JPEG-1 standard

© 1997-2015 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
http://cgg.mff.cuni.cz/~pepca/

JPEG-1 standard

- **Joint Photographic Experts Group** (‘86-’90)
  - multidiscipline board (ISO/CCITT)
- peak **compression ratio** at **good image quality**
  - user-controllable image quality
- universal application – arbitrary **images with continous waveform** (natural photographs)
  - should work well regardless of resolution, color depth or statistical characteristics (within „reasonable“ ranges)

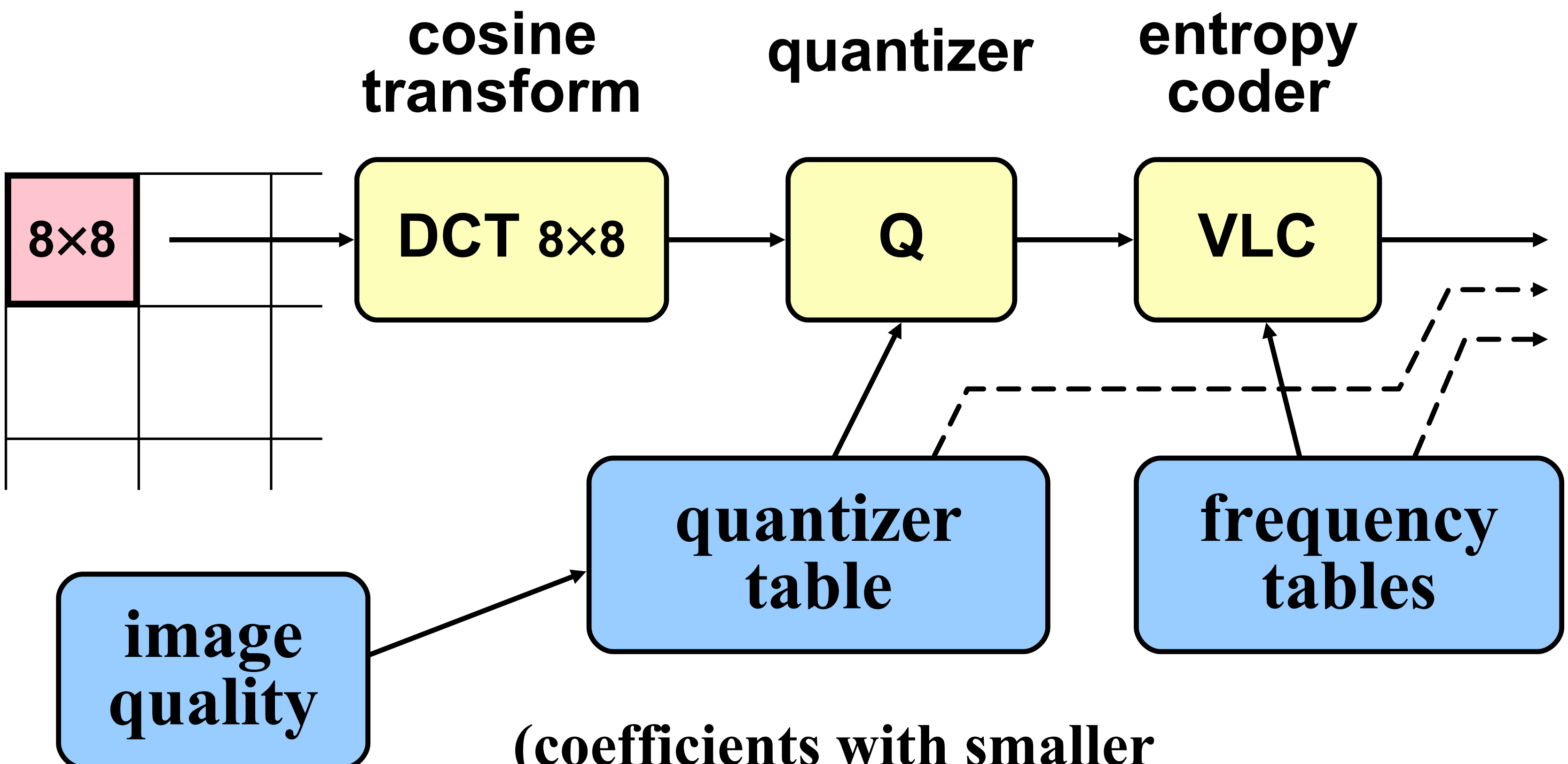
JPEG-1 standard

- reasonable **computing complexity** for SW & HW implementations
- four **coding modes**:
  - **losless** – less compression ratio (intended for science, medical data..)
  - **sequential coding** – single data pass in scanline order
  - **progressive coding** – several passes, gradual improvements of image quality
  - **hierarchical coding** – several image resolutions in one JPEG file, possibility of separate decoding

Selection of compression methods

- wide **selection procedure** (1987-88)
  - 12 methods were considered at the beginning
  - → narrowing candidate set to three best ones …
  - ⇒ and the winner was: **transform coding** based on DCT working on **8×8 pixel blocks**
- **quantizing** – physiologically optimized tables
- **channel coding** – Huffman or adaptive arithmetic codec (Q-coder patented by IBM)

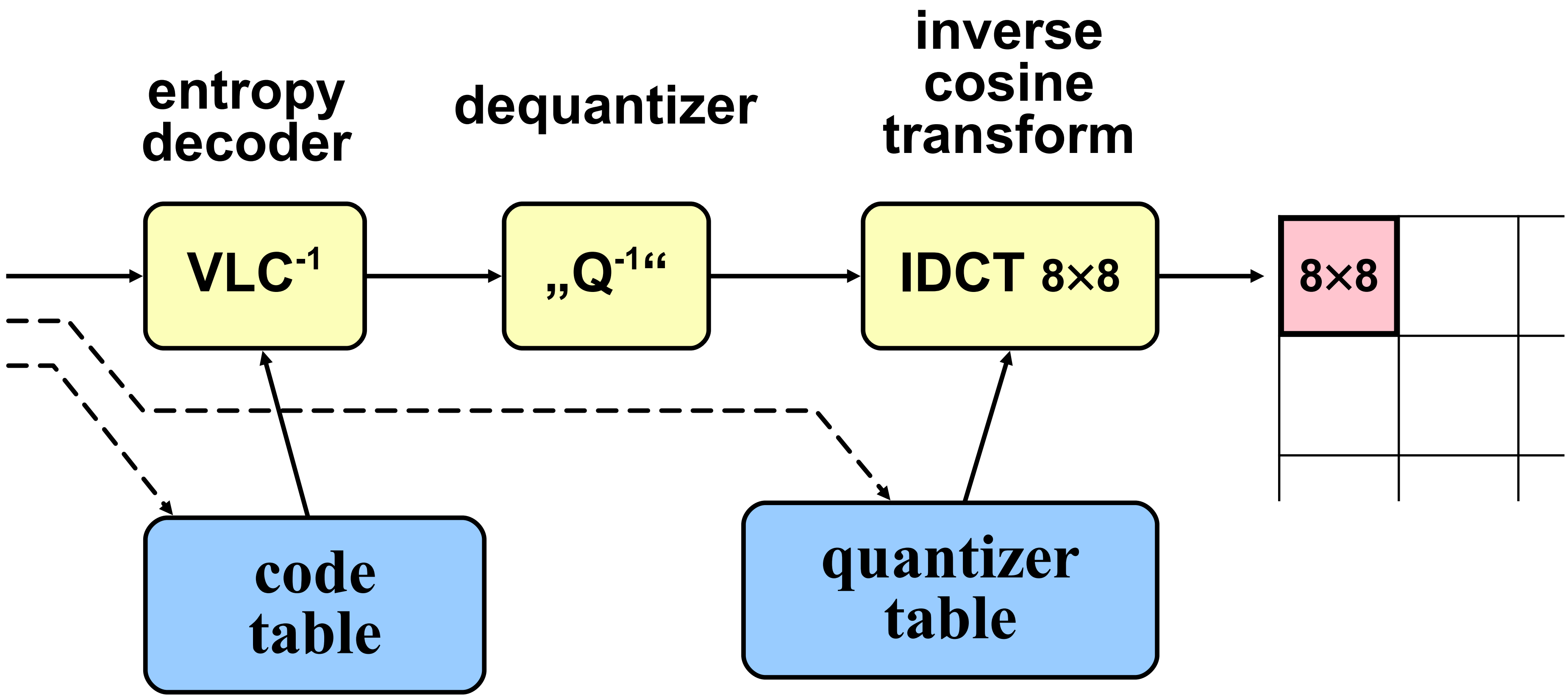
Lossy encoder (baseline JPEG-1)



(coefficients with smaller

i+j are preferred)

Lossy decoder (baseline JPEG–1)



Discrete cosine transform

Pre-conversion of all input samples – shifting values from interval $[0,2^{p}-1]$ to $[-2^{p-1},2^{p-1}-1]$.

$p=8$ or $p=12$

2D discrete cosine transform (DCT 8×8)

$$F(u,v) = C_u C_v \sum_{x,y=0}^{7} f(x,y) \cdot \cos\frac{\pi(2x+1)u}{16} \cos\frac{\pi(2y+1)v}{16}$$

$$C_u = \begin{cases} 1/2\sqrt{2} & \text{for } u=0 \\ 1/2 & \text{else} \end{cases}$$

Inverse cosine transform

2D inverse discrete cosine transform (IDCT 8×8)

$$f(x,y) = \sum_{u,v=0}^{7} C_u C_v \cdot F(u,v) \cdot \cos\frac{\pi(2x+1)u}{16} \cos\frac{\pi(2y+1)v}{16}$$

$$C_u = \begin{cases} 1/2\sqrt{2} & \text{for } u = 0 \\ 1/2 & \text{else} \end{cases}$$

Quantizing DCT coefficients

- basis of **lossy JPEG compression**
  - original precision of the coefficients is not preserved
- **transmitted precision** of individual coefficients depend on their visual importance
  - less important coefficients are transmitted with less precision or even dismissed
- **linear quantizers** controlled by **quantizer tables** (less value ⇔ higher importance)

Quantizing DCT coefficients

Integer **quantizer table:** $\left[Q(u,v) \right]_{u,v=0}^{7}$

less value of **Q(u,v)** ⇔ higher importance of **F(u,v)**

Quantizing:

$$F^{Q}(u,v) = \mathrm{round}\left(\frac{F(u,v)}{Q(u,v)} \right)$$

Dequantizing: (reconstruction)

$$F^{\bullet}(u,v) = F^{Q}(u,v) \cdot Q(u,v)$$

Recommended quantization tables

Luminance Y
(for 50% quality)

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

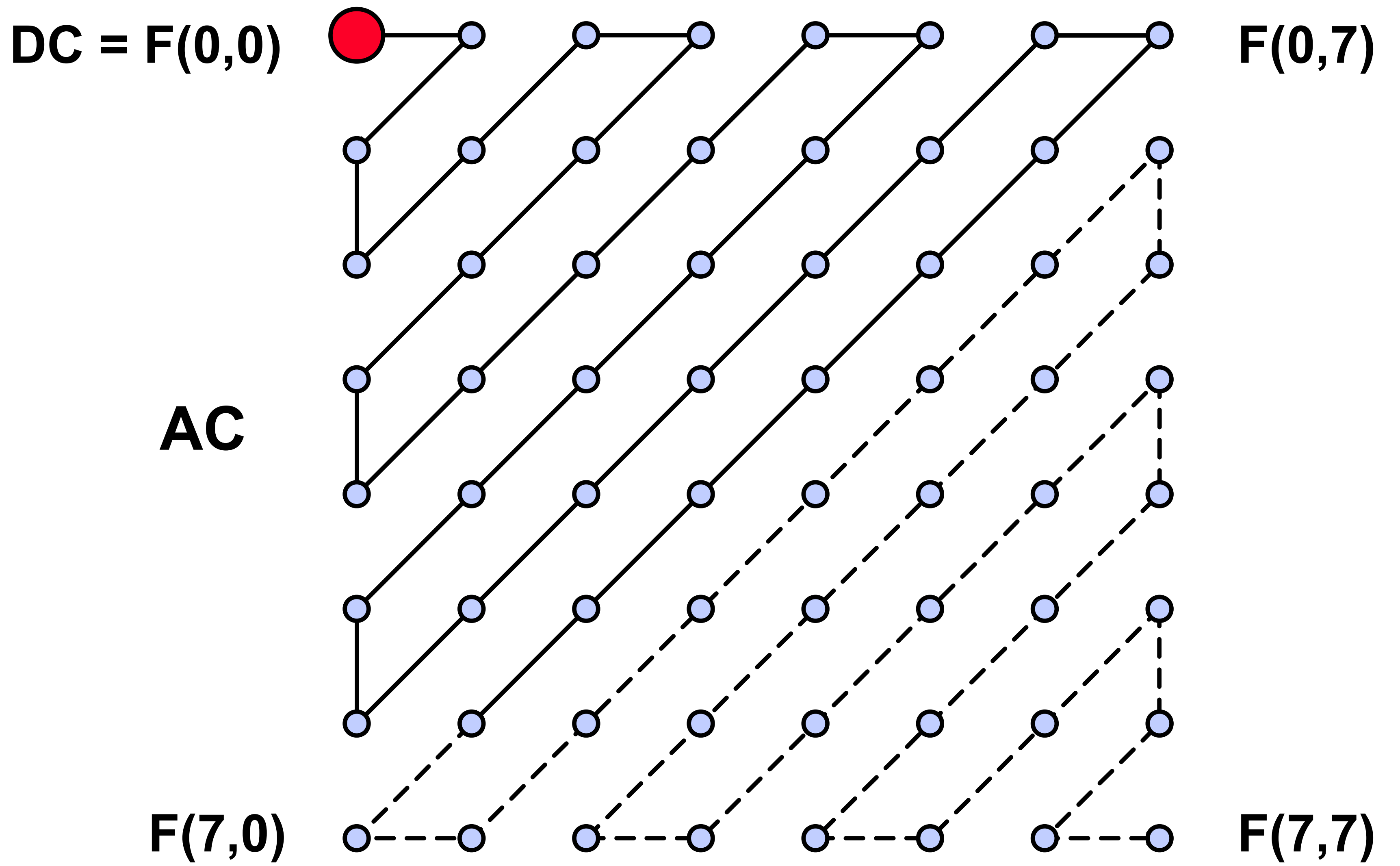
Color components C_b,C_r
(for 50% quality)

17	18	24	47	66	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	69	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

Coding of quantized coefficients

- **mean value, DC component** (**F(0,0)**) is coded individually
  - prediction from previous block, only difference is transmitted
- **frequency (AC) components** (**F(u,v)** for **u>0** $\vee$ **v>0**) are processed in „zig-zag" order
  - skipping over zero values (RLE)
  - nonzero values are coded using **VLI** (Variable Length Integer code)
- at the end, everything is subject of **entropy coding**
  - static Huffman coder or adaptive arithmetic coding

Zig-zag sequence



Intermediate code

- **DC**: **precision** (**S** .. in bits), **value** (VLI)
  - **S** symbol is encoded by entropic coder
  - the value itself is encoded using **VLI** (no entropy coding, will not have sense)

- **AC**: **[#zeroes, precision]** (**S**), **value** (VLI)
  - **S** symbol is a tuple – number of skipped zeroes and bit-precision of next nonzero coefficient
  - coefficient value is encoded using **VLI**
  - **S** symbol could iterate (more skipped zeroes)
  - special symbol **[0,0]** means „End Of Block“ (EOB)

VLI code (Variable Length Integer)

Precision in bits	Encoded values
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
11	-2047 .. -1024, 1024 .. 2047

Most frequent S-symbols

Symbol	Huffman	Symbol	Huffman
[0,1]	00	[0,6]	1111000
[0,2]	01	[1,3]	1111001
[0,3]	100	[5,1]	1111010
[EOB]	1010	[6,1]	1111011
[0,4]	1011	[0,7]	11111000
[1,1]	1100	[2,2]	11111001
[0,5]	11010	[7,1]	11111010
[1,2]	11011	[1,4]	111110110
[2,1]	11100	[3,2]	111110111
[3,1]	111010	[8,1]	1111110000
[4,1]	111011	[9,1]	1111110001

Example

Original image (512 bits):

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	162	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

DCT coefficients:

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Example

Quantizer table (q=50%):

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

Quantized coefficients:

15	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

Example

Intermediate: **(2)(3) - (1,2)(-2) - (0,1)(-1) - (0,1)(-1) - (0,1)(-1) - - (2,1)(-1) - (0,0)**

Huffman: **(011)(11) - (11011)(01) - (00)(0) - (00)(0) - (00)(0) - - (11100)(0) - (1010)**

Result: (31 bits .. 16.5 : 1)

0111111011010000000001110001010

Reconstruction error:

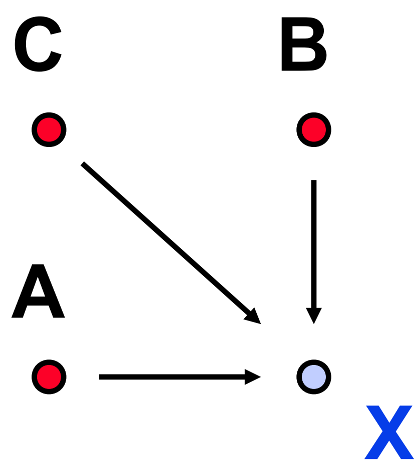
5	2	0	-1	-1	1	1	1
4	-1	-1	-2	-3	0	0	0
5	1	-3	-5	0	1	0	-1
1	0	-1	2	1	0	-2	-4
4	3	3	1	0	5	3	1
2	3	3	3	2	3	1	0
-2	-1	1	-1	0	4	2	1
-4	-3	0	0	-1	3	1	0

Lossless JPEG–1

- based on **different principle** (predictive codec)
  – lossless variants of DCT-based encoding turned to be not very practical
- suitable even for **high precision data** (up to 16 bpp)

- 1-2D **predictive method**
  – 7 predefined predictors to select from
- **predicition errors** are coded using **VLI**
  – precision (in bits) is encoded using Huffman code

Predictive method



Predictor	X
0	0
1	A
2	B
3	C
4	$A + B - C$
5	$A + \frac{1}{2}(B - C)$
6	$B + \frac{1}{2}(A - C)$
7	$\frac{1}{2}(A + B)$

Multi-component image

- JPEG can encode up to **255 image components**
  - all components must have the same depth = bpp (8 or 12 for DCT, 2 to 16 for losless method)
- individual components can have **different resolutions**
  - integer ratios 4 : 1 to 1 : 4
- user-defined **interlacing**
- optional switching of **quantizer** and **Huffman tables**
  - up to 4 quantizer and 4 frequency tables

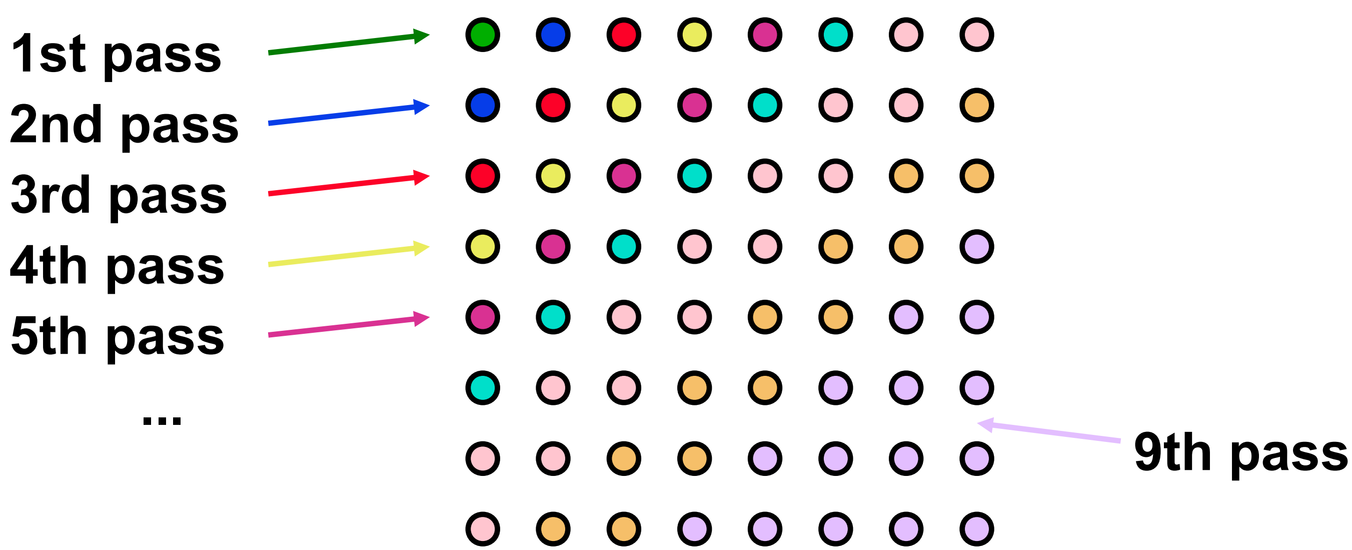
Progressive lossy mode

- the image is encoded in **multiple passes** with gradually improving quality
  - useful for on-line applications (low quality Internet)
  - transmission can be interrupted in a moment we realize that it is not what we want, or when quality is sufficient
- both encoder and decoder need **buffer memory** for the whole set of $\mathbf{F}^{\mathbf{Q}}$ coefficients (for the whole image)
  - 3 bits higher precision than original data
- ➡ **two progressive methods:** spectral selection and gradual approximation

Spectral selection

- only **some DCT coefficients** are transmitted in every pass
  - from most important ones to less important ones
  - every coefficient is transmitted in full precision

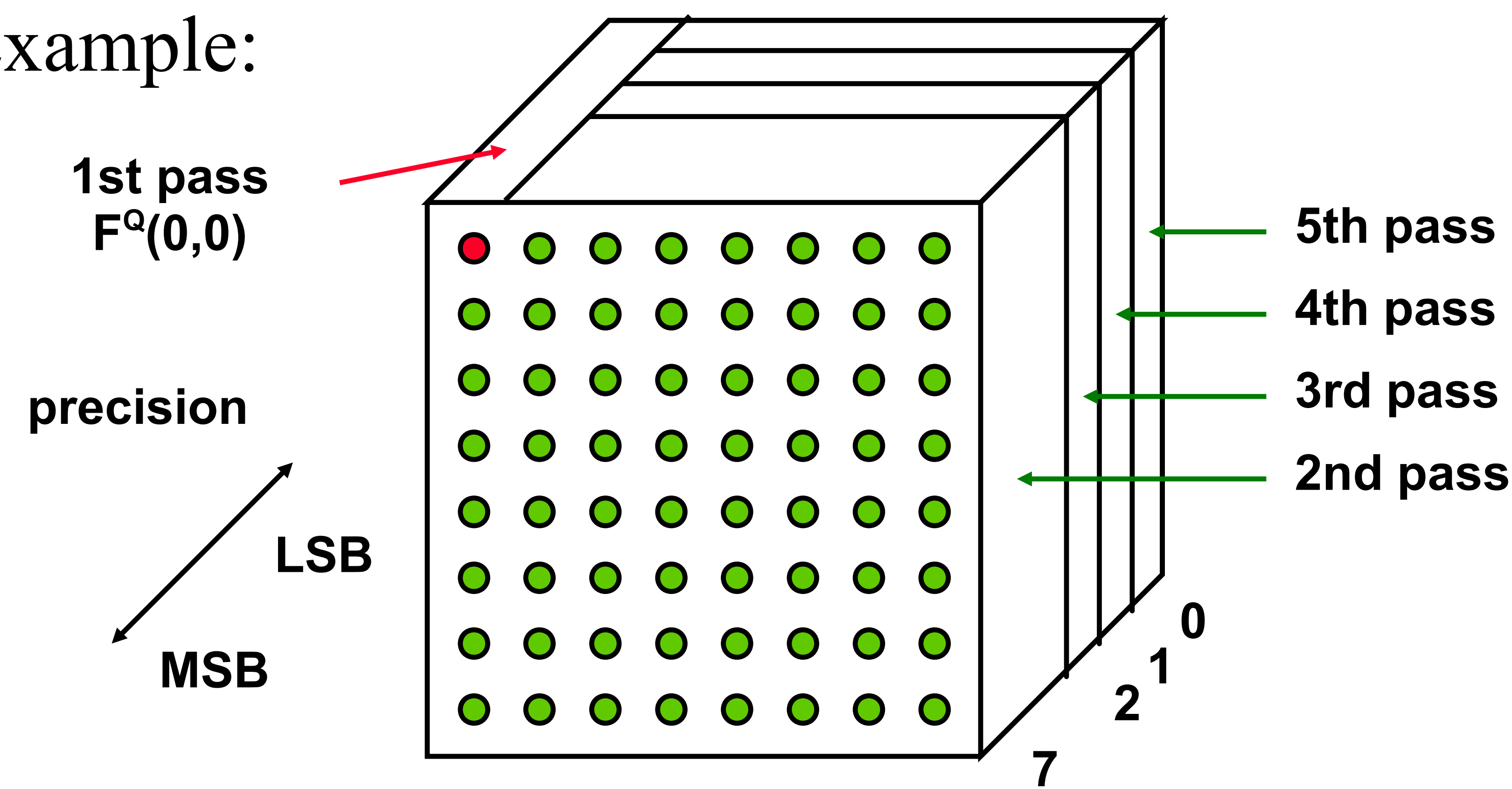
Example:



Gradual approximation

the whole set of DCT coefficients is transmitted, **they are refined gradually**

Example:



Hierarchical mode

- **pyramidal coding** – image is transmitted in several resolutions
  - previous resolution is used as prediction for the subsequent level
  - only residuals are coded

- individual levels (frames) can be coded using arbitrary **JPEG compression**
  - baseline DCT, progressive coding or lossless method
  - the last level is lossless ⇒ the whole hierarchy is lossless

JPEG File Interchange Format

- the simplest implementation of the JPEG recommend.
  - JFIF file-format: file-name extension **.JPG**
  - more complicated (more rich) is **TIFF JPEG**
- **compatibility** PC ↔ Mac ↔ UNIX
- standard **color system YC_bC_r** (as of CCIR 601)
- **preview image** - „thumbnail” (JPEG)
- support for **different resolutions** of individual image components
  - popular 4:2:0 color coding to save even more space

Color space YC_bC_r

- recommendation **CCIR 601** (video-signal coding)
  – color TV signal transmission (4:2:2 system: YC_bYC_rY..)

- **8 bits for each component**:
  – **Y**: **luminance** (grayscale value)
  – C_b resp. C_r: **color components** (mostly for **B** resp. **R**)

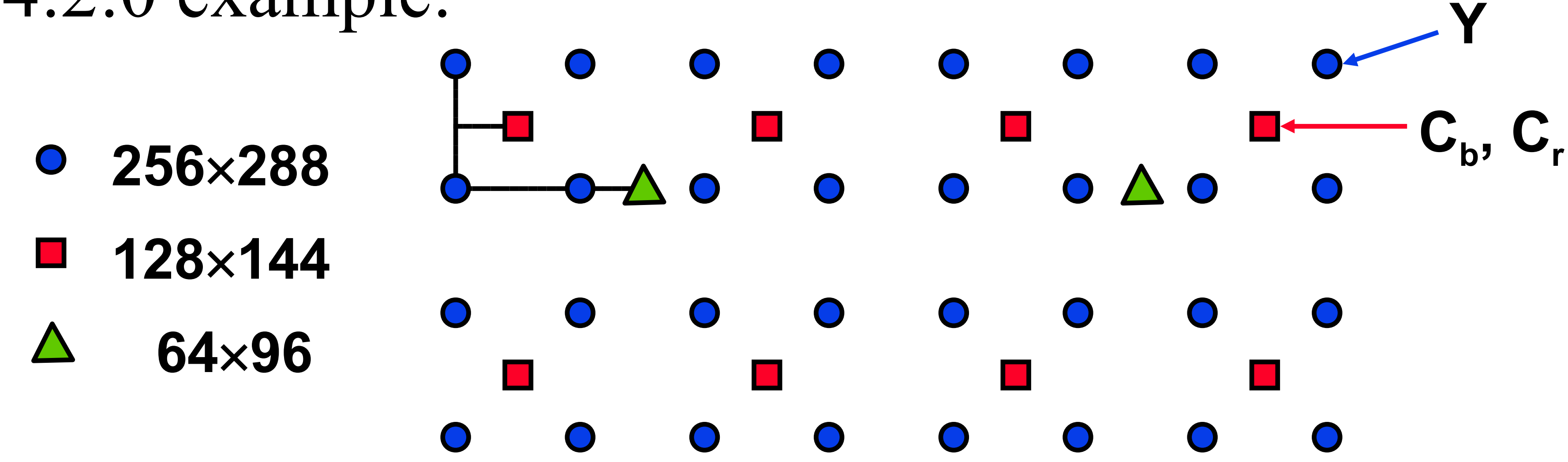
$$Y = 0.299\,R + \underline{0.587\,G} + 0.114\,B$$
$$C_b = -0.1687\,R - 0.3313\,G + \underline{0.5\,B} + 128$$
$$C_r = \underline{0.5\,R} - 0.4187\,G - 0.0813\,B + 128$$

Different resolutions

- **drift** of the first sample **[0,0]**:

$$\underline{X_{offset}} = \frac{X_{max}}{2X_{res}} - \frac{1}{2} \qquad \underline{Y_{offset}} = \frac{Y_{max}}{2Y_{res}} - \frac{1}{2}$$

4:2:0 example:



Baseline JPEG-1 examples

1. **RAYTRACE**: CGI image, ray-tracing (1200×900×24 RGB)
2. **PAINTING**: impressionist painting, image acquired by an flatbed image scanner (600×685×24 RGB)
3. **BWPHOTO**: black-and-white prhoto (portrait), scanned (624×735×8 gray)
4. **COLPHOTO**: color photograph (landscape scene), professional digitization (512×480×24 RGB)

Compression results

image	❶	❷	❸	❹
original	3240 KB	1233 KB	458 KB	737 KB
ARJ (comparison)	41%	74%	71%	86%
JPEG - 100%	18.6%	31.0%	52.0%	34.0%
JPEG - 90%	7.3%	11.3%	20.0%	14.0%
JPEG - 70%	4.0%	5.1%	10.8%	7.0%
JPEG - 50%	3.0%	3.5%	7.5%	4.7%

Sources

- **ISO/IEC JTC1 CD 10918**: ***Digital Compression and Coding of continuous-tone still images***, ISO 1993
- **G. Wallace**: ***The JPEG Still Picture Compression Standard***, Communications of the ACM, April 1991
- **E. Hamilton**: ***JPEG File Interchange Format, version 1.02***, September 1992

The End

More information:

- **V. Bhaskaran, K. Konstantinides**: ***Image and Video Compression Standards, Algorithms and Architectures***, Kluwer Academic Publishers, Boston 1995, 129-159
- **ed. by H.-M. Hang, J. Woods**: ***Handbook of Visual Communications***, Academic Press, San Diego 1995, 242-246, 366-375