Digital Image Processing (CSE/ECE 478)

Lecture # 21: Image Compression II

Avinash Sharma

Center for Visual Information Technology (CVIT),

IIIT Hyderabad

Lossless compression: Dictionary coding

- Important in presence of recurring patterns
- Static and adaptive
- Static example: a b r a c a d a b r a

- Dynamic dictionary
  - Build during compression after observing the data
  - Rebuild at the decompression step
  - LZW (*Lempel-Ziv-Welch*) is the commonly used algorithm

$$A = \{a, b, c, d, r\}$$

Code	Entry
000	a
001	b
010	c
011	d
100	r
101	ab
110	ac
111	ad

K. Sayood, Introduction to Data Compression, 2nd edition Morgan Kaufmann,2000

LZW coding

Lets try to compress the string 'thisisthe'

ASCII codes:
t(116), h(104), i(105),s(115), e(101)

Current	Next	Output	Add to dictionary

LZW coding

Lets try to compress the string ‘thisisthe’

ASCII codes:
t(116), h(104), i(105),s(115), e(101)

Current	Next	Output	Add to dictionary
t(116)	h(104)	t(116)	th(256)
h(104)	i(105)	h(104)	hi(257)
i(105)	s(115)	i(105)	is(258)
s(115)	i(105)	s(115)	si(259)
*is(258)			

LZW coding

ASCII codes:
t(116), h(104), i(105),s(115), e(101)

Lets try to compress the string ‘thisisthe’

Current	Next	Output	Add to dictionary
t(116)	h(104)	t(116)	th(256)
h(104)	i(105)	h(104)	hi(257)
i(105)	s(115)	i(105)	is(258)
s(115)	i(105)	s(115)	si(259)
*is(258)	t(116)	is(258)	ist(260)
*th(256)	e(101)	th(256)	the(261)
e (101)	-	e(101)	-

LZW decoding

Decompress 116, 104, 105, 115, 258, 256, 101

ASCII codes:
t(116), h(104), i(105),s(115), e(101)

Current	Next	Output	Add to dictionary
116	104	116	116 104 (256)

LZW decoding

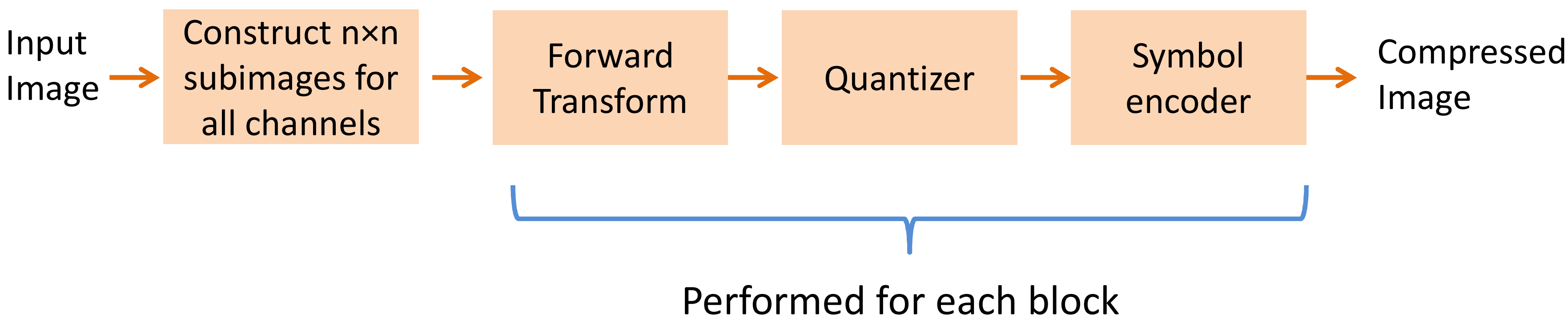
Decompress 116, 104, 105, 115, 258, 256, 101

ASCII codes:
t(116), h(104), i(105),s(115), e(101)

Current	Next	Output	Add to dictionary
116	104	116	116 104 (256)
104	105	104	104 105 (257)
105	115	105	105 115 (258)
115	258	115	115 105 115 (259)
258	256	105 115	105 115 116 104 (260)
256	101	116 104	116 104 101 (261)
101	-	101	-

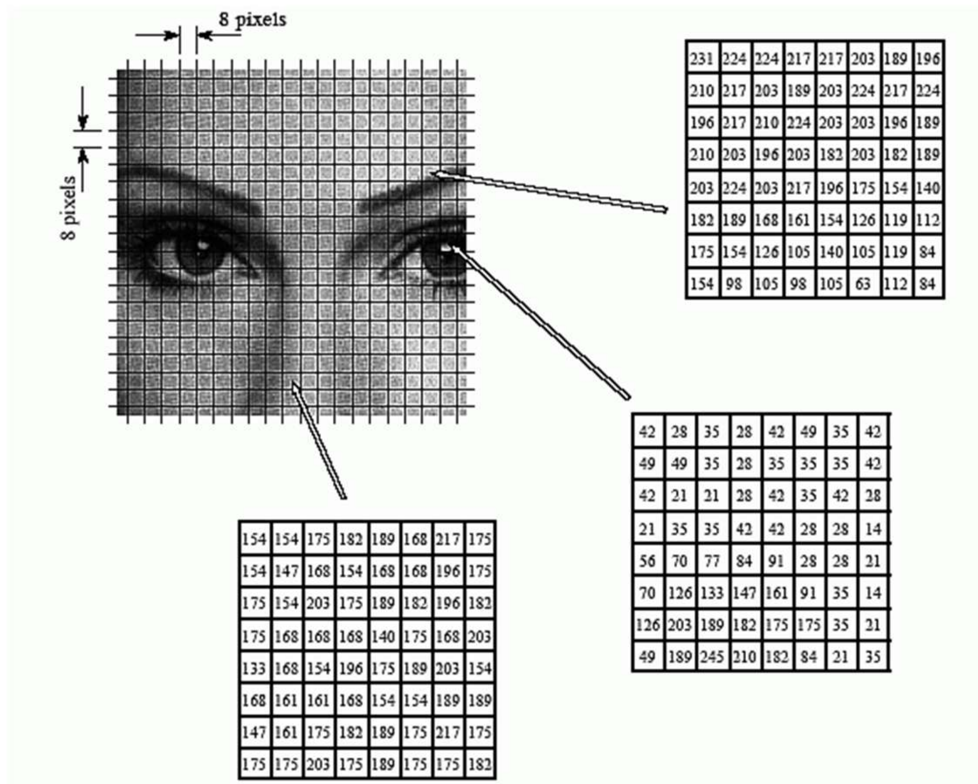
Lossy compression (with a case study of JPEG)

Lossy compression: JPEG



Block transform coding

- Partition the image into small non overlapping n×n blocks
  - 8×8 blocks in JPEG



231	224	224	217	217	203	189	196
210	217	203	189	203	224	217	224
196	217	210	224	203	203	196	189
210	203	196	203	182	203	182	189
203	224	203	217	196	175	154	140
182	189	168	161	154	126	119	112
175	154	126	105	140	105	119	84
154	98	105	98	105	63	112	84

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

154	154	175	182	189	168	217	175
154	147	168	154	168	168	196	175
175	154	203	175	189	182	196	182
175	168	168	168	140	175	168	203
133	168	154	196	175	189	203	154
168	161	161	168	154	154	189	189
147	161	175	182	189	175	217	175
175	175	203	175	189	175	175	182

Image source: dspguide.com

Block Transform coding

- Process blocks using 2D transforms
- General forward transform of image g, size n×n:

$$T(u,v) = \sum_{x=0}^{n-1} \sum_{y=o}^{n-1} g(x,y)\, r(x,y,u,v)$$

- Inverse transform

$$g(x,y) = \sum_{x=0}^{n-1} \sum_{y=o}^{n-1} T(u,v)\, s(x,y,u,v)$$

- $r(x,y,u,v)$ and s(x,y,u,v) are basis functions or transformation kernels

Block Transform coding

$$T(u,v) = \sum_{x=0}^{n-1}\sum_{y=o}^{n-1} g(x,y)\, r(x,y,u,v) \qquad g(x,y) = \sum_{x=0}^{n-1}\sum_{y=o}^{n-1} T(u,v)\, s(x,y,u,v)$$

- $r(x,y,u,v) = e^{-j2\pi(ux+vy)/n}$ and $\mathrm{s}(x,y,u,v) = \frac{1}{n^2} e^{j2\pi(ux+vy)/n}$

  Discrete Fourier Transform (DFT)

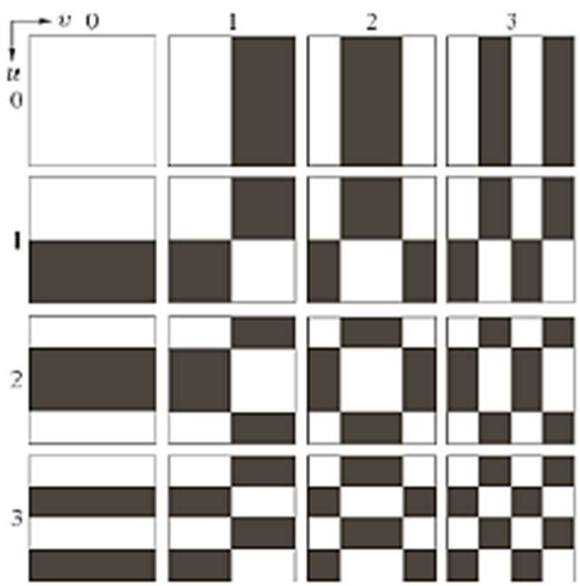
- $r(x,y,u,v) = \mathrm{s}(x,y,u,v) = \frac{1}{n}(-1)^{\sum_{i=0}^{m-1}\lfloor b_i(x)p_i(u)+b_i(y)p_i(v)\rfloor}$

  Walsh-Hadamard Transform (WHT)

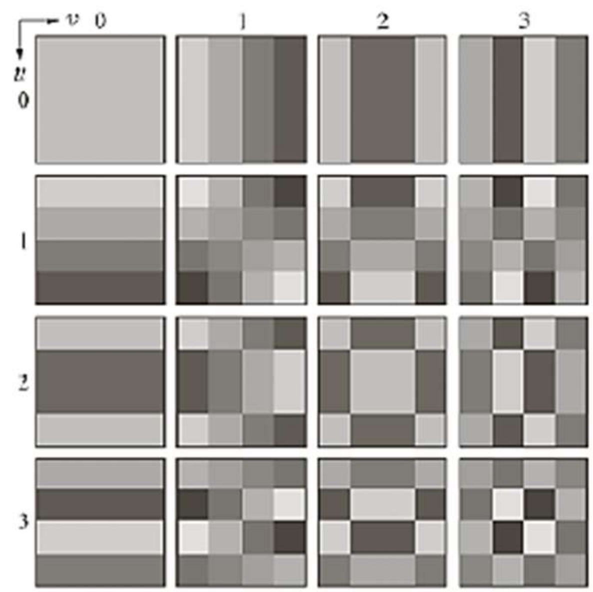
- $r(x,y,u,v) = \mathrm{s}(x,y,u,v) = \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$

  Discrete Cosine Transform (DCT)

Block Transform coding: which transform to use?



Walsh-Hadamard Transform Basis
for block of size 4×4



Discrete Cosine Transform Basis for
block of size 4×4

Block Transform coding: which transform to use?

Apply transform to each 8×8 block

Keep highest 50% of the coefficients in each block

Reconstruct using the inverse transform on each block

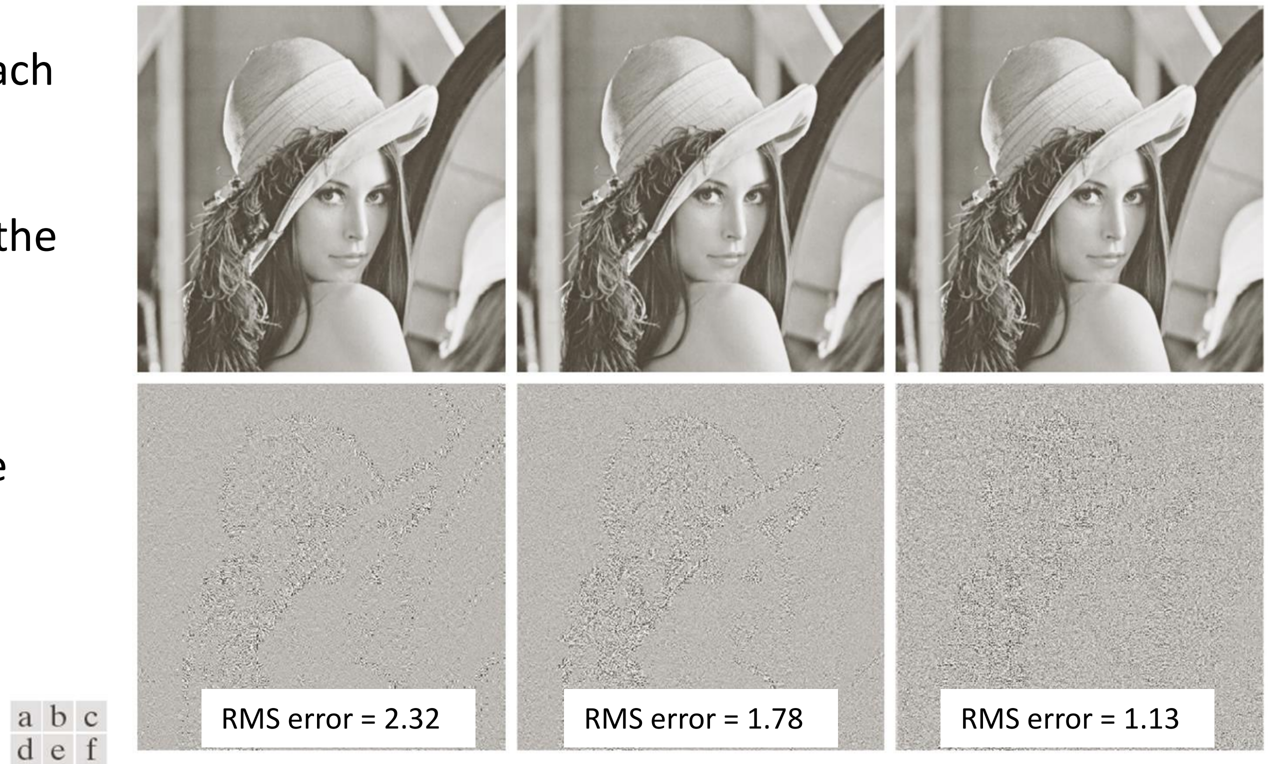


FIGURE 8.24 Approximations of Fig. 8.9(a) using the (a) Fourier, (b) Walsh-Hadamard, and (c) cosine transforms, together with the corresponding scaled error images in (d)–(f).

Block Transform coding: which transform to use?

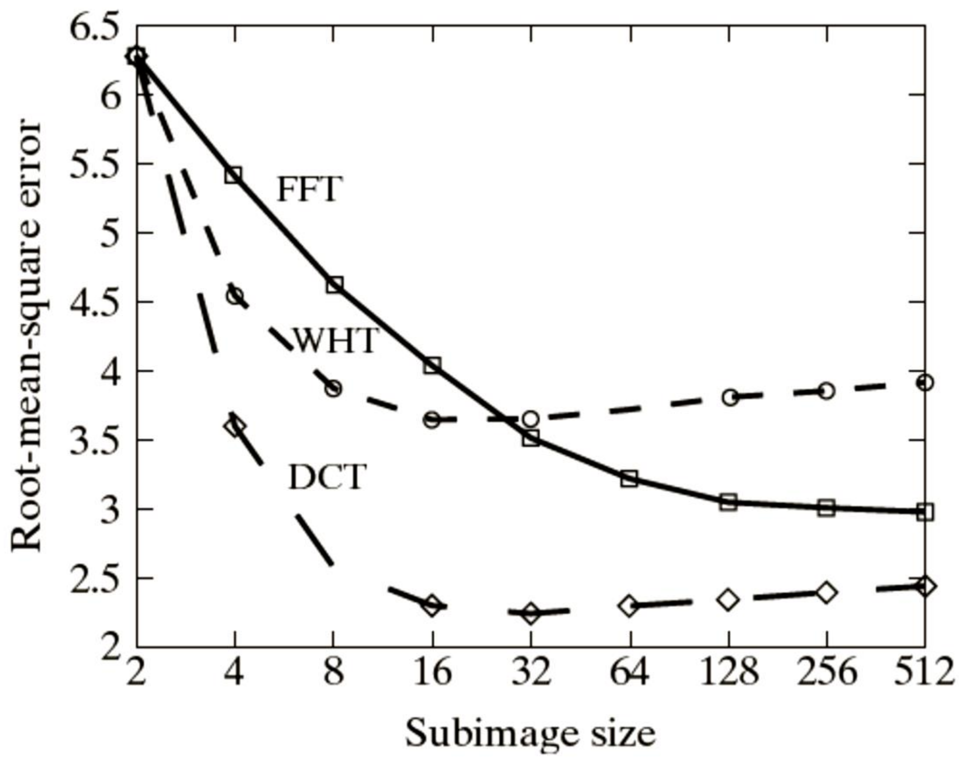
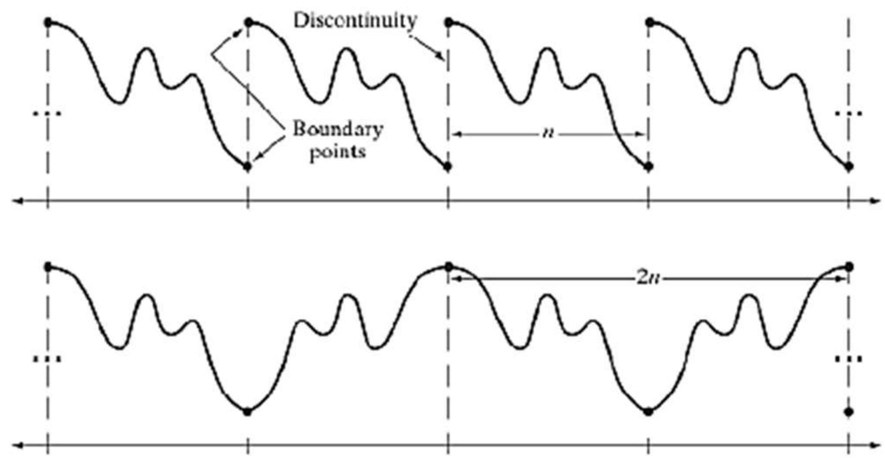


FIGURE 8.26 Reconstruction error versus subimage size.

Block Transform coding: which transform to use?



DFT v/s DST: Issue of *blocking artifact*

Quantization

Need of Quantization

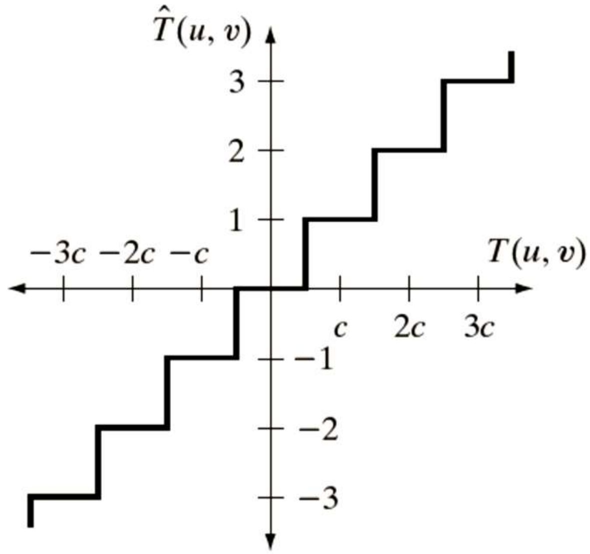
$$
\begin{pmatrix}
245 & 239 & 227 & 203 & 174 & 150 & 116 & 92 \\
229 & 216 & 197 & 172 & 150 & 119 & 85 & 69 \\
201 & 180 & 164 & 152 & 141 & 102 & 77 & 69 \\
174 & 153 & 148 & 146 & 140 & 112 & 93 & 91 \\
161 & 145 & 144 & 146 & 141 & 133 & 120 & 114 \\
149 & 139 & 143 & 144 & 142 & 139 & 133 & 133 \\
134 & 128 & 131 & 132 & 134 & 134 & 139 & 137 \\
119 & 114 & 112 & 111 & 111 & 119 & 131 & 141
\end{pmatrix}
\rightarrow
\begin{pmatrix}
7 & 17 & -2 & 1 & 1 & 0 & 0 & 0 \\
9 & 16 & -2 & 1 & 0 & 0 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

$$
\begin{matrix}
5.3 & 3.5 \\
0.6 & 0.3 \\
-1.3 & -2.7 \\
2.7 & -3.3 \\
1.1 & -2.5 \\
-1.4 & 2.4 \\
0.1 & -0.1 \\
-1.8 & 0.2
\end{matrix}
\Bigg)
$$

Three ways to quantize (threshold) a transformed sub-image (block):

- A single global threshold for all blocks
- A different threshold for all blocks (*N-largest coding*, gives constant bit rate)
- Threshold varied as the function of location of each coefficient within the block

Quantization: Threshold Coding



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

Different coefficients quantized with different step-size

$$kc - \frac{c}{2} \leq T(u,v) \leq kc + \frac{c}{2}$$

Finally encode the quantized output!

Quantization (example)

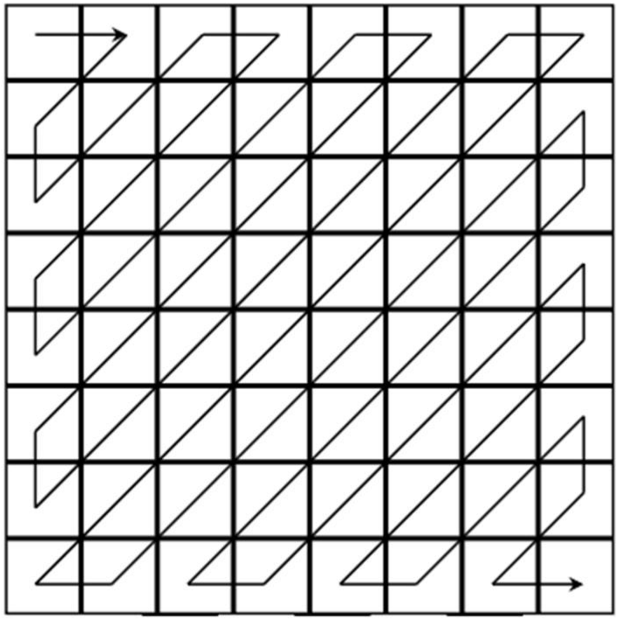
−415	−29	−62	25	55	−20	−1	3
7	−21	−62	9	11	−7	−6	6
−46	8	77	−25	−30	10	7	−5
−50	13	35	−15	−9	6	0	3
11	−8	−13	−2	−1	1	−4	1
−10	1	3	−3	−1	0	2	−1
−4	−1	2	−1	2	−3	1	−2
−1	−1	−1	−2	−1	−1	0	−1

$\xrightarrow{Q}$

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	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

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

Symbol encoding (Zigzag ordering)

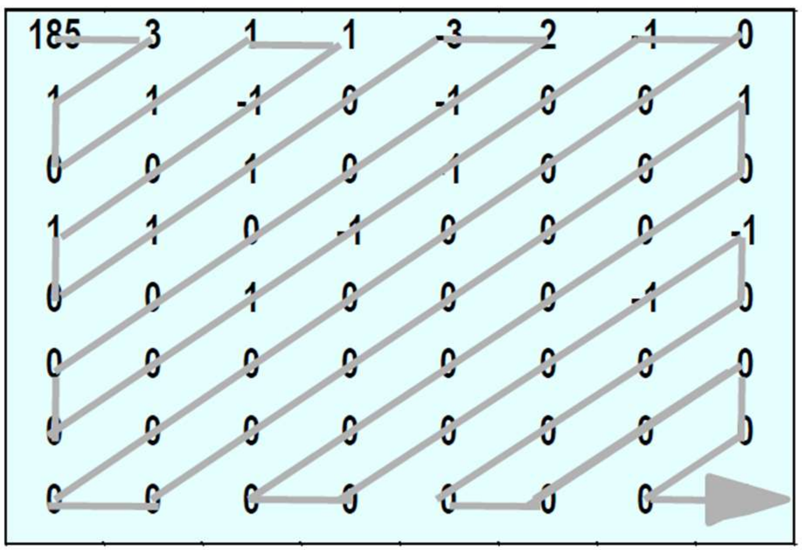


0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

JPEG uses run length encoding!

Symbol coding example

- Zigzag scan (additional example)

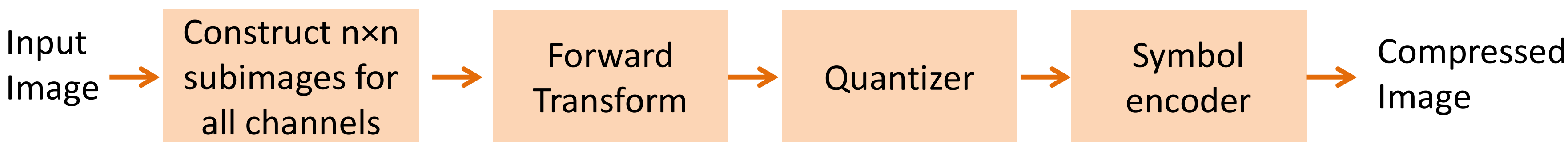


Run length coding

Mean of Block: 185

(0,3) (0,1) (1,1) (0,1) (0,1) (0,-1) (1,1)
(1,1) (0,1) (1,-3) (0,2) (0,-1) (6,1) (0,-1) (0,-1)
(1,-1) (14,1) (9,-1) (0,-1) EOB

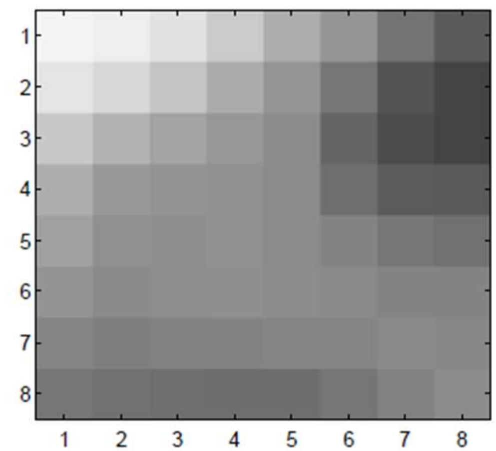
Lossy compression: JPEG



Lest understand the entire procedure with an example

- Consider a single 8×8 pixel block **B**:

$$\begin{pmatrix} 245 & 239 & 227 & 203 & 174 & 150 & 116 & 92 \\ 229 & 216 & 197 & 172 & 150 & 119 & 85 & 69 \\ 201 & 180 & 164 & 152 & 141 & 102 & 77 & 69 \\ 174 & 153 & 148 & 146 & 140 & 112 & 93 & 91 \\ 161 & 145 & 144 & 146 & 141 & 133 & 120 & 114 \\ 149 & 139 & 143 & 144 & 142 & 139 & 133 & 133 \\ 134 & 128 & 131 & 132 & 134 & 134 & 139 & 137 \\ 119 & 114 & 112 & 111 & 111 & 119 & 131 & 141 \end{pmatrix}$$



- Intensity range → [0 255]
- Subtract 127 from each entry and computer 2D DCT

Forward transform and quantization

- DCT of image block

$$\hat{\mathsf{B}} = \begin{pmatrix} 118.9 & 187.7 & -17.7 & 16.8 & 14.4 & 2.4 & 5.3 & 3.5 \\ 104.1 & 187.1 & -30.8 & 10.0 & -1.0 & -4.7 & 0.6 & 0.3 \\ 46.3 & 10.4 & 9.1 & -9.0 & -15.7 & 0 & -1.3 & -2.7 \\ 76.8 & -12.1 & -10.7 & -0.2 & -10.4 & 4.8 & 2.7 & -3.3 \\ 6.4 & -15.3 & 1.7 & -1.7 & -1.1 & 2.5 & 1.1 & -2.5 \\ 10.6 & -5.6 & -6.5 & -0.6 & 2.6 & 0.9 & -1.4 & 2.4 \\ 0.4 & -2.3 & 1.2 & -1.7 & 2.3 & -0.5 & 0.1 & -0.1 \\ 3.2 & -0.7 & -0.9 & 2.6 & -1.1 & 1.5 & -1.8 & 0.2 \end{pmatrix}$$

- Quantization and rounding

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

$$q(\hat{\mathsf{B}}) = \begin{pmatrix} 7 & 17 & -2 & 1 & 1 & 0 & 0 & 0 \\ 9 & 16 & -2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

More than 75% entries are zero (notice their placement)

Encoding

- Zigzag scan

$$q(\hat{\mathrm{B}}) = \begin{pmatrix} 7 & 17 & -2 & 1 & 1 & 0 & 0 & 0 \\ 9 & 16 & -2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

[**7** 17 9 3 16 -2 1 -2 1 5 0 -1 1 1 1 0 0 0 0 -1 EOB]

Lets try to reconstruct

Reconstruction: Decoding + Dequantization

[**7** 17 9 3 16 -2 1 -2 1 5 0 -1 1 1 1 0 0 0 0 -1 EOB]

$$\begin{pmatrix} 7 & 17 & -2 & 1 & 1 & 0 & 0 & 0 \\ 9 & 16 & -2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

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

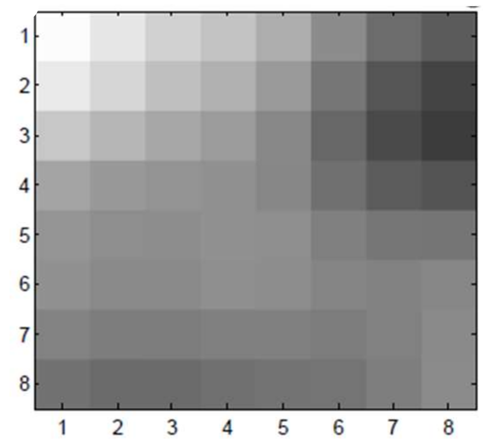
- Dequantization

$$\tilde{B} = \begin{pmatrix} 112 & 187 & -20 & 16 & 24 & 0 & 0 & 0 \\ 108 & 192 & -28 & 19 & 0 & 0 & 0 & 0 \\ 42 & 13 & 16 & 0 & 0 & 0 & 0 & 0 \\ 70 & -17 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -22 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Decoding

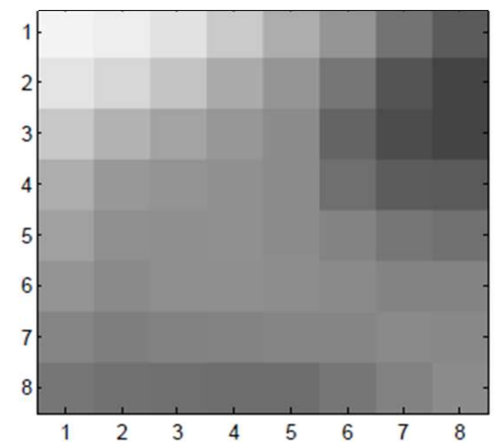
- Compute IDCT and add 127

$$\begin{pmatrix} 254 & 233 & 211 & 197 & 175 & 142 & 110 & 93 \\ 236 & 216 & 194 & 179 & 156 & 120 & 87 & 70 \\ 201 & 184 & 169 & 158 & 138 & 105 & 76 & 62 \\ 166 & 155 & 149 & 147 & 137 & 114 & 94 & 86 \\ 151 & 144 & 143 & 147 & 144 & 130 & 120 & 119 \\ 147 & 140 & 140 & 145 & 143 & 135 & 132 & 137 \\ 133 & 127 & 126 & 130 & 130 & 126 & 131 & 140 \\ 116 & 109 & 109 & 114 & 117 & 118 & 128 & 141 \end{pmatrix}$$



- Compare with original

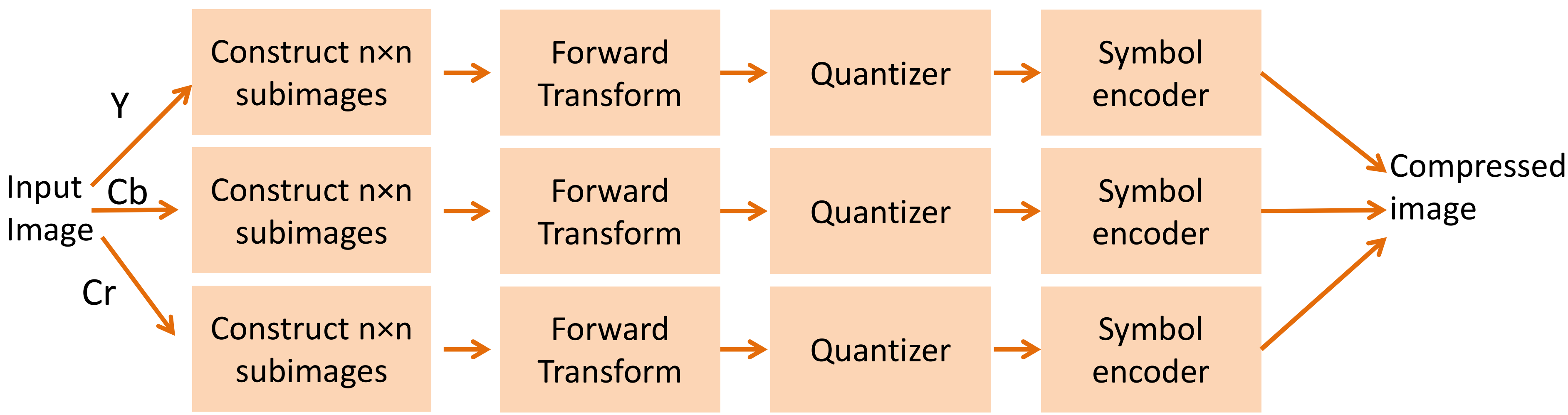
$$\begin{pmatrix} 245 & 239 & 227 & 203 & 174 & 150 & 116 & 92 \\ 229 & 216 & 197 & 172 & 150 & 119 & 85 & 69 \\ 201 & 180 & 164 & 152 & 141 & 102 & 77 & 69 \\ 174 & 153 & 148 & 146 & 140 & 112 & 93 & 91 \\ 161 & 145 & 144 & 146 & 141 & 133 & 120 & 114 \\ 149 & 139 & 143 & 144 & 142 & 139 & 133 & 133 \\ 134 & 128 & 131 & 132 & 134 & 134 & 139 & 137 \\ 119 & 114 & 112 & 111 & 111 & 119 & 131 & 141 \end{pmatrix}$$



Summary JPEG

- Divide into 8×8 subimages
- Compute DCT on each
- Quantize the coefficients
- Order coefficients in zigzag pattern
- Encode 1D sequence using run-length coding and Huffman coding

Color Images



Color Images

Y Cb Cr

- Different quantization matrices for chrominance and luminance
- Chroma subsampling (use reduced resolution of chroma channels)

Quantization matrices

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

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

Chrominance

These matrices are scaled for higher compression!