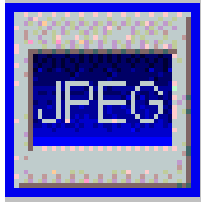


Outline

- Introduction
- Lossless compression
- Lossy compression
 - metrics
 - general methods
 - scalar
 - vector
 - differential
 - transform
 - Haar
 - JPEG
 - wavelet
 - MPEG

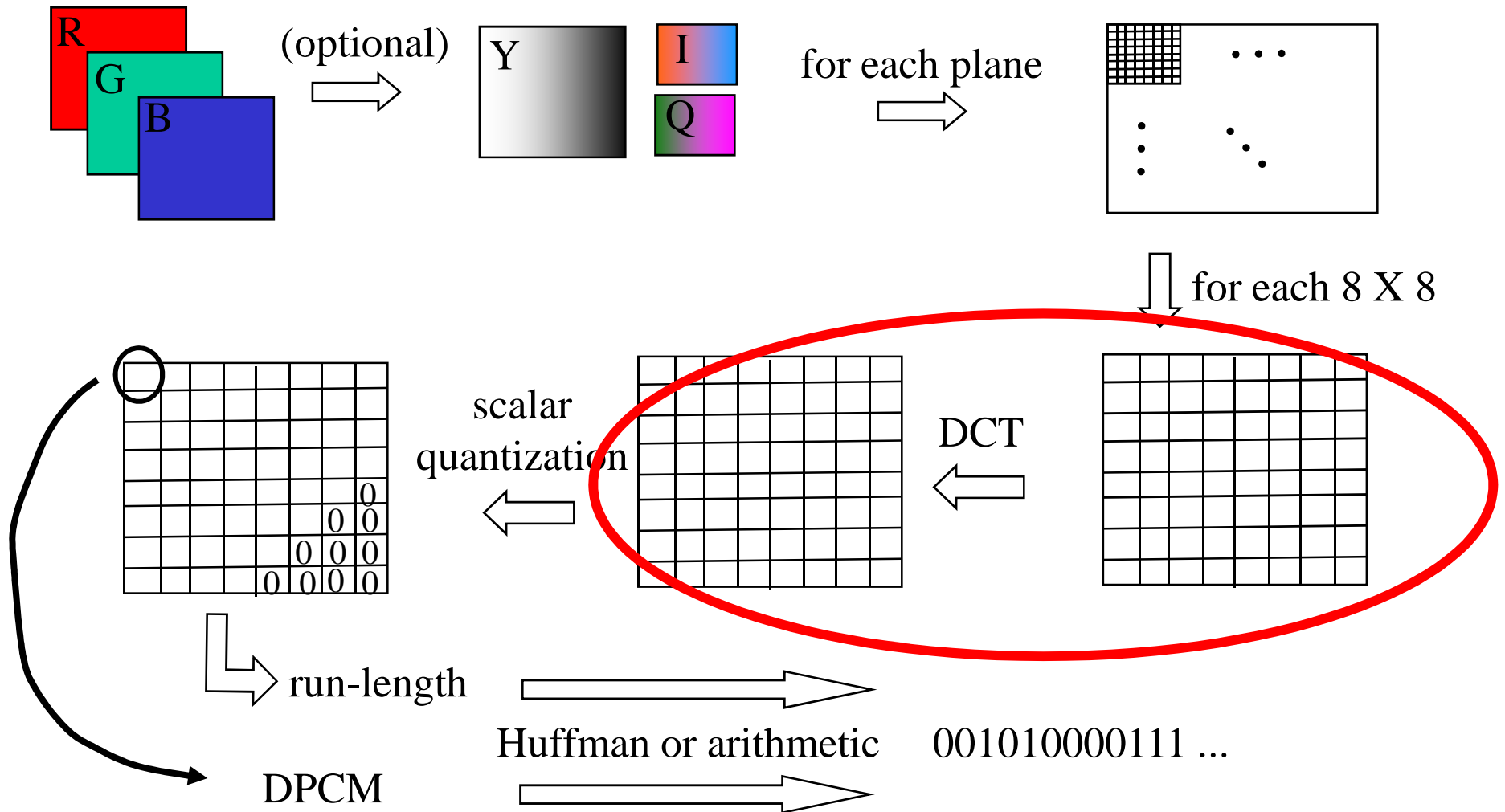


General methods - transform

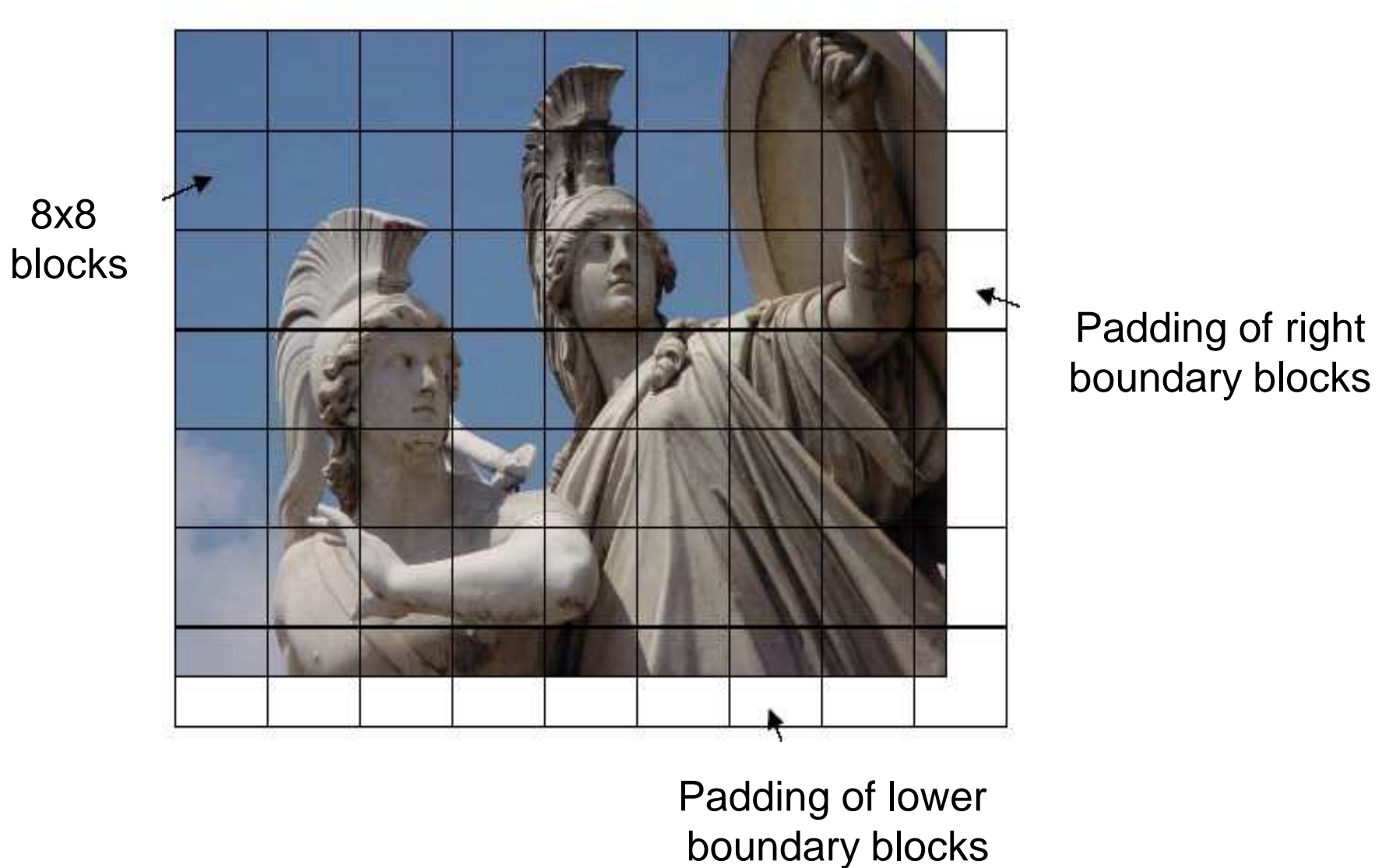


- Joint Photographics Expert Group
 - started 1986; standard published 1990-91
 - lossy (also lossless - rarely used)
 - includes:
 - **DCT - discrete cosine transform**
 - others: Fourier, Karhunen-Loeve, discrete sine, discrete Walsh-Hadamard
 - scalar, difference, run-length, and Huffman or arithmetic

JPEG



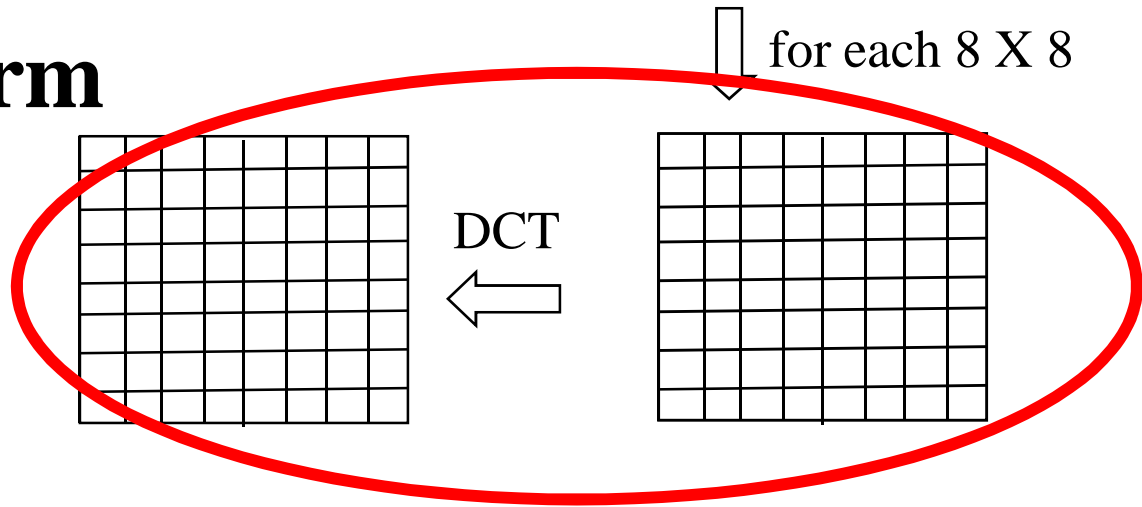
JPEG: image partition into 8x8 block



1-Dimension DCT

- Equation 13.43 in textbook equation for 1-D DCT transform for $N=8$
- Figure 13.3 shows the values from Equation 12.43 in picture format
- Section 13.9 question 2 asks for transforming using 1-D

DCT transform



Discrete Cosine Transform

$$F(u, v) = \frac{\Lambda(u)\Lambda(v)}{4} \sum_{i=0}^7 \sum_{j=0}^7 \cos \frac{(2i+1) \cdot u\pi}{16} \cdot \cos \frac{(2j+1) \cdot v\pi}{16} \cdot f(i, j)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

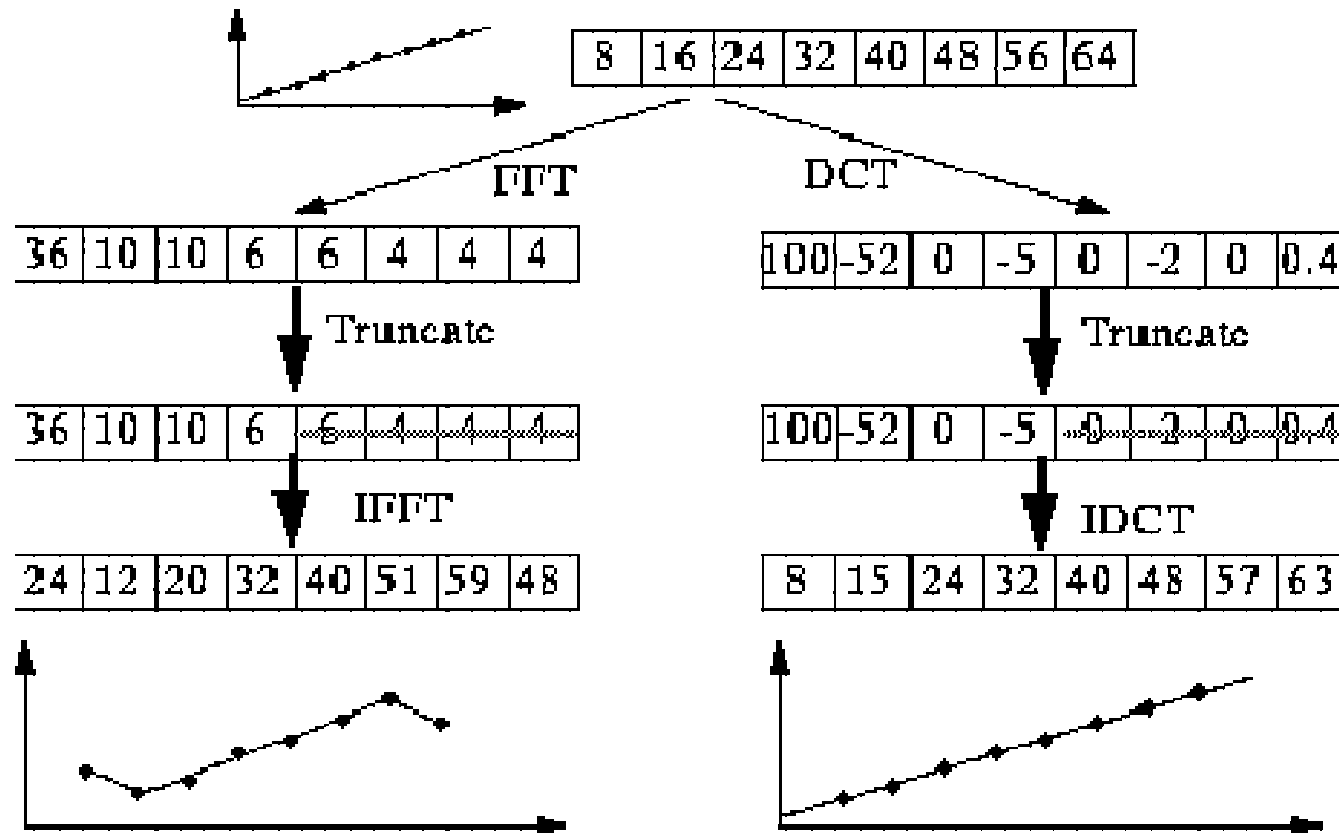
Inverse Discrete Cosine Transform

$$\hat{f}(i, j) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 \Lambda(u)\Lambda(v) \cos \frac{(2i+1) \cdot u\pi}{16} \cdot \cos \frac{(2j+1) \cdot v\pi}{16} \cdot F(u, v)$$

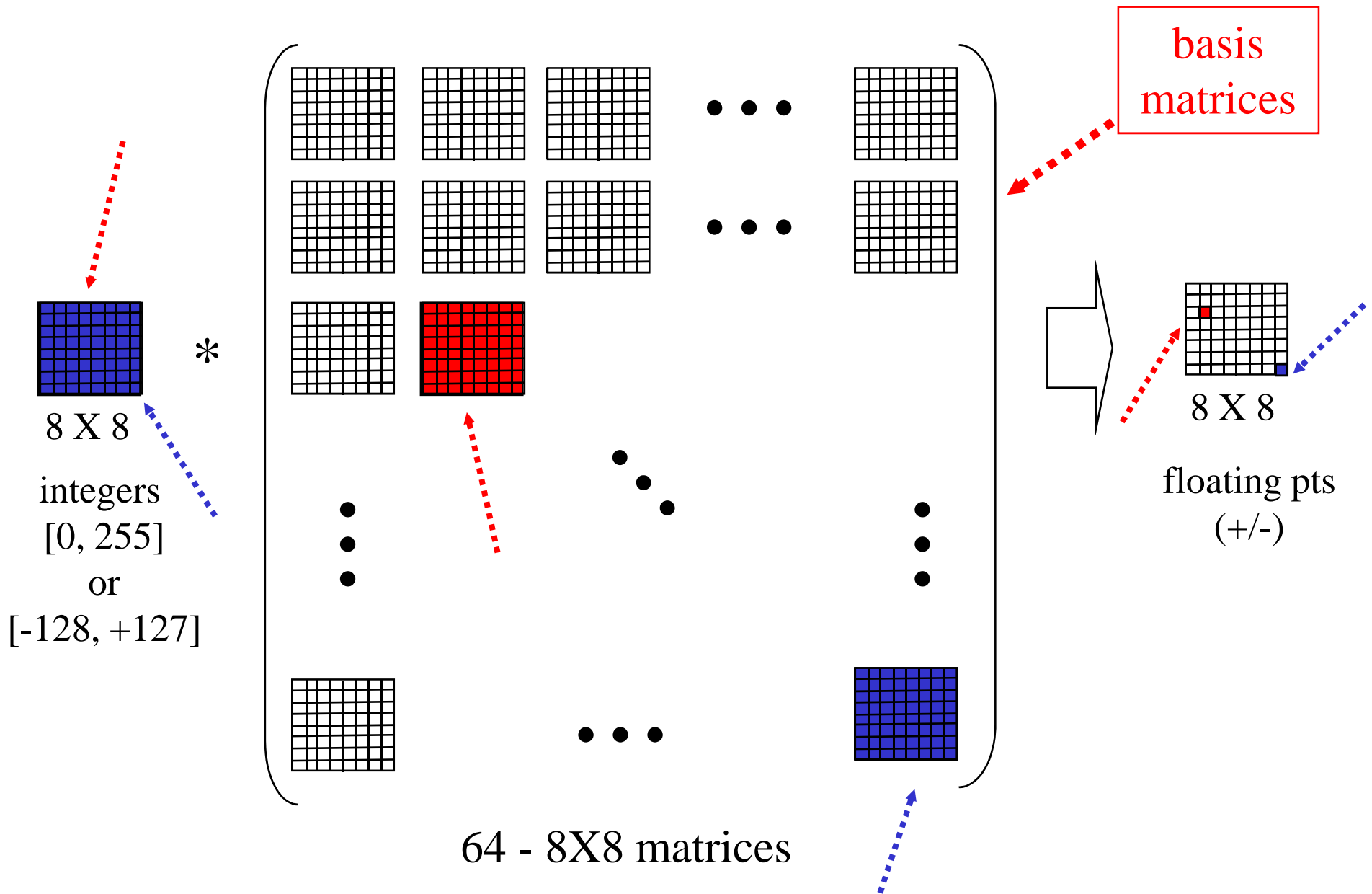
$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

Why DCT, not FFT?

Answer: DCT is similar to FFT, but can approximate linear signals well with few coefficients.

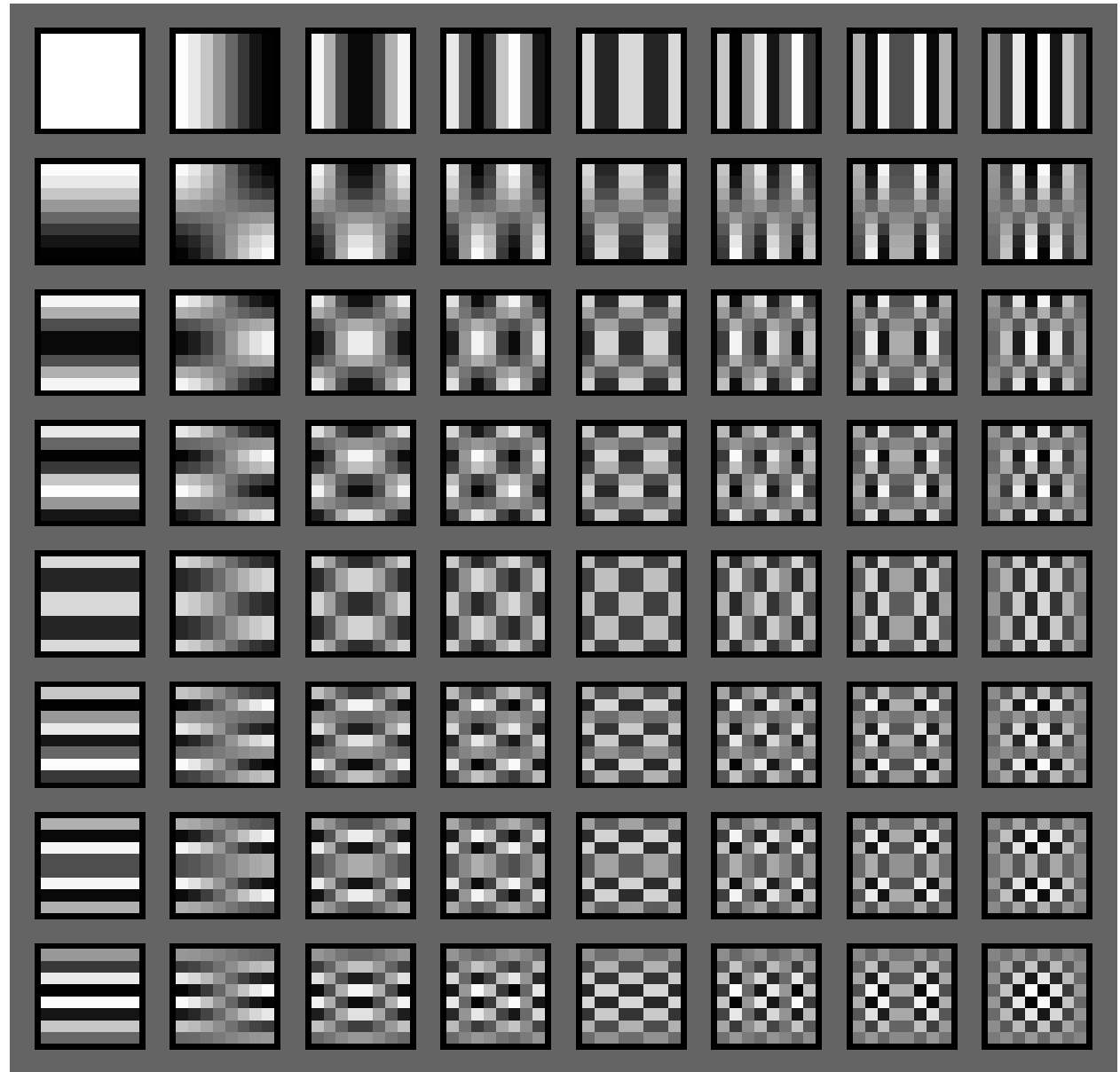


Discrete Cosine Transform



DCT basis matrices

white is + value
black is - value



JPEG example - coding

original 8X8 pixels

coefficients

124	125	122	120	122	119	117	118
121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	120
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

39.88	6.56	-2.24	1.22	-0.37	-1.08	0.79	1.13
-102.43	4.56	2.26	1.12	0.35	-0.63	-1.05	-0.48
37.77	1.31	1.77	0.25	-1.50	-2.21	-0.10	0.23
-5.67	2.24	-1.32	-0.81	1.41	0.22	-0.13	0.17
-3.37	-0.74	-1.75	0.77	-0.62	-2.65	-1.30	0.76
5.98	-0.13	-0.45	-0.77	1.99	-0.26	1.46	0.00
3.97	5.52	2.39	-0.55	-0.05	-0.84	-0.52	-0.13
-3.43	0.51	-1.07	0.87	0.96	0.09	0.33	0.01



÷

channel

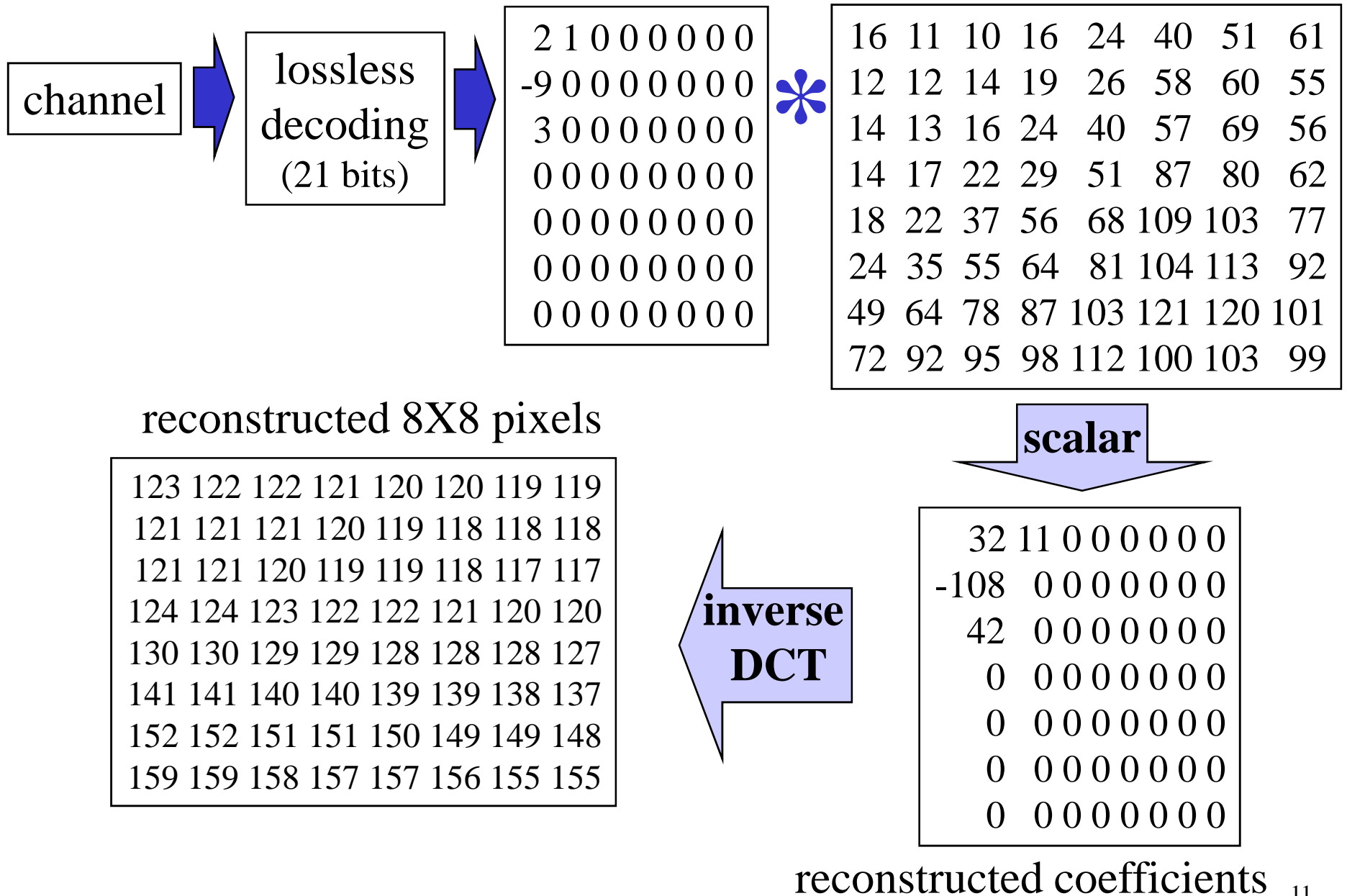
lossless
encoding
(21 bits !)

2	1	0	0	0	0	0	0
-9	0	0	0	0	0	0	0
3	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

scalar

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

JPEG example - decoding



JPEG example - explanation

original

124	125	122	120	122	119	117	118
121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	120
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

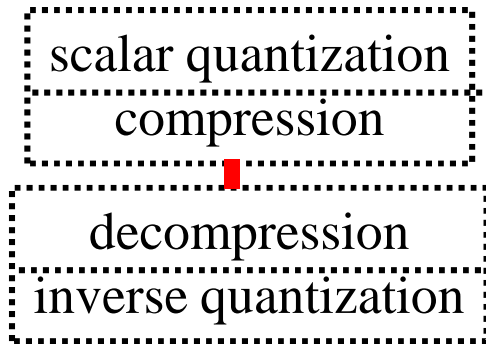
39.88	6.56	-2.24	1.22	-0.37	-1.08	0.79	1.13
-102.43	4.56	2.26	1.12	0.35	-0.63	-1.05	-0.48
37.77	1.31	1.77	0.25	-1.50	-2.21	-0.10	0.23
-5.67	2.24	-1.32	-0.81	1.41	0.22	-0.13	0.17
-3.37	-0.74	-1.75	0.77	-0.62	-2.65	-1.30	0.76
5.98	-0.13	-0.45	-0.77	1.99	-0.26	1.46	0.00
3.97	5.52	2.39	-0.55	-0.05	-0.84	-0.52	-0.13
-3.43	0.51	-1.07	0.87	0.96	0.09	0.33	0.01



123	122	122	121	120	120	119	119
121	121	121	120	119	118	118	118
121	121	120	119	119	118	117	117
124	124	123	122	122	121	120	120
130	130	129	129	128	128	128	127
141	141	140	140	139	139	138	137
152	152	151	151	150	149	149	148
159	159	158	157	157	156	155	155



32	11	0	0	0	0	0	0
-108	0	0	0	0	0	0	0
42	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



reconstructed

JPEG details - quantization

Non-uniform Quantization - Eye is most sensitive to low frequencies (upper left), less sensitive to high frequencies (lower right)

Luminance Quantization Table

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

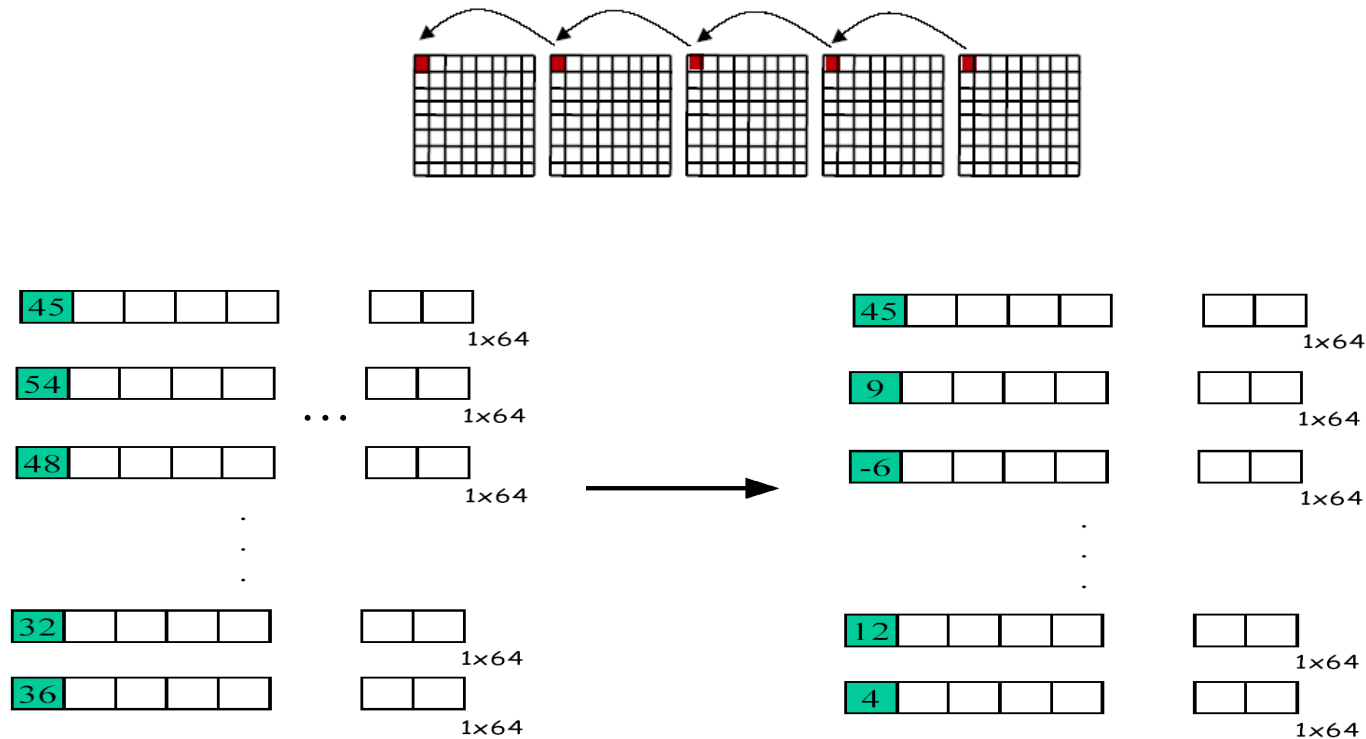
Chrominance Quantization Table

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

- Quantization tables values can be scaled up/down to adjust the *quality factor*
- Custom quantization tables can also be put in image header

JPEG Details: Entropy Encoding of DC Components

- Model: For photographs, DC value in each 8x8 block is often close to previous block.
- Coding Scheme: use Differential Pulse Code Modulation (DPCM):
 - Encode the difference between the current and previous 8x8 block.
 - Remember, encoding smaller numbers generally requires fewer bits



JPEG Details: Entropy Encoding of DC Components (cont'd)

Size-Value Encoding Table

Size	Code	Value Range	Code
0	00	0	---
1	010	-1, 1	0,1
2	011	-3,-2, 2,3	00,01, 10,11
3	100	-7,-6,-5,-4, 4,5,6,7	000,...,011, 100,...,111
4	101	-15,-14,-13,...,-8, 8,...,13,14,15	0000,...,0111, 1000,...,1111
5	110	-31,...,-16, 16,...,31	00000,...,01111, 10000,...,11111
6	1110	-63,...,-32, 32,...,63	000000,...,011111, 100000,...,111111
7	11110	-127,...,-64, 64,...,127	0000000,...,1111111
8	111110	-255,...,-128, 128,...,255	00000000,...,11111111
9	1111110		
10	11111110		
11	111111110	-2047,...,-1024, 1024,...,2047	00000000000,...,11111111111

Example: If a DC component is 40, and the previous DC component is 48.



The difference is -8. Therefore 40 gets coded as: **1010111**

0111: value representing -8

101: size from the same table reads 4

Entropy Coding: Example

40	12	0	0	0	0	0	0
10	-7	-4	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
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


 12, 10, 1, -7, 0, 0, -4, 0, 0, 0, ..., 0
 

0-0s, 12: (0/4) 12 → 10111100
 1011: code for 0/4 from AC code table (textbook Table 13.10)
 1100: code for 12 from Size-Value table (textbook Table 13.9)

0-0s, 10: (0/4) 10 → 10111010
 1011: code for 0/4 from AC code table
 1100: code for 10 from Size-Value table

0-0s, 1: (0/1) 1 → 001
 00: code for 0/1 from AC code table
 1: code for 1 from Size-Value table

0-0s, -7: (0/3) -7 → 100000
 100: code for 0/3 from AC code table
 000: code for -7 from Size-Value table

2-0s, -4: (2/3) -4 → 1111110111011
 1111110111: code for 2/3 from AC code table (not shown in Table 13.10)
 011: code for -4 from Size-Value table

56-0s: (0,0) → 1010 (special code for all 0's until EOB)

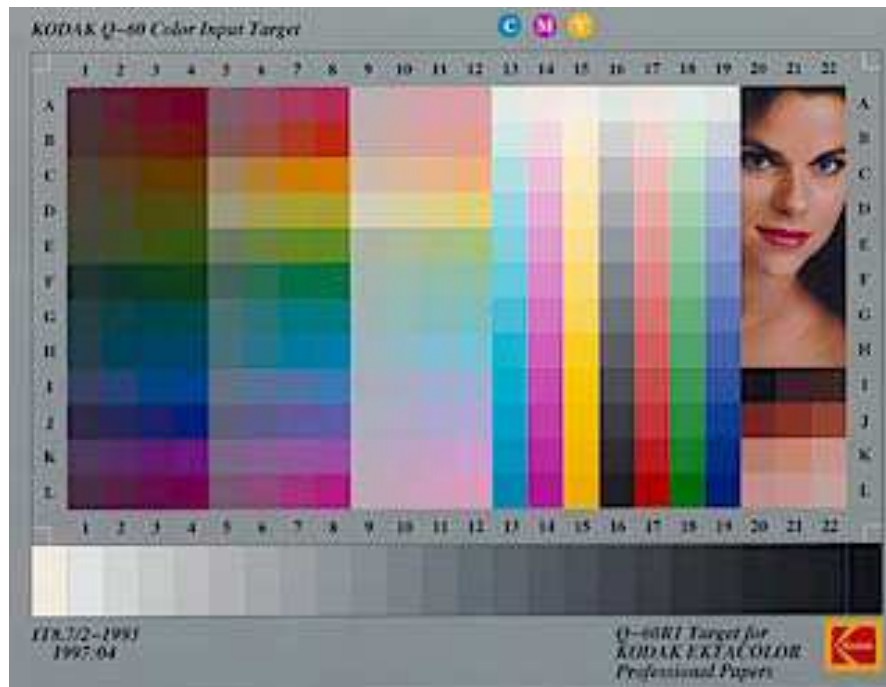
JPEG default AC code for luminance

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	1010 (= EOB)	4			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	11111111000000	17
0/3	100	6	8/3	111111110110111	19
0/4	1011	8	8/4	111111110111000	20
0/5	11010	10	8/5	111111110111001	21
0/6	111000	12	8/6	111111110111010	22
0/7	1111000	14	8/7	111111110111011	23
0/8	111110110	18	8/8	111111110111100	24
0/9	111111110000010	25	8/9	111111110111101	25
0/A	111111110000011	26	8/A	111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	111111110111111	18
1/3	1111001	10	9/3	111111111000000	19
1/4	111110110	13	9/4	111111111000001	20
1/5	11111110110	16	9/5	111111111000010	21
1/6	111111110000100	22	9/6	111111111000011	22
1/7	111111110000101	23	9/7	111111111000100	23
1/8	111111110000110	24	9/8	111111111000101	24
1/9	111111110000111	25	9/9	111111111000110	25
1/A	111111110001000	26	9/A	111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	111111111001000	18
2/3	1111110111	13	A/3	111111111001001	19
2/4	111111110001001	20	A/4	111111111001010	20
2/5	111111110001010	21	A/5	111111111001011	21
2/6	111111110001011	22	A/6	111111111001100	22
2/7	111111110001100	23	A/7	111111111001101	23

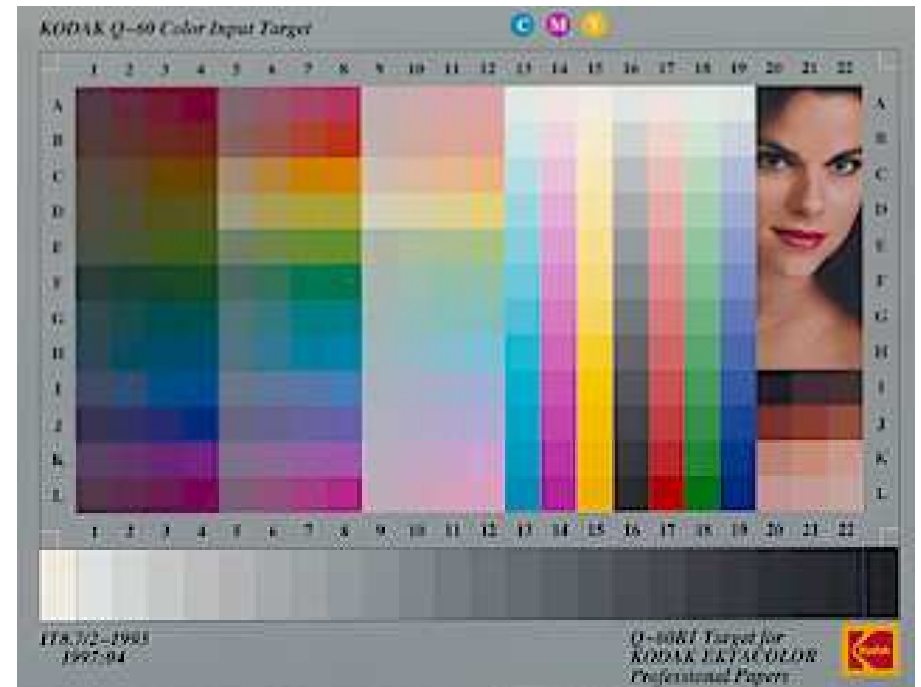
JPEG default AC code for luminance (cont'd)

2/8	111111110001101	24	A/8	111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	111111111010001	18
3/3	1111110111	14	B/3	111111111010010	19
3/4	111111110010000	20	B/4	111111111010011	20
3/5	1111111110010001	21	B/5	111111111010100	21
3/6	1111111110010010	22	B/6	111111111010101	22
3/7	1111111110010011	23	B/7	111111111010110	23
3/8	1111111110010100	24	B/8	111111111010111	24
3/9	1111111110010101	25	B/9	111111111011000	25
3/A	1111111110010110	26	B/A	111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	111111111011010	18
4/3	1111111110010111	19	C/3	111111111011011	19
4/4	1111111110011000	20	C/4	111111111011100	20
4/5	1111111110011001	21	C/5	111111111011101	21
4/6	1111111110011010	22	C/6	111111111011110	22
4/7	1111111110011011	23	C/7	111111111011111	23
4/8	1111111110011100	24	C/8	111111111100000	24
4/9	1111111110011101	25	C/9	111111111100001	25
4/A	1111111110011110	26	C/A	111111111100010	26

JPEG compression results

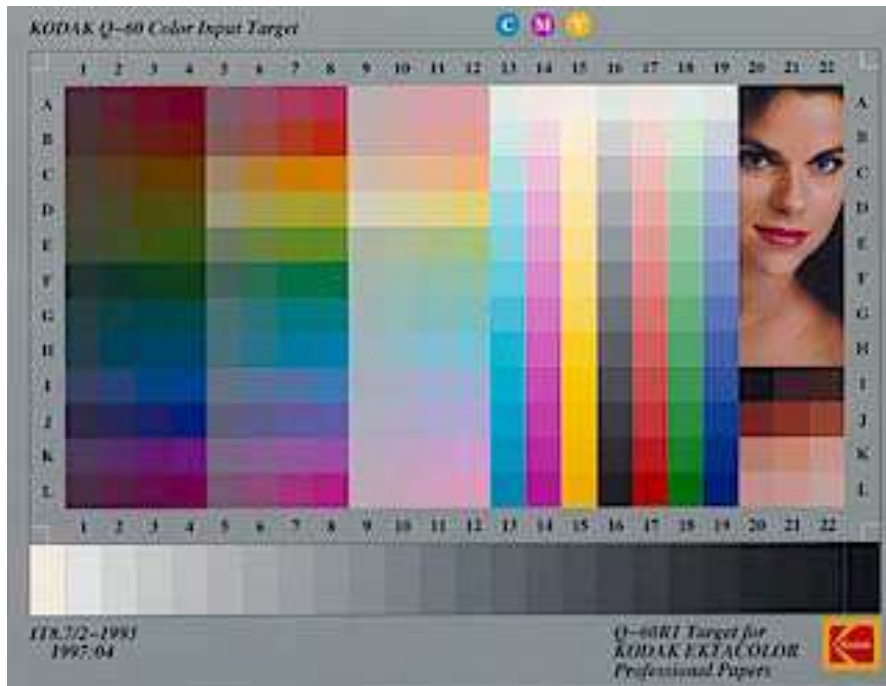


231KB original 320 X 240 X 24bit



74KB 3.24 : 1 compression

JPEG compression results

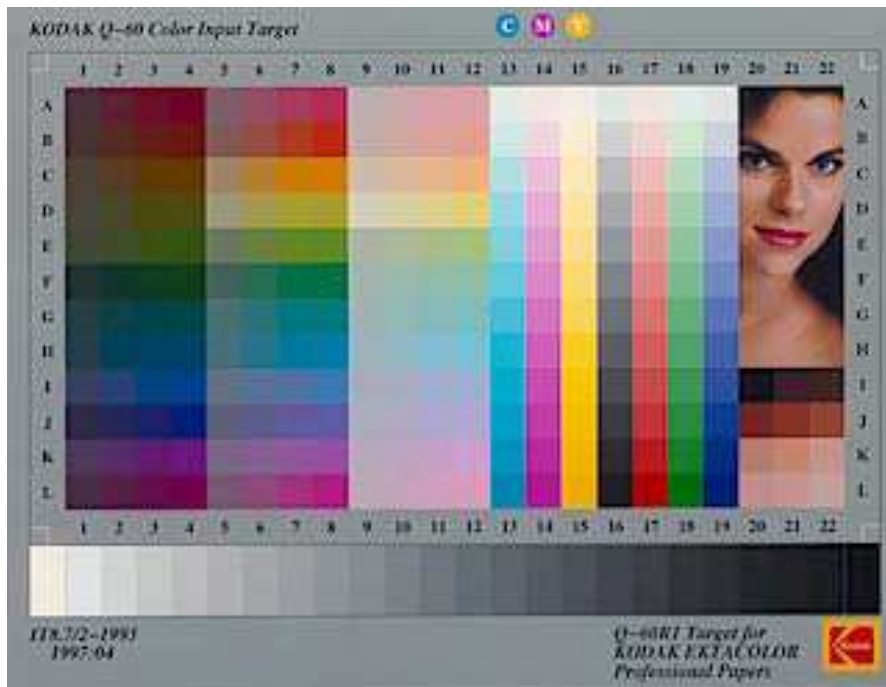


231KB original 320 X 240 X 24bit

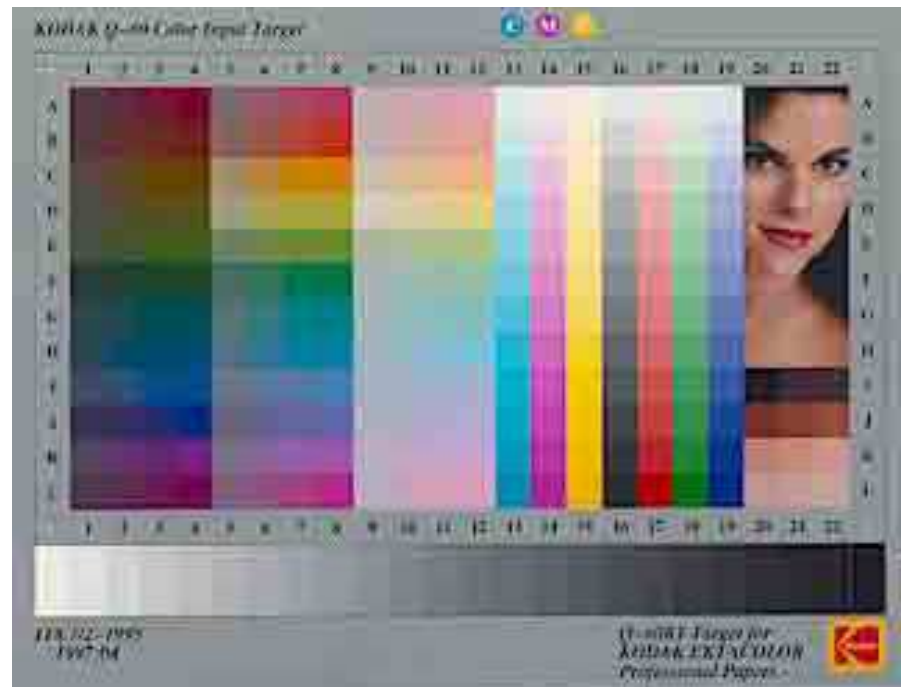


38KB 6.08 : 1 compression

JPEG compression results



231KB original 320 X 240 X 24bit



11KB 21 : 1 compression

ROI for GSM-Transmission at 9.6 kBit/s

Universität Klagenfurt - IWAS
Multimedia Kommunikation (VK 622.776)
Dr. Sebestyen/Dr. Hundt; Mai 2001

Progressive Download (CIF 352 x 288)



0.5 s



2 s

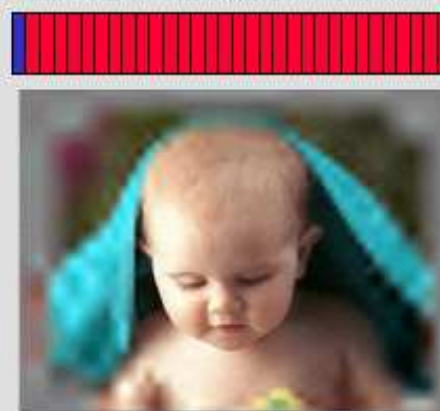


3 s

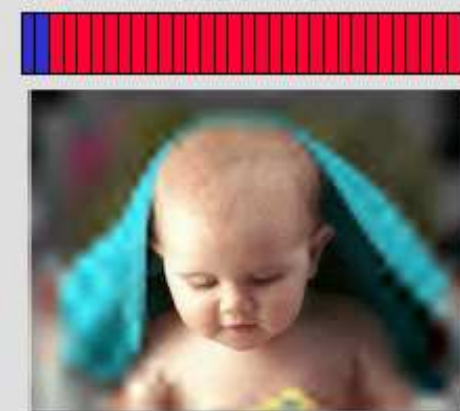
Progressive Download with Region of Interest (CIF 352 x 288)



0.2 s



0.5 s



1 s