

Image Compression (Chapter 8)



CS474/674 – Prof. Bebis

Image Compression

- The goal of image compression is to reduce the amount of data required to represent a digital image.

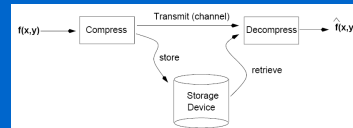


Image Compression (cont'd)

- Lossless**
 - Information preserving
 - Low compression ratios
- Lossy**
 - Not information preserving
 - High compression ratios

Trade-off: information loss **vs** compression ratio

Data ≠ Information

- Data and information are not synonymous terms!
- Data** is the means by which **information** is conveyed.
- Data compression aims to reduce the amount of data while preserving as much information as possible.

Data vs Information (cont'd)

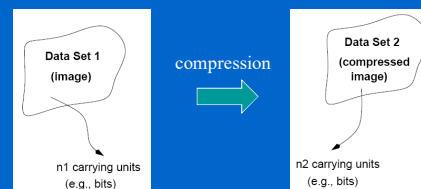
- The same **information** can be represented by different amount of **data** – for example:

Ex1: *Your wife, Helen, will meet you at Logan Airport in Boston at 5 minutes past 6:00 pm tomorrow night*

Ex2: *Your wife will meet you at Logan Airport at 5 minutes past 6:00 pm tomorrow night*

Ex3: *Helen will meet you at Logan at 6:00 pm tomorrow night*

Compression Ratio



Compression ratio:
$$C_R = \frac{n_1}{n_2}$$

Relevant Data Redundancy

$$R_D = 1 - \frac{1}{C_R}$$

Example:

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10} = 0.9$

(90% of the data in dataset 1 is redundant)

if $n_2 = n_1$, then $C_R = 1$, $R_D = 0$

if $n_2 \ll n_1$, then $C_R \rightarrow \infty$, $R_D \rightarrow 1$

Types of Data Redundancy

- (1) Coding Redundancy
- (2) Interpixel Redundancy
- (3) Psychovisual Redundancy

- Data compression attempts to reduce one or more of these redundancy types.

Coding - Definitions

- Code:** a list of symbols (letters, numbers, bits etc.)
- Code word:** a sequence of symbols used to represent some information (e.g., gray levels).
- Code word length:** number of symbols in a code word.

Example: (binary code, symbols: 0,1, length: 3)

0: 000 4: 100
1: 001 5: 101
2: 010 6: 110
3: 011 7: 111

Coding - Definitions (cont'd)

$N \times M$ image

r_k : k-th gray level

$l(r_k)$: # of bits for r_k

$P(r_k)$: probability of r_k

$$\text{Average \# of bits: } L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k) P(r_k)$$

$$\text{Total \# of bits: } NML_{avg}$$

$$\text{Expected value: } E(X) = \sum_x xP(X=x)$$

Coding Redundancy

- Case 1:** $l(r_k) = \text{constant length}$

Example:

r_k	$p(r_k)$	Code 1	$l(r_k)$
$r_0 = 0$	0.19	000	3
$r_1 = 1/7$	0.25	001	3
$r_2 = 2/7$	0.21	010	3
$r_3 = 3/7$	0.16	011	3
$r_4 = 4/7$	0.08	100	3
$r_5 = 5/7$	0.06	101	3
$r_6 = 6/7$	0.03	110	3
$r_7 = 1$	0.02	111	3

Assume an image with $L = 8$

Assume $l(r_k) = 3$, $L_{avg} = \sum_{k=0}^7 3P(r_k) = 3 \sum_{k=0}^7 P(r_k) = 3 \text{ bits}$

Total number of bits: $3NM$

Coding Redundancy (cont'd)

- Case 2:** $l(r_k) = \text{variable length}$

Table 6.1 Variable-Length Coding Example					
r_k	$p(r_k)$	Code 1	$l(r_k)$	Code 2	$l(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

$$L_{avg} = \sum_{k=0}^7 l(r_k) P(r_k) = 2.7 \text{ bits}$$

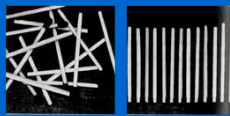
$$C_R = \frac{3}{2.7} = 1.11 \text{ (about 10\%)}$$

Total number of bits: $2.7NM$

$$R_D = 1 - \frac{1}{1.11} = 0.099$$

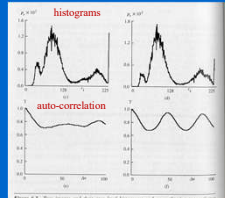
Interpixel redundancy

- Interpixel redundancy implies that pixel values are correlated (i.e., a pixel value can be reasonably predicted by its neighbors).



$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f(x)g(x+a)da$$

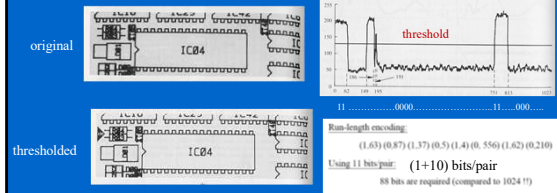
auto-correlation: $f(x)=g(x)$



Interpixel redundancy (cont'd)

- To reduce interpixel redundancy, some kind of transformation must be applied on the data (e.g., thresholding, DFT, DWT)

Example:



Psychovisual redundancy

- The human eye is more sensitive to the **lower** frequencies than to the **higher** frequencies in the visual spectrum.
- Idea: discard data that is perceptually insignificant!

Psychovisual redundancy (cont'd)

Example: **quantization**



Measuring Information

- A key question in image compression is:
"What is the minimum **amount of data** that is sufficient to describe completely an image without loss of information?"
- How do we measure the **information content** of an image?

Measuring Information (cont'd)

- We assume that **information generation** is a probabilistic process.
- Idea: associate information with probability!

A random event E with probability $P(E)$ contains:

$$I(E) = \log\left(\frac{1}{P(E)}\right) = -\log(P(E)) \text{ units of information}$$

Note: $I(E)=0$ when $P(E)=1$

How much information does a pixel contain?

- Suppose that gray level values are generated by a random process, then r_k contains:

$$I(r_k) = -\log(P(r_k)) \quad \text{units of information!}$$

(assume statistically independent random events)

How much information does an image contain?

- Average information content of an image:

$$E = \sum_{k=0}^{L-1} I(r_k) P(r_k)$$

using $I(r_k) = -\log(P(r_k))$

Entropy: $H = -\sum_{k=0}^{L-1} P(r_k) \log(P(r_k))$ units/pixel (e.g., bits/pixel)

Redundancy - revisited

- Redundancy: $R = L_{avg} - H$

where: $L_{avg} = E(I(r_k)) = \sum_{k=0}^{L-1} I(r_k) P(r_k)$

Note: if $L_{avg} = H$, then $R=0$ (no redundancy)

Entropy Estimation

- It is not easy to estimate H reliably!

image

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Gray Level	Count	Probability
21	12	3/8
95	4	1/8
169	4	1/8
243	12	3/8

Entropy Estimation (cont'd)

- First order estimate of H:

$$H = -\sum_{k=0}^3 P(r_k) \log(P(r_k)) = 1.81 \text{ bits/pixel}$$

Total bits: $4 \times 8 \times 1.81 = 58$ bits

$$L_{avg} = 8 \text{ bits/pixel} \quad R = L_{avg} - H$$

The first-order estimate provides only a lower-bound on the compression that can be achieved.

Estimating Entropy (cont'd)

- Second order estimate of H:
 - Use relative frequencies of pixel blocks :

image

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Gray Level Pair	Count	Probability
(21, 21)	8	1/4
(21, 95)	4	1/8
(95, 169)	4	1/8
(169, 243)	4	1/8
(243, 243)	8	1/4
(243, 21)	4	1/8

$$H = 2.5/2 = 1.25 \text{ bits/pixel}$$

Differences in Entropy Estimates

- Differences between higher-order estimates of entropy and the first-order estimate indicate the presence of **interpixel redundancy**!
- Need to apply some **transformation** to deal with interpixel redundancy!

Differences in Entropy Estimates (cont'd)

- For example, consider pixel differences:

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243



21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

Gray Level or Difference	Count	Probability
0	16	1/2
21	4	1/8
74	12	3/8

Differences in Entropy Estimates (cont'd)

- What is the entropy of the difference image?

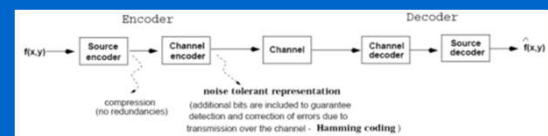
$$H = - \sum_{k=0}^2 P(r_k) \log(P(r_k)) = 1.41 \text{ bits/pixel}$$

(better than the entropy of the original image $H=1.81$)

- An even better transformation is possible since the second order entropy estimate is lower:

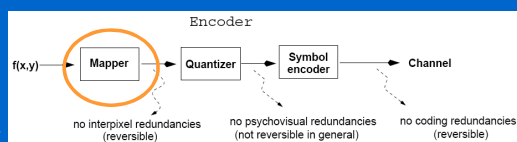
$$1.41 \text{ bits/pixel} > 1.25 \text{ bits/pixel}$$

Image Compression Model



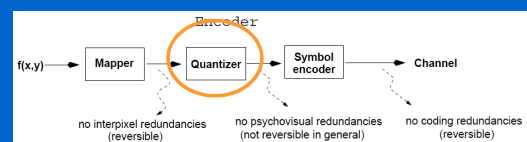
We will focus on the **Source Encoder/Decoder** only.

Image Compression Model (cont'd)

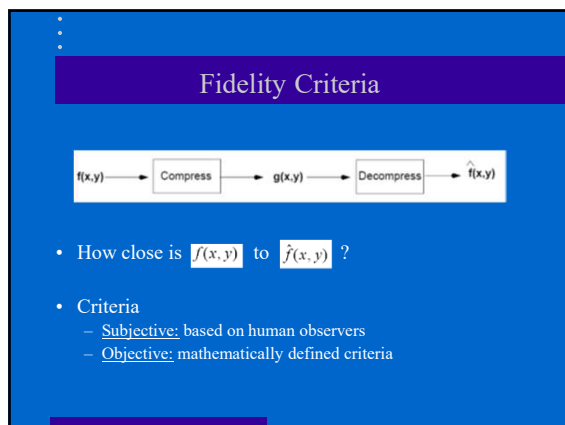
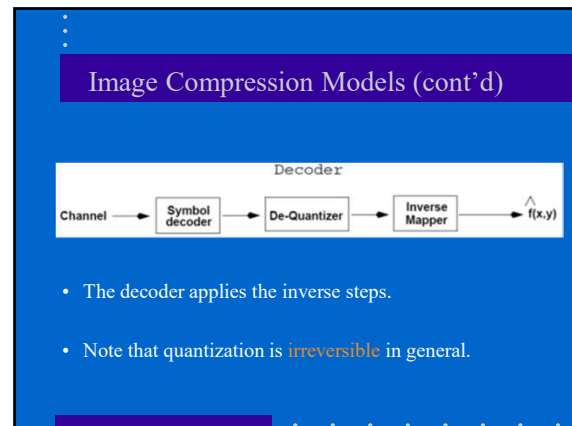
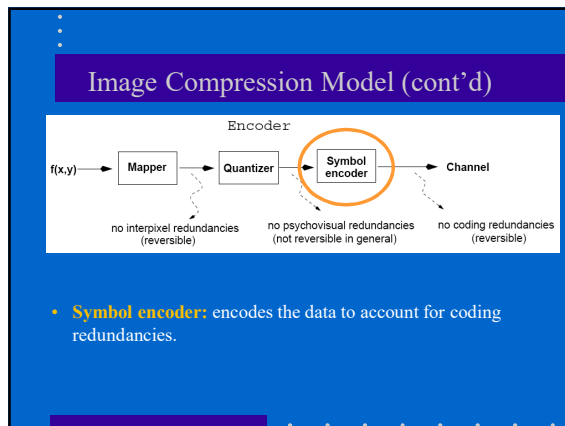


- Mapper**: transforms data to account for interpixel redundancies.

Image Compression Model (cont'd)

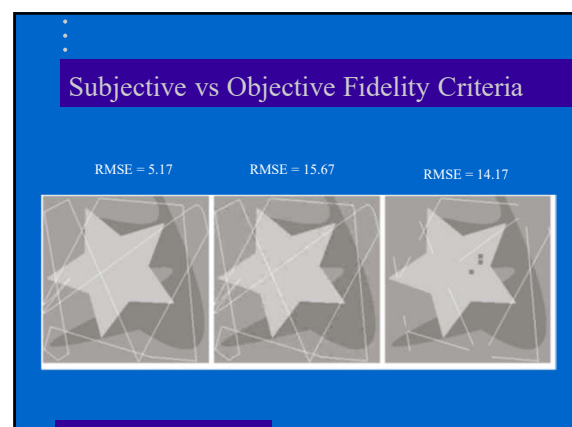
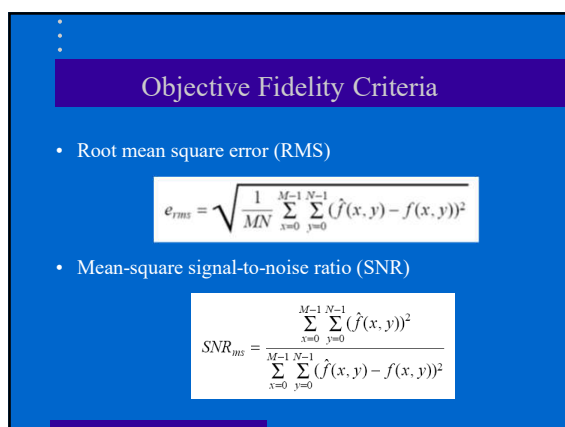


- Quantizer**: quantizes the data to account for psychovisual redundancies.



Subjective Fidelity Criteria

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

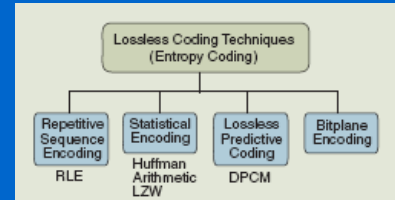
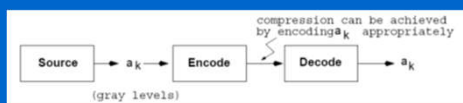


Lossless Compression



$$e(x, y) = \hat{f}(x, y) - f(x, y) = 0$$

Taxonomy of Lossless Methods

Huffman Coding
(addresses coding redundancy)

- A **variable-length coding** technique.
- Source symbols are encoded **one** at a time!
 - There is a **one-to-one correspondence** between source symbols and code words.
- **Optimal code** - minimizes code word length per source symbol.

Huffman Coding (cont'd)

- **Forward Pass**
 1. Sort probabilities per symbol
 2. Combine the lowest two probabilities
 3. Repeat Step2 until only two probabilities remain.

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	
a_6	0.3	0.3	0.3	0.3	
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1	0.1	
a_3	0.06				
a_5	0.04				

Huffman Coding (cont'd)

- **Backward Pass**
Assign code symbols going backwards

Original source		Source reduction			
Sym.	Prob.	Code	1	2	3
a_2	0.4	1	0.4 1	0.4 1	0.4 1
a_6	0.3	00	0.3 00	0.3 00	0.3 00
a_1	0.1	011	0.1 011	0.2 010	0.3 01
a_4	0.1	0100	0.1 0100	0.1 011	0.3 01
a_3	0.06	01010	0.1 0101		
a_5	0.04	01011			

Huffman Coding (cont'd)

- L_{avg} assuming **Huffman coding**:

$$L_{avg} = E(l(a_k)) = \sum_{k=1}^6 l(a_k)P(a_k) =$$

$$3 \times 0.1 + 1 \times 0.4 + 5 \times 0.06 + 4 \times 0.1 + 5 \times 0.04 + 2 \times 0.3 = 2.2 \text{ bits/symbol}$$

- L_{avg} assuming **binary coding**:

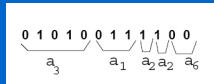
6 symbols, we need a 3-bit code

(a_1 : 000, a_2 : 001, a_3 : 010, a_4 : 011, a_5 : 100, a_6 : 101)

$$L_{avg} = \sum_{k=1}^6 l(a_k)P(a_k) = \sum_{k=1}^6 3P(a_k) = 3 \sum_{k=1}^6 P(a_k) = 3 \text{ bits/symbol}$$

Huffman Coding/Decoding

- Coding/Decoding can be implemented using a **look-up table**.
- Decoding can be done unambiguously.



Original source		
Sym.	Prob.	Code
a_3	0.4	1
a_5	0.3	00
a_1	0.1	011
a_2	0.1	0100
a_4	0.06	01010
a_5	0.04	01011

Arithmetic (or Range) Coding (addresses coding redundancy)

- The main weakness of Huffman coding is that it encodes source symbols **one** at a time.
- Arithmetic coding encodes **sequences** of source symbols together.
 - There is **no** one-to-one correspondence between source symbols and code words.
- Slower than Huffman coding but can achieve better compression.

Arithmetic Coding (cont'd)

- A sequence of source symbols is assigned to a **sub-interval** in $[0,1]$ which can be represented by an **arithmetic code**, e.g.:

$a_1 a_2 a_3 a_4$
→
sub-interval $[0.06752, 0.0688)$
→
arithmetic code 0.068

- Start with the interval $[0, 1]$; a sub-interval is chosen to represent the message which becomes **smaller** and **smaller** as the number of symbols in the message increases.

Arithmetic Coding (cont'd)

Encode message: $a_1 a_2 a_3 a_4$

Source Symbol	Probability
a_1	0.2
a_2	0.2
a_3	0.4
a_4	0.2

- Start with interval $[0, 1]$

0 ————— 1

- Subdivide $[0, 1]$ based on the probabilities of a_1

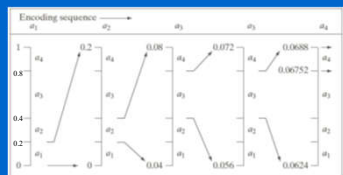


Initial Subinterval
$[0.0, 0.2)$
$[0.2, 0.4)$
$[0.4, 0.8)$
$[0.8, 1.0)$

- Update interval by processing source symbols

Example

Source Symbol	Probability	Initial Subinterval
a_1	0.2	$[0.0, 0.2)$
a_2	0.2	$[0.2, 0.4)$
a_3	0.4	$[0.4, 0.8)$
a_4	0.2	$[0.8, 1.0)$



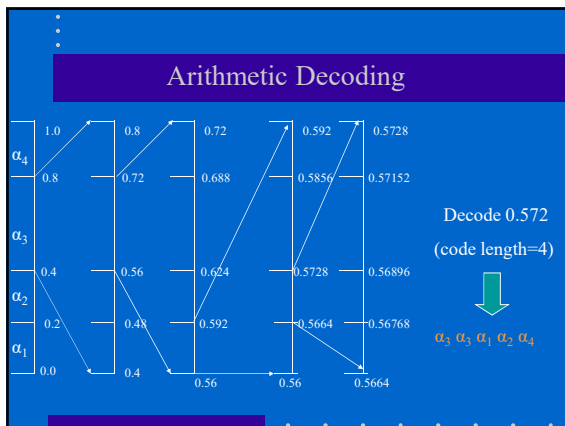
Encode
 $a_1 a_2 a_3 a_4$
↓
 $[0.06752, 0.0688)$
 or
 0.068
 (must be inside sub-interval)

Example (cont'd)

- The message $a_1 a_2 a_3 a_4$ is encoded using 3 decimal digits **or** $3/5 = 0.6$ decimal digits per source symbol.
- The entropy of this message is: $H = -\sum_{k=0}^3 P(r_k) \log(P(r_k))$

$$-(3 \times 0.2 \log_{10}(0.2) + 0.4 \log_{10}(0.4)) = 0.5786 \text{ digits/symbol}$$

Note: finite precision arithmetic might cause problems due to truncations!



LZW Coding (addresses interpixel redundancy)

- Requires no prior knowledge of symbol probabilities.
- Assigns **fixed length** code words to **variable length** symbol sequences.
 - There is **no** one-to-one correspondence between source symbols and code words.
- Included in GIF, TIFF and PDF file formats

LZW Coding

- A **codebook** (or **dictionary**) needs to be constructed.
- Initially, the first 256 entries of the dictionary are assigned to the gray levels 0,1,2,...,255 (i.e., assuming 8 bits/pixel)

Consider a 4x4, 8 bit image

Dictionary Location	Entry
0	0
1	1
...	...
255	255
256	-
...	...
511	-

LZW Coding (cont'd)

As the encoder examines image pixels, gray level sequences (i.e., blocks) that are not in the dictionary are assigned to a new entry.

Dictionary Location	Entry
0	0
1	1
...	...
255	255
256	39-39
...	...
511	-

39 39 126 126
 39 39 126 126
 39 39 126 126
 39 39 126 126

- Is 39 in the dictionary.....Yes
 - What about 39-39.....No
 * Add 39-39 at location 256

Example

39 39 126 126
39 39 126 126
39 39 126 126
39 39 126 126

CR = empty
repeat
P=next pixel
CS=CR + P

If CS is found:
(1) No Output
(2) CR=CS

else:
(1) Output D(CR)
(2) Add CS to D
(3) CR=P

Concatenated Sequence: CS = CR + P

Currently Recognized Sequence (CR)	Pixel Being Processed (P)	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39-39	126	256	260	39-39-126
126-126	39	258	261	126-126-39
39	39	39	262	39-39-126-126
39-39-126	126	259	263	126-39-39
126-39	39	257	264	39-126-126
39-126	126	126		

Decoding LZW

- Use the dictionary for decoding the “encoded output” sequence.
- The dictionary need not be sent with the encoded output.
- Can be built on the “fly” by the decoder as it reads the received code words.

Run-length coding (RLC) (addresses interpixel redundancy)

- Reduce the size of a repeating string of symbols (i.e., runs):

1 1 1 1 1 0 0 0 0 0 1 \rightarrow (1,5) (0, 6) (1, 1)
a a a b b b b b c c \rightarrow (a,3) (b, 6) (c, 2)

- Encodes a run of symbols into two bytes: (symbol, count)
- Can compress any type of data but cannot achieve high compression ratios compared to other compression methods.

Combining Huffman Coding with Run-length Coding

- Assuming that a message has been encoded using Huffman coding, additional compression can be achieved using run-length coding.

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

e.g., (0,1)(1,1)(0,1)(1,0)(0,2)(1,4)(0,2)

Bit-plane coding (addresses interpixel redundancy)

- Process each bit plane individually.

- Decompose an image into a series of binary images.
- Compress each binary image (e.g., using run-length coding)

Lossy Methods - Taxonomy

Lossy Compression

- Transform the image into some other domain to reduce interpixel redundancy.

$\sim (N/n)^2$ subimages

Example: Fourier Transform

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{\frac{j2\pi(ux+vy)}{N}}, \quad x, y=0, 1, \dots, N-1$$

Note that the magnitude of the FT decreases, as u, v increase!

$K \ll N$

$$\hat{f}(x, y) = \frac{1}{N} \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} F(u, v) e^{\frac{j2\pi(ux+vy)}{N}}, \quad x, y=0, 1, \dots, N-1$$

$\sum_{x,y} (\hat{f}(x, y) - f(x, y))^2$ is very small !!

Transform Selection

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

- $T(u, v)$ can be computed using various transformations, for example:
 - DFT
 - DCT (Discrete Cosine Transform)
 - KLT (Karhunen-Loeve Transformation) or Principal Component Analysis (PCA)
- JPEG using DCT for handling interpixel redundancy.

DCT (Discrete Cosine Transform)

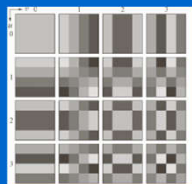
Forward: $C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right),$
 $u, v=0, 1, \dots, N-1$

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

$$\alpha(u) = \begin{cases} \sqrt{1/N} & \text{if } u=0 \\ \sqrt{2/N} & \text{if } u>0 \end{cases} \quad \alpha(v) = \begin{cases} \sqrt{1/N} & \text{if } v=0 \\ \sqrt{2/N} & \text{if } v>0 \end{cases}$$

DCT (cont'd)

- Basis functions for a 4x4 image (i.e., cosines of different frequencies).



DCT (cont'd)

Using 8 x 8 sub-images yields 64 coefficients per sub-image.

Reconstructed images by truncating 50% of the coefficients

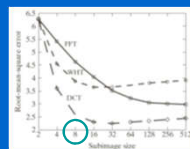
More compact transformation



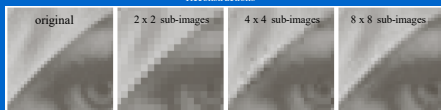
RMS error: 2.32 1.78 1.13

DCT (cont'd)

- Sub-image size selection:



Reconstructions

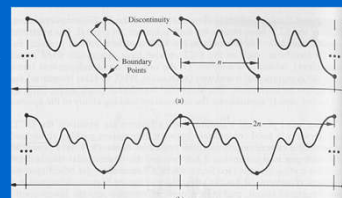


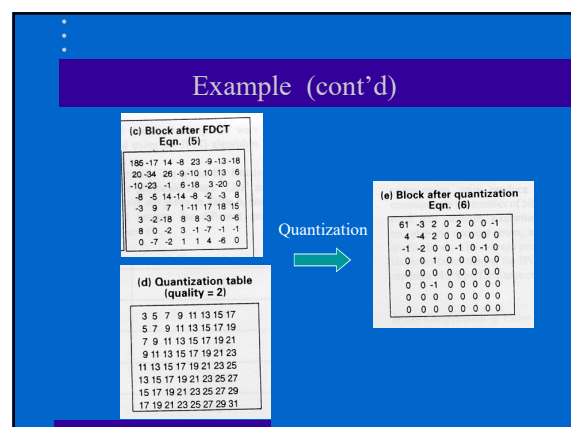
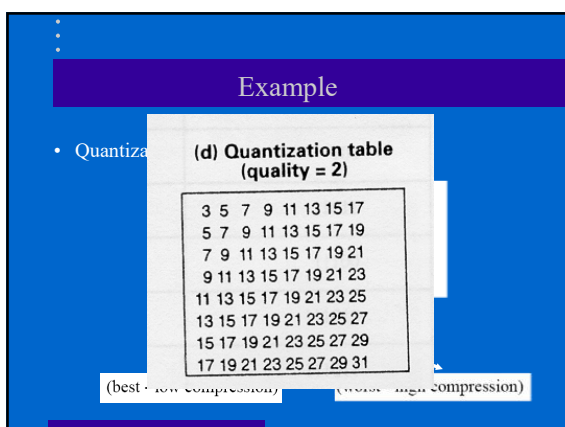
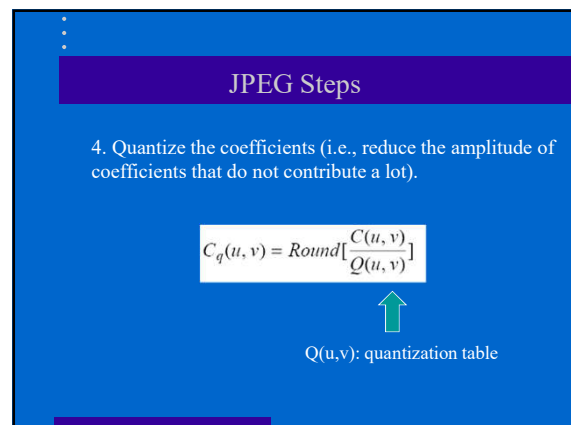
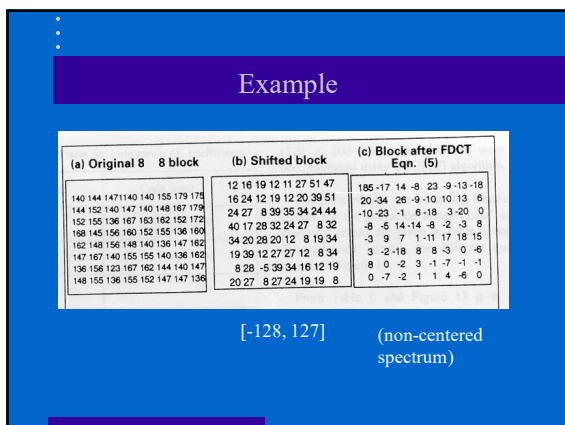
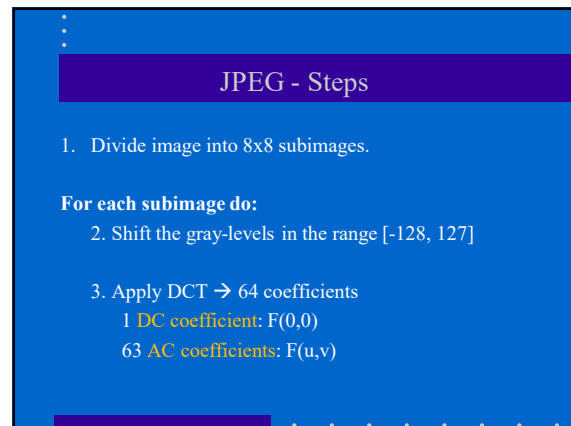
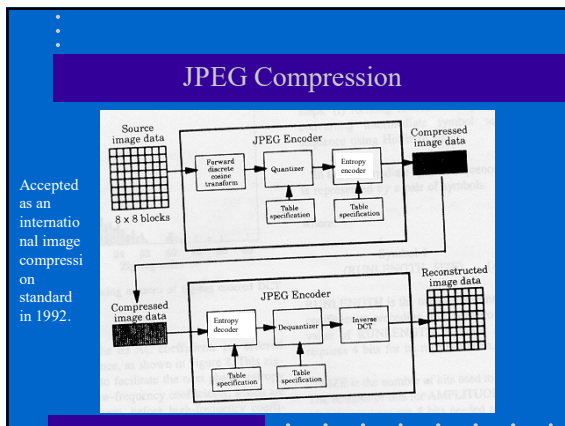
DCT (cont'd)

- DCT minimizes "blocking artifacts" (i.e., boundaries between subimages do not become very visible).

DFT has n -point periodicity

DCT has $2n$ -point periodicity





Final Symbol Sequence

(g) Intermediate symbol sequence
 (6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
 (5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

↓

(e) Encoded bit sequence (total 98 bits)
 111011110100100100000100011011011011100101111111011
 1101110101111011011100011101101111101001010

What is the effect of the “Quality” parameter?

(58k bytes) (21k bytes) (8k bytes)

lower compression higher compression

$1 \leq \text{quality} \leq 25$

What is the effect of the “Quality” parameter? (cont'd)

Table 6. Results of JPEG Compression for Grayscale Image ‘Lena’ (320 × 240 pixels)

Quality factors	Original number of bits	Compressed number of bits	Compression ratio (Cr)	Bits/pixel (Nb)	RMS error
1	512,000	48,021	10.66	0.75	2.25
2	512,000	30,490	16.79	0.48	2.75
4	512,000	20,264	25.27	0.32	3.43
8	512,000	14,162	36.14	0.22	4.24
15	512,000	10,479	48.85	0.16	5.36
25	512,000	9,034	56.64	0.14	6.40

Effect of Quantization: homogeneous 8 x 8 block

An 8 × 8 block from the Y image of ‘Lena’

200 202 189 188 189 175 175 175	515 65 -12 4 1 2 -8 5
200 203 198 188 189 182 178 175	-16 3 2 0 0 -11 -2 3
203 200 200 195 200 187 185 175	-12 6 11 -1 3 0 1 -2
200 200 200 200 197 187 187 187	-8 3 -4 -2 -2 -3 -5 -2
200 205 200 200 195 188 187 175	0 -2 7 -5 4 0 -1 -4
200 200 200 200 200 190 187 175	0 -3 -1 0 4 1 -1 0
205 200 199 200 191 187 187 175	3 -2 -3 3 3 -1 -1 3
210 200 200 200 188 185 187 186	-2 5 -2 4 -2 2 -3 0

$f(i, j)$ $F(u, v)$

Fig. 9.2: JPEG compression for a smooth image block.

Effect of Quantization: homogeneous 8 x 8 block (cont'd)

Quantized	De-quantized
32 6 -1 0 0 0 0 0	512 66 -10 0 0 0 0 0
-1 0 0 0 0 0 0 0	-12 0 0 0 0 0 0 0
-1 0 1 0 0 0 0 0	-14 0 16 0 0 0 0 0
-1 0 0 0 0 0 0 0	-14 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
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

$\hat{F}(u, v)$ $\hat{F}(u, v)$

Effect of Quantization: homogeneous 8 x 8 block (cont'd)

Reconstructed

199 196 191 186 182 178 177 176	199 196 191 186 182 178 177 176
201 199 196 192 188 183 180 178	201 199 196 192 188 183 180 178
203 203 202 200 195 189 183 180	203 203 202 200 195 189 183 180
202 203 204 203 198 191 183 179	202 203 204 203 198 191 183 179
200 201 202 201 196 189 182 177	200 201 202 201 196 189 182 177
200 200 199 197 192 186 181 177	200 200 199 197 192 186 181 177
204 202 199 195 190 186 183 181	204 202 199 195 190 186 183 181
207 204 200 194 190 187 185 184	207 204 200 194 190 187 185 184


Original

200 202 189 188 189 175 175 175	200 202 189 188 189 175 175 175
200 203 198 188 189 182 178 175	200 203 198 188 189 182 178 175
203 200 200 195 200 187 185 175	203 200 200 195 200 187 185 175
200 200 200 200 197 187 187 187	200 200 200 200 197 187 187 187
200 205 200 200 195 188 187 175	200 205 200 200 195 188 187 175
200 200 200 200 200 190 187 175	200 200 200 200 200 190 187 175
205 200 199 200 191 187 187 175	205 200 199 200 191 187 187 175
210 200 200 200 188 185 187 186	210 200 200 200 188 185 187 186

Error is low!

1 6 -2 2 7 -3 -2 -1
-1 4 2 -4 1 -1 -2 -3
0 -3 -2 -5 5 -2 2 -5
-2 -3 -4 -3 -1 -4 4 8
0 4 -2 -1 -1 -1 5 -2
0 0 1 3 8 4 6 -2
1 -2 0 5 1 1 4 -6
3 -4 0 6 -2 -2 2 2

Effect of Quantization: non-homogeneous 8 x 8 block



Another 8 x 8 block from the Y image of 'Lena'

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

$f(i,j)$

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

$F(u,v)$

Effect of Quantization: non-homogeneous 8 x 8 block (cont'd)

Quantized	De-quantized														
-5	-80	-44	90	-80	48	40	51	0							
-11	-5	-2	0	1	0	0	-1	-132	-60	-28	0	26	0	0	-55
3	-6	4	0	-3	-1	0	1	42	-78	64	0	-120	-57	0	56
0	1	-1	0	1	0	0	0	0	17	-22	0	51	0	0	0
0	0	-1	0	0	1	0	0	0	0	-37	0	0	109	0	0
0	-1	1	-1	0	0	0	0	0	-35	55	-64	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

$\hat{F}(u,v)$

Effect of Quantization: non-homogeneous 8 x 8 block (cont'd)

Reconstructed

70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

Original:


70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

Error is high!

0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

Case Study: Fingerprint Compression

- FBI is digitizing fingerprints at 500 dots per inch with 8 bits of grayscale resolution.
- A single fingerprint card turns into about 10 MB of data!




A sample fingerprint image
768 x 768 pixels = 589,824 bytes

WSQ Fingerprint Compression

- An image coding standard for digitized fingerprints employing the **Discrete Wavelet Transform** (*Wavelet/Scalar Quantization or WSQ*).
- Developed and maintained by:
 - FBI
 - Los Alamos National Lab (LANL)
 - National Institute for Standards and Technology (NIST)

Need to Preserve Fingerprint Details



The "white" spots in the middle of the black ridges are *sweat pores* and they are admissible points of identification in court.

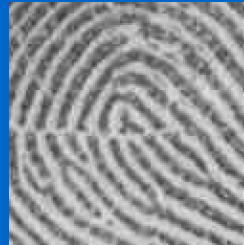
These details are just a couple pixels wide!

What compression scheme should be used?

- Lossless or lossy compression?
- In practice lossless compression methods haven't done better than 2:1 on fingerprints!
- Does JPEG work well for fingerprint compression?

Results using JPEG compression

file size 45853 bytes
compression ratio: 12.9



Fine details have been lost.
Image has an artificial "blocky" pattern superimposed on it.
Artifacts will affect the performance of fingerprint recognition.

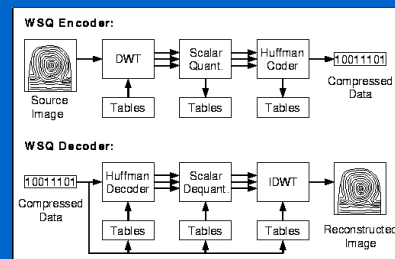
Results using WSQ compression

file size 45621 bytes
compression ratio: 12.9



Fine details are better preserved.
No "blocky" artifacts.

WSQ Algorithm



Compression ratio

- FBI's target bit rate is around 0.75 bits per pixel (bpp)
- This corresponds to a compression ratio of $8/0.75=10.7$
- Target bit rate can set via a parameter, similar to the "quality" parameter in JPEG.



Varying compression ratio (cont'd)

Original image 768 x 768 pixels (589824 bytes)





Varying compression ratio (cont'd)

0.9 bpp compression

<p>WSQ image, file size 47619 bytes, compression ratio 12.4</p> 	<p>JPEG image, file size 49658 bytes, compression ratio 11.9</p> 
---	--



Varying compression ratio (cont'd)

0.75 bpp compression

<p>WSQ image, file size 39270 bytes, compression ratio 15.0</p> 	<p>JPEG image, file size 40780 bytes, compression ratio 14.5</p> 
--	--

Varying compression ratio (cont'd)

0.6 bpp compression



<p>WSQ image, file size 30987 bytes, compression ratio 19.0</p> 	<p>JPEG image, file size 30081 bytes, compression ratio 19.6</p> 
---	--

JPEG Modes

- JPEG supports several different modes
 - Sequential Mode
 - Progressive Mode
 - Hierarchical Mode
 - Lossless Mode
- The default mode is "sequential"
 - Image is encoded in a **single scan** (left-to-right, top-to-bottom).

Progressive JPEG

- Image is encoded in **multiple scans**, in order to produce a quick, rough decoded image when transmission time is long.

Sequential	
Progressive	

Progressive JPEG (cont'd)

- Each scan encodes a **subset** of DCT coefficients.
- We'll examine the following algorithms:
 - (1) Progressive spectral selection algorithm
 - (2) Progressive successive approximation algorithm
 - (3) Combined progressive algorithm

Progressive JPEG (cont'd)

(1) Progressive spectral selection algorithm

- Group DCT coefficients into several spectral bands
- Send low-frequency DCT coefficients first
- Send higher-frequency DCT coefficients next

Band 1: DC coefficient only
 Band 2: AC_1 and AC_2 coefficients
 Band 3: AC_3, AC_4, AC_5, AC_6 coefficients
 Band 4: AC_7, \dots, AC_{63} coefficients

Example

Table 8. Progressive spectral selection JPEG (Image "Cherub": 320 × 240 pixels → 512,000 bits)

Scan number	Bits transmitted	Compression ratio	Bits/pixel	RMS error
1	29,005	17.65	0.45	19.97
2	37,237	7.73	1.04	13.67
3	71,259	3.72	2.15	7.90
4	32,489	3.01	2.68	4.59
Sequential JPEG	172,117	2.97	2.69	4.59

Progressive JPEG (cont'd)

(2) Progressive successive approximation algorithm

- Send all DCT coefficients but with lower precision.
- Refine DCT coefficients in later scans.

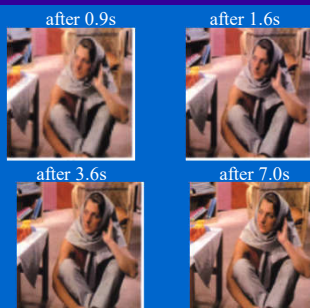
Band 1: All DCT coefficients (divided by four)
 Band 2: All DCT coefficients (divided by two)
 Band 3: All DCT coefficients (full resolution)

Example

Table 9. Progressive successive approximation JPEG (Image "Cherub": 320 × 240 pixels → 512,000 bits)

Scan number	Bits transmitted	Compression ratio	Bits/pixel	RMS error
1	28,215	19.53	0.41	22.48
2	34,506	8.43	0.95	12.75
3	63,792	4.11	1.95	7.56
4	95,267	2.33	2.43	4.59
Sequential JPEG	172,117	2.97	2.69	4.59

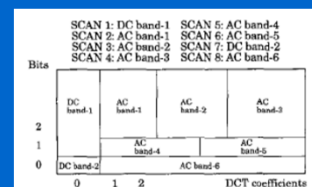
Example



Progressive JPEG (cont'd)

(3) Combined progressive algorithm

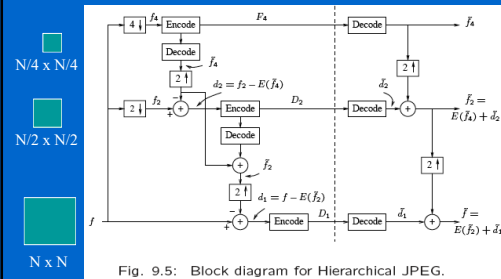
- Combines spectral selection and successive approximation.



Hierarchical JPEG

- Hierarchical mode encodes the image at different resolutions.
- Image is transmitted in multiple passes with increased resolution at each pass.

Hierarchical JPEG (cont'd)



More Methods ...

- See "Image Compression Techniques", IEEE Potentials, February/March 2001