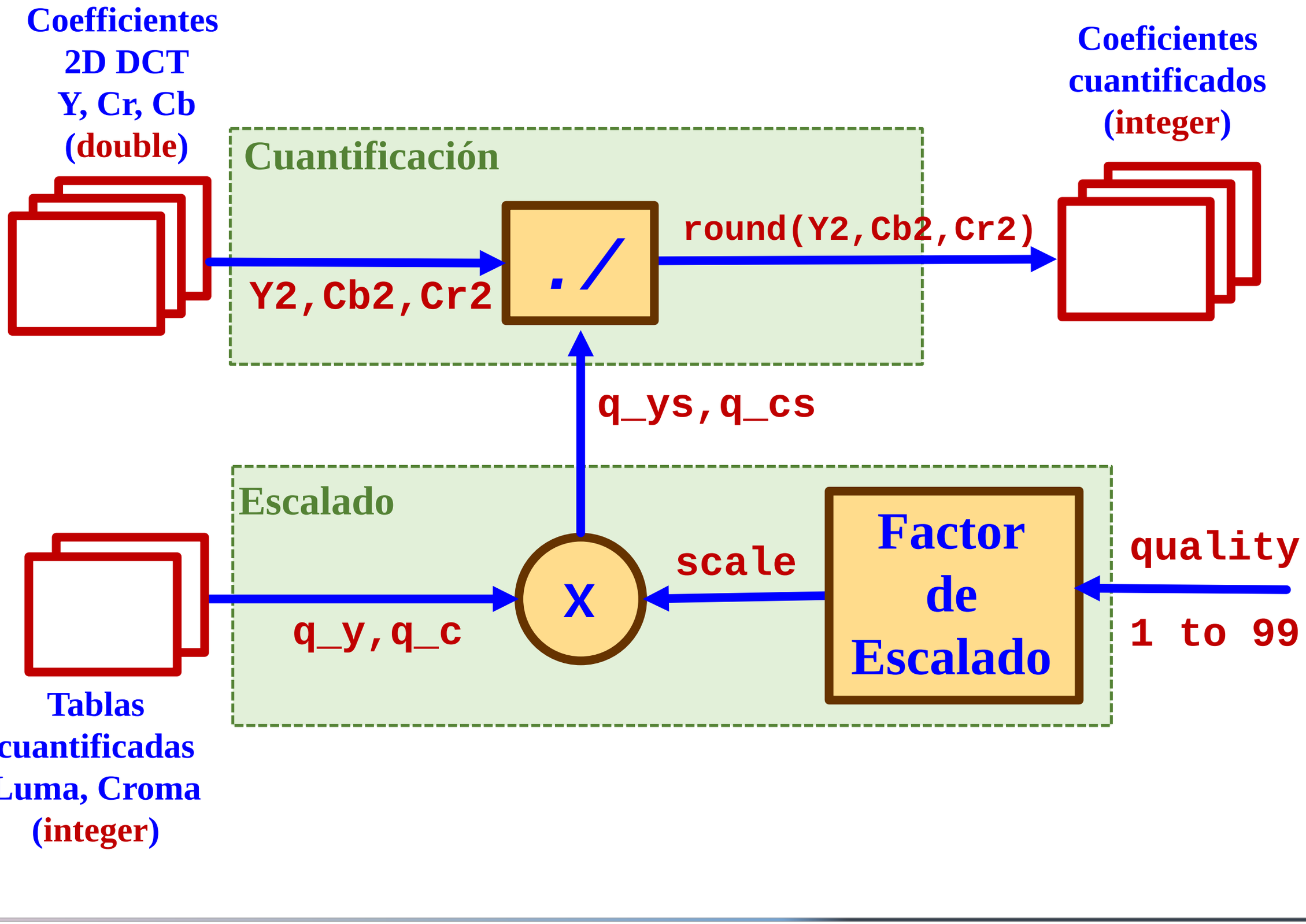
LUMA

q_y

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

CROMAS

q_c


Coefficientes
2D DCT
Y, Cr, Cb
(double)
Cuantificación
Y2,Cb2,Cr2
./
round(Y2,Cb2,Cr2)
Coeficientes
cuantificados
(integer)
q_ys,q_cs
Escalado
q_y,q_c
X
scale
Factor
de
Escalado
quality
1 to 99
Tablas
cuantificadas
Luma, Croma
(integer)

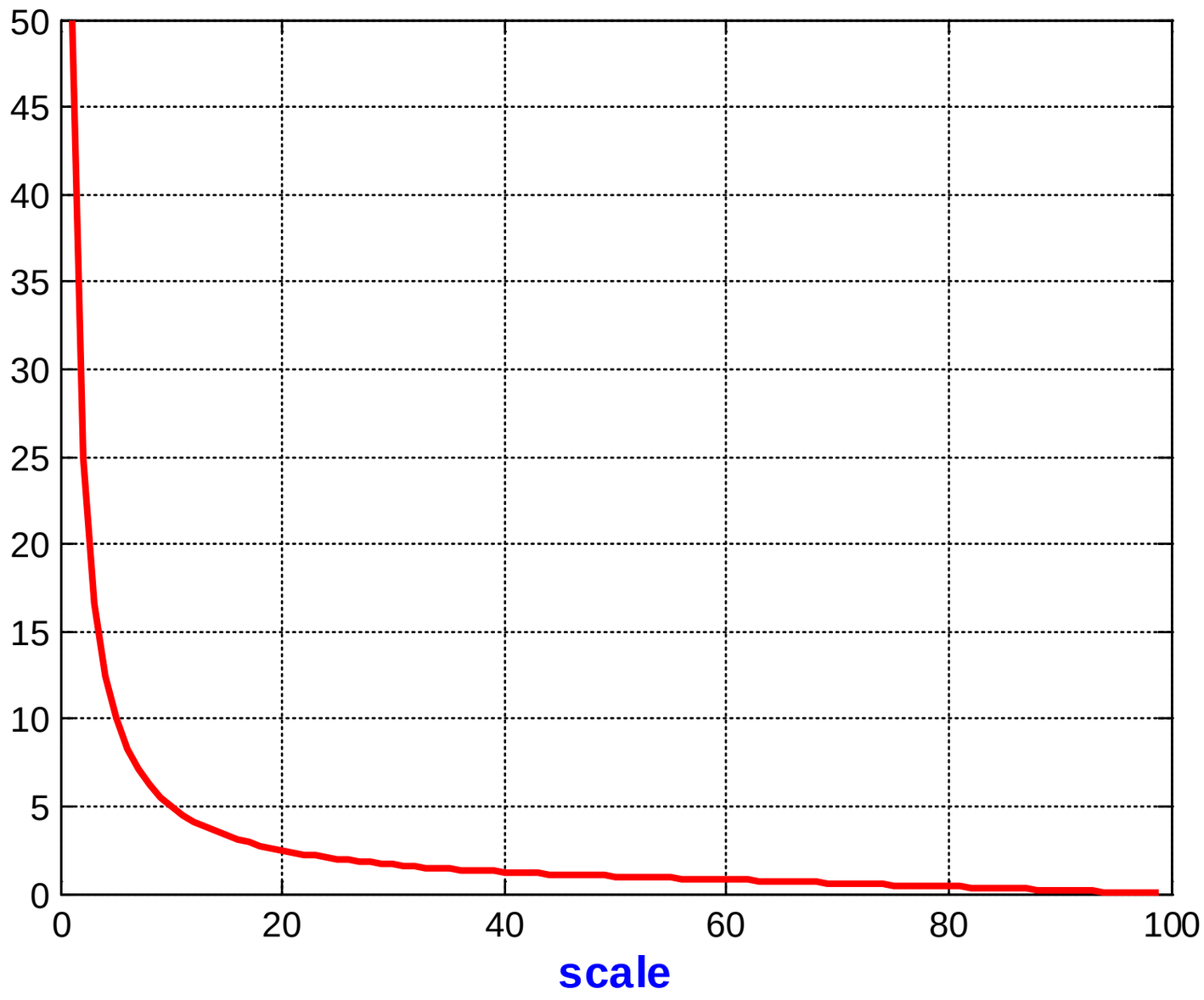
```
% Calculo del factor de escalado
if( quality <= 50)
    scale=50/quality;
else
    scale=2-2*quality/100;
end

% Escalado
q_ys=q_y*scale;
q_cs=q_c*scale;
```

Escalado

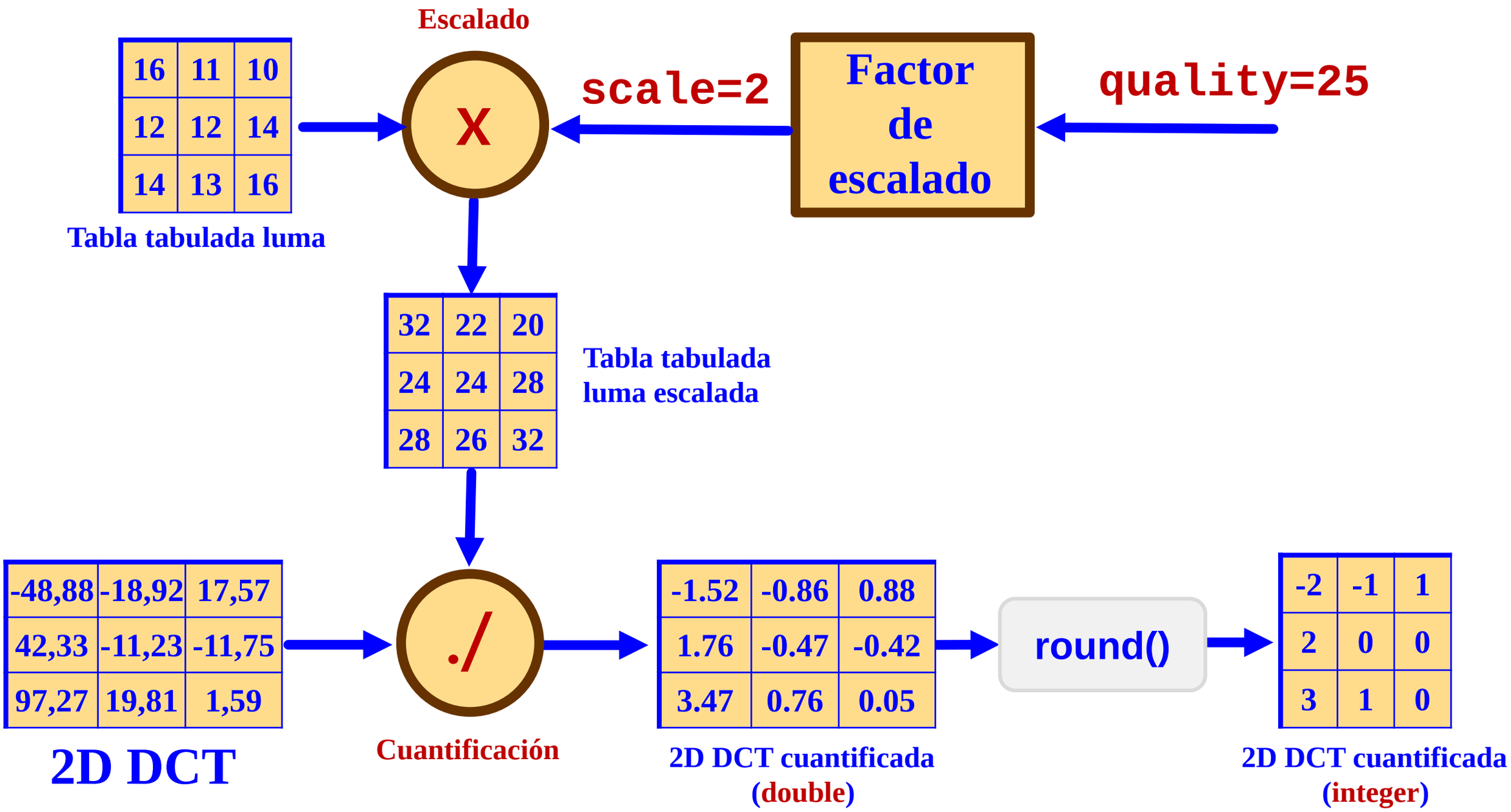
```
% Cuantificación
Y3=blockproc (Y2,   [8 8], @(block_struct) round(round(
block_struct.data ./ q_ys)) );
Cb3=blockproc (Cb2, [8 8], @(block_struct) round(round(
block_struct.data ./ q_cs)) );
Cr3=blockproc (Cr2, [8 8], @(block_struct) round(round(
block_struct.data ./ q_cs)) );
```

Cuantificación

```
if( quality <= 50)
    scale=50/quality;
else
    scale=2-2*quality/100;
end
```



Ejemplo simple de la cuantificación luma (3x3)



Ejemplo simple de la cuantificación luma

-48,88	-18,92	17,57	3,16	8,38	3,78	7,47	0,18
42,33	-11,23	-11,75	-5,24	1,24	-5,98	-5,99	0,92
97,27	19,81	1,59	0,27	-2,70	1,57	-6,59	8,14
8,06	11,42	20,33	-4,41	-4,72	-1,54	-3,80	-1,93
-52,88	-11,36	9,73	9,19	-0,63	4,31	4,22	-10,65
-26,55	2,10	-13,56	0,19	2,46	2,99	7,12	2,49
21,54	3,14	-3,84	-4,53	-2,27	-1,66	-4,59	2,48
27,85	10,15	1,88	3,41	-4,00	0,59	-5,46	-1,85

2D DCT

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

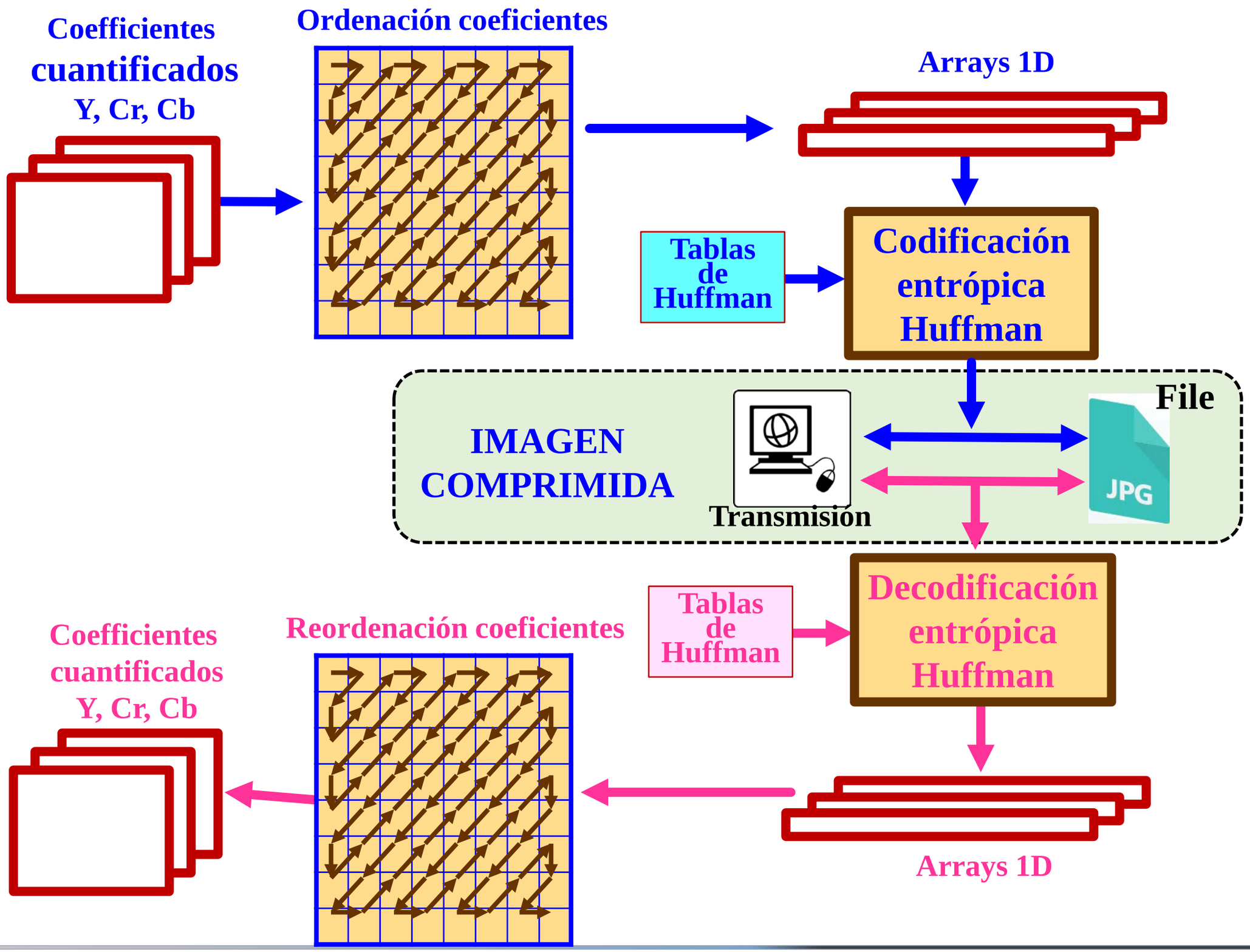
Tabla tabulada luma

Escalado + Cuantificación

-3	-2	2	0	0	0	0	0
4	-1	-1	0	0	0	0	0
7	2	0	0	0	0	0	0
1	1	1	0	0	0	0	0
-3	-1	0	0	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

Transformada Cuantizada

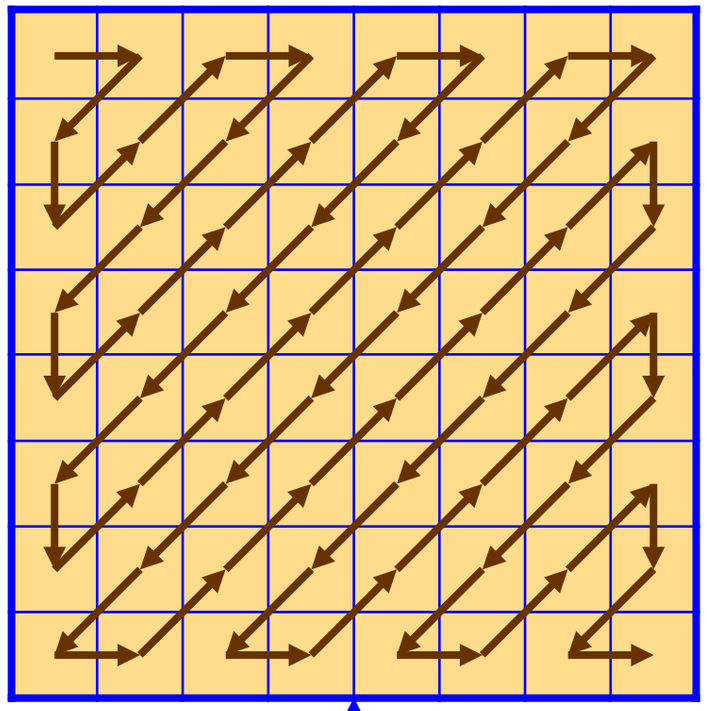
Step 4. Codificación/Decodificación entrópica Huffman



➢ Ordenación Zigzag JPEG

Tabla de Ordenación

-3	-2	2	0	0	0	0	0
4	-1	-1	0	0	0	0	0
7	2	0	0	0	0	0	0
1	1	1	0	0	0	0	0
-3	-1	0	0	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

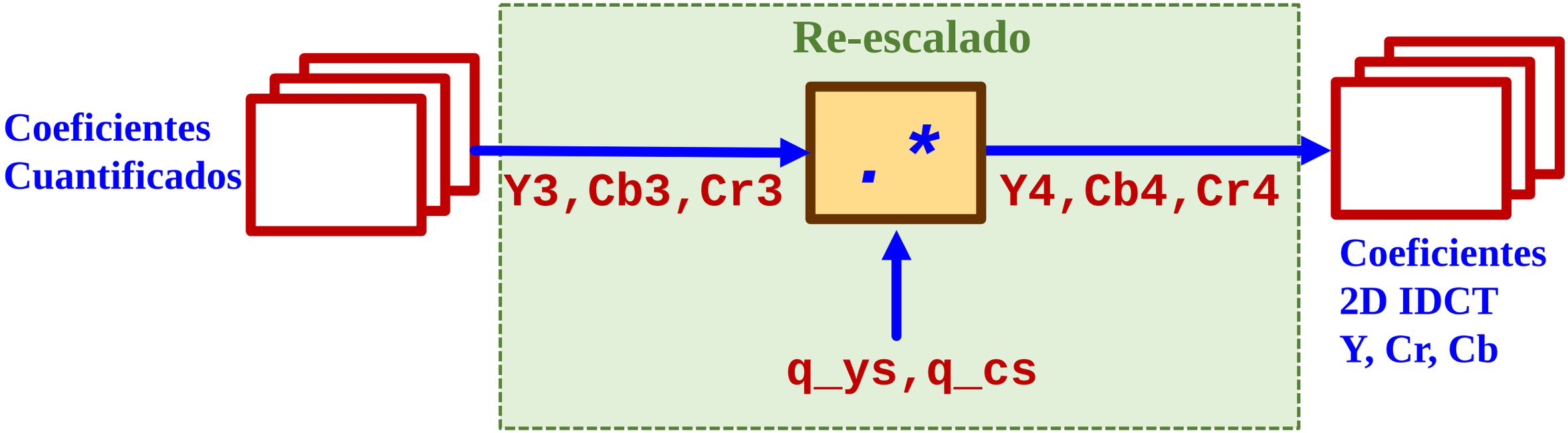


Transformada Cuantizada

-3	-2	4	7	-1	2	0	-1	2	1	-3	1	0	0	0	0	0	0	1	-1	-1	•••

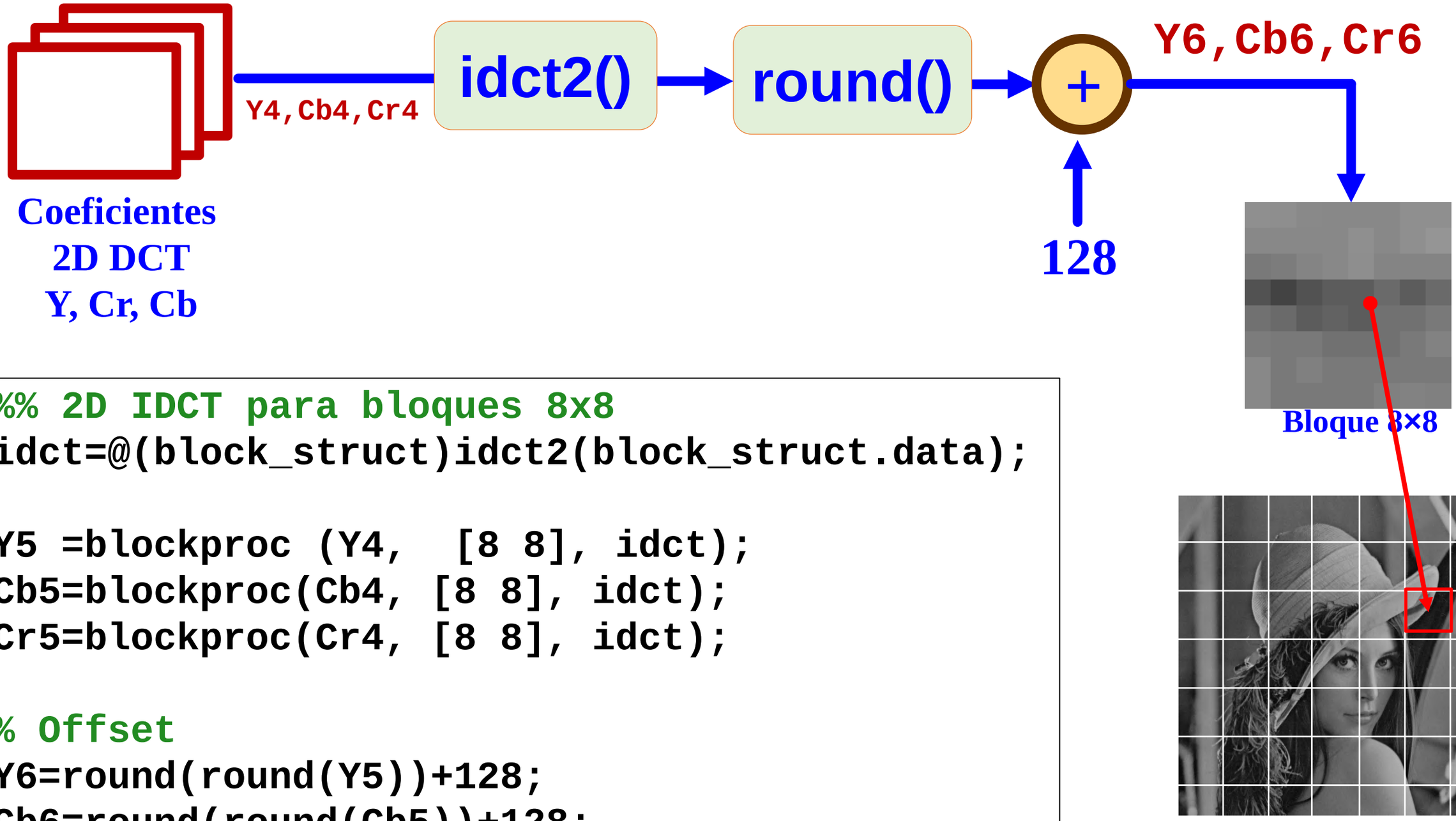
Array1-D

Step 5. Cuantificación inversa: Re-escalado



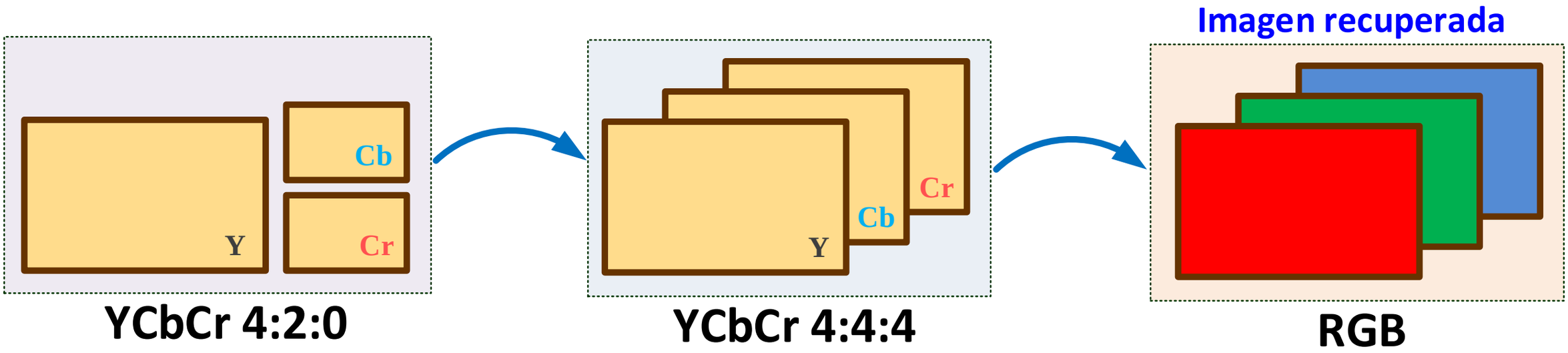

```
%% Cuantificación inversa: Re-Escalado
Y4=blockproc (Y3,   [8 8], @(block_struct) round(round(
block_struct.data .* q_ys)) );
Cb4=blockproc (Cb3, [8 8], @(block_struct) round(round(
block_struct.data .* q_cs)) );
Cr4=blockproc (Cr3, [8 8], @(block_struct) round(round(
block_struct.data .* q_cs)) );
```

Step 6. 2D IDCT




```
%% 2D IDCT para bloques 8x8
idct=@(block_struct)idct2(block_struct.data);

Y5 =blockproc (Y4,  [8 8], idct);
Cb5=blockproc(Cb4, [8 8], idct);
Cr5=blockproc(Cr4, [8 8], idct);

% Offset
Y6=round(round(Y5))+128;
Cb6=round(round(Cb5))+128;
Cr6=round(round(Cr5))+128;
```

Step 7. Conversión YCbCr 4:2:0 → RGB



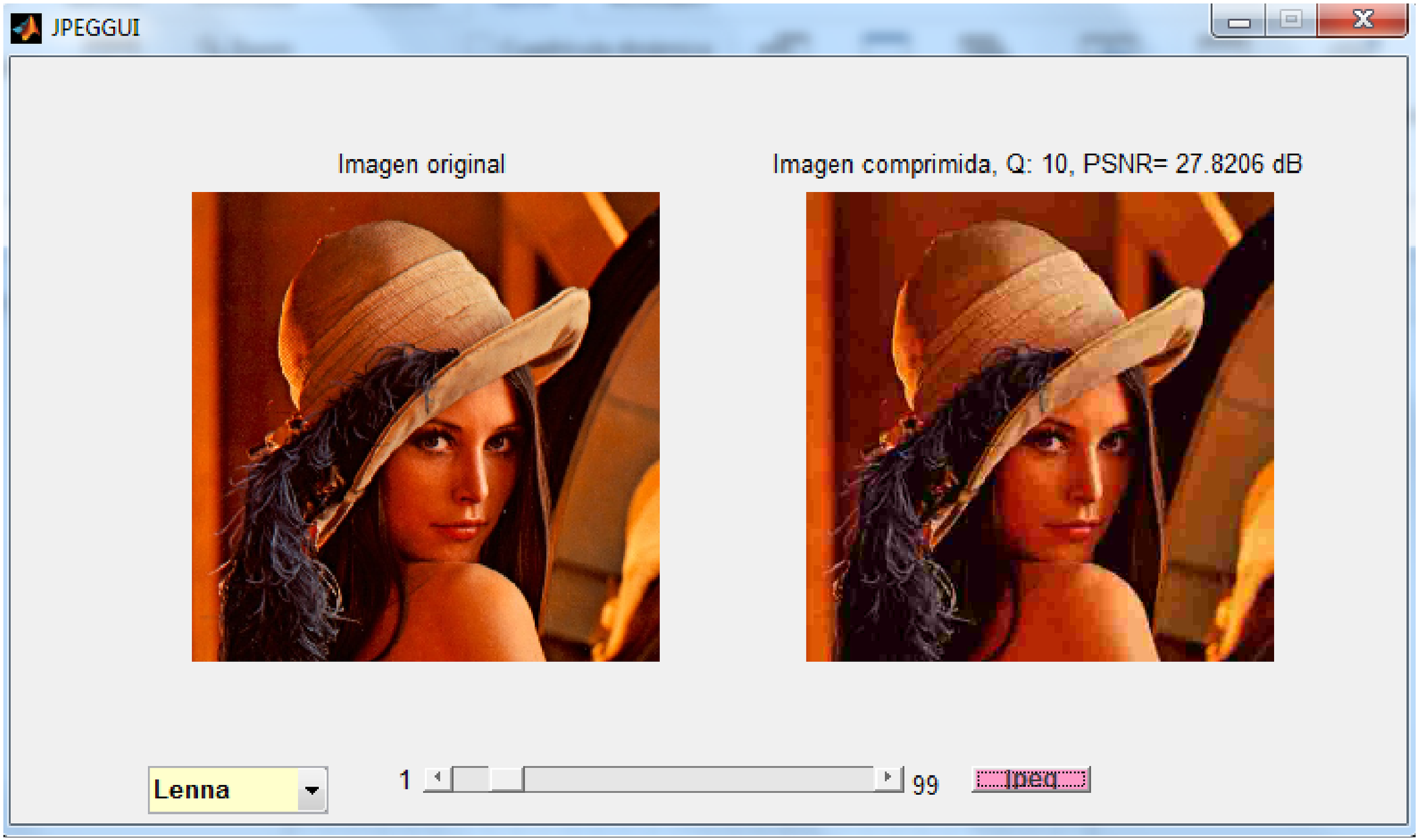
YCbCr 4:2:0 → YCbCr 4:4:4

```
%% Upsampling cromas. Conversion YCbCr 4:2:0 a 4:4:4
Cb6=imresize(Cb6,2);
Cr6=imresize(Cr6,2);
```

YCbCr 4:4:4 → RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.3441 & -0.7141 \\ 1.0 & 1.772 & -0.0001 \end{bmatrix} \left(\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \right)$$

APLICACIÓN MATLAB jpegSlice.m

```
----------------------
---  JPEG BASICO   ---
----------------------
Step 1. Conversion RGB -> YcbYcr 4:2:0
Step 2. 2D DCT
Step 3. Cuantificación
Step 4. Codificación/Decodificación entrópica Huffman
Step 5. Cuantificación inversa: Re-escalado
Step 6. 2D IDCT
Step 7. Conversión YCbCr 4:2:0 -> RGB


Fichero de salida (quality=78): lennaRec.tif
PSNR=32.008496 dB
**********************
```

EDICIÓN DE UNA INTERFASE GUI CON MATLAB