FUNDAMENTALS OF IMAGE AND VIDEO PROCESSING

Part 8: Image Coding (short version)

Why coding images

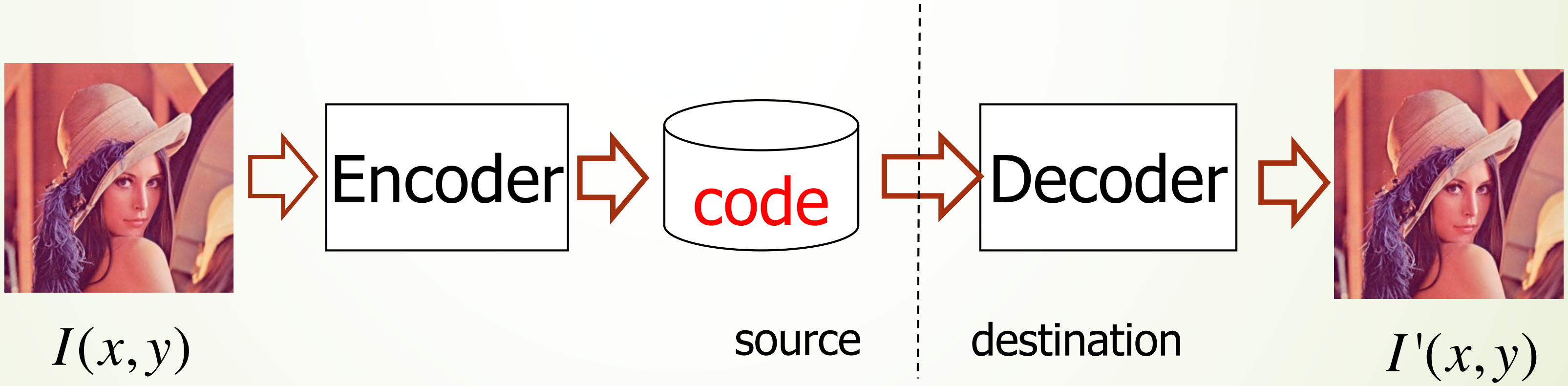
- Visual data are large object to store or transmit
  - Consider an image at HDTV resolution (1920x1080 pixels), 24 bits per pixel (8xRGB): in raw format it will require 1920x1080x24 = **49766400 bits** (approximately 6Mbytes)
  - If we have a sequence of such images, at 30fps progressive, the bitrate that we get is about **1.5 Gbps** (gigabit per second)
- Big problems both for transmission and storage!
- To allow transmitting an image in realtime or streaming a video we need to reduce the above amounts of information of 1-2 orders magnitude
  - This can be done thanks to **image and video compression**
  - The objective of compression is to obtain media with similar quality using a significantly lower number of bits

How to compress

- This goal may be achieved by exploiting some intrinsic (and inter-related) characteristics of image and video data:
  - **Low frequency content**: images are a low-pass signals, their frequency content is limited
  - **Slow variation**: image signals are locally stationary, they change slowly along space (and time, for video), except for specific transition points (edges), which are however quite sparse
  - **High-correlation**: if we measure the spatial (temporal) correlation of an image (video) signal, we see that it is a long-memory process
  - **Predictability**: future values of image signals can be often predicted with high accuracy from previous value. This is even more evident for video signals along time
  - **Self-similarity**: parts of the image are sometimes repetitive, even if they show significant variations (think about textures)

CoDec

- We call CoDec (acronym of Coder+Decoder) a system that transforms an image in a coded (compressed) format and then back to an image
  - The compression module (**encoder**) generates a code that (typically) requires less bits than the original image
  - The decompression module (**decoder**) reconstructs an image (close to, but in general different from the original) from the code



NB. The code is not an image (pixel), but a set of data that is "meaningful" only for an appropriate decoder

CoDec performance

- We typically measure the performance of a CoDec using two parameters:
  - **Compression factor**: it is expressed by **quantitative parameters** that indicate how effective the compression was in terms of bit reduction
  - **Quality**: it is expressed by **quantitative or qualitative parameters** that indicate how accurate is the reconstruction I'(x,y) with respect to the original image I(x,y)

NB. While the compression can be measured exactly, the quality is more difficult to evaluate, as it is influenced by subjective factors (human perception)

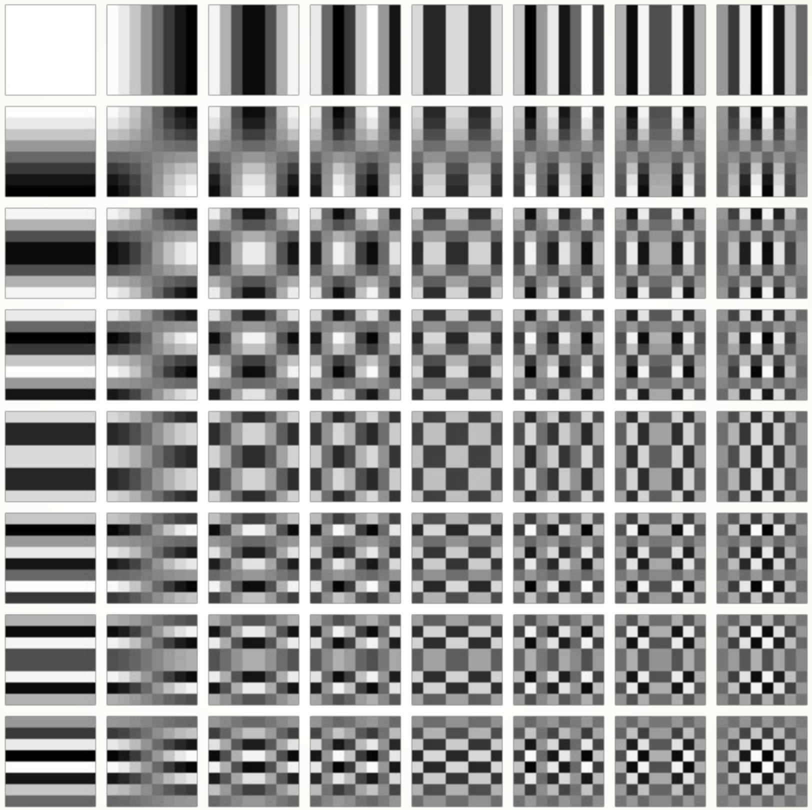
Transform coding

- The term transform coding is associated to a large class of methods that use as a primary tool the transformation of the image in a different domain
  - The transform we have learnt in this course is Fourier, but there are many other transforms that are more suitable for image coding
  - The two most important ones are DCT and DWT (we'll see them)
- Different methods are characterized by:
  - Which image transform they adopt
  - How the transform is applied (whole image, blocks)
  - How the transformed coefficients are coded
- Tipically the three things are related each-other

Transform coding using DCT

- A largely used transform in image coding is DCT
- DCT stands for Discrete Cosine Transform. It is a rather 'young' transform, invented in 1972 by Nasir Ahmed
- DCT is quite similar to DFT but:
  - It is real and not complex (simpler and better for compression)
  - It is more effective in compressing the signal energy in few coefficients
  - It can be calculated with integer computation and it is very fast
- From a mathematical viewpoint, DCT is a transformation using real cosine waves at multiples of a bases frequency along x and y as basis functions
  - In the following slide the basis images of the DCT are shown

2D DCT basis functions



8x8 2D-DCT basis functions (64 8x8 images)

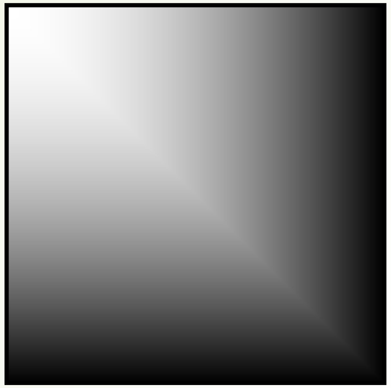
DCT: Example

- This is how the DCT of Lena appears:
  - Significant coefficients are very few, concentrated in the top-left corner (the DC coefficient)
  - The number of coefficients is equal to the image size (as for the DFT)

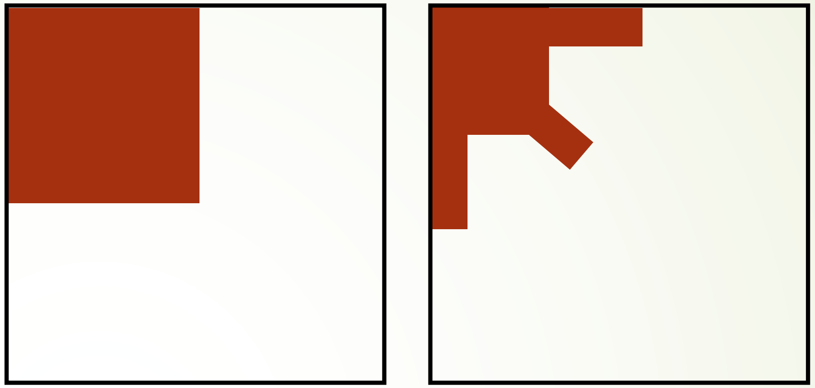
Coding of transformed image

- Since the coefficients are sparse and collected into a small portion of the transformed images, there are 2 main ways to encode them efficiently:
  - **Zonal coding**: only the coefficients belonging to the area that most probably will contain significant values are saved, all the others are set to zero
  - **Threshold coding**: only the coefficients that have a value above a given threshold are saved, all the others are set to zero
- Both methods have **pros and cons**:
  - Zonal: does not require storing the position of saved coefficients (more effective in compression) BUT introduces a sever blurring
  - Threshold: more adaptive (better quality) but wastes a lot of bits to store the coefficients' positions

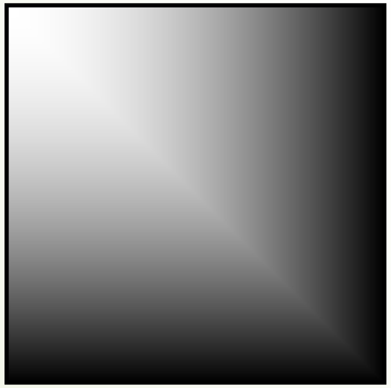
Zonal vs. threshold coding



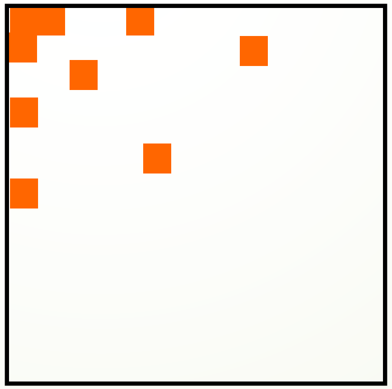
coefficients image



a couple of typical zonal masks



coefficients image



1100100000000000000
1000000000001000000
0010000000000000000
1000000000000000000
0000001000000000000
1000000000000000000
0000000000000000000
0000000000000000000
0000000000000000000

selected coefficients and relevant bit-map

The JPEG standard

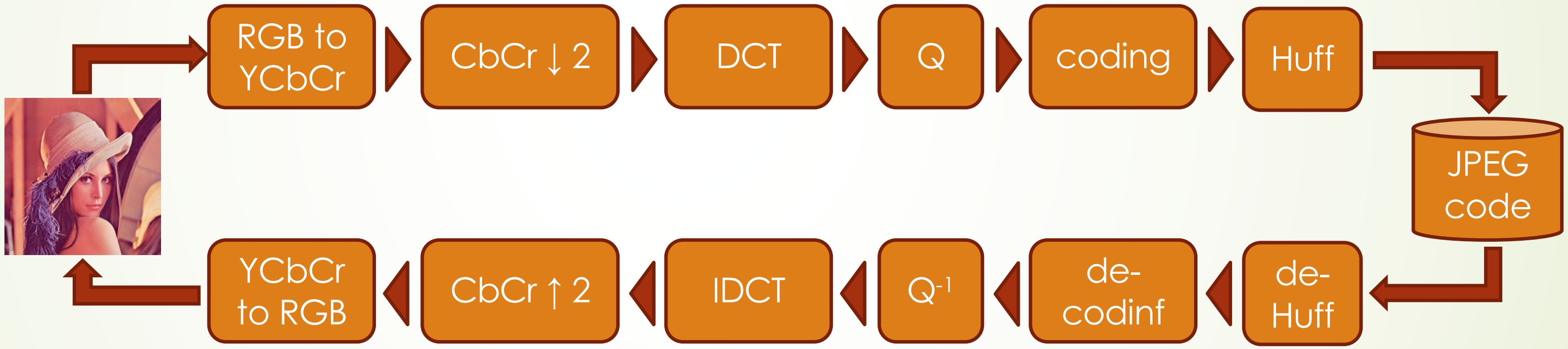
- JPEG is the acronym of **Joint Photographic Experts Group**, a Committee established by ISO (International Standardization Organization) in 1986 and still active
- The first objective of the Committee was to standardize a technique for still picture coding
- The standard known as JPEG (.jpg format) has been released in 1992 and collected the best of the state-of-art worldwide in image compression
- It has been (and still is) extremely successful and widely used, even if several more efficient coders have been made available since then (e.g., JPEG XR, JPEG2000, JPEG XS, etc.)
- It is used on digital cameras, smartphones, social networks, web, professional/amateur repositories, medical archives, etc.

JPEG standard

- As all standards it is formulated in a very detailed way
  - To ensure interoperability, the format should be very precise
  - The encoder instead, has some degree of freedom
- It includes variants that are rarely implemented (commercial implementations are typically subsets of the standard)
- The target are photographic pictures (photos), but it works also on professional pictures (e.g., biomedical, remote sensing, documents) to some extent
- The target compression is around 1:10-20, with acceptable quality (low artifacts, PSNR > 30 dB)
- It is easy-to-use by design (it has a single parameter)

JPEG: The algorithm

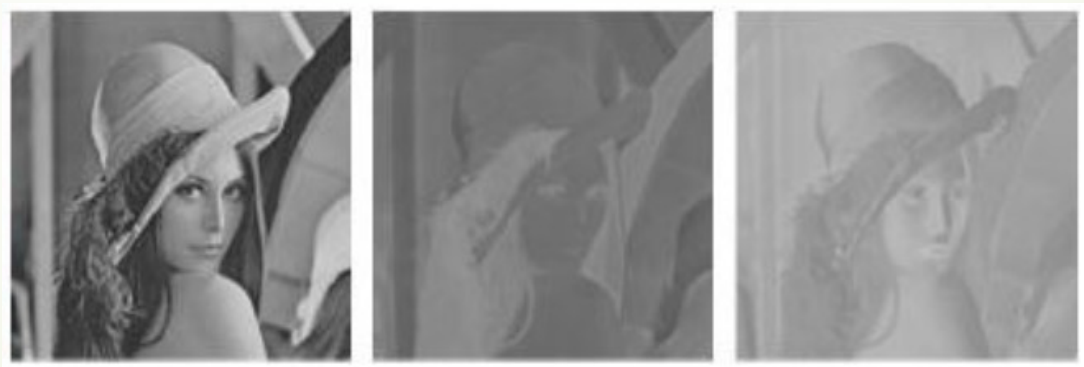
- We will just focus on the algorithmic part of the standard
  - JPEG is just a sophisticated **block transform coder** using DCT
  - The co-dec scheme is depicted below:



- We will analyze step by step the encoding process
  - It will be also useful to review some processing techniques we've seen along the course

JPEG: step by step

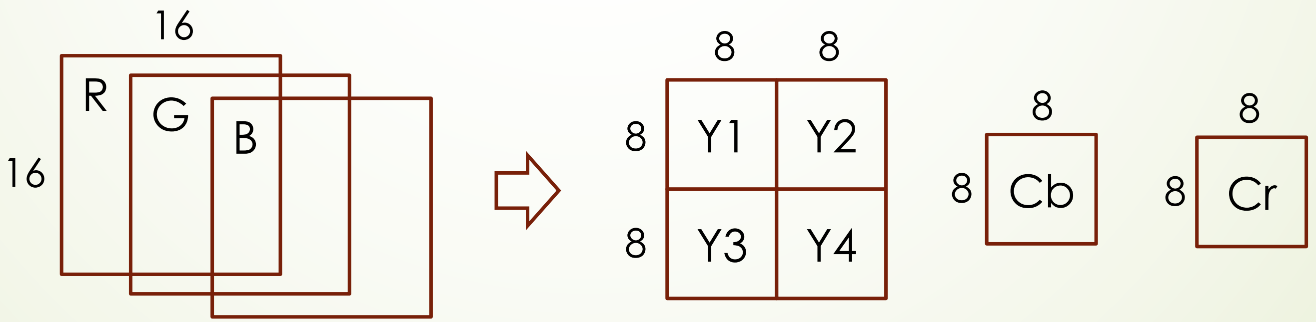
- STEP1: **Chromatic conversion**
  - JPEG takes in input standard RGB images
  - RGB representation is redundant (there is a significant correlation among color components)
  - Since we would like to remove redundancy as much as possible, JPEG converts the image in YCbCr format
  - In YCbCr (similar to YUV used in old TV systems), Y is the luminance component, while Cb and Cr are chromatic components and contain less information



Y Cb Cr

JPEG: step by step

- Chromatic components downsampling
  - Our visual system is less sensitive to Cb and Cr spatial variations, then we can downsample them
  - Both images are downsampled by a factor 2 along x and y
  - For a NxM input image, we obtain 3 images: Y (NxM), Cb,Cr (N/2, M/2)
- Then, the image is divided into 8x8 blocks.
  - Given the different resolution, for every 4 Y blocks we will have 1 Cb and 1 Cr block (this represents a 16x16 image block)
  - This configuration is called **4:1:1 scheme**



JPEG: step by step

- DCT Transform
  - Every 8x8 block (independently if Y or CbCr) is transformed by DCT
  - The transform is calculated in integer form using a look-up table
  - The upper left coefficient (DC)will be proportional to the mean value
  - The remaining coefficients will contain increasing frequencies and will typically show sharply decreasing values

$$f(j,k)=\begin{bmatrix}139&144&149&153&155&155&155&155\\144&151&153&156&159&156&156&156\\150&155&160&163&158&156&156&156\\159&161&162&160&160&159&159&159\\159&160&161&161&162&155&155&155\\161&161&161&161&160&157&157&157\\162&162&161&163&162&157&157&157\\162&162&161&161&163&158&158&158\end{bmatrix} \Rightarrow F(u,v)=\begin{bmatrix}1260&-1&-12&-5&2&-2&-3&1\\-23&-17&-6&-3&-3&0&0&-1\\-11&-9&-2&2&0&-1&-1&0\\-7&-2&0&1&1&0&0&0\\-1&-1&1&2&0&-1&1&1\\2&0&2&0&-1&1&1&-1\\-1&0&0&-1&0&2&1&-1\\-3&2&-4&-2&2&1&-1&0\end{bmatrix}$$

pixel-domain values DCT-domain values

JPEG: step by step

- Quantization of coefficients
  - To exploit the energy compaction properties of the DCT, we need to select significant coefficients
  - JPEG does not use either zonal or threshold coding, but rather a combination of the two
- Coefficients are integer-divided by a pre-defined matrix Q, whose values are arranged according to the coefficient position
  - Before dividing, matrix Q is multiplied by a factor q, which determines the impact of the operation and then the compression
  - Most coefficients, especially in the high-frequency area, are cut. The remaining ones are strongly quantized

$$F_Q(u,v) = \left\lfloor \frac{F(u,v)}{q \cdot Q(u,v)} \right\rfloor$$

JPEG: step by step

- The standard Q matrix is the following: $$Q(u,v)=\begin{bmatrix}16 & 11 & 10 & 16 & 24 & 40 & 51 & 61\\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55\\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56\\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62\\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77\\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92\\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101\\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99\end{bmatrix}$$
- It has been defined as to produce effective compression, while at the same time preserving the perceptual quality of the image
  - Coefficients have been refined using panels of users
  - It may be customized by the specific implementation

$$F(u,v)=\begin{bmatrix}1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1\\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1\\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0\\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0\\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1\\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1\\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1\\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0\end{bmatrix} \xrightarrow{q=1} F_Q(u,v)=\begin{bmatrix}79 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

JPEG: step by step

- After quantization:
  - The DC coefficient has usually larger values, and it is the only one to maintain a correlation across blocks
  - The AC coefficients are very sparse and usually located near to the upper left corner
- The coding block treats differently DC and AC coefficients:
  - The DC coefficient is coded in DPCM with the previous one to exploit residual correlation
  - AC coefficients are zig-zag scanned and coded with special keywords using RLE and coefficient categories (we'll see in detail)

NB. As you can see, JPEG is indeed a combination of different methods: it includes transform coding, RLE, DPCM and more

JPEG: step by step

- Zig-zag coding is used to order the coefficients in such a way that non-null values have higher probability to be scanned first, thus creating long sequences of zeros

$$F_Q(u,v) = \begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Scanned sequence: 0, -2, -1, -1, -1, 0, 0, -1, 0…

JPEG: step by step

- Each coefficient is associated to a 8 bit codeword 'NNNNSSSS'
  - the 4 bits **NNNN** encode the previous run of zeros in RLE (this allows representing runs from 0 to 15 in length)
  - the 4 bits **SSSS** encode the coefficient range (see table below): only 10 categories are needed as the maximum value is +/-1024

Category	Range of AC value
1	-1, +1
2	-3,-2,+2+,3
3	-7,…,-4,+4,…,+7
4	-15,…,-8,+8,…,+15
…	…
10	-1023,…,-512,+512,…,+1023

*NB. The specific value within the category is sent separately. The i-th category requires exactly i bits for indexing (**precision bits**)*

JPEG: step by step

- The number of codewords used is 16x10=160
- Part of the remaining codewords are used for special cases:
  - **Continuation code**: if a run is longer than 15, a run continuation code is sent with the word 11110000 (category 0 does not exist!)
  - **End of block**: after the last non-null coefficient, and EOB code is sent to indicate that nothing will follow, with codeword 00000000

In our example, the scanned sequence was: 0, -2, -1, -1, -1, 0, 0, -1, 0s

Then we will have:

run 1, category 2: 00010010

run 0, category 1: 00000001

run 2, catagory 1: 00100001

+ precision bits: 01, 0, 0, 0, 0

run 0, category 1: 00000001

run 0, category 1: 00000001

EOB: 00000000

JPEG Step by step

- Finally, some redundancy still remains in the codewords
  - Some of them are much more frequent than other
  - For instance, EOB is always present, lower categories and shorter runs are more probable, etc.
- To exploit this redundancy we can use an entropy coder
  - the standard uses an Huffman LUT
  - alternatives are also possible

In our example we will have:

11100101, 000, 000, 000, 110110, 1010

+ precision bits 01, 0, 0, 0, 0

Zero	Cat.	Code	Codeword
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
2	1	5	11011
2	2	8	11111000
3	1	6	111010
3	2	9	111110111
4	1	6	111011
5	2	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
11	1	9	111111010
EOB		4	1010

JPEG: Decoding

- Decoding will follow the same processing chain in inverse order (the technique is symmetric)
  - Apply Huffmans decoding to reconstruct NNNNSSSS codewords
  - Reconstruct the sequence and values of coefficients from NNNNSSSS codewords + precision bits, and place it into the block following the zig-zag order
  - De-quantize the block multiplying by the q·Q matrix
  - Apply the inverse DCT transform to reconstruct pixel values
  - Upsample Cb and Cr components
  - Apply the color transform YCbCr → RGB
- The decoded image will be different from the original
  - The two lossy (irreversible) operations are CbCr downsampling and especially, coefficients quantization

JPEG: Performance

- To compression factor is highly variable and depends indirectly on the q parameter
  - In JPEG we cannot set the compression, we set the desired quality. The resulting compression depends on the image content
- The code contains an header (negligible), the DPCM bits, the Huffman code bits, and the precision bits
  - In our example we have (for 1 image block):
    - *DPCM: depends on correlation with previous block, let's use an average of 4 bits*
    - *Huffman: 111001010000000001101101010, total 27 bits*
    - *Precision: 010000: 6 bits*
  - Total: 37 bits, instead of 8x8x8=512 bits
    - *Compression factor* $C_f = 512/37 = 13.8$
    - *Bitrate* $r_b = 8/13.8 = 0.58$ *bpp*

JPEG Performance

- The result of decoding in our sample block will be:

$$\hat{f}(j,k)=\begin{bmatrix}144 & 146 & 149 & 152 & 154 & 156 & 156 & 156\\ 148 & 150 & 152 & 154 & 156 & 156 & 156 & 156\\ 155 & 156 & 157 & 158 & 158 & 157 & 156 & 155\\ 160 & 161 & 161 & 162 & 161 & 159 & 157 & 155\\ 163 & 163 & 164 & 163 & 162 & 160 & 158 & 156\\ 163 & 163 & 164 & 164 & 162 & 160 & 158 & 157\\ 160 & 161 & 162 & 162 & 162 & 161 & 159 & 158\\ 158 & 159 & 161 & 161 & 162 & 161 & 159 & 158\end{bmatrix}$$

- Subtracting it from the original we obtain:

$$e(j,k)=\begin{bmatrix}-5 & -2 & 0 & 1 & 1 & -1 & -1 & -1\\ -4 & 1 & 1 & 2 & 3 & 0 & 0 & 0\\ -5 & -1 & 3 & 5 & 0 & -1 & 0 & 1\\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4\\ -4 & -3 & -3 & -1 & 0 & -5 & -3 & -1\\ -2 & -2 & -3 & -3 & -2 & -3 & -1 & 0\\ 2 & 1 & -1 & 1 & 0 & -4 & -2 & -1\\ 4 & 3 & 0 & 0 & 1 & -3 & -1 & 0\end{bmatrix} \Rightarrow$$

MSE = 5.1
PSNR = 41 dB

JPEG: Examples

- JPEG coding of Lena at maximum quality (100)

original

Cf = 3, PSNR=37.50 dB

JPEG: Examples

- JPEG coding of Lena at medium and low quality (50, 10)

C_f = 22, PSNR=29.81 dB

C_f = 55, PSNR=25.28 dB

*NB. **tiling** and **ringing artifacts** appear, due to block transform coding!*

JPEG2000

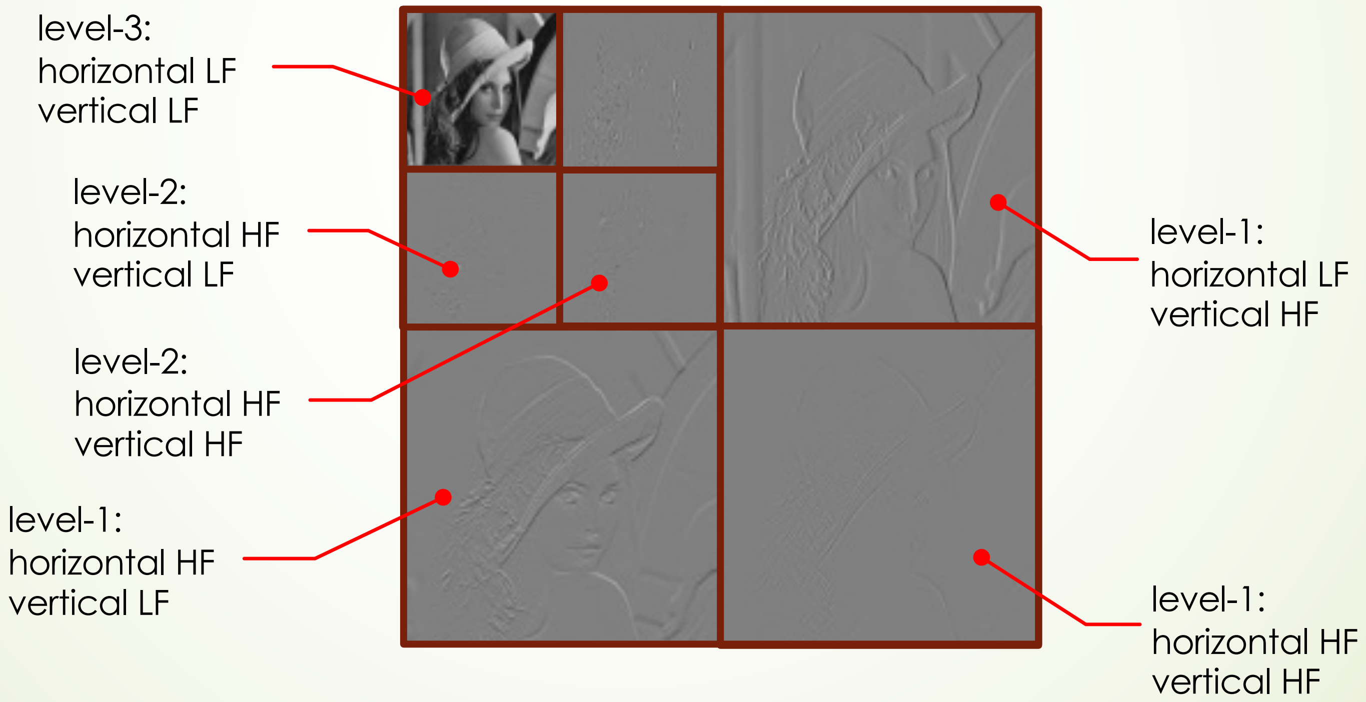
- JPEG2K is an evolution of JPEG standardized almost 10 year later
  - At that time, the main technological breakthrough in this field was the Wavelet Transform (DWT)
  - DWT is pretty different from DFT and DCT, in that it produces a transformed signal which is in a mixed **space-frequency domain**
  - Besides this, the wavelet domain is said to be better adapted to the characteristics of the human visual system
- This led to the development of a rather different approach of transform coding
  - Not only a different transform is used, with better performances, but also a number of **additional functionalities** are introduced (e.g., scalability, resilience, progressive reconstruction, regions of interest, …)

JPEG2000 and Wavelets

- A complete study of wavelets is out of the scope of this course
- A very simple introduction of the concept is the following:
  - We want to detect the presence of some frequency content in a given area of the image
  - We can do that by analyzing the image with appropriate functions (wavelets) containing localized frequencies at various intensities
  - Wavelets can be generated by scaling and translating a unique function, called **wavelet mother function**

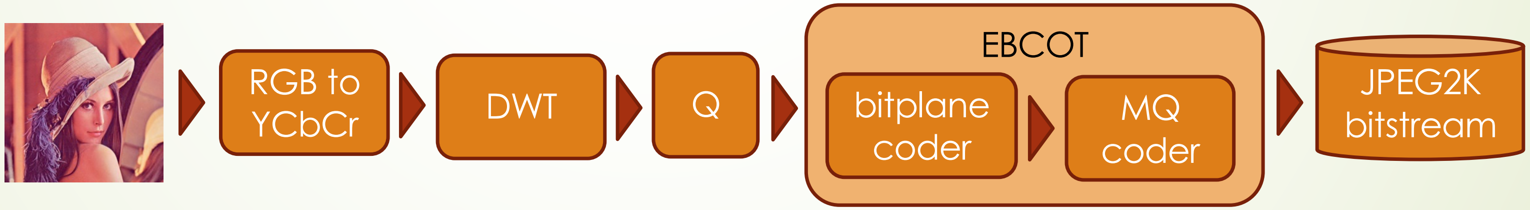
JPEG2000 and Wavelets

- In practical terms, what we do is to use special filters and to analyze the image at different scales factors
  - What we get is a kind of **pyramidal representation**

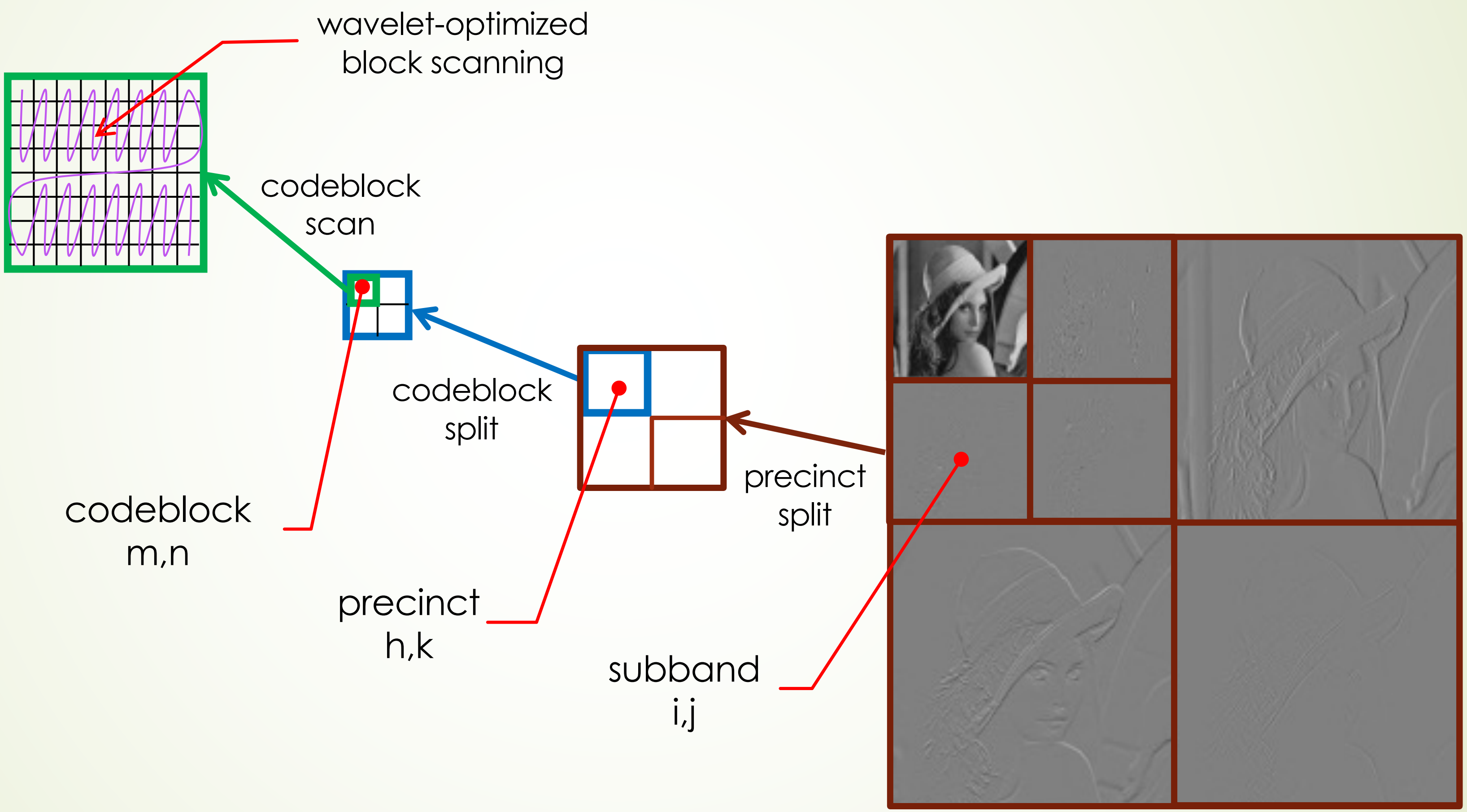


JPEG2000: Scheme

- The encoder is apparently quite similar to JPEG
  - The biggest difference is the encoder (EBCOT: embedded block coding with optimized truncation)
  - The transform is applied to blocks within each sub-band of the pyramidal structure
  - The bits associated to the various blocks have different importance according to the local content: an optimized allocation of bits allows maximizing the rate-distortion function for any given stream size

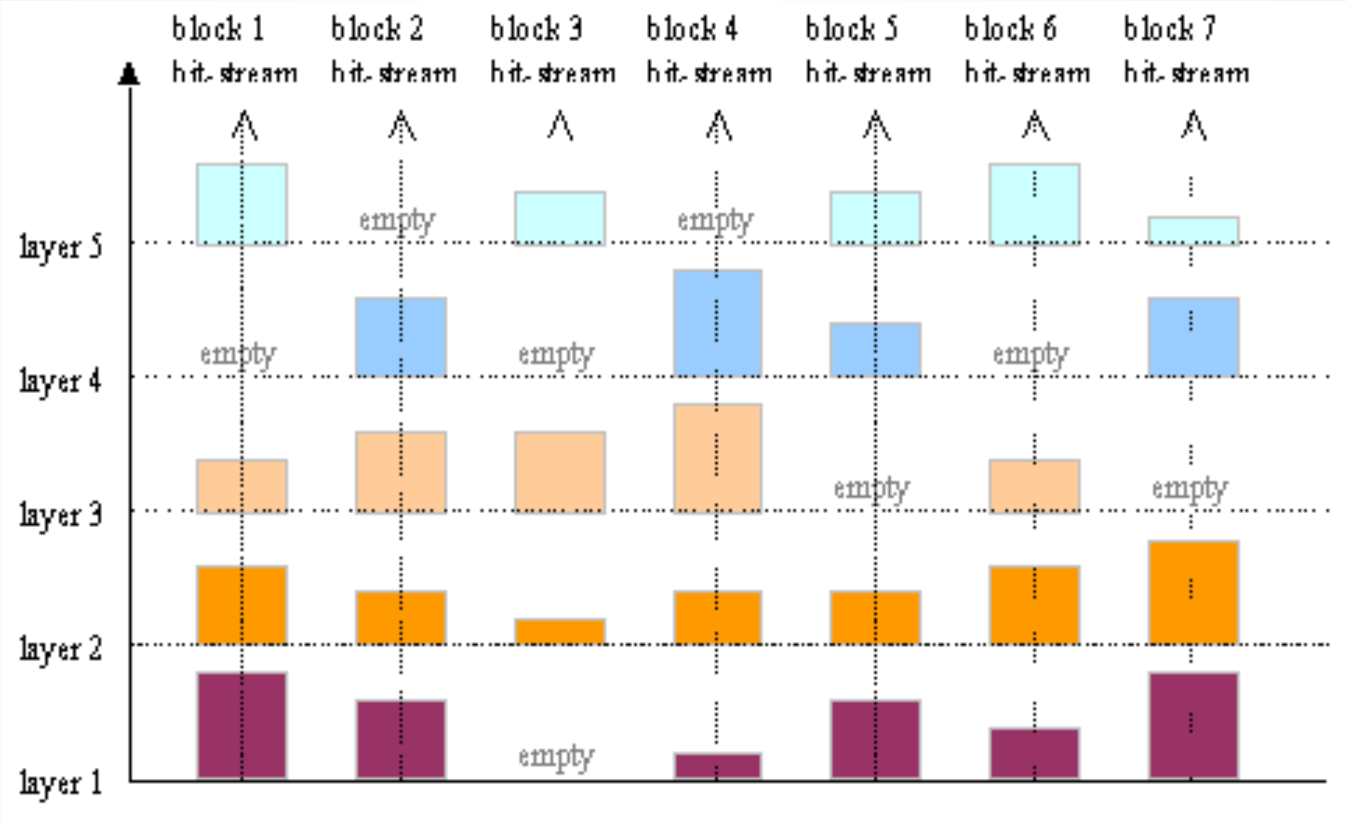


JPEG2000: Block coding



Wavelet decomposition

JPEG2000: Bit allocation



JPEG2000: Scalability and RoIs

- Thanks to the code structure we can transmit an image at different resolution or quality in a progressive way
- We can also transmit an image with differentiated quality (better quality in the area of interest)

Original with RoI

Reconstructed

JPEG2000: Performance

- JPEG2000 provides better visual and SNR quality wrt JPEG and allows to achieve much higher compression factors

JPEG at C_f 55

JPEG2000 at C_f 55

What we've learned in this section

- Images are huge data objects, they need compression
- Images can be compressed because they are redundant
- Coding may introduce degradation or not. If it does, we have to measure the image quality
- Quality is related to our (human) perception
- Coding without losing information has strong limitations
- Lossy coding allows larger compressions at the price of some image degradation (it's a trade-off)
- Current standards are based on sophisticated implementations of block transform coders
- Even if it has almost 30 years and there are better performing techniques, JPEG is still the most diffused image coding standard