

# DIGITAL STILL IMAGING: FIRST GENERATION



*Fernando Pereira*

*Instituto Superior Técnico*



# Multilevel Photographic Image Coding (gray and colour)

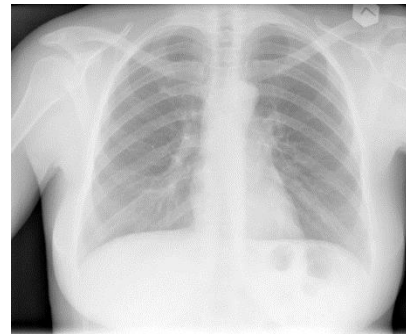


## OBJECTIVE

**Efficient representation of multilevel photographic images  
(still pictures) for storage and transmission.**

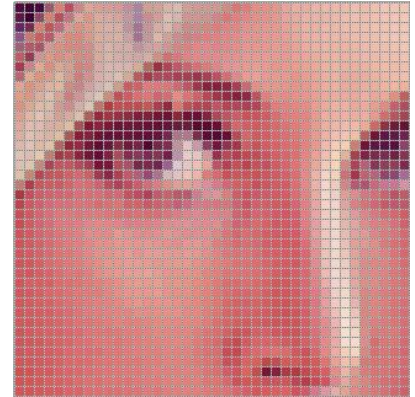
# Applications

- ★ Digital cameras
- ★ Image databases, e.g. museums, maps
- ★ Desktop publishing
- ★ Colour fax
- ★ Medical images
- ★ ...
- ★ and Digital cinema (!)



# The Image Representation Problem ...

A image is represented as a set of  $M \times N$  luminance and chrominance samples (*spatial sampling and quantization*) with a certain number of bits per sample,  $P$  (*PCM coding*).

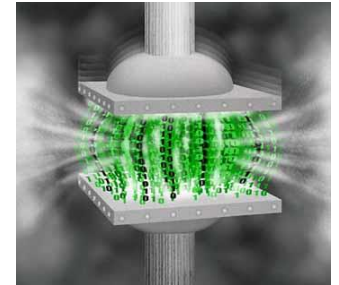


Thus, the total number of bits ( $M \times N \times P$ )  
- and so the memory and bandwidth –  
necessary to PCM digitally represent an  
image is **HUGE !!!**

**This is the so-called *RAW / ORIGINAL image !***



# Image (Source) Coding Objective



**Image coding/compression deals with the efficient representation of images, satisfying the relevant requirements.**

**And these requirements keep changing, e.g., coding efficiency, error resilience, random access, interaction, editing, to address new applications and functionalities ...**

# Where does Compression come from ?

★ **REDUNDANCY** – Regards the similarities, correlation and predictability of samples and symbols corresponding to the image/audio/video data.

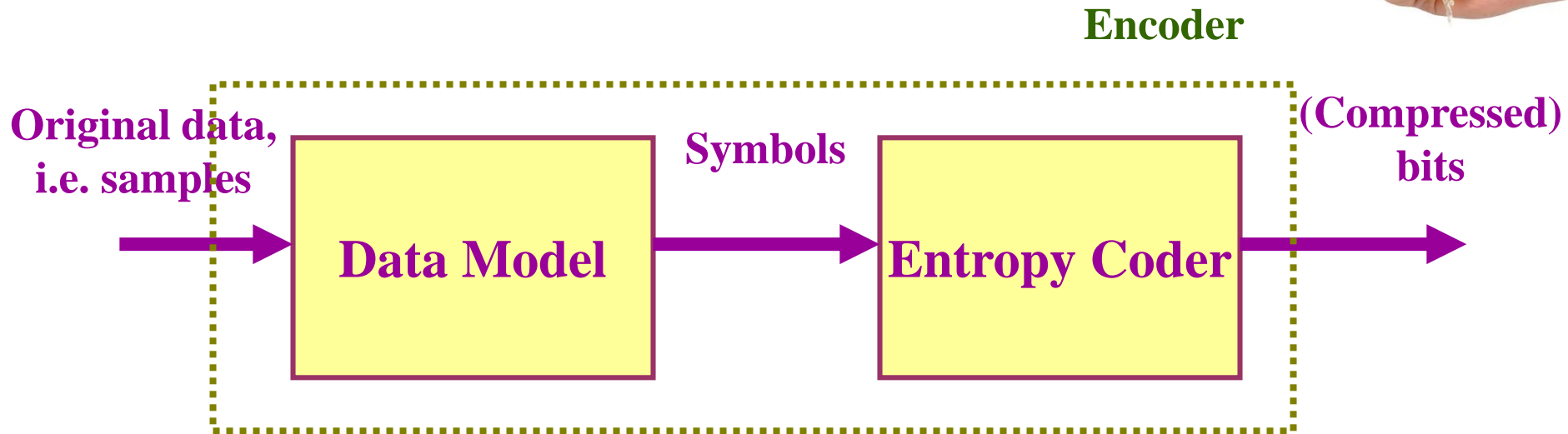
-> **redundancy reduction does not involve any information loss, implying it is a reversible process -> *lossless coding***

★ **IRRELEVANCY** – Regards the part of the information which is imperceptible for the visual or auditory human systems.

-> **irrelevancy reduction involves removing non-redundant information, implying it is an irreversible process -> *lossy coding***

**Source coding exploits these two concepts: for this, it is necessary to know the source statistics and the human visual/auditory systems characteristics.**

# Source Coding: Original Data, Symbols and Bits



**Source Coding implies two main steps:**

- ★ **Data modeling** – By adopting a more powerful data representation model the raw PCM symbols are converted into more efficient and ‘sophisticated’ symbols, notably exploiting spatial and temporal redundancies as well as irrelevancy, targeting the relevant representation requirements.
- ★ **Entropy coding** – By exploiting the statistical characteristics of the symbols produced by the data modeling process, a set of bits is produced.

# Image Coding: Multiple Solutions



- ★ DCT-based transform coding, e.g. JPEG standard

JPEG

- ★ Fractal-based coding

- ★ Vector quantization coding

JPEG  
2000

- ★ Wavelet-based coding, e.g. JPEG 2000 standard

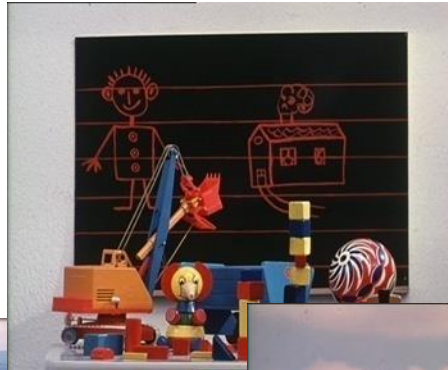
- ★ Lapped biorthogonal-based transform coding, e.g. JPEG XR standard

- ★ ...

**~1990**

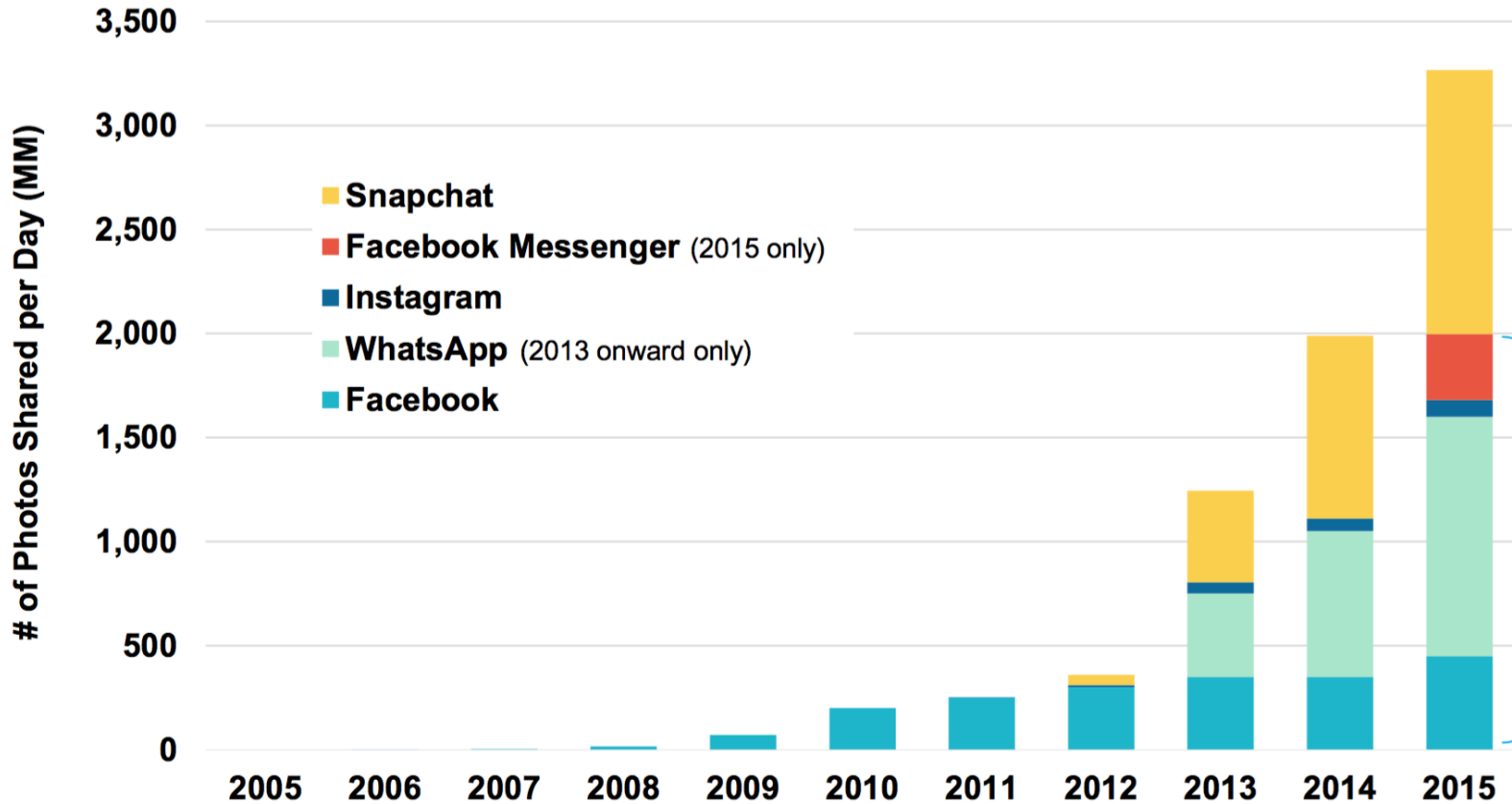
# **The JPEG Standard**

(Joint Photographic Experts Group, joint ISO & ITU-T)



**Definition of a generic compression standard for multilevel photographic images considering the requirements of most applications.**

# After 25 Years, JPEG Still Rocks ...



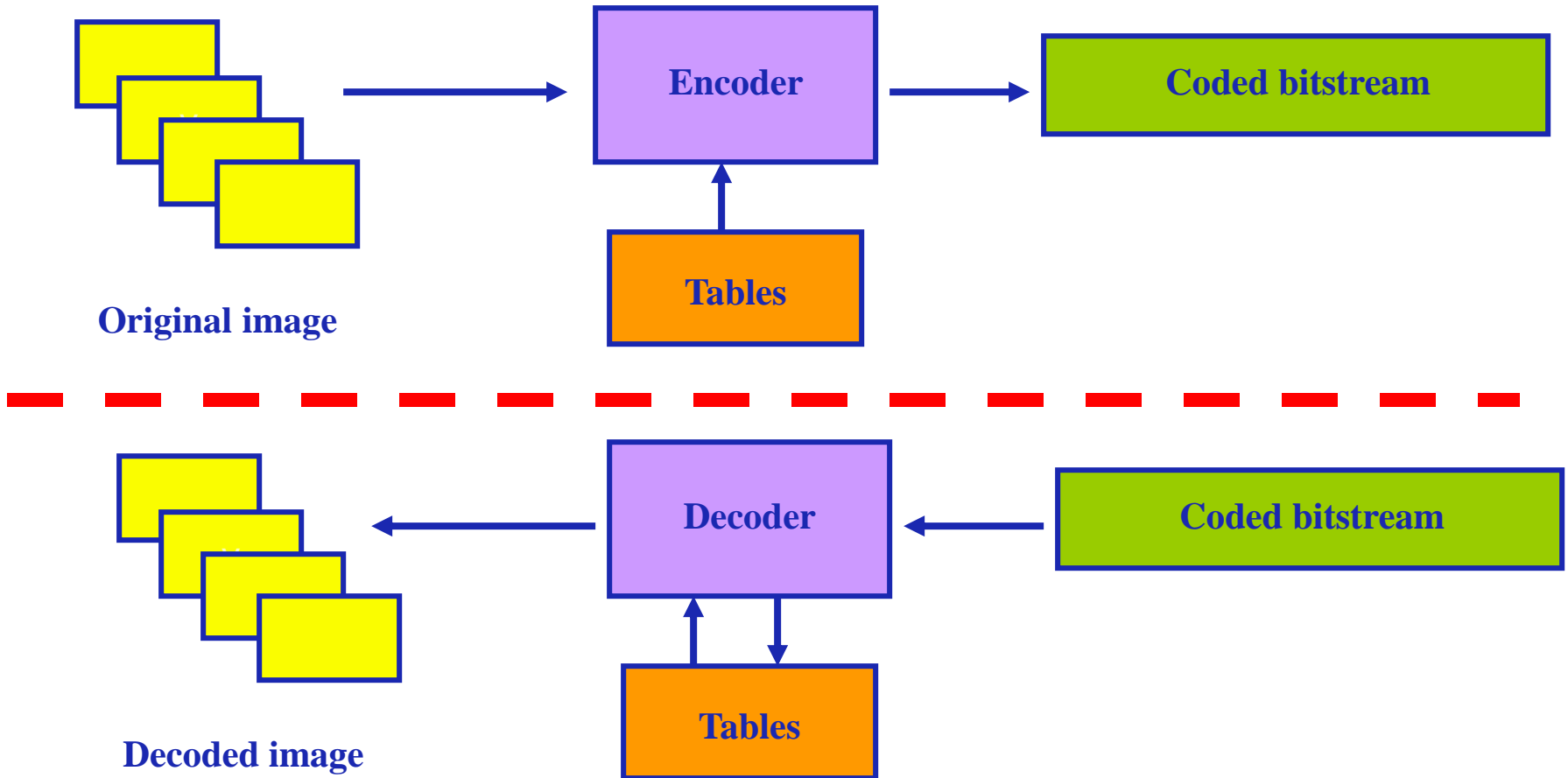
Source: KPCB Internet Trends 2016 (June 2016)

# JPEG Standard Major Requirements

≈1985

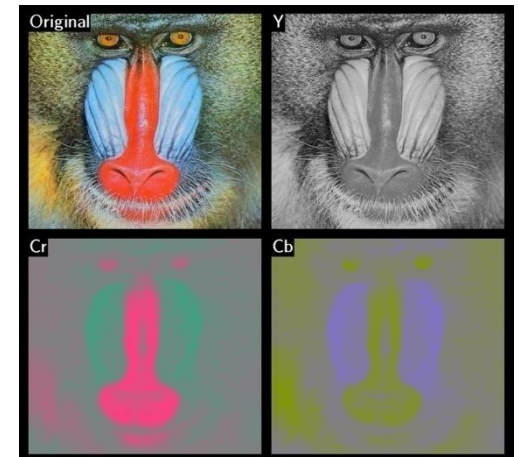
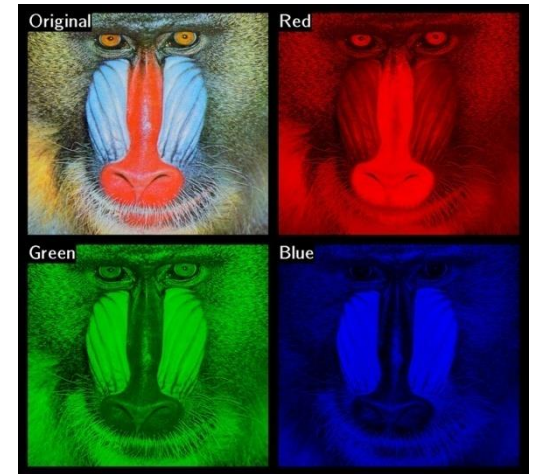
- ★ **Efficiency** - The standard must be based on the most efficient compression techniques, notably for high quality.
- ★ **Compression/Quality Tunable** - The standard shall allow tuning the quality versus compression efficiency.
- ★ **Generic** - The standard must be applicable to any type of multilevel photographic images without restrictions in resolution, aspect ratio, color space, content, etc.
- ★ **Low Complexity** - The standard must be implementable with a reasonable complexity; notably, its software implementation on a large range of CPUs must be possible.
- ★ **Functional Flexibility** - The standard must provide various relevant operation modes, notably sequential, progressive, lossless and hierarchical.

# JPEG Elements



# What Images can JPEG Encode ?

- ★ Size between  $1 \times 1$  and  $65535 \times 65535$
- ★ 1 to 255 colour components or spectral bands (typically  $Y C_R C_B$  or RGB)
- ★ Each component,  $C_i$ , consists of a matrix with  $x_i$  columns and  $y_i$  lines
- ★ 8 or 12 bits per sample for (lossy) DCT based compression
- ★ 2 to 16 bits per sample for lossless compression



# Types of JPEG Compression



- ★ **LOSSLESS** - The image is reconstructed with no losses, this means it is mathematically equal to the original; compression factors of about 2-3 may be achieved, depending on the image content.
- ★ **LOSSY** - The image is reconstructed with losses but, if desired, with a very high fidelity to the original (transparent coding); this type of coding allows achieving higher compression factors, e.g. 10, 20 or more; in the JPEG standard, this type of coding is based on the Discrete Cosine Transform (DCT).

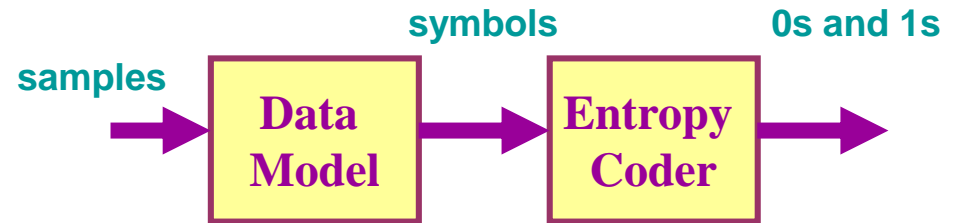
# JPEG Baseline Process

The most used JPEG coding solution is DCT based (lossy), called

**BASELINE SEQUENTIAL PROCESS**

and it is appropriate to inumerous applications. This process is mandatory for all systems claiming JPEG compliance.

# DCT Based Coding

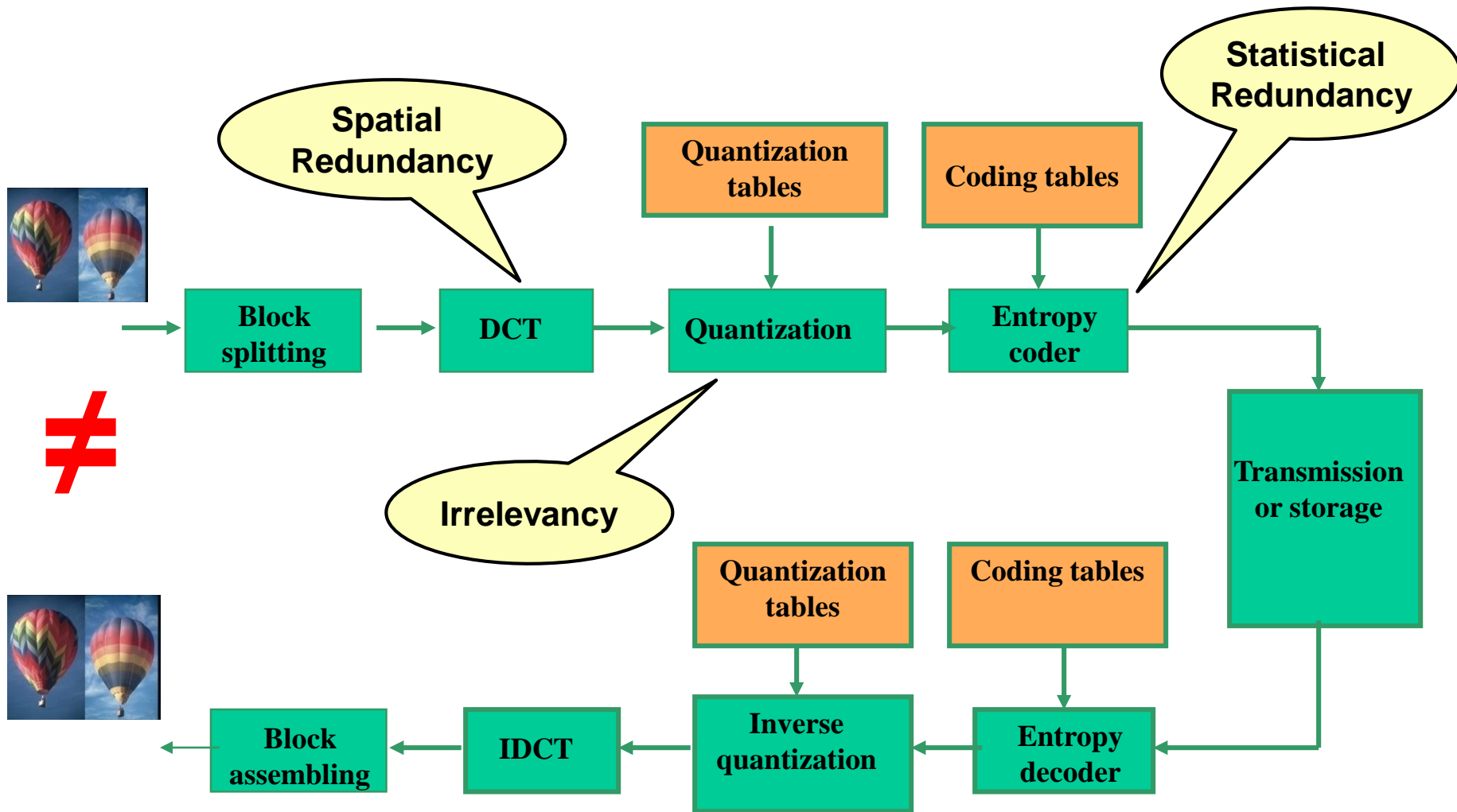


**The joint action of the various JPEG Baseline encoder modules targets the reduction of the redundancy and irrelevancy contained in the images.**

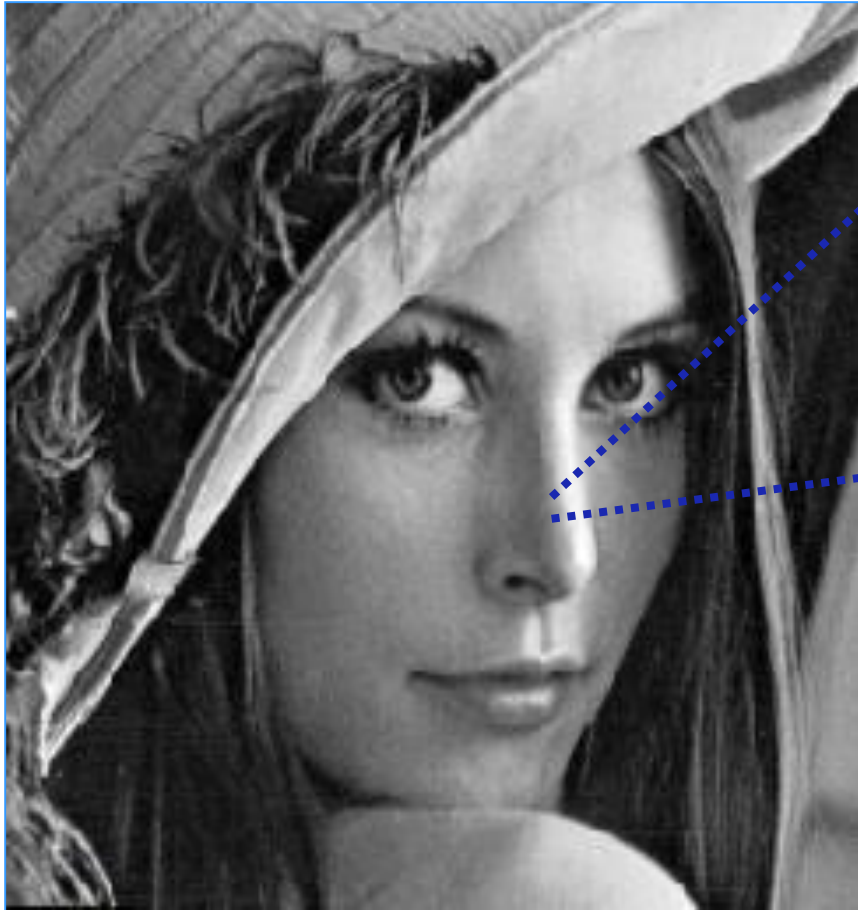
**The first encoder part (data modeling) targets the generation of a signal without memory (*elimination of spatial redundancy*) and *without irrelevancy*.**

**The final entropy coding module targets the exploitation of the symbol statistics to minimize the rate to transmit (*elimination of statistical redundancy*).**

# DCT Based Image Coding



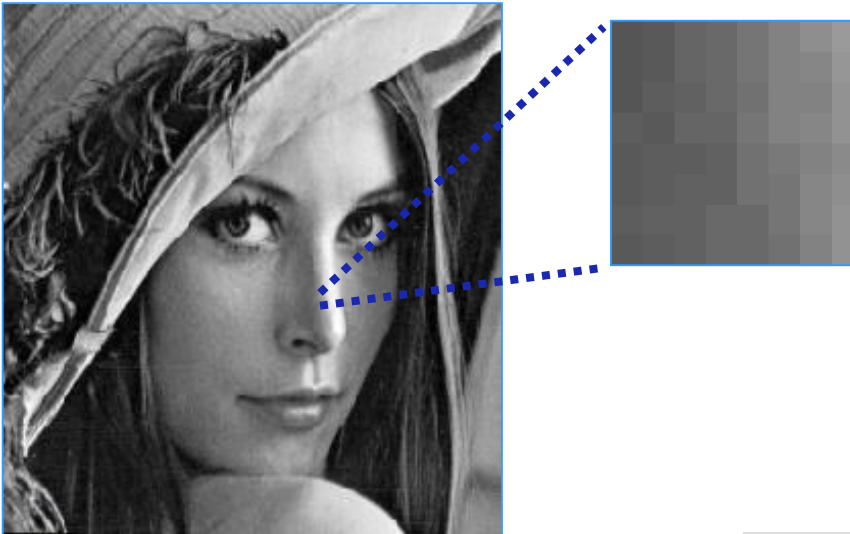
# What is Really a 8×8 Block ...



Imagine a block where all the samples are similar, this means have the same luminance value ...

It costs  $8 \times 8 \times 8 = 512$  PCM bits to say 64 times the same thing!

# What is Transformed ?



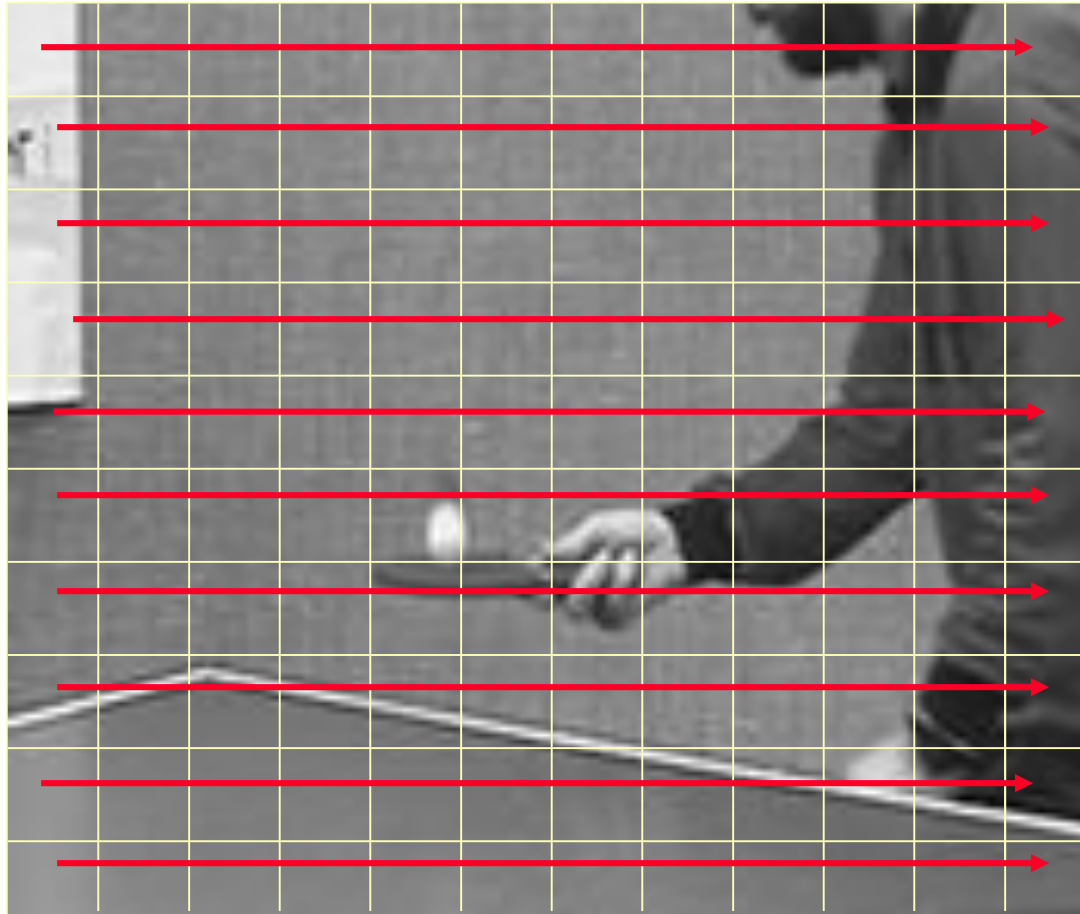
Transform is  
 applied block  
 after block in the  
 image ...

Same process (in  
 parallel) for  
 luminance and the  
 chrominances !

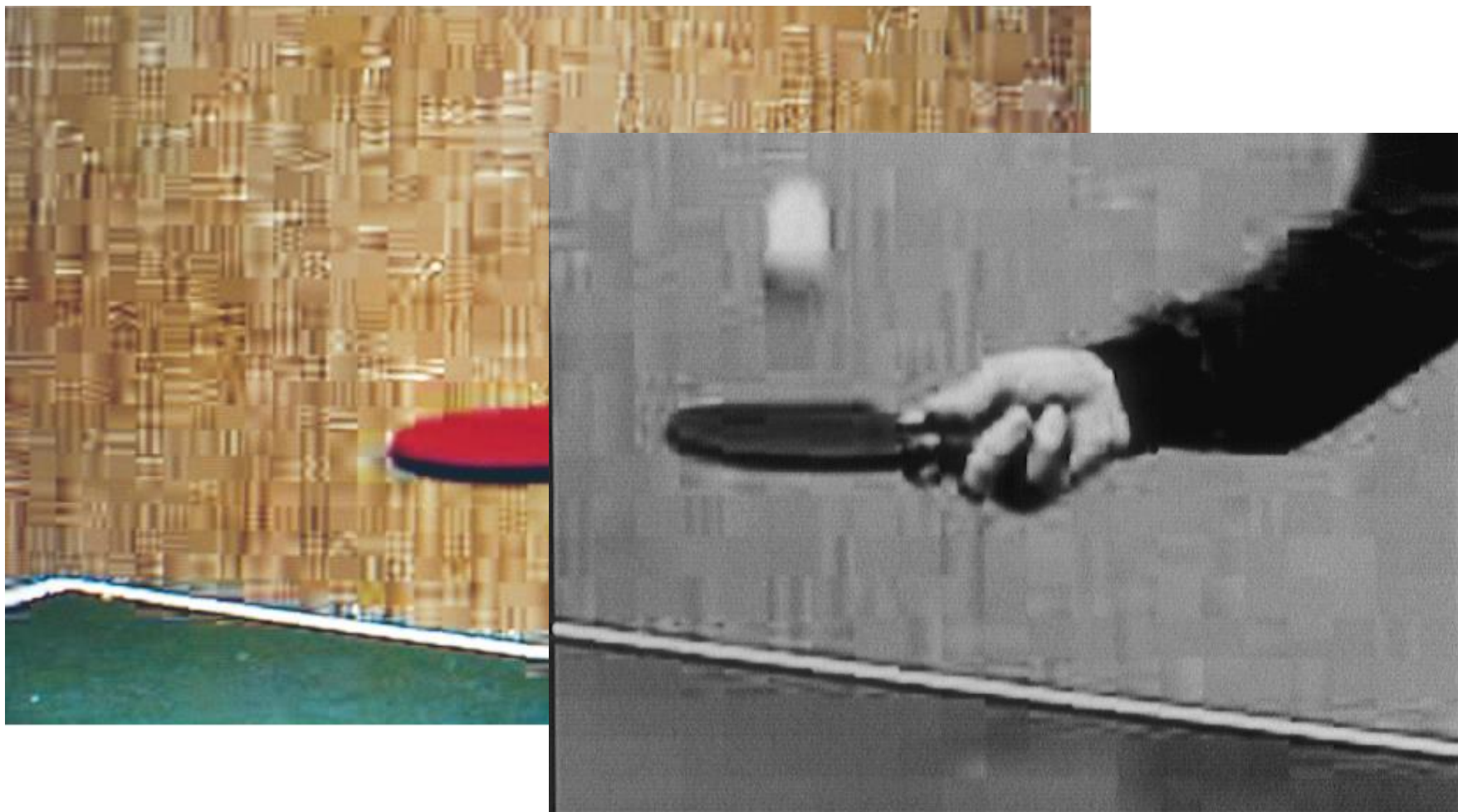
Y =

87	89	101	106	118	130	142	155
85	91	101	105	116	129	135	149
86	92	96	105	112	128	131	144
92	88	102	101	116	129	135	147
88	94	94	98	113	122	130	139
88	95	98	97	113	119	133	141
92	99	98	106	107	118	135	145
89	95	98	107	104	112	130	144

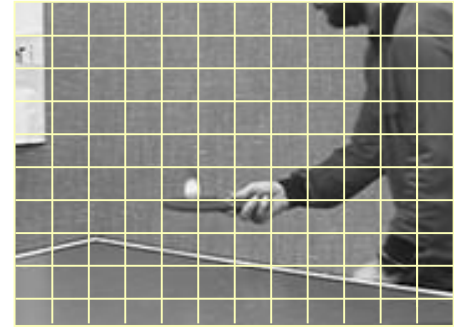
# JPEG Block Coding Sequence



# The Block Effect ...



# Why do we Transform Blocks ?



**Basically, the transform represents the original signal in another domain where there is less spatial redundancy.**

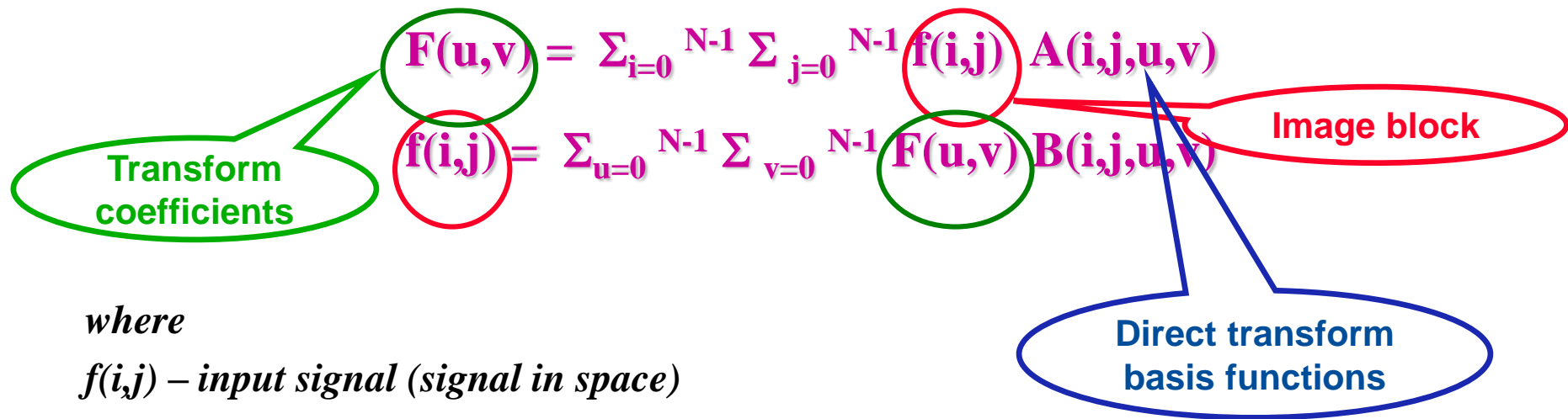
- ★ **The full exploitation of the spatial redundancy in the image would require applying the transform to blocks as big as possible, ideally to the full image; however, the redundancy is rather ‘local’ ...**
- ★ **The computational effort associated to the transform grows quickly with the size of the block used ... and the added spatial redundancy decreases ... So some trade-off is needed ...**

**Applying the transform to blocks, typically of  $8 \times 8$  samples, was a good trade-off between the exploitation of the spatial redundancy and the associated computational effort.**

Transform coding involves the division of the image into blocks of  $N \times N$  samples to which the transform is applied, producing blocks with  $N \times N$  transform coefficients.

A transform is formally defined by its direct and inverse transform equations:

$$F(u,v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) A(i,j,u,v)$$

$$f(i,j) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) B(i,j,u,v)$$


where

$f(i,j)$  – input signal (signal in space)

$A(i,j,u,v)$  – direct transform basis functions

$F(u,v)$  – transform coefficients (signal in frequency)

$B(i,j,u,v)$  – inverse transform basis functions

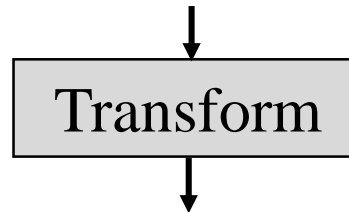
# What Shall the Transform Provide in Image Compression ?

- ★ **REVERSIBILITY** – The transform must be reversible since the image to transform has to be recovered again in the spatial domain.
- ★ **INCORRELATION** – The ideal transform shall provide coefficients which are uncorrelated this means each one carries additional/novel information.
- ★ **ENERGY COMPACTATION** – The major part of the signal energy shall be compacted in a small number of coefficients.
- ★ **IMAGE INDEPENDENT TRANSFORM BASIS FUNCTIONS** – Since images show significant statistical variations, the optimal transform should be image dependent; however, the use of image dependent transforms would require its computation as well as its storage and transmission; thus, an image independent transform is desirable even if at some cost in coding efficiency.
- ★ **LOW COMPLEXITY IMPLEMENTATIONS** – Due to the high number of operations involved, the transform shall allow low complexity/fast implementations.

Luminance  
Samples,  $Y =$

87	89	101	106	118	130	142	155
85	91	101	105	116	129	135	149
86	92	96	105	112	128	131	144
92	88	102	101	116	129	135	147
88	94	94	98	113	122	130	139
88	95	98	97	113	119	133	141
92	99	98	106	107	118	135	145
89	95	98	107	104	112	130	144

**But what is the  
*killer*  
 transform ?**



Transform  
 Coefficients =

898.0000	-149.5418	26.6464	-14.0897	0.7500	-5.7540	3.5750	0.0330
12.1982	-16.5235	-7.6122	5.2187	-0.2867	-1.9909	8.4265	1.2591
5.3355	-2.6557	2.3410	-9.9277	2.4614	4.4558	-3.1945	-3.1640
1.9463	-2.7271	1.5106	2.8421	-2.1336	-2.7203	-2.7510	5.4051
0.7500	-2.0745	0.8610	0.2085	2.5000	1.8446	2.0787	2.4750
7.9536	-2.6624	2.6308	0.4010	0.4772	3.3000	1.7394	0.3942
-4.1042	-0.1650	-0.6945	0.0601	0.0628	-0.7874	-0.8410	0.3496
-3.4688	2.3804	0.1559	0.8696	0.1142	-0.5240	-3.9974	-5.6187

# Karhunen-Loève Transform (KLT)

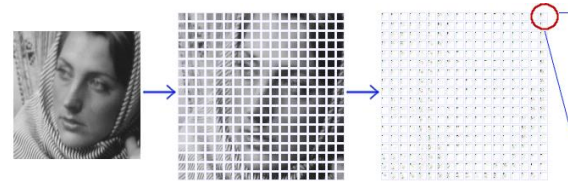
The Karhunen-Loève Transform is typically considered the ideal transform because it achieves the

## MAXIMUM ENERGY COMPACTATION

this means, if a certain limited number of coefficients is coded, the KLT coefficients are always those containing the highest percentage of the total signal energy.

The KLT base functions are based on the *eigen vectors of the covariance matrix for the image blocks ... and thus depend on each image block being transformed !*

# Why is KLT Never Used ?



**In practice, the use of KLT for image compression is negligible because:**

- ★ **KLT basis functions are image dependent requiring the computation of the image covariance matrix as well as its storage and/or transmission to allow decoding.**
- ★ **Fast algorithms for its computation are not as good as for other transforms.**
- ★ **There are other transforms without the drawbacks above but still with a energy compactation performance only slightly lower than KLT.**





The DCT is one of the several sinusoidal transforms available; its basis functions correspond to discretized sinusoidal functions.

Transform coefficients

$$F(u, v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

Image block

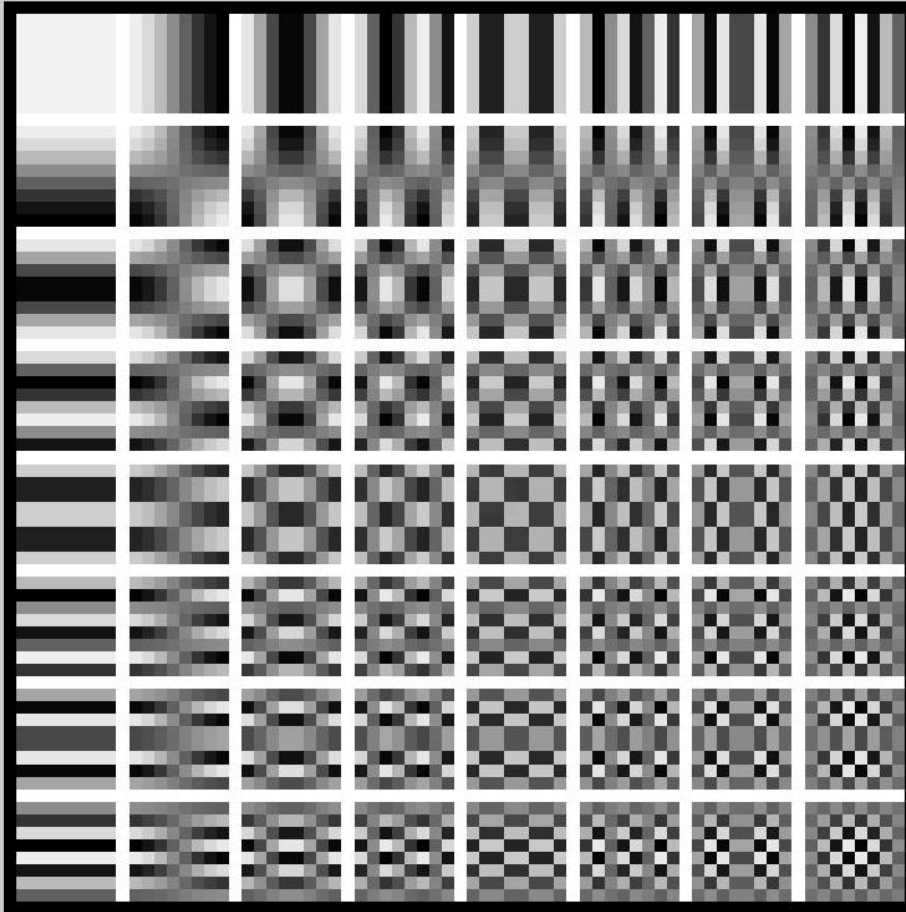
Image block

$$f(i, j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v) F(u, v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

Transform coefficients

The DCT is the most used transform for image and video compression since its performance is close to the KLT performance for highly correlated signals; moreover, there are fast implementation algorithms available.

# DCT Bidimensional Basis Functions (N=8)



All existing and future image blocks can be rather efficiently represented with these 64 (8x8) basic images !!!

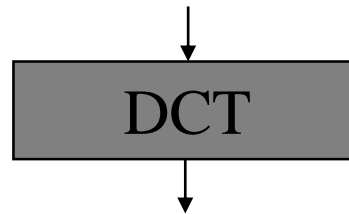


Luminance  
Samples, Y =

87	89	101	106	118	130	142	155
85	91	101	105	116	129	135	149
86	92	96	105	112	128	131	144
92	88	102	101	116	129	135	147
88	94	94	98	113	122	130	139
88	95	98	97	113	119	133	141
92	99	98	106	107	118	135	145
89	95	98	107	104	112	130	144

**64 PCM samples are transformed into 64 DCT coefficients !**

**But more perceptual compression friendly !**



DCT  
Coefficients =

898.0000	-149.5418	26.6464	-14.0897	0.7500	-5.7540	3.5750	0.0330
12.1982	-16.5235	-7.6122	5.2187	-0.2867	-1.9909	8.4265	1.2591
5.3355	-2.6557	2.3410	-9.9277	2.4614	4.4558	-3.1945	-3.1640
1.9463	-2.7271	1.5106	2.8421	-2.1336	-2.7203	-2.7510	5.4051
0.7500	-2.0745	0.8610	0.2085	2.5000	1.8446	2.0787	2.4750
7.9536	-2.6624	2.6308	0.4010	0.4772	3.3000	1.7394	0.3942
-4.1042	-0.1650	-0.6945	0.0601	0.0628	-0.7874	-0.8410	0.3496
-3.4688	2.3804	0.1559	0.8696	0.1142	-0.5240	-3.9974	-5.6187

# How to Interpret a Transform ?



The formula for the inverse transform

$$f(i,j) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \cdot B(i,j,u,v)$$

Image  
block

Weights

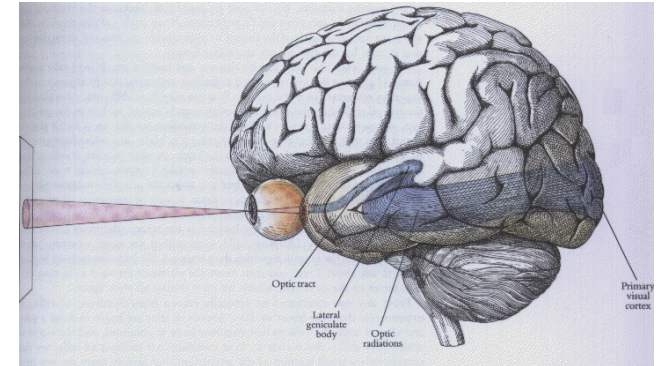
Basic image  
blocks

expresses that the inverse transform process may be interpreted as a reconstruction/synthesis of each image block by adding the relevant set of basic functions – *the transform basis functions* – adequately weighted by the transform coefficients.

*The Spectral Interpretation – As most transforms use basis functions with different frequencies (in a broad sense), the decomposition in basis functions assumes a spectral meaning where each coefficient represents the fraction of energy in the image corresponding to a certain basis function/frequency.*

# Advantages of the Spectral Interpretation

The spectral interpretation allows to easily introduce in the coding process some relevant characteristics of the human visual system which are essential for efficient (lossy) coding.



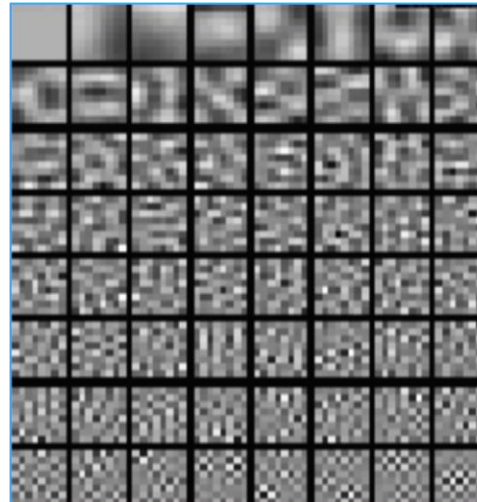
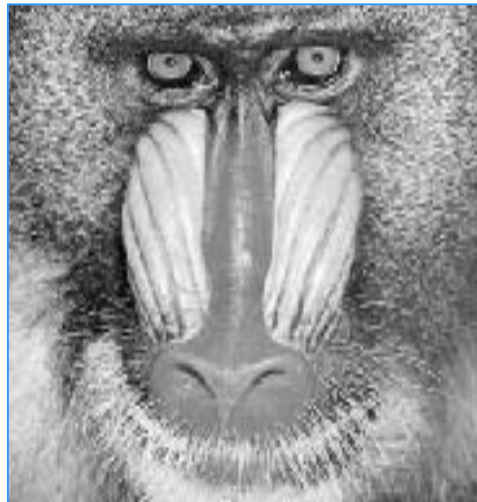
★ The human visual system is less sensitive to the higher spatial frequencies

->> coarser coding (through quantization) of the corresponding transform coefficients

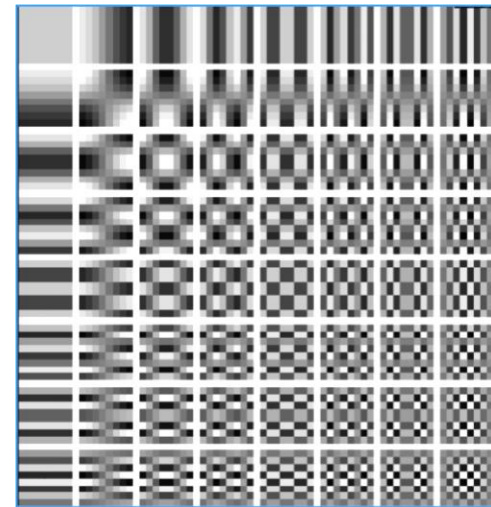
★ According to the Weber's Law, *the smallest change in the intensity of a stimulus capable of being perceived is proportional to the intensity of the original stimulus*, e.g. it is more difficult to see a change in a very white background

->> coarser coding (through quantization) of the coefficients depending on the amount of variation and context

# DCT versus KLT ...



**DCT: Same basis functions for all image blocks !**



**DCT for all blocks**

**KLT for a block**

# How Does the DCT Work ?

**Spatial Domain, samples**

**Frequency Domain, DCT coefficients**

X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X

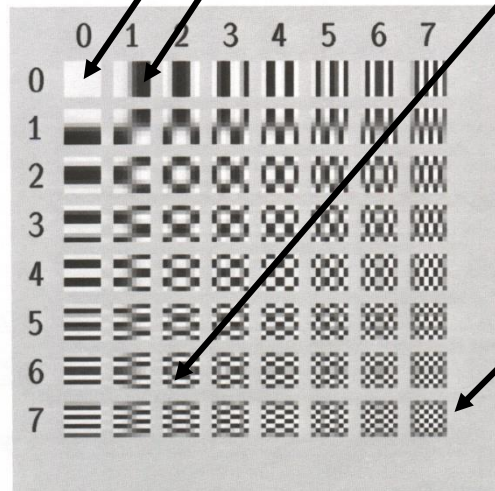
**DCT**



x	y		a				
C	f	d					c
	H						k
Y			i				p
	q	d					
	n	m					
							z

*Low-frequency, high energy*

*High-frequency, low energy*



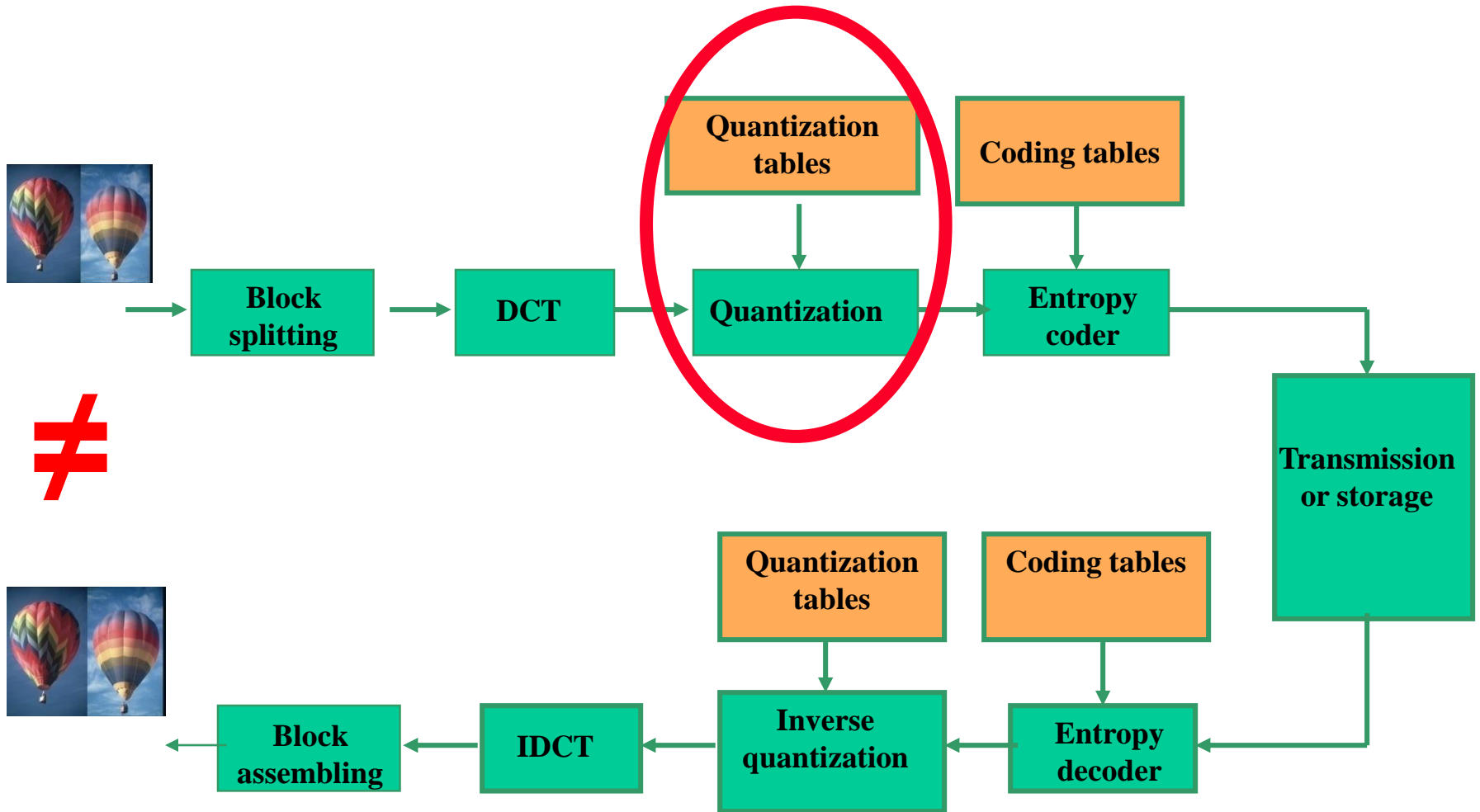


Since the DCT uses sinusoidal functions, it is impossible to perform computations with full precision. This leads to (slight) differences in the results for different implementations, so-called *decoding mismatch*.

- ★ To accommodate future (simpler) implementation developments, the JPEG recommendation does not specify any specific DCT or IDCT implementation.
- ★ The JPEG recommendation specifies a fidelity/accuracy test regarding a reference implementation in order to limit the differences caused by the freedom in terms of DCT and IDCT implementation.

*Note: The DCT is applied to the signal samples with  $P$  bits, with values between  $-2^{P-1}$  and  $2^{P-1}-1$  in order the DC coefficient is distributed around zero.*

# DCT Based Image Coding



# Quantization: Making the Codec Lossy ...

Quantization is the process by which irrelevancy or perceptual redundancy is reduced.

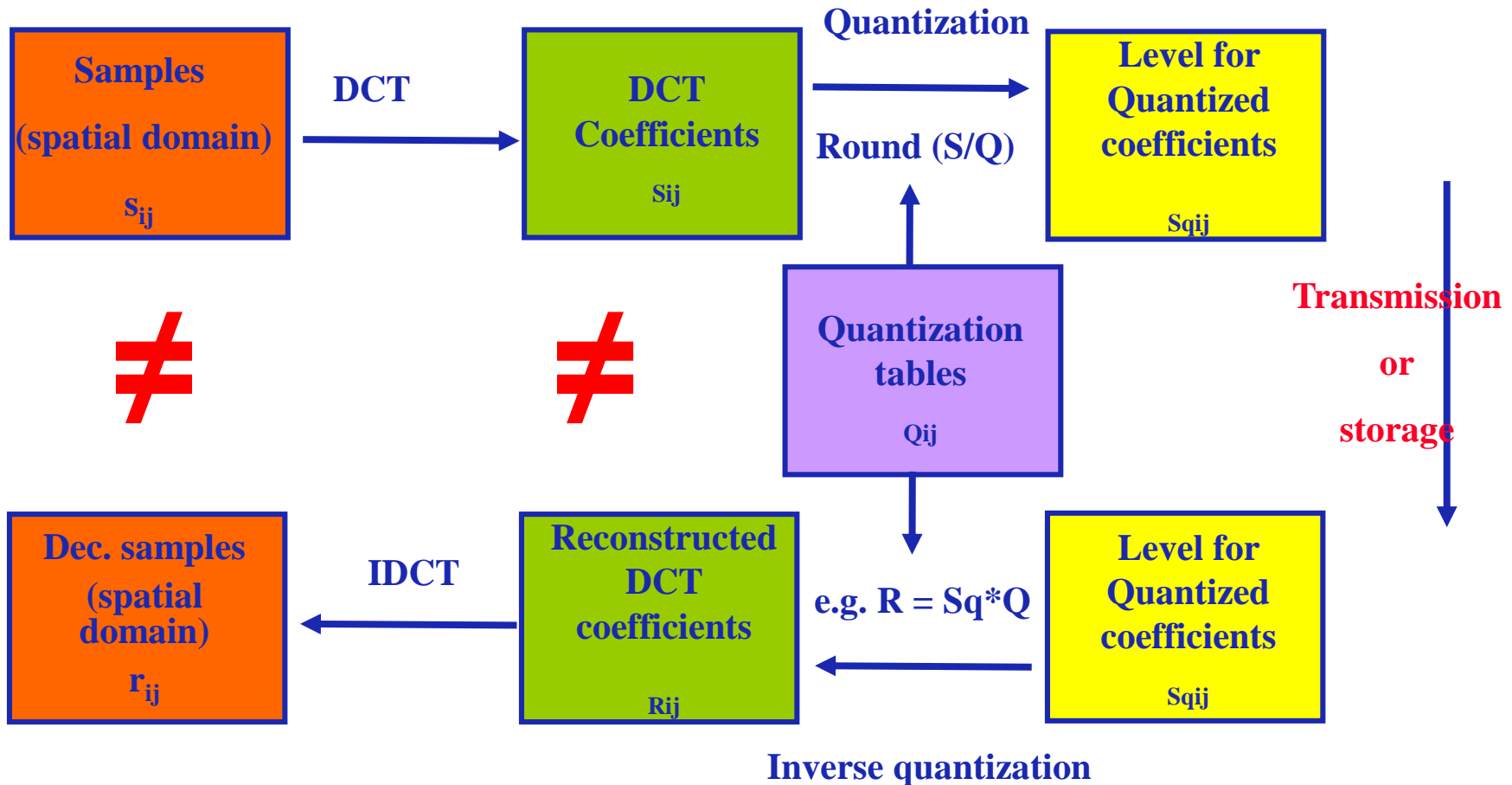
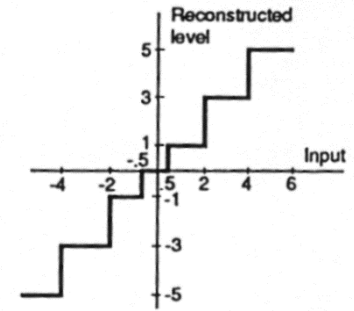
This process is the main responsible for the quality losses (but also the increased compression factors) in DCT based codecs (*but quality may be transparent even with quantization*).



For transparent quality, each quantization step may be selected taking into account the ‘minimum perceptual difference’ characteristics of the human visual system for the coefficient in question.

The quantization matrixes are not standardized but there is a suggested solution for ITU-R 601 resolution images (which still has to be coded).

# How Does DCT Coding Work ?



# Quantization Matrices

For transparent quality, JPEG suggests to quantize the DCT coefficients using the values for the ‘minimum perceptual difference’ (for each coefficient) multiplied by 2; for more compression, a multiple of them may be used.

The quantization matrixes have to be always transmitted or at least signalled.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	69	75
14	13	16	24	48	69	75	75
14	17	22	29	57	77	80	62
17	22	33	66	68	109	103	77
22	33	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

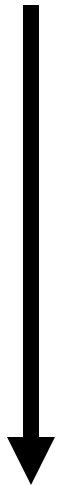
**Luminance**

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

**Chrominances**

**Situation: Luminance and crominance with 2:1 horizontal subsampling; samples with 8 bits (*Lohscheller*)**

898.0000	-149.5418	26.6464	-14.0897	0.7500	-5.7540	3.5750	0.0330
12.1982	-16.5235	-7.6122	5.2187	-0.2867	-1.9909	8.4265	1.2591
5.3355	-2.6557	2.3410	-9.9277	2.4614	4.4558	-3.1945	-3.1640
1.9463	-2.7271	1.5106	2.8421	-2.1336	-2.7203	-2.7510	5.4051
0.7500	-2.0745	0.8610	0.2085	2.5000	1.8446	2.0787	2.4750
7.9536	-2.6624	2.6308	0.4010	0.4772	3.3000	1.7394	0.3942
-4.1042	-0.1650	-0.6945	0.0601	0.0628	-0.7874	-0.8410	0.3496
-3.4688	2.3804	0.1559	0.8696	0.1142	-0.5240	-3.9974	-5.6187



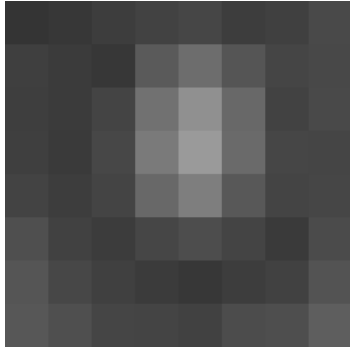
Quantizing ...

56	-14	3	-1	0	0	0	0
1	-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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Finally, the  
waited  
miracle !



# JPEG Coding: an Encoder Example



**Original PCM**

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

**Original PCM - 128**

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

$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

**DCT Coefficients**

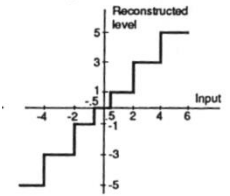
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Quantized DCT Coeffs**

**Quantization Steps**

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

# JPEG Coding: a Decoder Example



**Quantized DCT Coeffs**

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

**Dequantized DCT Coeffs**

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

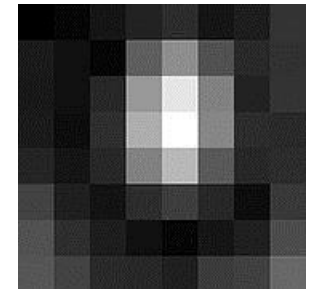
**Inverse DCT Output**

$$\begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -61 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$

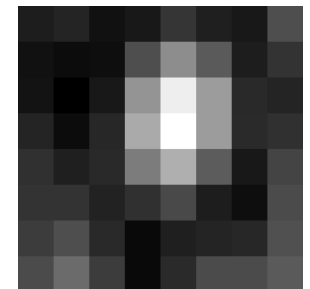
$$\begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 64 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 95 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}$$

$$\begin{bmatrix} -10 & -10 & 4 & 6 & 2 & 2 & 4 & -9 \\ 6 & 4 & -1 & 8 & 1 & -2 & 7 & 1 \\ 4 & 9 & 8 & 2 & -4 & -10 & -1 & 8 \\ -2 & 3 & 5 & 2 & -1 & -8 & 2 & -1 \\ -3 & -2 & 1 & 3 & 4 & 0 & 8 & -8 \\ 8 & -6 & -4 & 0 & -3 & 6 & 2 & -6 \\ 10 & -11 & -3 & 5 & -8 & -4 & -1 & 0 \\ 6 & -15 & -6 & 14 & -3 & -5 & -3 & 7 \end{bmatrix}$$

**Original block**



**Decoded block**



**Inverse DCT Output + 128**

**Coding error**

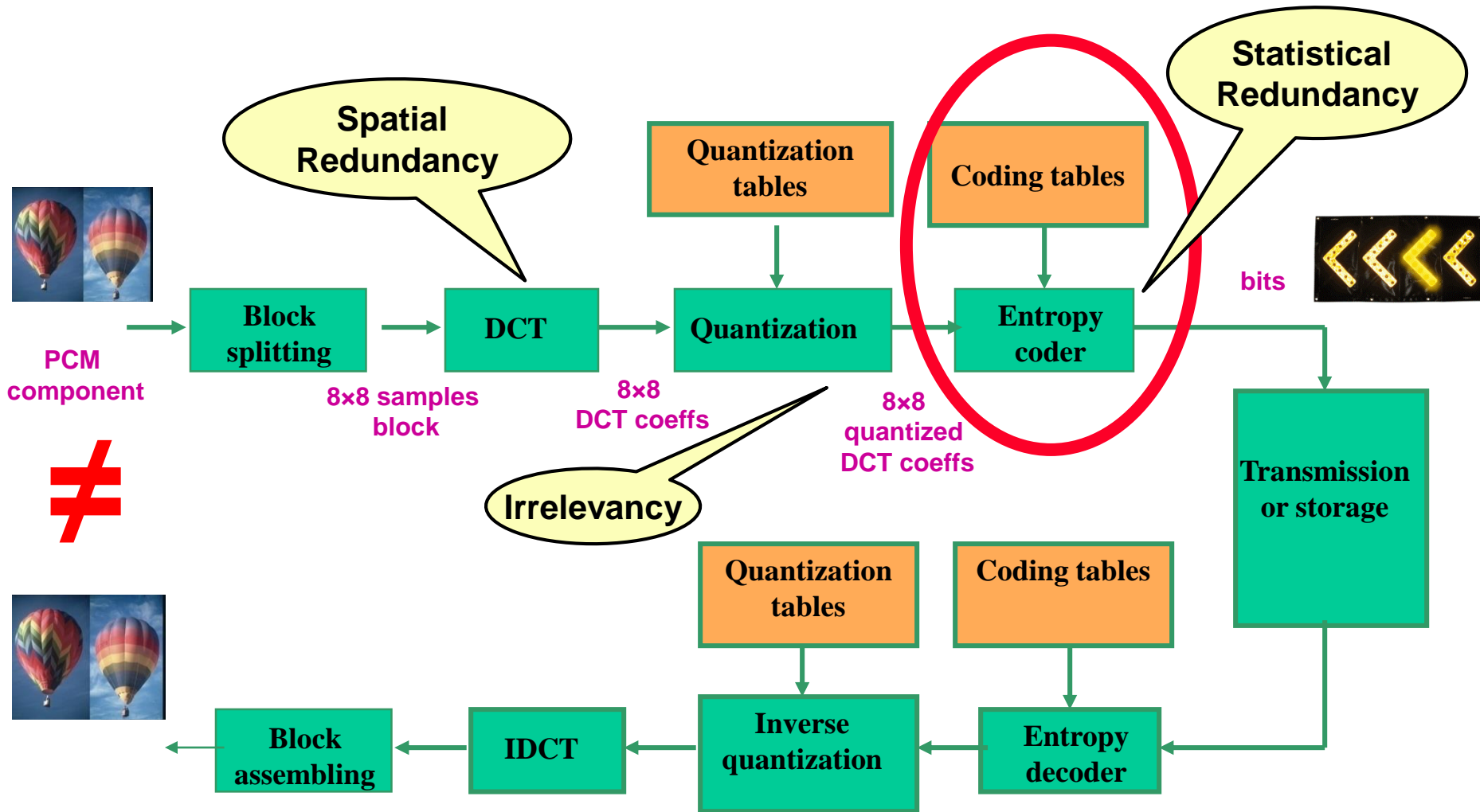
**Photo with the  
compression rate  
decreasing, and  
hence quality  
increasing, from  
left to right.**

**DCT coefficients  
selection and  
quantization are  
the secrets !**

**And not defined by  
the standard ...**



# DCT Based Image Coding



# Zig-Zag Serializing the Quantized Coefficients

124	25	0	0	0	0	23	0
147	0	13	0	0	78	190	248
126	147	0	0	0	0	0	0
0	10	0	0	15	0	183	119
40	0	0	0	83	0	0	0
94	0	0	173	0	0	0	0
0	0	0	56	0	0	0	0
203	0	0	0	0	0	0	0

Each DCT block is represented as a sequence of (run, level) pairs, e.g. (0,124), (0, 25), (0,147), (0, 126), (3,13), (0, 147), (1,40) ...

- ★ For the decoder to reconstruct the matrix with the quantized DCT coefficients, the position and amplitude of the non-null coefficients has to be coded, one after another.
- ★ The position of each quantized DCT coefficient may be sent in a relative or absolute way.
- ★ The JPEG solution is to send the position of each non-null quantized DCT coefficient through a run indicating the number of null DCT coefficients existing between the current and the previous non-null coefficient.

# The Symbols to Code for Each Block ...

56	-14	3	-1	0	0	0	0
1	-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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Each DCT block is represented as a sequence of (run, level) pairs, notably

**DC: (0, 56)**

**AC1: (0, -14)**

**AC2: (0, 1)**

**AC4: (1, -1)**

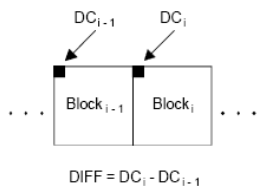
**AC5: (0, 3)**

**AC6: (0, -1)**

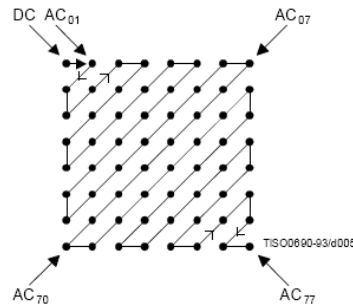
**AC7: (0, -1)**

**EOB (End of Block)**

Small  
*runs* and  
small  
*levels* are  
highly  
probable !



Differential DC encoding



Zig-zag order



**Entropy coding allows representing with a stream of bits the stream of symbols issued by a source, taking into account their statistical distribution.**

**Unless all symbols have the same probability, constant length coding is not the most efficient solution ...**

**Entropy coding:**

- (+) Increases the final compression efficiency**
- (+) Does not degrade the coded signal, this means it is lossless**
- (-) Produces a highly time varying bitstream**
- (-) Increases the sensibility to transmission errors**
- (\*) Provides compression in statistical terms, not necessarily symbol by symbol**

# Huffman (VLC) Coding

Símbolo	Probabilidade redução 0	Palav. Código redução 0	Probabilidade redução 1	Palav. Código redução 1
A	0.7	0	0.7	0
B	0.2	1 0	0.3	1
C	0.1	1 1		

Huffman coding allows obtaining a code with an average number of bits per symbol as close as desired to the source entropy.

But this requires knowledge on the source statistics, i.e., symbol probabilities.

**Entropy = 1.157 bit/symbol**  
 ( $H = \sum p_i \log_2 (1/p_i)$  bit/symbol)

**Average code length = 1.3 bit/symbol**

**Efficiency = 1.157/1.3 = 89%**



# The Symbols to Code for Each Block ...

56	-14	3	-1	0	0	0	0
-1	-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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Each DCT block is represented as a sequence of (run, level) pairs, notably

**DC: (0, 56)**

**AC1: (0, -14)**

**AC2: (0, 1)**

**AC4: (1, -1)**

**AC5: (0, 3)**

**AC6: (0, -1)**

**AC7: (0, -1)**

**EOB (End of Block)**

Small *runs* and small *levels* are highly probable !

These pairs/symbols have to be converted into bits considering their statistics.

# Entropy Coding: Intermediary Symbols

Each *(run, level)* pair associated to a non-null AC coefficient is represented by a pair of symbols, *Symbol 1* and *Symbol 2*:



- ★ *Run* - number of null DCT coefficients preceding the coefficient being coded in the zig-zag scanning
- ★ *Size* – number of bits used to code the *Level* (this means *Symbol 2*)
- ★ *Level* – quantization level of the AC coefficient to be coded

Each DC coefficient is represented in the same way, with the *run* equal to zero.

# The Symbols to Code for Each Block ...

Each DCT block is represented as a sequence of (run, level) pairs, notably

56	-14	3	-1	0	0	0	0
1	-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	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

**DC: (0, 56)**

**AC1: (0, -14)**

**AC2: (0, 1)**

**AC4: (1, -1)**

**AC5: (0, 3)**

**AC6: (0, -1)**

**AC7: (0, -1)**

<i>Run</i>	<i>Size</i>	<i>Level</i>
------------	-------------	--------------

**Symbol 1, Symbol2**

**Symbol 1, Symbol2**

**Symbol 1, Symbol2**

**Symbol 1, Symbol2**

**Symbol 1, Symbol2**

**Symbol 1, Symbol2**

**Symbol 1, Symbol2**

**EOB (End of Block)**

# Coding Tables for Symbols 1 and 2

	0	1	2	Size	9	10
<b>Runlength</b>	0	EOB	Run-size values			
	.	X				
	.	X				
	.	X				
	15	ZRL				

Size	Amplitude
1	-1, 1
2	-3, -2, 2, 3
3	-7 ... -4, 4 ... 7
4	-15 ... -8, 8 ... 15
5	-31 ... -16, 16 ... 31
6	-63 ... -32, 32 ... 63
7	-127 ... -64, 64 ... 127
8	-255 ... -128, 128 ... 255
9	-511 ... -256, 256 ... 511
10	-1023 ... -512, 512 ... 1023

*Symbol 1: (run, size)*

**Bidimensional**

**Huffman coding**

with some  
pragmatism for runs  
longer than 15 !

*Symbol 2: (level)*

**VLI (Variable Length  
Integer) coding**

# Symbol 2 (Level) VLI Coding Example



Symbol 1 - Huffman (bidimensional)

Symbol 2 - VLI

0000 -15

0001 -14

0010 -13

0011 -12

0100 -11

0101 -10

0110 -9

0001 -8

1000 8

1001 9

1010 10

1011 11

1100 12

1101 13

1110 14

1111 15



1100

+12 in binary

after 'inverting' all bits

The code for negative values is simply the 'inversion' of the code for positive values.



1100

+12 in binary

# Entropy Coding: Summary

(run, level) =>  
for each non-null DCT coefficient



Symbol 1 - Huffman (bidimensional)

Symbol 2 - VLI

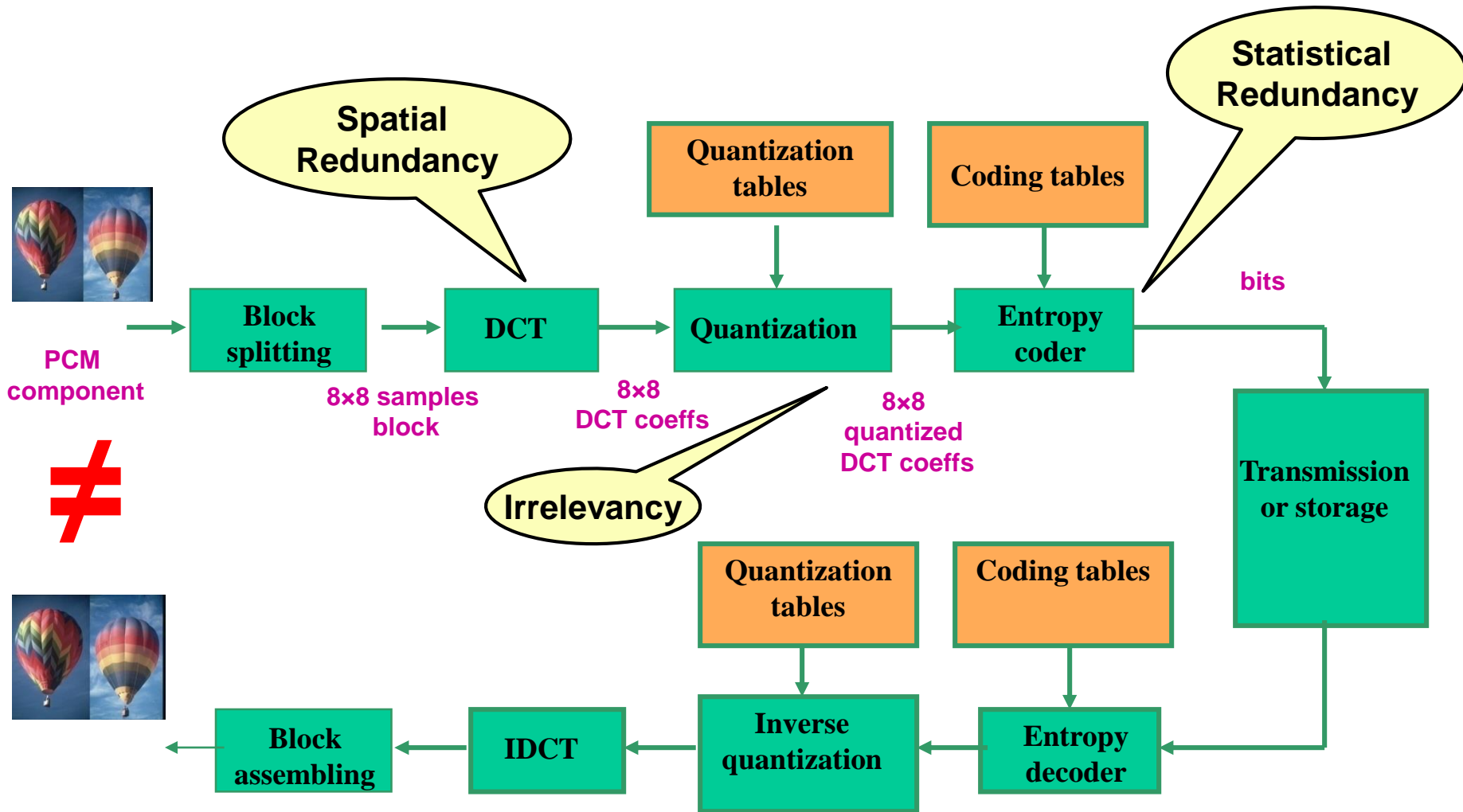
Symbol 1 - Huffman table

Symbol 2 – VLI table

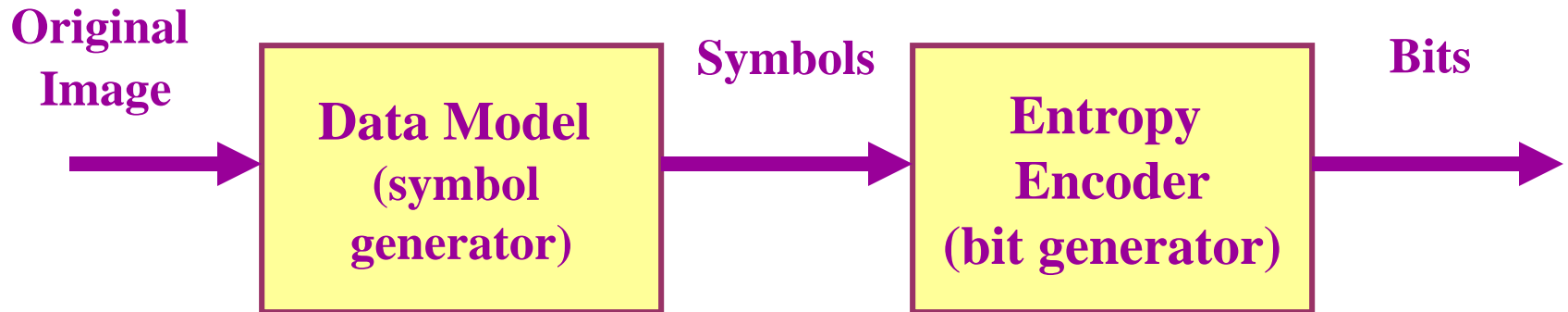
Runlength	0	1	2	Size	9	10
	EOB		Run-size values			
.	X					
.	X					
.	X					
15	ZRL					

Size	Level
1	-1, 1
2	-3, -2, 2, 3
3	-7 ... -4, 4 ... 7
4	-15 ... -8, 8 ... 15
5	-31 ... -16, 16 ... 31
6	-63 ... -32, 32 ... 63
7	-127 ... -64, 64 ... 127
8	-255 ... -128, 128 ... 255
9	-511 ... -256, 256 ... 511
10	-1023 ... -512, 512 ... 1023

# DCT Based Image Coding



# JPEG (Baseline) Coding Model



**JPEG Model:** An image is represented as a sequence of (almost) independent  $8 \times 8$  samples blocks with each block represented by means of a zig-zag sequence of quantized DCT coefficients using *(run, level)* pairs, terminated by a *End of Block*.

# Compression versus Quality



**JPEG offers the following levels of compression/quality for sequential DCT based coding, considering colour images with medium complexity:**

- ★ **0.25 - 0.5 bit/pixel – medium to good quality; enough for some applications**
- ★ **0.5 - 0.75 bit/pixel – good to very good quality; enough for many applications**
- ★ **0.75 - 1.5 bit/pixel – excellent quality; enough for most applications**
- ★ **1.5 - 2.0 bit/pixel – transparent quality; enough for the most demanding applications**

**These compression/quality levels are only indicative since the compression always depends on the specific image content, notably if there is more or less spatial redundancy and irrelevancy.**

**The quality level may be controlled through the quantization steps.**

# JPEG Test Images



**Barb 1**



**Barb 2**



**Board**



**Boats**



**Hill**



**Hotel**



**Zelda**



**Toys**

# Performance Results

Imagem	Coef. DC Lum (byte)	Coef DC crom (byte)	Coef AC Lum (byte)	Coef AC Crom (byte)	Global (byte)	Factor Comp.	Ritmo (bit/pel)	SNR Y (dB)	SNR U (dB)	SNR V (dB)
Zelda	4208	2722	19394	3293	29617	28.00	0.571	38.09	42.01	40.98
Barb1	4520	2926	40995	4878	53319	15.56	1.028	33.39	38.38	39.01
Boats	3833	2255	29302	3755	39145	21.19	0.755	35.95	41.13	40.13
Black	3497	2581	21260	6015	33353	24.87	0.643	37.75	40.09	38.23
Barb2	4223	2933	41613	7246	56014	14.81	1.080	32.37	37.05	36.09
Hill	4007	2206	34890	3727	44830	18.50	0.865	34.31	39.83	38.09
Hotel	4239	2708	35520	6658	49125	16.88	0.945	34.55	37.95	36.99

# Summary: How Does JPEG Compress ?

## ★ Spatial Redundancy

- Image samples statistically dependent are converted into uncorrelated DCT coefficients with the signal energy concentrated into the smallest possible number of coefficients

## ★ Irrelevancy

- DCT coefficients are quantized using psychovisual criteria

## ★ Statistical Redundancy

- The symbol statistics is exploited using *run-length* coding and Huffman entropy coding (or arithmetic coding)

# JPEG Extensions

*Making JPEG more powerful and flexible*

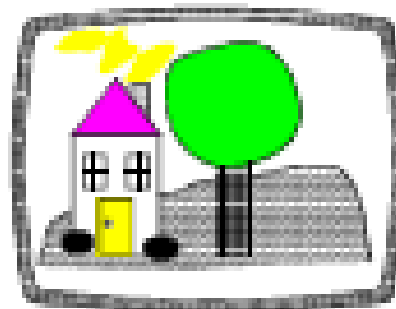
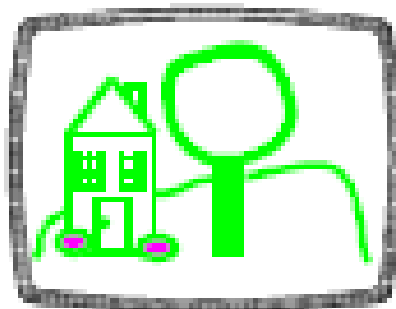
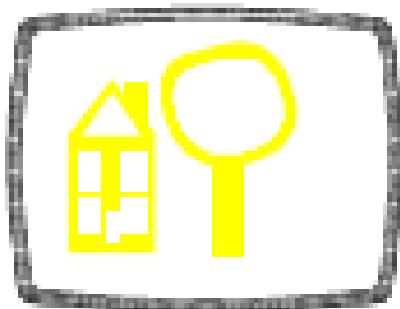


The various JPEG operational modes address the need to provide solutions for a large range of applications with different requirements.

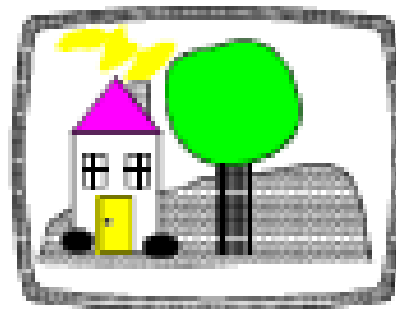
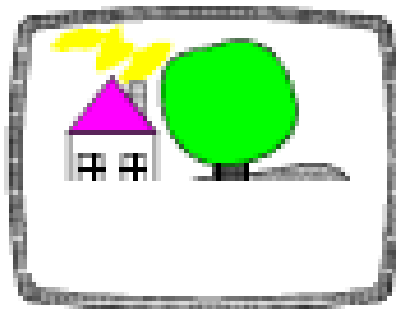
- ★ **SEQUENTIAL MODE** – Each image component is coded in a single scan (from top to bottom and left to right).
- ★ **PROGRESSIVE MODE (scalable)** - The image is coded with several scans which offer a successively better quality (but same spatial resolution).
- ★ **HIERARCHICAL MODE (scalable)** - The image is coded in several resolutions exploiting their mutual dependencies, with lower resolution images available without decoding higher resolution images.
- ★ **LOSSLESS MODE** – This mode guarantees the exact reconstruction of each sample in the original image (mathematical equality).

For each operation mode, one or more codecs are specified; these codecs are different in the sample precision (bit/sample) or the entropy coding method.

# Progressive versus Sequential Modes



Progressive

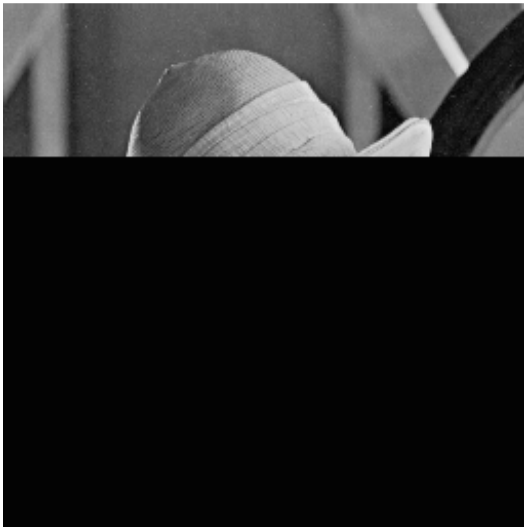
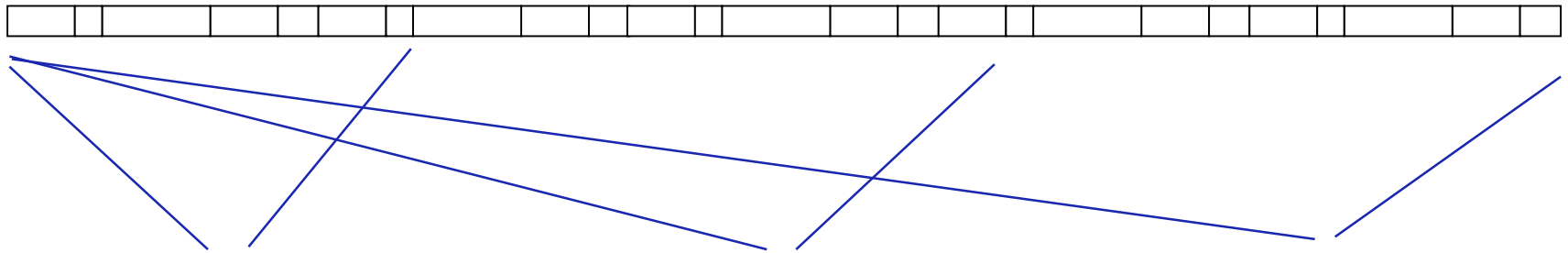


Sequential

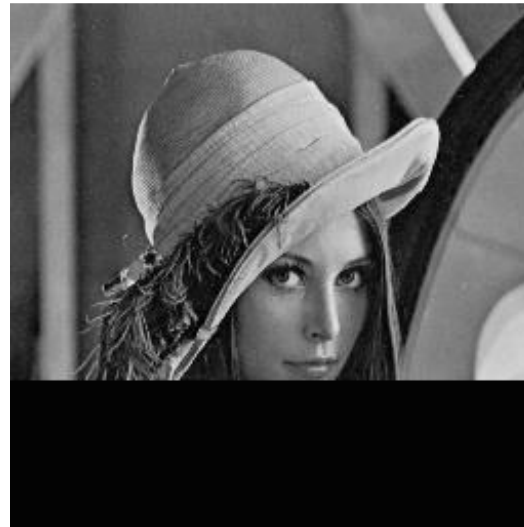
TG000733-63/2006

# Sequential Mode or No Scalability ...

NON scalable stream



Decoding 1



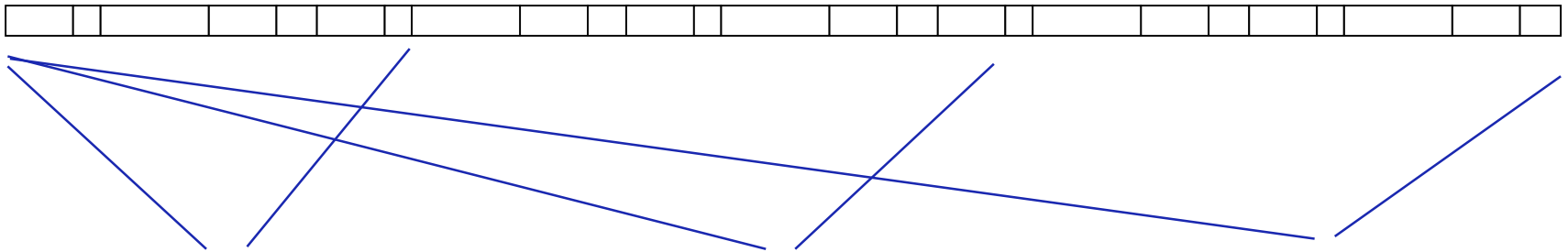
Decoding 2



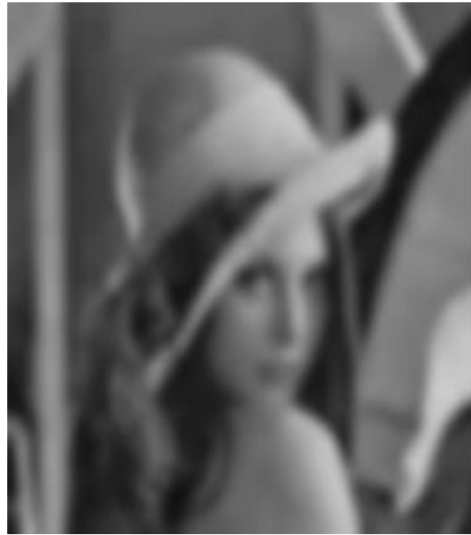
Decoding 3

# Progressively More Quality: Quality or SNR Scalability

Scalable stream



Decoding 1



Decoding 2



Decoding 3

# JPEG Progressive Mode



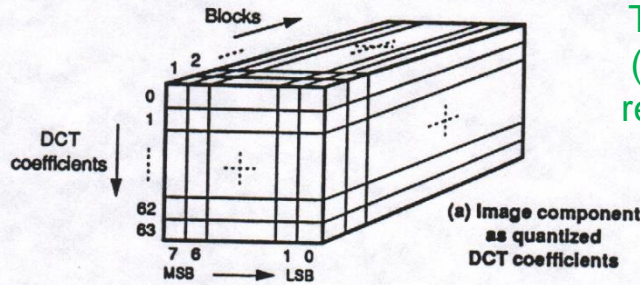
**The image is coded with successive scans. The first scan gives very quickly an idea about the image content; after, the quality of the decoded image is progressively improved with the successive scans (quality layers).**

**The implementation of the progressive mode requires a memory with the size of the image to store the quantized DCT coefficients (11 bits for the baseline process) which will be partially coded with each scan.**

**There are two methods of implementing the progressive mode:**

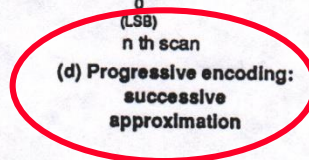
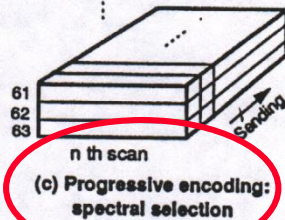
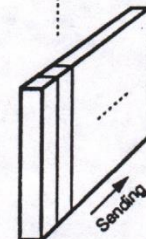
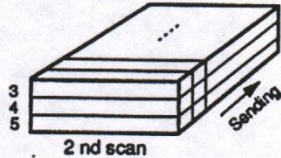
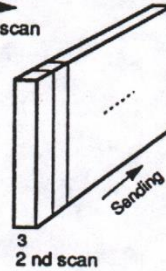
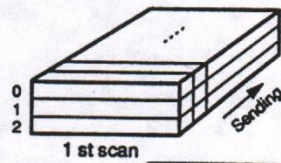
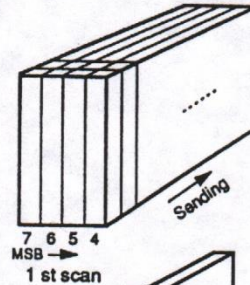
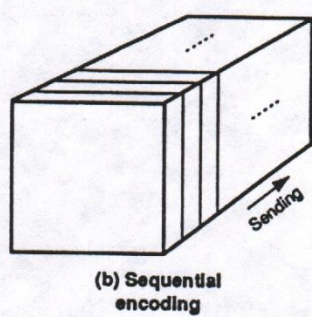
- ★ **SPECTRAL SELECTION** – Only a specified 'zone' of the DCT coefficients is coded in each scan (going from lower to higher frequencies)
- ★ **GROWING PRECISION** – DCT coefficients are coded with successively higher precision, bitplane after bitplane

**The spectral selection and successive approximations methods may be applied separately or combined.**



This cuboid includes all (quantized) information representing the image !

**Spectral selection:** Each layer brings an increasing number of DCT coefficients, and thus frequencies

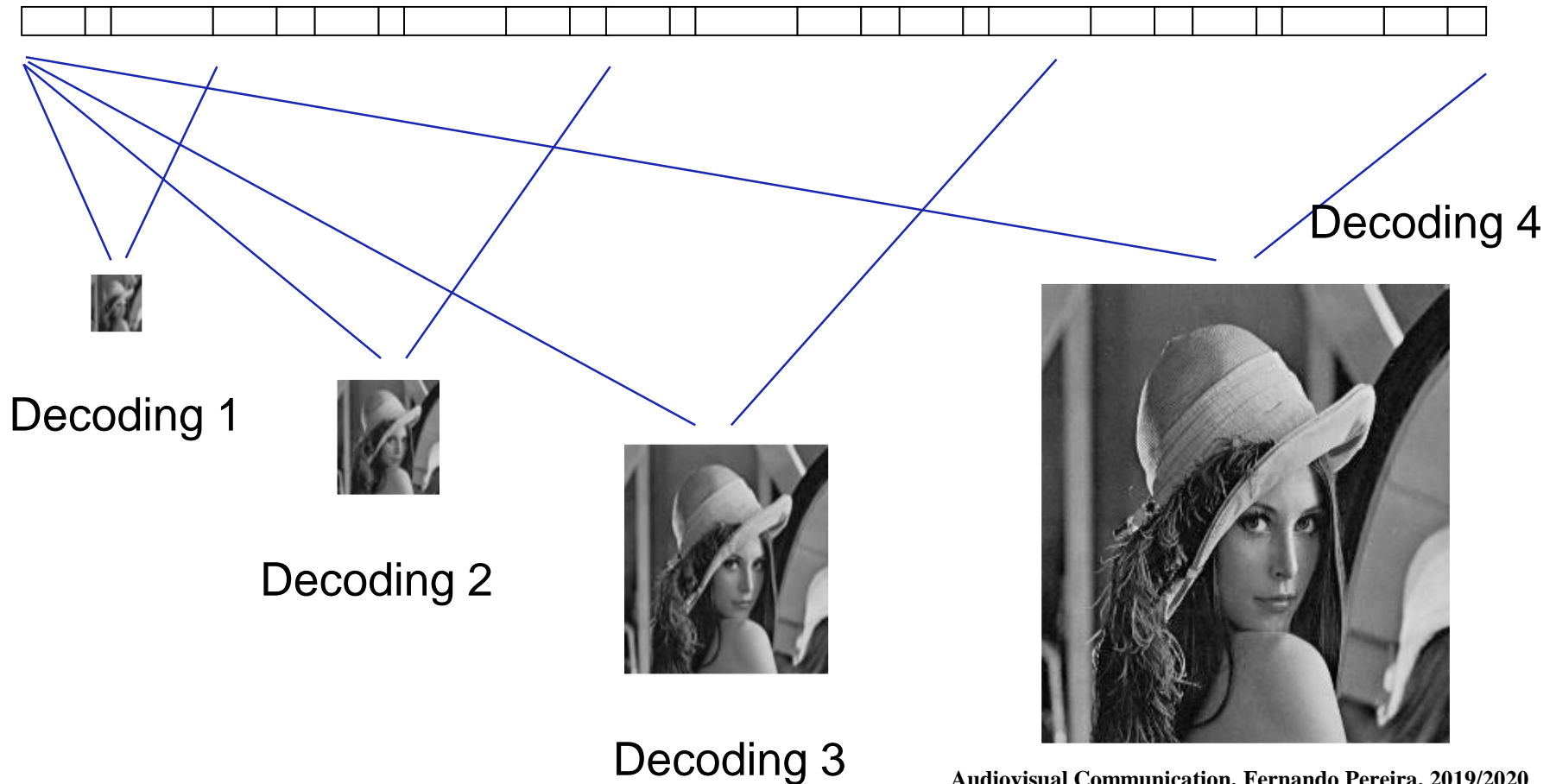


# Progressive Modes: Spectral Selection and Growing Precision

**Successive approximation:** Each layer brings an increasing (mathematical) precision for all coefficients

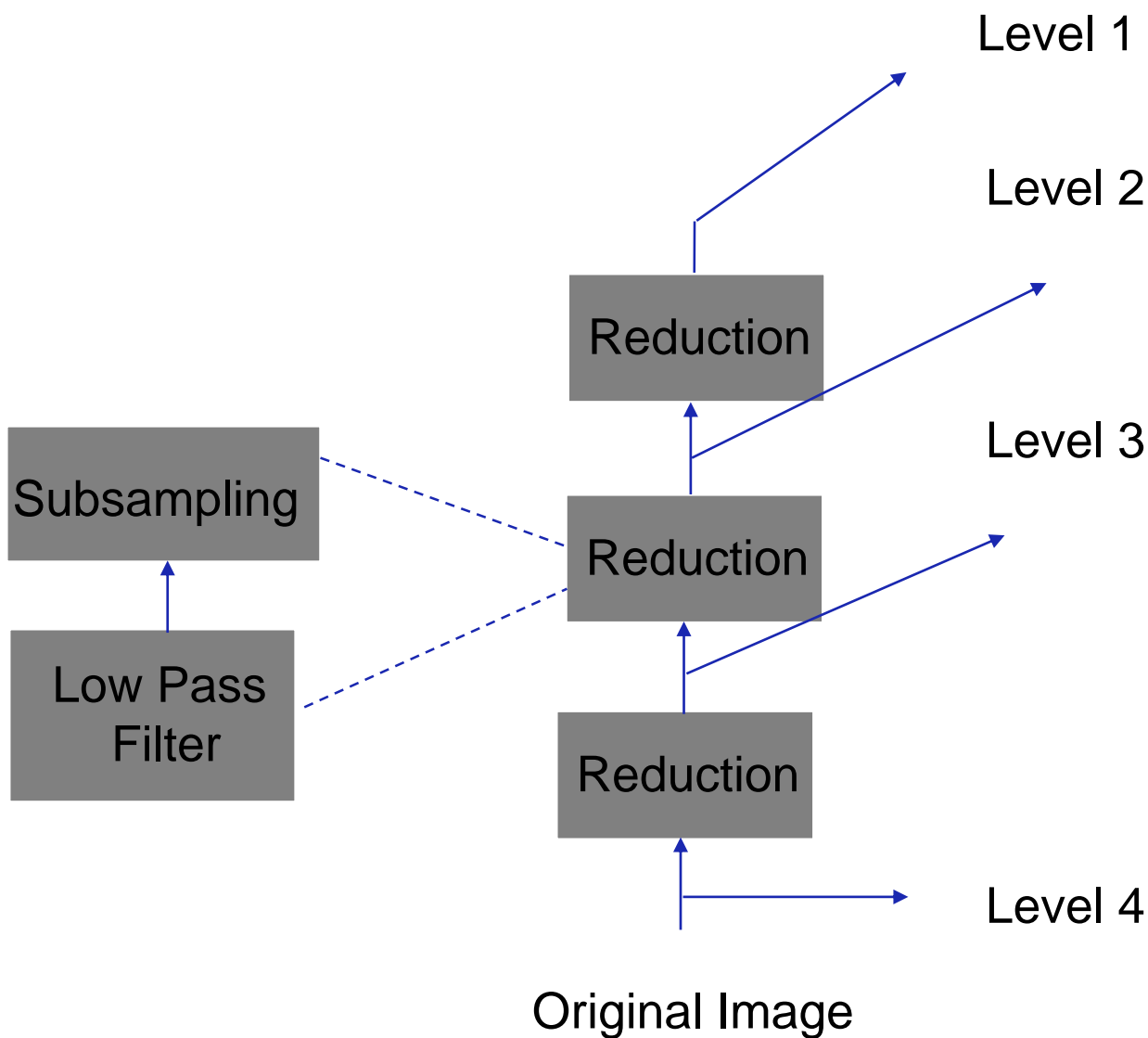
# Hierarchical Mode or Spatial Scalability ...

Scalable stream



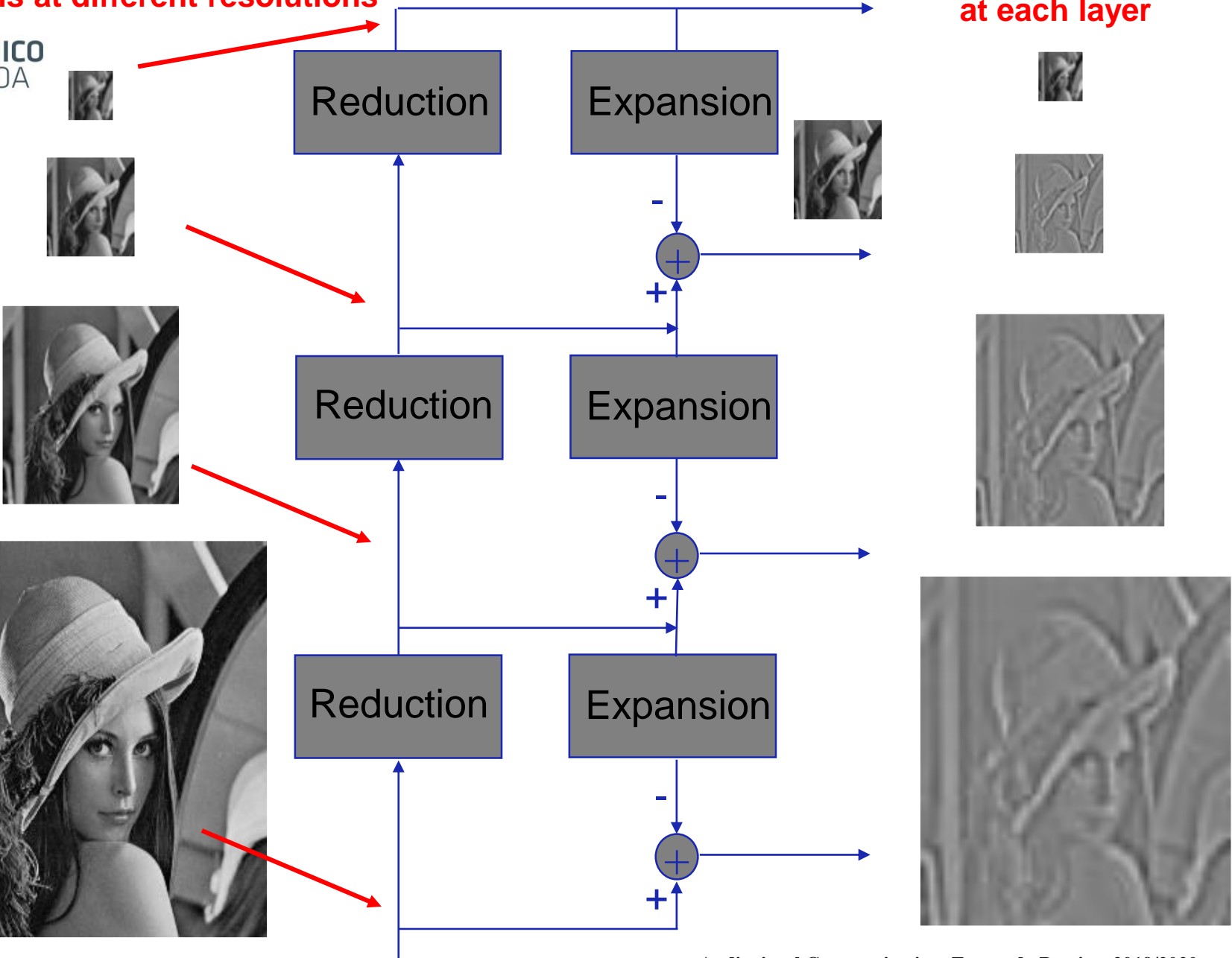
# Image Pyramid at Encoder

Originals at different resolutions



Originals at different resolutions

Additional information at each layer



Original Image

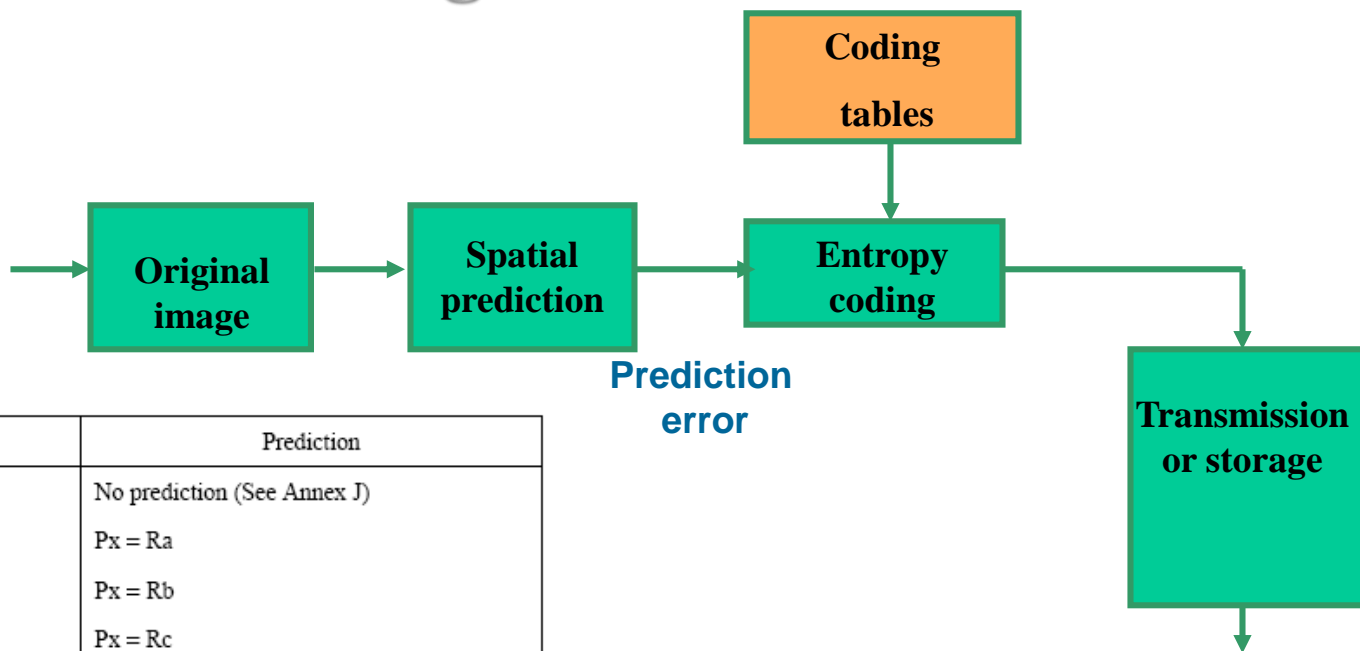


PERFECT

**The JPEG lossless mode is based on a spatial prediction scheme. The prediction combines the values of, at most, 3 adjacent samples. Finally, the *prediction mode* and the *prediction error* are coded.**

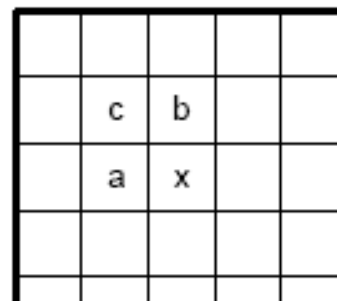
- ★ **This JPEG coding mode is rather popular for medical imaging.**
- ★ **The definition of a DCT based lossless mode would require a much more precise/rigid definition of the codecs, e.g. DCT implementation.**
- ★ **Two JPEG lossless codecs are specified, one using Huffman coding and another using arithmetic coding.**
- ★ **The codecs may use any precision between 2 and 16 bit/sample.**
- ★ **The JPEG lossless mode offers ≈ 2:1 compression for colour images of medium complexity.**
- ★ **There is also a JPEG-LS standard developed later, allowing to achieve better compression factors (≈ 3:1).**

# Lossless Coding



Selection-value	Prediction
0	No prediction (See Annex J)
1	$P_x = R_a$
2	$P_x = R_b$
3	$P_x = R_c$
4	$P_x = R_a + R_b - R_c$
5	$P_x = R_a + ((R_b - R_c)/2)^{a)}$
6	$P_x = R_b + ((R_a - R_c)/2)^{a)}$
7	$P_x = (R_a + R_b)/2$

a) Shift right arithmetic operation



***x is the sample to code***

- $P_x$  is the prediction
- $R_a$ ,  $R_b$ , and  $R_c$  are the reconstructed samples immediately to the left, above, and diagonally to the left of the current sample

# What Makes a Compression Technology Successful ?

- ★ Adoption in a standard
- ★ Compression performance
- ★ Encoder and decoder complexity
- ★ Error resilience
- ★ Random access
- ★ Scalability
- ★ Added value regarding alternative solutions/standards
- ★ Patents and licensing issues
- ★ Adoption companies
- ★ Marketing issues
- ★ ...



# Image Coding: Multiple Solutions

## JPEG Standards

- ★ JPEG - DCT-based transform coding
- ★ JPEG-LS - Lossless coding
- ★ JPEG 2000 - Wavelet-based coding
- ★ JPEG XR - Lapped biorthogonal-based transform coding

JPEG



JPEG  
2000

## Other Solutions

- ★ Fractal-based coding
- ★ Vector quantization coding
- ★ GIF, TIFF, PNG
- ★ H.264/AVC Intra, HEVC Intra

JPEG is nowadays almost a *raw format* as many cameras don't even make available the PCM format.

- ★ **JPEG: Still Image Data Compression Standard**, William Pennebaker, Joan Mitchell, Kluwer Academic Publishers, 1993
- ★ **Image and Video Compression Standards: Algorithms and Architectures**, Vasudev Bhaskaran and Konstantinos Konstantinides, Kluwer Academic Publishers, 1995
- ★ **Digital Image Compression Techniques**, Majid Rabbani, Paul W. Jones, SPIE Press, Tutorial texts on Optical Engineering, 1991