

IMAGE COMPRESSION-II

Week IX

IMAGE COMPRESSION

- Data redundancy
- Self-information and Entropy
- Error-free and lossy compression
- Huffman coding
- Predictive coding
- Transform coding

Data Redundancy

- **CODING**: Fewer bits to represent frequent symbols.
- **INTERPIXEL / INTERFRAME**: Neighboring pixels have similar values.
- **PSYCHOVISUAL**: Human visual system can not simultaneously distinguish all colors.

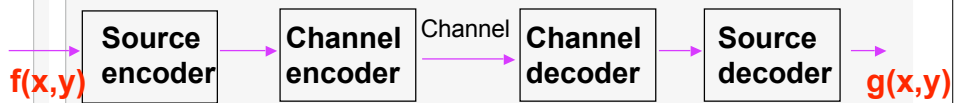
Coding Redundancy (contd.)

- Consider equation (A): It makes sense to assign fewer bits to those r_k for which $p_r(r_k)$ are large in order to reduce the sum.
- this achieves data compression and results in a variable length code.
- More probable gray levels will have fewer # of bits.

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k) \rightarrow (A)$$

General Model

General compression model



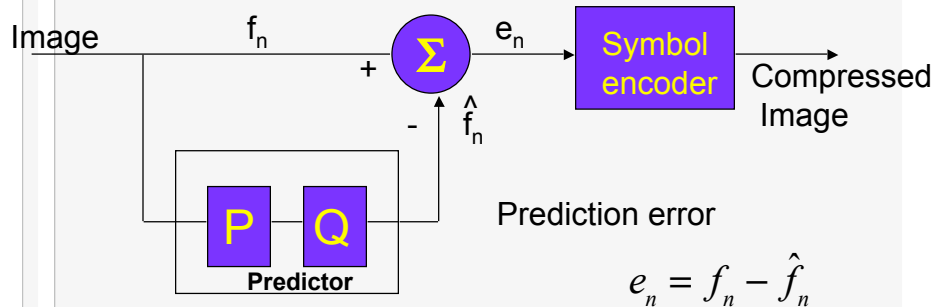
Source encoder



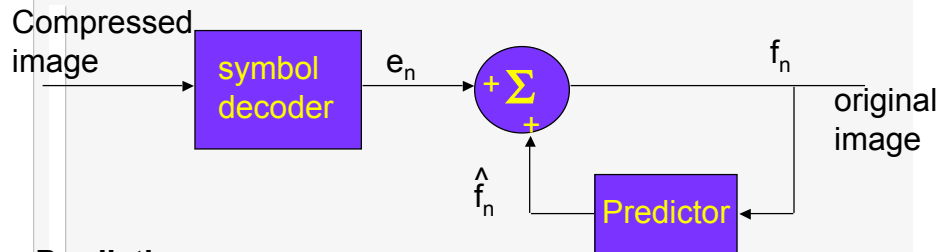
8.2.9: Predictive coding

To reduce / eliminate interpixel redundancies

Lossless predictive coding: ENCODER



Decoder



Prediction error:

$$e_n = f_n - \hat{f}_n$$

e_n is coded using a variable length code (symbol encoder)

$$f_n = e_n + \hat{f}_n$$

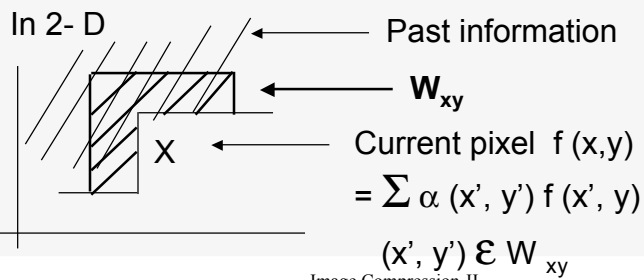
Example

Example 1: $\hat{f}_n = \text{Int} \left(\sum_{i=1}^m \alpha_i f_{n-i} \right)$

→ Linear predictor ; m = order of predictor

Example 2: $\hat{f}_n(x,y) = \text{Int} \left(\sum_{i=1}^m \alpha_i f(x,y-i) \right)$

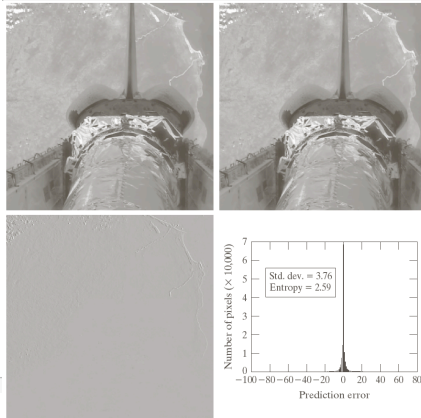
In 2- D



Predictive coding and temporal redundancy

$$\hat{f}(x, y, t) = \text{round}[\alpha f(x, y, t - 1)]$$

$$e(x, y, t) = f(x, y, t) - \hat{f}(x, y, t)$$



a b
c d

FIGURE 8.35

(a) and (b) Two views of Earth from an orbiting space shuttle video. (c) The prediction error image resulting from Eq. (8.2-36). (d) A histogram of the prediction error. (Original images courtesy of NASA.)

ession-II

9

An example of first order predictor



Histograms of original and error images.



Prediction error image

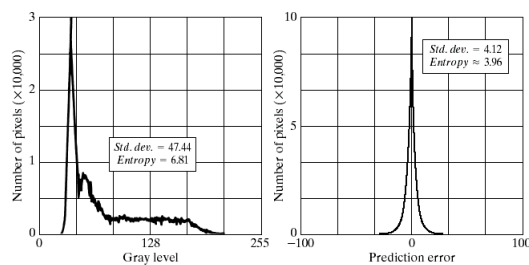


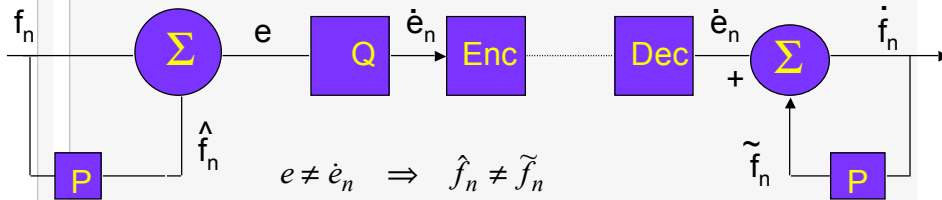
Image Compression-II

10

Lossy Compression (pp. 596)

Lossy compression: uses a quantizer to compress further the number of bits required to encode the 'error'.

First consider this:



Notice that, unlike in the case of loss-less prediction, in lossy prediction the predictors P "see" different inputs at the encoder and decoder

Quantization error

This results in a gradual buildup of error which is due to the quantization error at the encoder site.

In order to minimize this buildup of error due to quantization we should ensure that 'Ps' have the same input in both the cases.

f_n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\hat{f}_n = f_{n-1}$															
e_n	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
\hat{e}_n	0	2	2	2	2
\dot{f}_n	0	2	4	6	8	10

Predictive Coding With Feedback

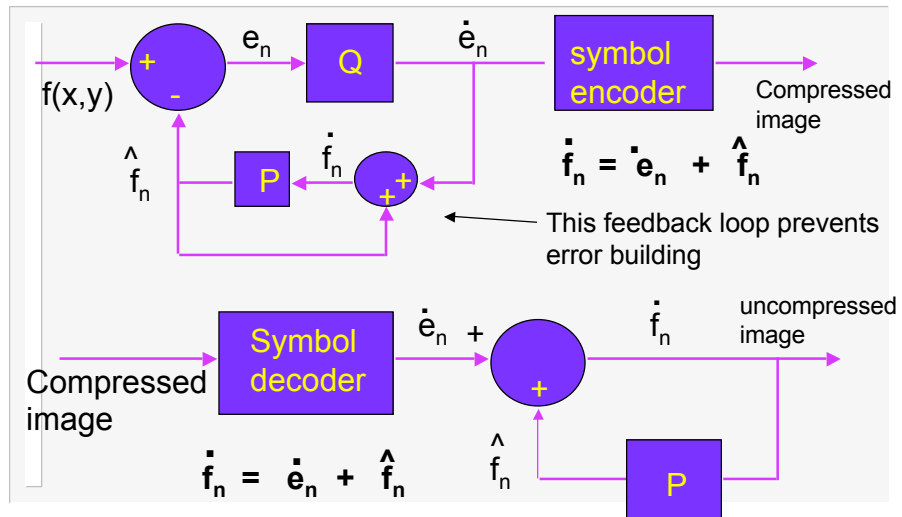


Image Compression-II

13

Example

Example:

$$\hat{f}_n = \alpha \dot{f}_{n-1}$$

$$\text{and } \dot{e}_n = \begin{cases} +\xi & e_n > 0 \\ -\xi & e_n < 0 \end{cases} \quad \begin{array}{l} 0 < \alpha < 1 \\ \text{prediction coefficient} \end{array}$$

$$\dot{f}_n = \dot{e}_n + \hat{f}_n$$

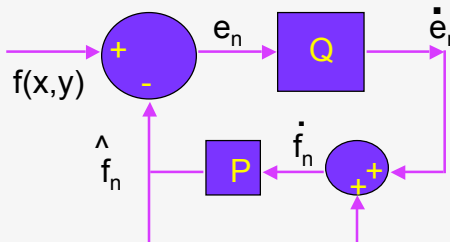
$$= \dot{e}_n + \alpha \dot{f}_{n-1}$$

Image Compression-II

14

Example

with feedback

$$\begin{aligned} f_n &= 0 & 1 & 2 & 3 & 4 \\ e_n &= - & 1 & 2 & 1 & 2 \\ \dot{e}_n &= - & 0 & 2 & 0 & 2 \\ \hat{f}_n &= 0 & 0 & 2 & 2 & 4 \\ \hat{f}_n &= - & 0 & 0 & 2 & 2 \end{aligned}$$


Note: The quantizer used here is-- $\text{floor}(e_n/2)*2$. This is different from the one used in the earlier example. Note that this would result in a worse response if used without Feedback (output will be flat at "0").

Another example

{14, 15, 14, 15, 13, 15, 15, 14, 20, 26, 27, 28, 27, 27, 29, 37, 37, 62, 75, 77, 78, 79, 80, 81, 81, 82, 82}

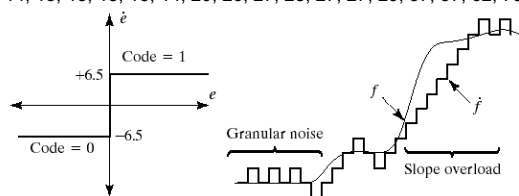


FIGURE 8.22 An example of delta modulation.

n	Input		Encoder			Decoder		Error [f - \hat{f}]
	f	\hat{f}	e	\dot{e}	\hat{f}	\hat{f}		
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	-5.5
2	14	20.5	-6.5	-6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	-5.5
·	·	·	·	·	·	·	·	·
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·

A comparison (Fig 8.23)

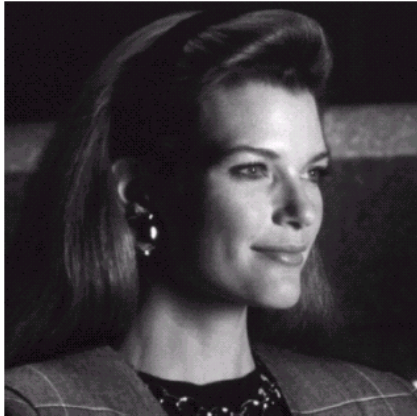
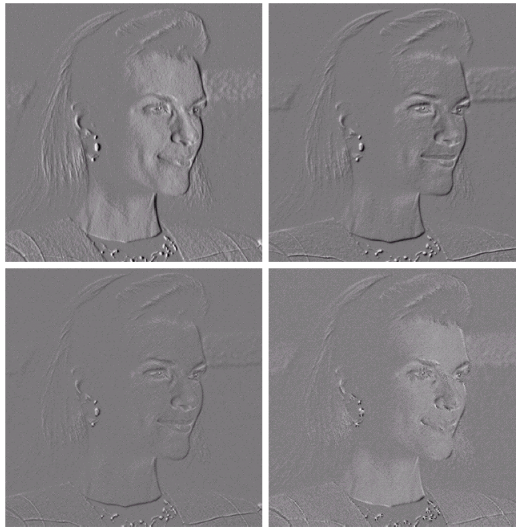


FIGURE 8.23 A
512 × 512 8-bit
monochrome
image.

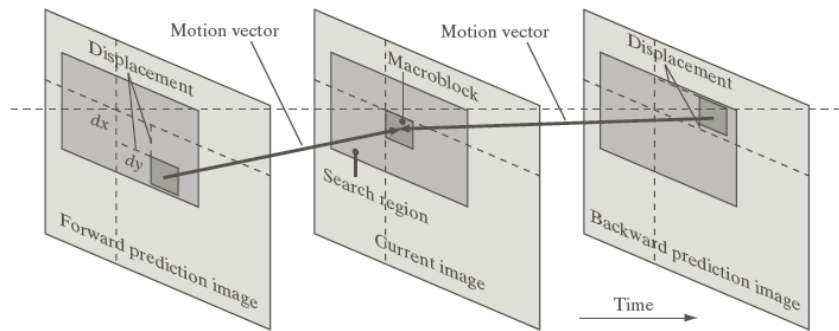
Four linear predictors

a b
c d

FIGURE 8.24 A
comparison of
four linear
prediction
techniques.



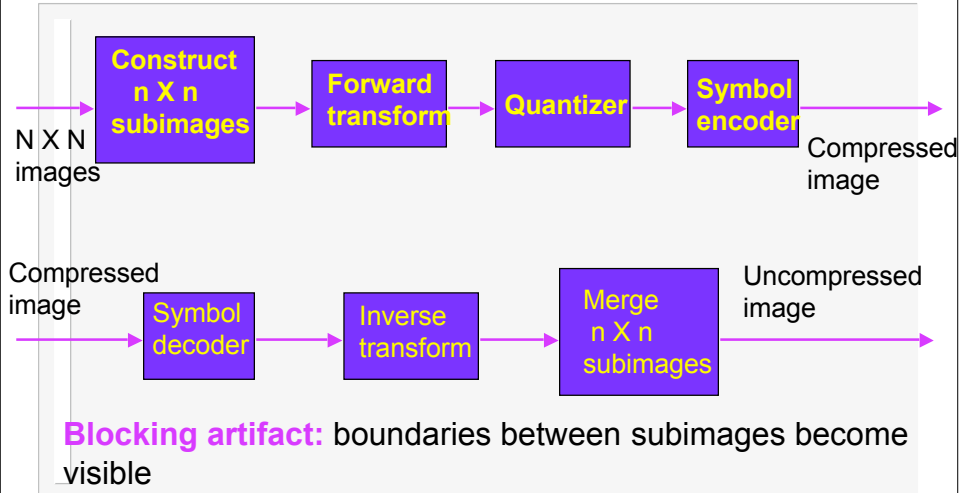
Motion compensation in Video



Block Transform Coding

Section 8.2.8

Transform coding



Transform Selection

- DFT
- Discrete Cosine Transform (DCT)
- Wavelet transform
- Karhunen-Loeve Transform (KLT)
-

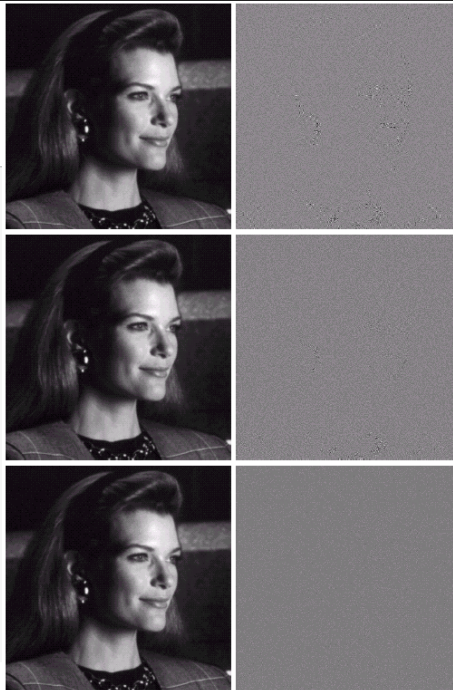
$$T(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x, y) r(x, y, u, v) \text{ --- (8.2.10)}$$

$$g(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) s(x, y, u, v) \text{ --- 8.2.11}$$

Transform Kernels

- Separable if $r(x, y, u, v) = r_1(x, u)r_2(y, v) \text{ --- (8.2.12)}$
- E.g. DFT: $r(x, y, u, v) = \exp(-j2\pi(ux + vy) / n)$
- E.g. Walsh-Hadamard transform (see page 568, text)
- E.g. DCT

Approximations using DFT, Hadamard and DCT, and the scaled error images



Discrete Cosine Transform

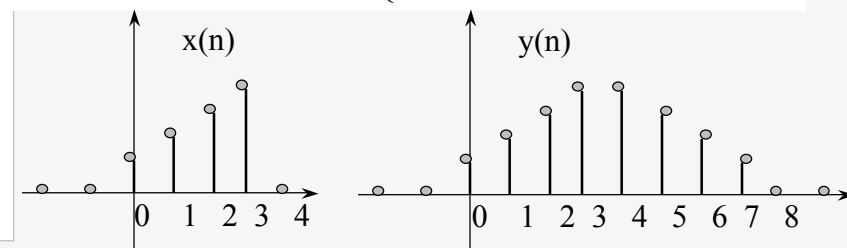
Ahmed, Natarajan, and Rao, IEEE T-Computers, pp. 90-93, 1974.

1-D Case: Extended 2N Point Sequence

Consider 1 - D first; Let $x(n)$ be a N point sequence $0 \leq n \leq N - 1$.

$$x(n) \leftrightarrow \begin{matrix} 2 - N \text{ point} \\ y(n) \end{matrix} \xleftrightarrow{DFT} \begin{matrix} 2 - N \text{ point} \\ Y(u) \end{matrix} \leftrightarrow \begin{matrix} N - \text{point} \\ C(u) \end{matrix}$$

$$y(n) = x(n) + x(2N - 1 - n) = \begin{cases} x(n), & 0 \leq n \leq N - 1 \\ x(2N - 1 - n), & N \leq n \leq 2N - 1 \end{cases}$$



DCT & DFT

$$\begin{aligned}
 Y(u) &= \sum_{n=0}^{2N-1} y(n) \exp\left(-j \frac{2\pi}{2N} un\right) \\
 &= \sum_{n=0}^{N-1} x(n) \exp\left(-j \frac{2\pi}{2N} un\right) + \sum_{n=N}^{2N-1} x(2N-1-n) \exp\left(-j \frac{2\pi}{2N} un\right) \\
 &= \sum_{n=0}^{N-1} x(n) \exp\left(-j \frac{2\pi}{2N} un\right) + \sum_{m=0}^{N-1} x(m) \exp\left(-j \frac{2\pi}{2N} u(2N-1-m)\right) \\
 &= \exp\left(j \frac{\pi}{2N} u\right) \sum_{n=0}^{N-1} x(n) \exp\left(-j \frac{\pi}{2N} u - j \frac{2\pi}{2N} un\right) \\
 &\quad + \exp\left(j \frac{\pi}{2N} u\right) \sum_{n=0}^{N-1} x(n) \exp\left(j \frac{\pi}{2N} u + j \frac{2\pi}{2N} un\right) \\
 &= \exp\left(j \frac{\pi}{2N} u\right) \sum_{n=0}^{N-1} 2x(n) \cos\left(\frac{\pi}{2N} u(2n+1)\right).
 \end{aligned}$$

Image Compression-II

27

DCT

The N-point DCT of $x(n)$, $C(u)$, is given by

$$C(u) = \begin{cases} \exp\left(-j \frac{\pi}{2N} u\right) Y(u), & 0 \leq u \leq N-1 \\ 0 & \text{otherwise.} \end{cases}$$

The unitary DCT transformations are:

$$F(u) = \alpha(u) \sum_{n=0}^{N-1} f(n) \cos\left(\frac{\pi}{2N} (2n+1)u\right), \quad 0 \leq u \leq N-1, \text{ where}$$

$$\alpha(0) = \frac{1}{\sqrt{N}}, \quad \alpha(u) = \sqrt{\frac{2}{N}} \text{ for } 1 \leq k \leq N-1.$$

The inverse transformation is

$$f(n) = \sum_{u=0}^{N-1} \alpha(u) F(u) \cos\left(\frac{\pi}{2N} (2n+1)u\right), \quad 0 \leq u \leq N-1.$$

Image Compression-II

28

Discrete Cosine Transform—in 2-D

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

for $u, v = 0, 1, 2, \dots, N-1$, where

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \dots, N-1. \end{cases} \quad \text{and}$$

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

for $x, y = 0, 1, 2, \dots, N-1$

DCT Summary

$$T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v) \dots \dots \dots 8.2.10$$

$$r(x, y, u, v) = s(x, y, u, v)$$

$$= \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right] \dots \dots \dots 8.2.18$$

DCT Basis functions

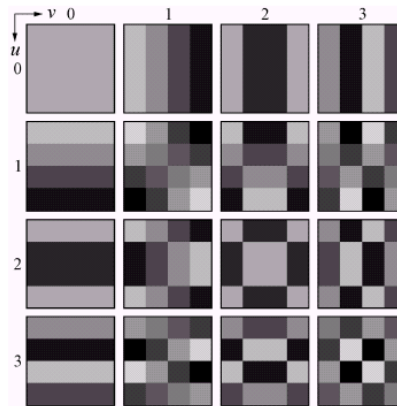


FIGURE 8.30 Discrete-cosine basis functions for $N = 4$. The origin of each block is at its top left.

Implicit Periodicity-DFT vs DCT (Fig 8.32)

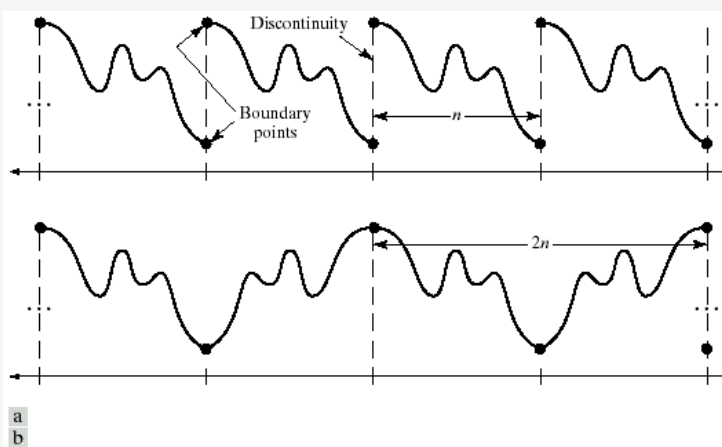


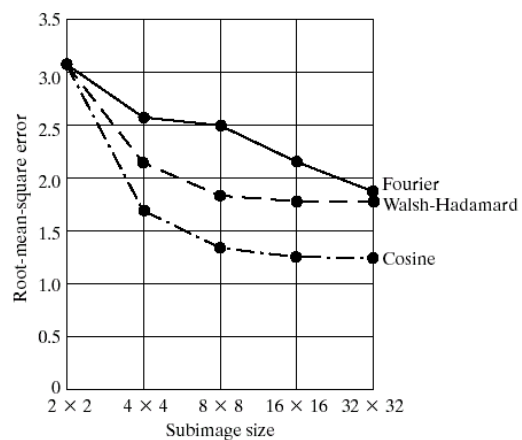
FIGURE 8.32 The periodicity implicit in the 1-D (a) DFT and (b) DCT.

Why DCT?

- Blocking artifacts less pronounced in DCT than in DFT.
- Good approximation to the Karhunen-Loeve Transform (KLT) but with basis vectors fixed.
- DCT is used in JPEG image compression standard.

Sub-image size selection (fig 8.26)

FIGURE 8.33
Reconstruction
error versus
subimage size.



Different sub-image sizes



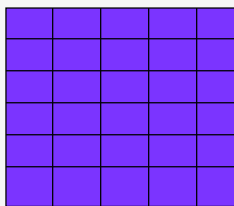
a b
c d
e f

FIGURE 8.34 Approximations of Fig. 8.23 using 25% of the DCT coefficients: (a) and (b) 8×8 subimage results; (c) zoomed original; (d) 2×2 result; (e) 4×4 result; and (f) 8×8 result.

Bit Allocation/Threshold Coding

- # of coefficients to keep
 - How to quantize them
- Threshold coding
Zonal coding

Threshold coding

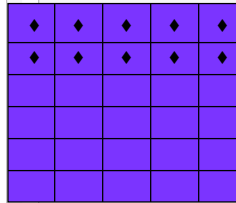


For each subimage i

- Arrange the transform coefficients in decreasing order of magnitude
- Keep only the top X% of the coefficients and discard rest.
- Code the retained coefficient using variable length code.

Zonal Coding

Zonal coding:



i^{th} coefficient
in each sub
image

- (1) Compute the variance of each of the transform coeff; use the subimages to compute this.
- (2) Keep X% of their coeff. which have maximum variance.
- (3) Variable length coding (proportional to variance)

Bit allocation: In general, let the number of bits allocated be made proportional to the variance of the coefficients. Suppose the total number of bits per block is B. Let the number of retained coefficients be M. Let $v(i)$ be variance of the i -th coefficient. Then

$$b(i) = \frac{B}{M} + \frac{1}{2} \log_2 v(i) - \frac{1}{2M} \sum_{i=1}^M \log_2 v(i)$$

Zonal Mask & bit allocation: an example

1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	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	0

8	7	6	4	3	0	0	0
7	6	5	4	0	0	0	0
6	5	4	0	0	0	0	0
4	4	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

Typical Masks (Fig 8.36)

a b
c d

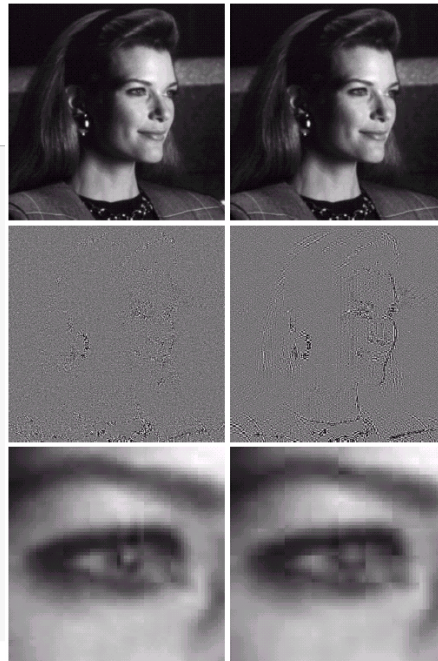
FIGURE 8.36 A typical (a) zonal mask, (b) zonal bit allocation, (c) threshold mask, and (d) thresholded coefficient ordering sequence. Shading highlights the coefficients that are retained.

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

Image Approximations

a b
c d
e f

FIGURE 8.35 Approximations of Fig. 8.23 using 12.5% of the 8×8 DCT coefficients: (a), (c), and (e) threshold coding results; (b), (d), and (f) zonal coding results



The JPEG standard

- The following is mostly from Tekalp's book, Digital Video Processing by M. Tekalp (Prentice Hall).
- For the new JPEG-2000 check out the web site www.jpeg.org.
- See Example 8.17 in the text

JPEG (contd.)

- JPEG is a lossy compression standard using DCT.
- Activities started in 1986 and the ISO in 1992.
- Four modes of operation: Sequential (baseline), hierarchical, progressive, and lossless.
- Arbitrary image sizes; DCT mode 8-12 bits/sample. Luminance and chrominance channels are separately encoded.
- We will only discuss the baseline method.

JPEG-baseline.

- DCT: The image is divided into 8x8 blocks. Each pixel is level shifted by 2^{n-1} where 2^n is the maximum number of gray levels in the image. Thus for 8 bit images, you subtract 128. Then the 2-D DCT of each block is computed. For the baseline system, the input and output data precision is restricted to 8 bits and the DCT values are restricted to 11 bits.
- Quantization: the DCT coefficients are threshold coded using a quantization matrix, and then reordered using zig-zag scanning to form a 1-D sequence.
- The non-zero AC coefficients are Huffman coded. The DC coefficients of each block are DPCM coded relative to the DC coefficient of the previous block.

JPEG -color image

- RGB to Y-Cr-Cb space
 - $Y = 0.3R + 0.6G + 0.1B$
 - $Cr = 0.5(B - Y) + 0.5$
 - $Cb = (1/1.6)(R - Y) + 0.5$
- Chrominance samples are sub-sampled by 2 in both directions.

Y1	Y2	Y3	Y4
Y5	Y6	Y7	Y8
Y9	Y10	Y11	Y12
Y13	Y14	Y15	Y16

Cr1	Cr2
Cr3	Cr4

Cb1	Cb2
Cb3	Cb4

Non-Interleaved

Scan 1: Y1, Y2, ..., Y16

Scan 2: Cr1, Cr2, Cr3, Cr4

Scan 3: Cb1, Cb2, Cb3, Cb4

Interleaved: Y1, Y2, Y3, Y4, Cr1, Cb1, Y5, Y6, Y7, Y8, Cr2, Cb2, ...

JPEG – quantization matrices

- Check out the matlab workspace (dctex.mat).
 - Quantization table for the luminance channel.
 - Quantization table for the chrominance channel.
 - JPEG baseline method
 - Consider the 8x8 image (matlab: array s.)
 - Level shifted ($s-128=sd$).
 - 2d-DCT: $dct2(sd)=dcts$
 - After dividing by quantization matrix qmat: $dctshat$.
 - Zigzag scan as in threshold coding.
- [20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, EOB].

An 8x8 sub-image (s)

s = (8x8block)

```
183 160 94 153 194 163 132 165
183 153 116 176 187 166 130 169
179 168 171 182 179 170 131 167
177 177 179 177 179 165 131 167
178 178 179 176 182 164 130 171
179 180 180 179 183 164 130 171
179 179 180 182 183 170 129 173
180 179 181 179 181 170 130 169
```

sd =(level shifted)

```
55 32 -34 25 66 35 4 37
55 25 -12 48 59 38 2 41
51 40 43 54 51 42 3 39
49 49 51 49 51 37 3 39
50 50 51 48 54 36 2 43
51 52 52 51 55 36 2 43
51 51 52 54 55 42 1 45
52 51 53 51 53 42 2 41
```

2D DCT (dcts) and the quantization matrix (qmat)

dcts=

```

312 56 -27 17 79 -60 26 -26
-38 -28 13 45 31 -1 -24 -10
-20 -18 10 33 21 -6 -16 -9
-11 -7 9 15 10 -11 -13 1
-6 1 6 5 -4 -7 -5 5
3 3 0 -2 -7 -4 1 2
3 5 0 -4 -8 -1 2 4
3 1 -1 -2 -3 -1 4 1
    
```

qmat=

```

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
    
```

Division by qmat (dctshat)=dcts/qmat

dcthat=

```

20 5 -3 1 3 -2 1 0
-3 -2 1 2 1 0 0 0
-1 -1 1 1 1 0 0 0
-1 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
    
```

dcts=

```

312 56 -27 17 79 -60 26 -26
-38 -28 13 45 31 -1 -24 -10
-20 -18 10 33 21 -6 -16 -9
-11 -7 9 15 10 -11 -13 1
-6 1 6 5 -4 -7 -5 5
3 3 0 -2 -7 -4 1 2
3 5 0 -4 -8 -1 2 4
3 1 -1 -2 -3 -1 4 1
    
```


JPEG – baseline method example

Zigzag scan as in threshold coding.

[20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -2, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, EOB].

- The DC coefficient is DPCM coded (difference between the DC coefficient of the previous block and the current block.)
- The AC coef. are mapped to run-level pairs.
(0,5), (0,-3), (0, -1), (0,-2),(0,-3),(0,1),(0,1),(0,-1),(0,-1), (2,1), (0,2), (0,3), (0,-2),(0,1),(0,1),(6,1),(0,1),(1,1),EOB.
- These are then Huffman coded (codes are specified in the JPEG scheme.)
- The decoder follows an inverse sequence of operations. The received coefficients are first multiplied by the same quantization matrix.
(recddctshat=dctshat.*qmat.)
- Compute the inverse 2-D dct. (recdsd=idct2(recddctshat); add 128 back.
(recds=recdsd+128.)

Decoder

Recddcthat=dcthat*qmat

Recdsd=

320	55	-30	16	72	-80	51	0	67	12	-9	20	69	43	-8	42
-36	-24	14	38	26	0	0	0	58	25	15	30	65	40	-4	47
-14	-13	16	24	40	0	0	0	46	41	44	40	59	38	0	49
-14	0	0	29	0	0	0	0	41	52	59	43	57	42	3	42
0	0	0	0	0	0	0	0	44	54	58	40	58	47	3	33
0	0	0	0	0	0	0	0	49	52	53	40	61	47	1	33
0	0	0	0	0	0	0	0	53	50	53	46	63	41	0	45
0	0	0	0	0	0	0	0	55	50	56	53	64	34	-1	57

Received signal

Reconstructed S=

```

195 140 119 148 197 171 120 170
186 153 143 158 193 168 124 175
174 169 172 168 187 166 128 177
169 180 187 171 185 170 131 170
172 182 186 168 186 175 131 161
177 180 181 168 189 175 129 161
181 178 181 174 191 169 128 173
183 178 184 181 192 162 127 185
    
```

s = (8x8block)

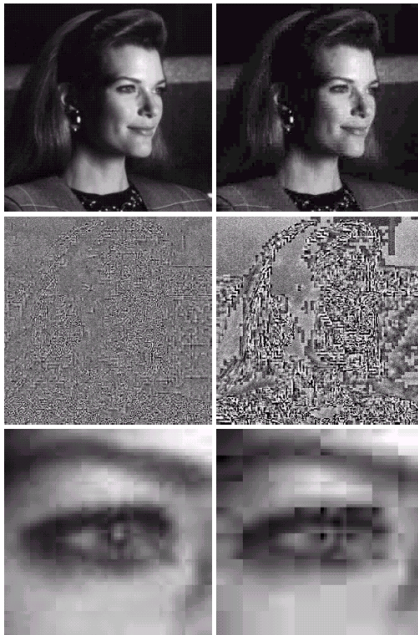
```

183 160 94 153 194 163 132 165
183 153 116 176 187 166 130 169
179 168 171 182 179 170 131 167
177 177 179 177 179 165 131 167
178 178 179 176 182 164 130 171
179 180 180 179 183 164 130 171
179 179 180 182 183 170 129 173
180 179 181 179 181 170 130 169
    
```

Example

a b
c d
e f

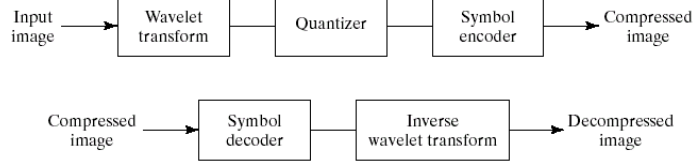
FIGURE 8.38 Left column: Approximations of Fig. 8.23 using the DCT and normalization array of Fig. 8.37(b). Right column: Similar results for 4Z.



Wavelet Compression

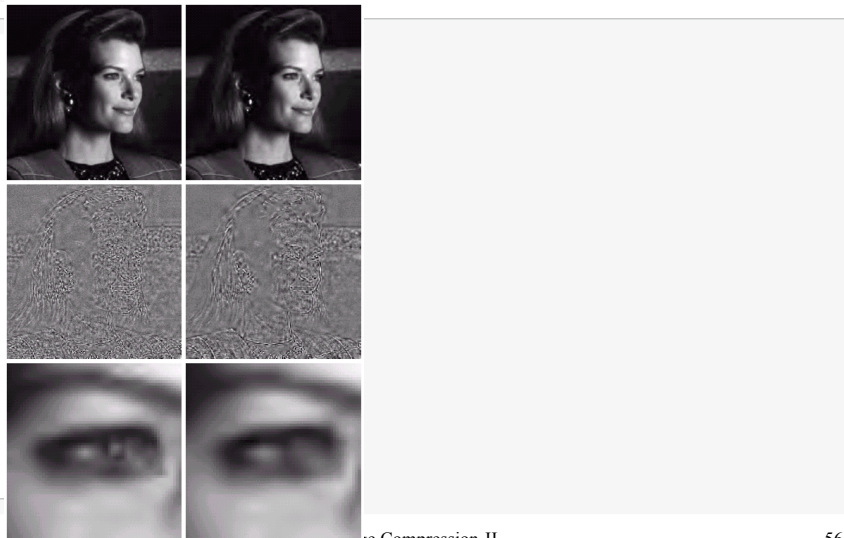
a
b

FIGURE 8.39 A wavelet coding system: (a) encoder; (b) decoder.



a b
c d
e f

FIGURE 8.41 (a), (c), and (e) Wavelet coding results with a compression ratio of 108 to 1; (b), (d), and (f) similar results for a compression of 167 to 1.



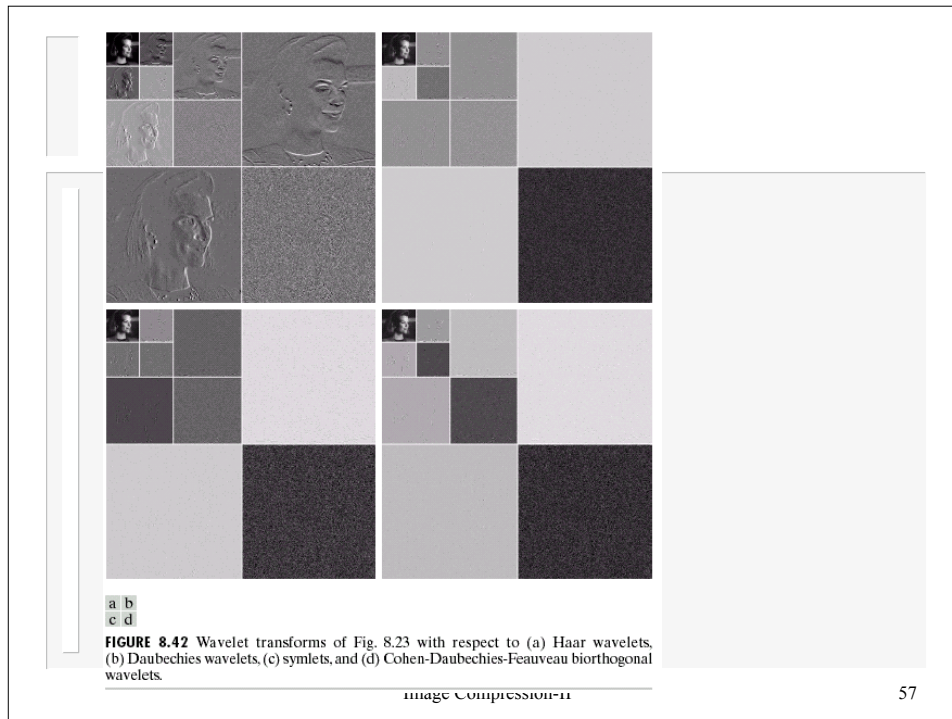


Image Compression: Summary

- Data redundancy
- Self-information and Entropy
- Error-free compression
 - Huffman coding, Arithmetic coding, LZW coding, Run-length encoding
 - Predictive coding
- Lossy coding techniques
 - Predictive coding (Lossy)
 - Transform coding
 - DCT, DFT, KLT, ...
- JPEG image compression standard