ELEG 467/667 - Imaging and Audio Signal Processing

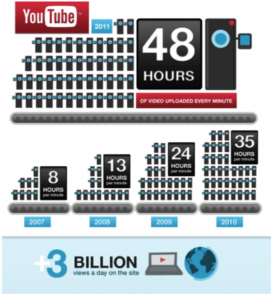
Gonzalo R. Arce

Image Compression

Department of Electrical and Computer Engineering
University of Delaware
Newark, DE, 19716
Spring 2013

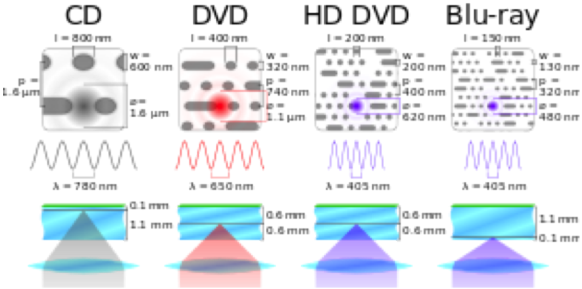
Motivation

- Over 50 hours of video content uploaded onto YouTube every minute!
- People are watching everything from online content to TV and movies online.
- Cisco predicts that 90 percent of all Internet traffic will be video in the near future.



The Challenge

Blue Ray Video Content has:



- 30 frames/sec
  1920 x 1080 pixels
  3 x 8 bits per pixel
- 1.5 Gigabits/sec
- LTE download rates (mobile) 100 Megabits/sec
- 15 Cell Phones needed

- Data compression involves encoding information using fewer bits than the original representation.
- Compression can be either lossy or lossless.
- Lossless compression
  - Identify and eliminate statistical redundancy. No information is lost (formal name is source coding)
  - Exploits statistical redundancy to represent data more concisely
  - An image may have areas of color that do not change locally; instead of coding “red pixel, red pixel, ...” it is encoded as “279 red pixels” (run-length encoding)
  - Many schemes to reduce file size by eliminating redundancy: Lempel-Ziv (LZ) method used in PKZIP, Gzip and PNG.

- Lossy data compression is the converse
  - Some loss of information. Human eye is more sensitive to subtle variations in luminance than to variations in color.
- JPEG image compression rounds off nonessential bits of information.
  - Trade-off between information lost and the size reduction.
- Lossy image compression used to increase storage capacities with minimal degradation of picture quality.
- DVDs use the lossy MPEG-2 Video codec for video compression.

(a) JPEG Q=100 Compression 2.6→1

(b) JPEG Q=50 Compression 15→1

(c) JPEG Q=10 Compression 46→1

Fundamentals

n_1 and n_2: the number of information-carrying units in two data sets that represent the same information.

R_D : Relative data redundancy of the first data set (n_1)

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

C_R : Compression ratio

$$C_R = \frac{n_1}{n_2}$$

$n_2 = n_1$ ➡ $C_R = 1, R_D = 0$

The first representation contains no redundant data.

$n_2 \ll n_1$ ➡ C_R ➔ ∞, R_D ➔ 1

significant compression and highly redundant data

$n_2 \gg n_1$ ➡ C_R ➔ $0, R_D$ ➔ $-\infty$

data expansion

Fundamentals

$C_R : (0, \infty)$
$R_D : (-\infty, 1)$

Compression ratio 10 (or 10:1) means that the fist data set has 10 bits for every 1 bit in the second or compressed data set.

The corresponding redundancy of 0.9 implies that 90% of the data in the first data set is redundant.

Three basic data redundancies

- *coding* redundancy
- *interpixel* redundancy
- *psychovisual* redundancy

Coding Redundancy

$$
\begin{aligned}
r_k &: \text{the gray levels of an image}[0,1] \\
p_r(r_k) &: \text{probability that each } r_k \text{ occurs} \\
L &: \text{the number of gray levels} \\
n_k &: \text{the number of times that the } kth \text{ gray level appears in the image} \\
n &: \text{the total number of pixels in the image} \\
l(r_k) &: \text{the number of bits used to represent each value of } r_k \\
L_{avg} &: \text{the average number of bits required to represent each pixel}
\end{aligned} \tag{1}
$$

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0, 1, 2, \ldots . L-1 \tag{2}$$

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k) \tag{3}$$

The total number of bits required to code an $M \times N$ image is

$$MNL_{avg}$$

Coding Redundancy

Natural m-bit binary code

$\rightarrow L_{avg} = m$

r_k	$p_r(r_k)$	**Code 1**	$l_1(r_k)$	**Code 2**	$l_2(r_k)$
$r_0 = 0$	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

TABLE 8.1 Example of variable-length coding.

Code 1: $L_{avg} = 3$ bits

Code 2:

$$
\begin{aligned}
L_{avg} &= \sum_{k=0}^{7} l_2(r_k)p_r(r_k) \\
&= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08) \\
&\quad + 5(0.06) + 6(0.03) + 6(0.02) \\
&= 2.7 \text{ bits}
\end{aligned}
$$

Coding Redundancy

$$C_R = 3/2.7 = 1.11$$

Approximately 10% of the data in code 1 is redundant.

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

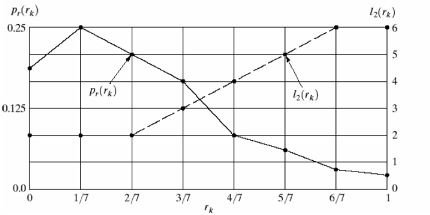


FIGURE 8.1 Graphic representation of the fundamental basis of data compression through variable-length coding.

The histogram of the image and $l_2(r_k)$.
These two functions are inversely proportional.
The shortest code words in code 2 are assigned to the gray levels that occur most frequently.

Coding Redundancy

Assigning fewer bits to the more probable gray levels than to the less probable ones achieves data compression.

$\rightarrow$ *variable-length coding*

Coding redundancy is present when the codes assigned to a set of events are not selected to take full advantage of the events probabilities.

Interpixel Redundancy

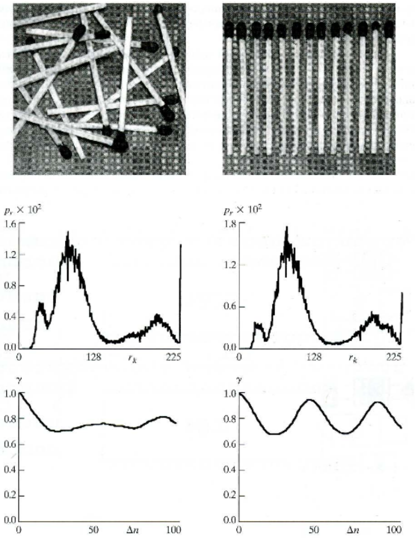


FIGURE 8.2 Two images and their gray-level histograms and normalized autocorrelation coefficients along one line.

Left and right images have identical histograms.

The codes representing the gray levels of each image have nothing to do with the correlation between pixels.

Correlations result from the geometric relationships between the objects in the image.

Interpixel Redundancy

- The value of any given pixel can be somewhat predicted from the value of its neighbors.
- Information carried by individual pixels is relatively small.
- Much of visual contribution of a single pixel to an image is redundant.

  spatial redundancy
  interframe redundancy
  interpixel redundancy

Interpixel Redundancy

- To reduce the interpixel redundancies, the image must be transformed into a more efficient format.
  Ex the differences between adjacent pixels can be used to represent an image.

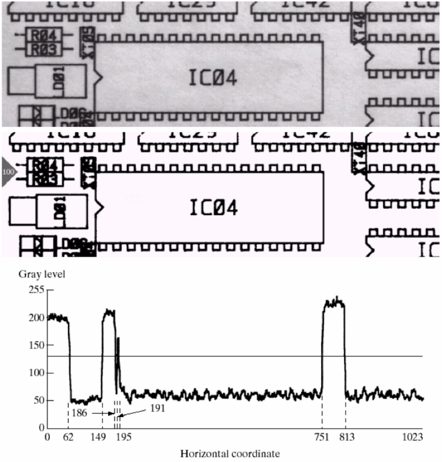
- Reversible mapping
  (the original image elements can be reconstructed)

(d) Run-length encoded data.

1 bit for the type (black or white)
10 bits for the length $(0 \sim 1023)$

Only 88 bits $(8 * (1 + 10))$ are needed to represent the 1024 bits of binary data.

Entire 1024x343 section is reduced to 12,166 runs.



Line 100: (1, 63) (0, 87) (1, 37) (0, 5) (1, 4) (0, 556) (1, 62) (0, 210)

a
b
c
d

FIGURE 8.3 Illustration of run-length coding: (a) original image. (b) Binary image with line 100 marked. (c) Line profile and binarization threshold. (d) Run-length code.

Interpixel Redundancy

As 11 bits are required to represent each run-length pair, the resulting compression ratio and corresponding relative redundancy are

$$C_R = \frac{(1024)(343)(1)}{(12166)(11)} = 2.63 \tag{4}$$

and

$$R_D = 1 - \frac{1}{2.63} = 0.62 \tag{5}$$

Psychovisual Redundancy

- The eye does not respond with equal sensitivity to all visual information.
- Certain information has less importance than other information in normal visual processing.
  $\rightarrow$ *psychovisually redundant*
- It can be eliminated without significantly impairing the quality of image perception.
- Elimination of psychovisually redundant data results in a loss of quantitative information, commonly done by quantization.

Coding Redundancy

Improved Gray Scale (IGS)

(a) 8-bit (256 levels)
(b) 4-bit (16 levels) - Contouring
(c) IGS quantization

a b c
FIGURE 8.4
(a) Original image.
(b) Uniform quantization to 16 levels. (c) IGS quantization to 16 levels.

Pixel	Gray level	Sum	IGS
$i-1$	N/A	00000000	N/A
i	01101100	01101100	0110
$i+1$	10001011	10010111	1001
$i+2$	10000111	10001110	1000
$i+3$	11110100	11110100	1111

Fidelity Criteria

Two classes of criteria:
(1) Objective fidelity criteria

Cost function can be expressed as a function of the original image and the compressed and subsequently decompressed output image, it is an objective fidelity criterion.

(2) Subjective fidelity criteria

$f(x, y)$: an input image
$\hat{f}(x, y)$: approximation of $f(x, y)$ resulting from compression and subsequently decompressing the input.
$e(x, y)$: the error between $f(x, y)$ and $\hat{f}(x, y)$.

Fidelity Criteria

$$e(x,y) = \hat{f}(x,y) - f(x,y) \tag{6}$$

Total error between the two images (size $M \times N$) is

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right] \tag{7}$$

The *root-mean-square error*, e_{rms} is

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2 \right]^{1/2} \tag{8}$$

The *mean-square signal-to-noise ratio* of the compressed-decompressed image,

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2} \tag{9}$$

Fidelity Criteria

- Most decompressed images ultimately are viewed by humans. $\rightarrow$ measuring image quality by the subjective evaluations of a human observer often is more appropriate.
- Evaluations made using an *absolute rating scale* or by means of *side-by-side comparisons*.

a b c

FIGURE 8.4 Three approximations of the image in Fig. 8.1(a).

Elements of Information Theory

Is there a minimum amount of data that is sufficient to describe completely an image without loss of information?

$\rightarrow$ Information theory

1. Homework due soon.

2. Midterm exam next class.

Measuring Information

An event E that occurs with probability $P(E)$ is said to contain

$$I(E) = \log_2 \frac{1}{P(E)} = -\log_2 P(E) \qquad \text{information bits.}$$

$I(E)$: *self-information* of E.

If $P(E) = 1 \quad \rightarrow I(E) = 0$ (no information) no uncertainty is associated with the event.

If $P(E) = 0.99 \quad \rightarrow$ some small amount of information.

If $P(E) = 1/2$, $I(E) = -\log_2 1/2$, or 1 bit.
$\rightarrow$ ex. Flipping a coin and communicating the result

The Information Channel

The average information per source output is

Shannon Entropy:

$$H(z) = -\sum_{j=1}^{L-1} P(a_j) \log P(a_j) \tag{10}$$

where a_j is gray level j, and L is the number of gray levels.

- Defines the average amount of information bits obtained per single source output.
- If magnitude increases
  $\rightarrow$ more uncertainty and thus more information
- If symbols are equally probable, the entropy is maximized and the source provides the greatest possible average information per source symbol.

Using Information Theory

A method of estimating the information content is to construct a relative frequency of occurrence of the gray levels.

Model the probabilities of the source using the gray-level histogram.

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

First-order estimate
entropy = 1.81 bits/pixel or 58 total bits

Using Information Theory

<u>Better estimation</u>: Examine the relative frequency of pixel blocks in the sample image.

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

Second order estimate

the resulting entropy estimate is 2.5/2, or 1.25 bits/pixel

As block size approaches infinity, the estimate approaches the source's true entropy.

Variable-Length Coding

- Used to reduce coding redundancy.
- A variable-length code assigns the shortest possible code words to the most probable gray levels.

  Huffman coding

- Yields the smallest possible number of bits per source symbol.
- The resulting code is optimal for a fixed value of n, subject to the constraint that the source symbols be coded one at a time.
- two steps :-
  - *source reduction*
  - *code assignment*

Variable-Length Coding

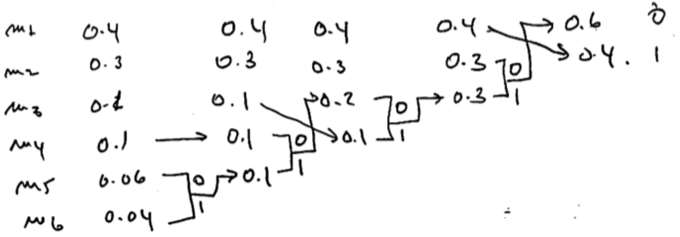
Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

FIGURE 8.7 Huffman source reductions.

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
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				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

FIGURE 8.8 Huffman code assignment procedure.

Variable-Length Coding



$m_1 \equiv 1$

$m_2 \equiv 00$

$m_3 \equiv 011$

$m_4 = 0100$

$m_5 = 01010$

$m_6 \quad 01011$

The Huffman code

- Yields the smallest possible number of unique code symbols per source symbol.
- Step 1.
  1. Sort the gray levels by decreasing probability.
  2. Add the two smallest probabilities.
  3. Sort the new value into the list.
  4. Repeat until only two probabilities remain.
- Step 2.
  1. Give the code 0 to the highest probability, and the code 1 to the lowest probability in the present node.
  2. Go backwards through the tree and add 0 to the highest and 1 to the lowest probability in each node until all gray levels have a unique code.

Variable-Length Coding

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

- The entropy of the source is 2.14 bits/symbol.
- The resulting Huffman code efficiency is 0.973.

Block code: each source symbol is mapped into a fixed sequence of bits.
Instantaneous: each code word in a string of code symbols can be decoded without referencing succeeding symbols.
Uniquely decodable: any string of code symbols can be decoded uniquely.

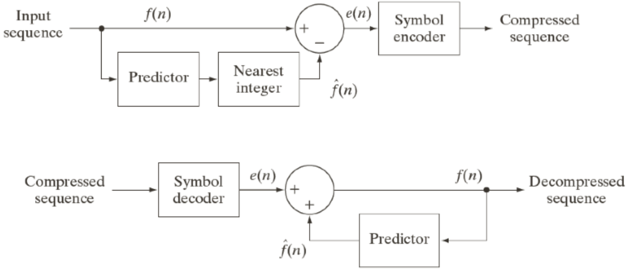
Ex. $\boxed{01010}\boxed{011}\boxed{1}\boxed{1}\boxed{00} \rightarrow$ 01010 011 1 1 00

$a_3\ a_1\ a_2\ a_2\ a_6$

LZW Coding

- Lempel-Ziv-Welch (LZW) coding assigns fixed length code words to variable length sequences of source symbols but requires no a priori knowledge of the probability of occurrence of the symbols to be encoded.
- LZW compression has been integrated into a various imaging file formats, including the *graphic interchange format* (GIF), *tagged image file format* (TIFF), and the *portable document format* (PDF).

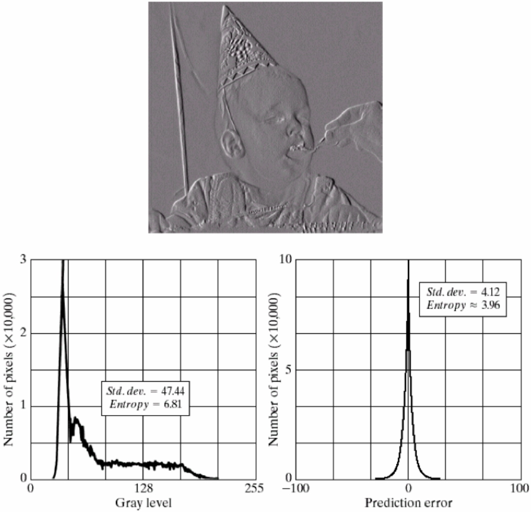
Lossless Predictive Coding



a
b

FIGURE 8.33
A lossless predictive coding model:
(a) encoder;
(b) decoder.

Lossless Predictive Coding



a
b c

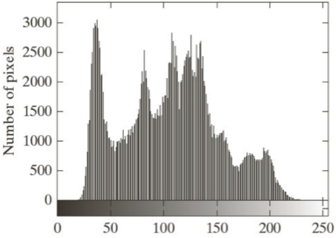
FIGURE 8.20 (a) The prediction error image resulting from Eq. (8.4-9). (b) Gray-level histogram of the original image. (c) Histogram of the prediction error.

Lossy Compression

- Lossy encoding is based on the concept of compromising the accuracy of the reconstructed image in exchange for increased compression.
- If the resulting distortion (which may or may not be visually apparent) can be tolerated, the increase in compression can be significant.

$$10{:}1 \text{ to } 50{:}1 \rightarrow \text{more than } 100{:}1$$

Lossy Predictive Coding



a b

FIGURE 8.9 (a) A 512 × 512 8-bit image, and (b) its histogram.

Predictors

$$\hat{f}(x,y) = 0.97f(x,y-1) \tag{11}$$

$$\hat{f}(x,y) = 0.5f(x,y-1) + 0.5f(x-1,y) \tag{12}$$

$$\hat{f}(x,y) = 0.75f(x,y-1) + 0.75f(x-1,y) - 0.5f(x-1,y-1) \tag{13}$$

$$\hat{f}(x,y) = \begin{cases} 0.97f(x,y-1) & \text{if } \triangle h \leq \triangle v \\ 0.97f(x-1,y) & \text{otherwise} \end{cases} \tag{14}$$

Lossy Predictive Coding

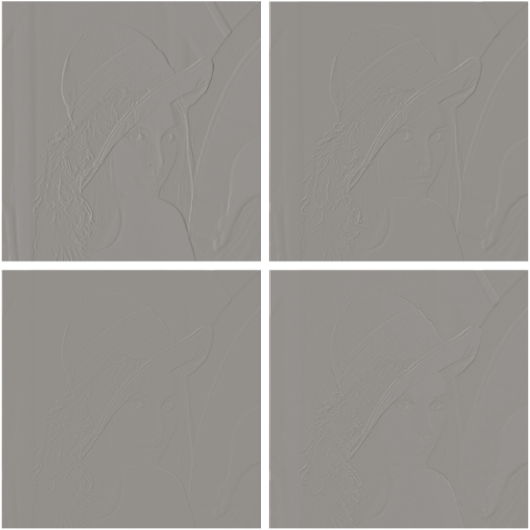


FIGURE 8.43 A comparison of four linear prediction techniques.

Lossy Predictive Coding

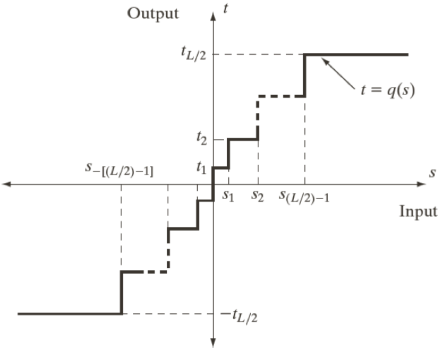


FIGURE 8.44 A typical quantization function.

Levels	2		4		8	
i	s_i	t_i	s_i	t_i	s_i	t_i
1	∞	0.707	1.102	0.395	0.504	0.222
2			∞	1.810	1.181	0.785
3					2.285	1.576
4					∞	2.994
θ	1.414		1.087		0.731	

TABLE 8.12 Lloyd-Max quantizers for a Laplacian probability density function of unit variance.

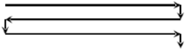
Difference coding

$$f(X_i) = \begin{cases} X_i & \text{if } i = 0, \\ X_i - X_{i-1} & \text{if } i > 0. \end{cases} \quad (15)$$

- E.g.,

Original	56	56	56	82	82	82	83	80	80	80	80
Code $f(X_i)$	56	0	0	26	0	0	1	-3	0	0	0

- The code is calculated row by row.



- Both run-length coding, and difference coding are reversible, and can be combined with, e.g., Huffman coding.

Example of combined difference and Huffman coding

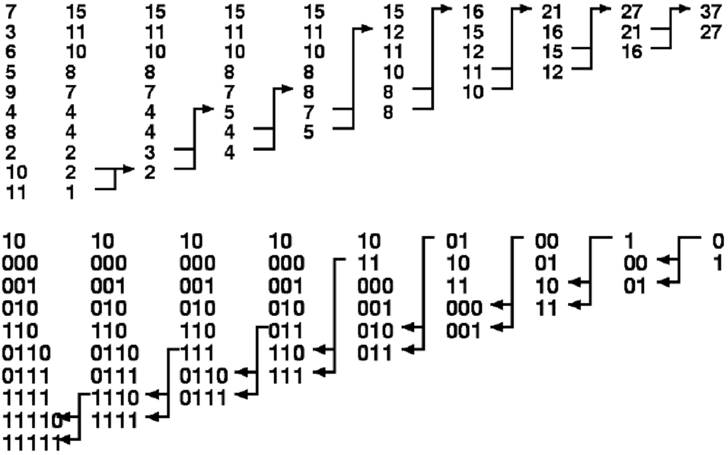
Original image.

9	8	7	7	7	5	5	5
7	7	7	7	4	4	5	5
6	6	6	9	9	9	6	6
6	6	7	7	7	9	9	9
3	7	7	8	8	8	3	3
3	3	3	3	3	3	3	3
10	10	11	7	7	7	6	6
4	4	5	5	5	2	2	6

Difference image.

9	-1	-1	0	0	-2	0	0
0	0	0	3	0	-1	0	0
-1	0	0	3	0	0	-3	0
0	-1	0	0	-2	0	0	3
-3	4	0	1	0	0	-5	0
0	0	0	0	0	0	0	0
7	0	1	-4	0	0	-1	0
0	-1	0	0	3	0	-4	0

Huffman code of original image



$L_{avg} = 3.1$

Huffman code of Difference image

0	42	42	42	42	42	42	42	42	42
-1	7	7	7	7	7	7	8	14	22
3	4	4	4	4	4	7	7	8	
-2	2	2	3	4	4	4	7		
1	2	2	2	3	4	4			
-3	2	2	2	2	3				
-4	2	2	2	2					
-5	1	2	2						
7	1	1							
9	1								

0	0	0	0	0	0	0	0	0
100	100	100	100	100	100	11	10	1
110	110	110	110	110	101	100	11	
10100	10100	1011	111	111	110	101		
10101	10101	10100	1011	1010	111			
1110	1110	10101	10100	1011				
1111	1111	1110	10101					
10111	10110	1111						
101100	10111							
101101								

$L_{avg} = 2$

Transform Coding

- Predictive coding techniques operate directly on image pixels and thus are spatial domain methods.
- Transform coding uses linear transforms (such as Fourier transform) to map the image into a set of transform coefficients, which are then quantized and coded.
- A significant number of coefficients have small magnitudes and can be coarsely quantized (or discarded entirely) with little image distortion.

- Unitary transform packs as much information as possible into the smallest number of transform coefficients.
- The quantization stage eliminates coefficients that carry the least information.
- The encoding process uses a variable length code to quantize coefficients.

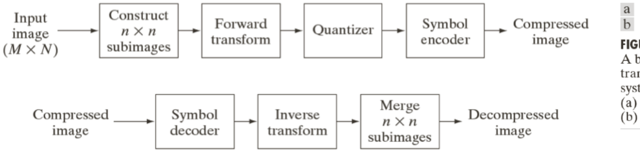


FIGURE 8.21 A block transform coding system: (a) encoder; (b) decoder.

Transform selection

Walsh-Hadamard transform (WHT)

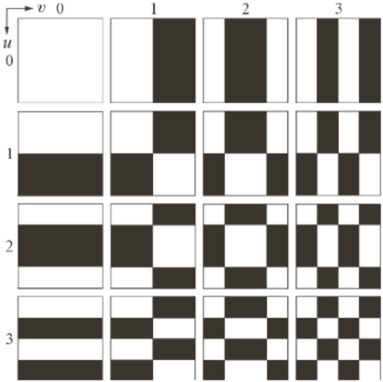


FIGURE 8.22 Walsh-Hadamard basis functions for $n = 4$. The origin of each block is at its top left.

Transform selection
Discrete cosine transform (DCT)

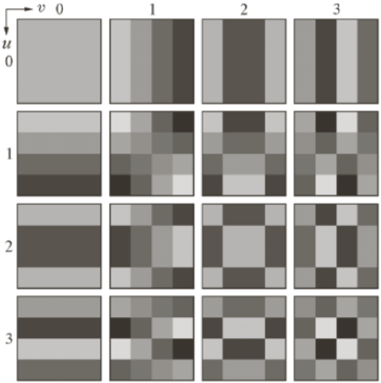


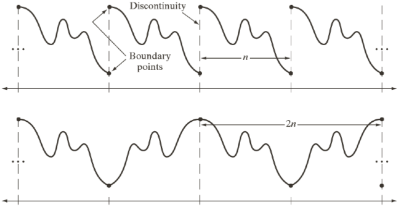
FIGURE 8.23 Discrete-cosine basis functions for $n = 4$. The origin of each block is at its top left.

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).

Three approximations of the 512 x 512 image:

1. Divide the original image into subimages of size 8 x 8,
2. Transforms
3. truncate 50% of the resulting coefficients (minimum magnitude).
4. inverse transform

- The information packing of DCT is superior to that of the DFT and WHT.
- The *Karhunen-Loeve transform* (KLT) is the optimal transform.
  $\rightarrow$ the KLT minimizes the mean-square error for any input image and any number of retained coefficients.
- However, because the KLT is data dependent
  $\rightarrow$ the KLT is seldom used in practice for image compression.



a
b

FIGURE 8.25 The periodicity implicit in the 1-D (a) DFT and (b) DCT.

- The *DCT* provides a good compromise between information packing ability and computational complexity.
- The DCT has become an international standard for transform coding.
- The DCT has the advantages of having been implemented in a single integrated circuit, packing the most information into the fewest coefficients, and minimizing the blocklike appearance, called *blocking artifact*, that results when the boundaries between subimages become visible.

Subimage size selection

- Another significant factor affecting transform coding error is subimage size.
- The level of compression and computational complexity increase as the subimage size increases.
- The most popular subimage sizes are 8×8 and 16×16.

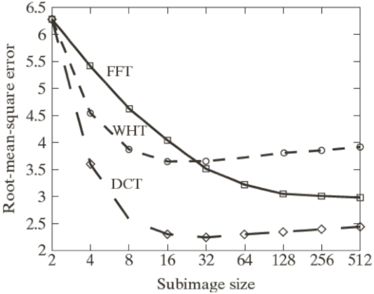
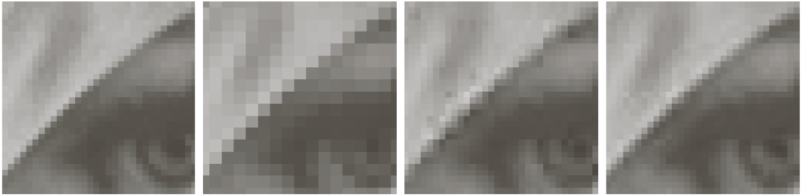


FIGURE 8.26 Reconstruction error versus subimage size.



a b c d

FIGURE 8.27 Approximations of Fig. 8.27(a) using 25% of the DCT coefficients and (b) 2×2 subimages, (c) 4×4 subimages, and (d) 8×8 subimages. The original image in (a) is a zoomed section of Fig. 8.9(a).

Bit allocation

The overall process of truncating, quantizing, and coding the coefficients of a transformed subimage is commonly called *bit allocation*.

- Zonal coding implementation the retained coefficients are selected on the basis of maximum variance.
- Threshold coding implementation the retained coefficients are selected on the basis of maximum magnitude.

a b
c d

FIGURE 8.28 Approximations of Fig. 8.9(a) using 12.5% of the 8×8 DCT coefficients: (a)—(b) threshold coding results; (c)—(d) zonal coding results. The difference images are scaled by 4.

The threshold coding difference image of Fig.8.28(b) contains far less error than the zonal coding difference image of Fig.8.28(d).

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

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

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

a b
c d

FIGURE 8.29 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.

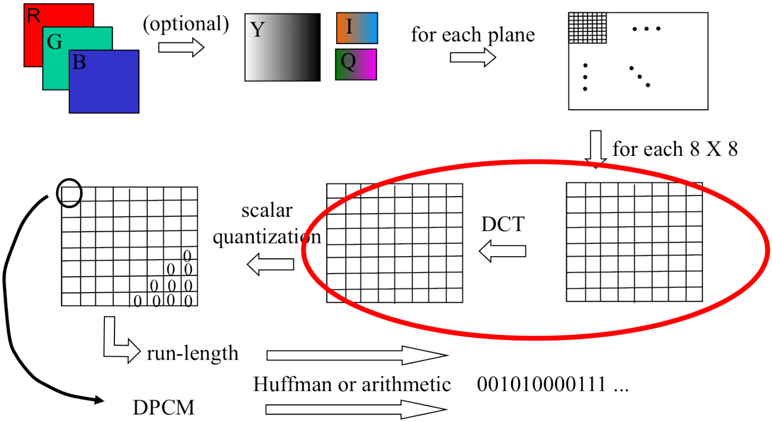
Zonal coding

- A single *fixed* mask for all subimages
- Coefficients of maximum variance are located around the origin.

Threshold coding

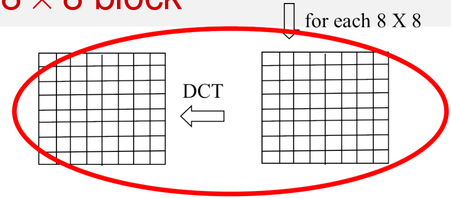
- Inherently adaptive where the location of the transform coefficients retained for each subimage vary from one subimage to another.

JPEG: Joint Photographic Experts Group



JPEG: Image partition into 8×8 block

JPEG: Image partition into 8×8 block



Discrete Cosine Transform

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

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

Inverse Discrete Cosine Transform

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

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

S

8 X 8

$$\bullet \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \end{bmatrix}$$

transform EACH row
with 1-D DCT basis matrix

8 X 8

$$\bullet \begin{bmatrix} .1913 & .2778 & .3536 & .4157 & .4619 & .4904 & .3536 \\ -.4619 & -.4904 & -.3536 & -.0975 & .1913 & .4157 & .3536 \\ .4619 & .0975 & -.3536 & -.4904 & -.1913 & .2778 & .3536 \\ -.1913 & .4157 & .3536 & -.2778 & -.4619 & .0975 & .3536 \\ -.1913 & -.4157 & .3536 & .2778 & -.4619 & -.0975 & .3536 \\ .4619 & -.0975 & -.3536 & .4904 & -.1913 & -.2778 & .3536 \\ -.4619 & .4904 & -.3536 & .0975 & .1913 & -.4157 & .3536 \\ .1913 & -.2778 & .3536 & -.4157 & .4619 & -.4904 & .3536 \end{bmatrix}$$

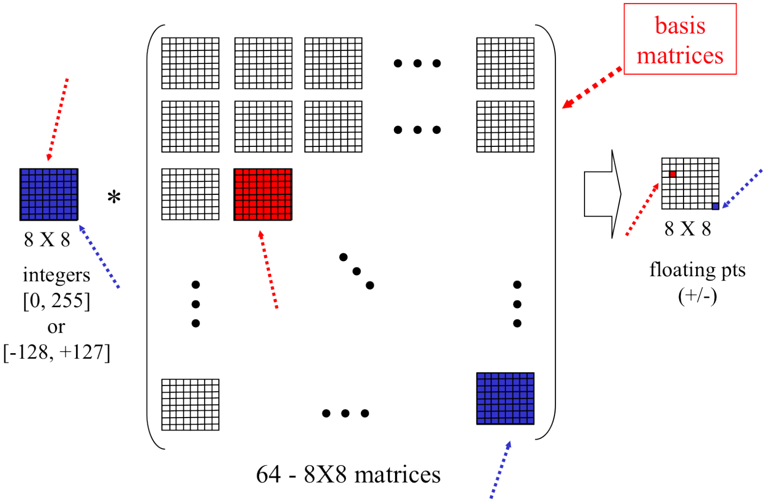
transform EACH column
with rotated 1-D DCT
basis matrix

2-D DCT transform one
dimension at a time

S

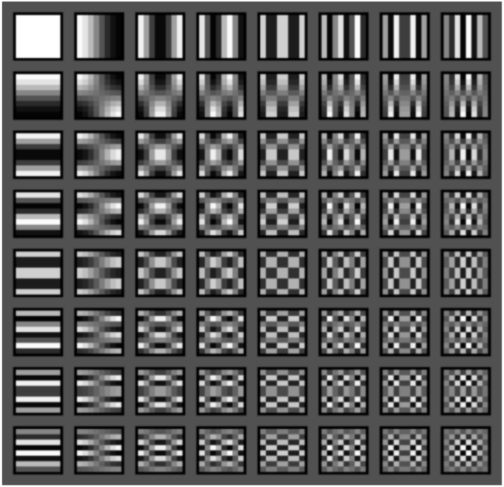
8 X 8

2-D Discrete Cosine Transform



DCT basis matrices

white is + value
black is - value



JPEG example - coding

original 8X8 pixels

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

DCT

coefficients

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

$\div$

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

scalar

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

lossless encoding (21 bits !)

channel

JPEG example - decoding

channel → lossless decoding (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

*

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

scalar

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

reconstructed coefficients

inverse DCT

reconstructed 8X8 pixels

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

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

DCT

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

scalar quantization

compression

decompression

inverse quantization

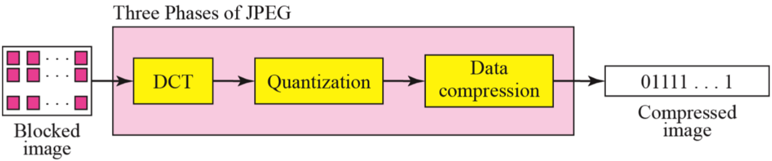
32	11	0	0	0	0	0	0
-108	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

inverse DCT

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

reconstructed

JPEG process



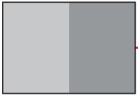
Picture

20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20

P(x,y)

160	**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**

T(m,n)



Picture

20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**
20	20	20	20	**50**	**50**	**50**	**50**

P(x,y)

280	**2109**	**0**	**39**	**0**	**225**	**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**	**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**

T(m,n)

Picture

20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90

P(x,y)

400	**2146**	**0**	**231**	**21**	**3**	**21**	**28**
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**

T(m,n)

JPEG Details - quantization

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

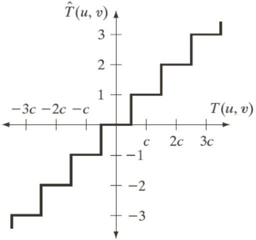
Luminance Quantization Table

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



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

a b

FIGURE 8.30 (a) A threshold coding quantization curve [see Eq. (8.2-29)]. (b) A typical normalization matrix.

- Typical normalization array, which is used in JPEG.
- Array weighs each coefficient of a transformed subimage according to heuristically determined perceptual or psychovisual importance.

FIGURE 8.31 Approximations of Fig. 8.9(a) using the DCT and normalization array of Fig. 8.30(b): (a) **Z**, (b) 2**Z**, (c) 4**Z**, (d) 8**Z**, (e) 16**Z**, and (f) 32**Z**.

- Compression ratio

12:1	19:1	30:1
49:1	85:1	182:1

JPEG

Ex. 8×8 subimage with the JPEG baseline standard

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

EXAMPLE 8.17: JPEG baseline coding and decoding.

JPEG

Ex. 256 or 2^8 gray levels, $\rightarrow$ level shifting by 128 or -2^7 gray levels

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

JPEG

Ex. Transformed in accordance with the forward DCT for $N = 8$, becomes

$$\begin{array}{rrrrrrrr}
-415 & -29 & -62 & 25 & 55 & -20 & -1 & 3 \\
7 & -21 & -62 & 9 & 11 & -7 & -6 & 6 \\
-46 & 8 & 77 & -25 & -30 & 10 & 7 & -5 \\
-50 & 13 & 35 & -15 & -9 & 6 & 0 & 3 \\
11 & -8 & -13 & -2 & -1 & 1 & -4 & 1 \\
-10 & 1 & 3 & -3 & -1 & 0 & 2 & -1 \\
-4 & -1 & 2 & -1 & 2 & -3 & 1 & -2 \\
-1 & -1 & -1 & -2 & -1 & -1 & 0 & -1
\end{array}$$

JPEG

JPEG used the normalization array to quantize the transformed array. The scaled and truncated coefficients are

$$
\begin{array}{cccccccc}
-26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\
1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

$$
\begin{aligned}
\hat{T}(0,0) &= round\left[\frac{T(0,0)}{Z(0,0)}\right] \\
&= round\left[\frac{-415}{16}\right] \\
&= -26
\end{aligned}
\tag{16}
$$

To decompress the JPEG compressed subimage,

$$
\begin{array}{rrrrrrrr}
-26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\
1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Denormalization,

$$\begin{array}{rrrrrrrr}
-416 & -33 & -60 & 32 & 48 & 0 & 0 & 0 \\
12 & -24 & -56 & 0 & 0 & 0 & 0 & 0 \\
-42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\
-56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

$$\begin{aligned}
\overline{T}(0,0) &= \hat{T}(0,0)Z(0,0) \\
&= (-26)(16) \\
&= -416
\end{aligned} \tag{17}$$

Inverse DCT,

$$\begin{array}{cccccccc}
-70 & -64 & -61 & -64 & -69 & -66 & -58 & -50 \\
-72 & -73 & -61 & -39 & -30 & -40 & -54 & -59 \\
-68 & -78 & -58 & -9 & 13 & -12 & -48 & -64 \\
-59 & -77 & -57 & 0 & 22 & -13 & -51 & -60 \\
-54 & -75 & -64 & -23 & -13 & -44 & -63 & -56 \\
-52 & -71 & -72 & -54 & -54 & -71 & -71 & -54 \\
-45 & -59 & -70 & -68 & -67 & -67 & -61 & -50 \\
-35 & -47 & -61 & -66 & -60 & -48 & -44 & -44
\end{array}$$

Level shifting each pixel by $+2^7$ (or $+128$),

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

- The errors (the differences between the original and reconstructed subimage) range from -14 to $+11$.

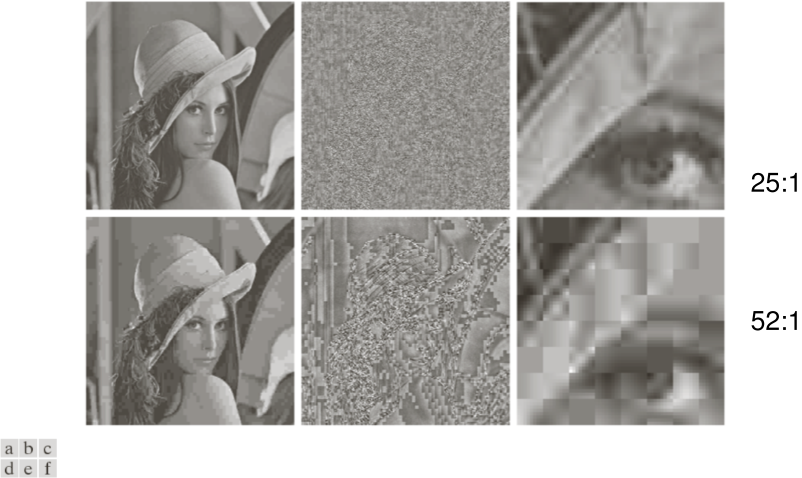
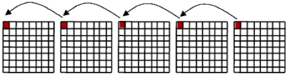
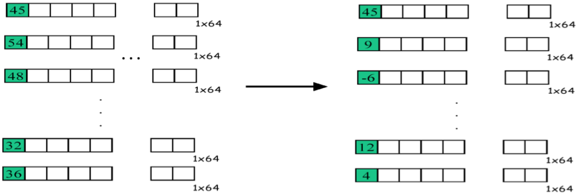


FIGURE 8.32 Two JPEG approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image.

JPEG Details: Entropy Encoding of DC Components

- Model: For photographs, DC value in each 8×8 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

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

Figure : Size-Value Encoding Table

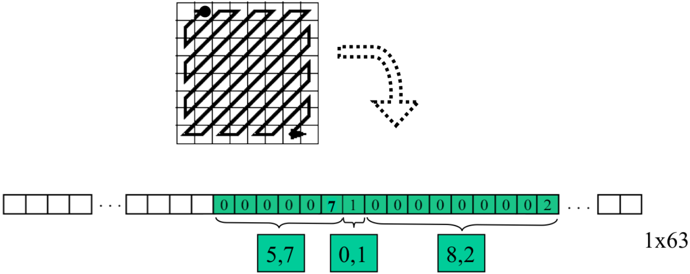
Example: If a DC component is 40, and the previous DC component is 48. The difference is -8. Therefore 40 gets coded as: 1010111

0111: value representing -8

101: size from the same table reads 4

JPEG Details - Entropy Encoding of AC Components

- Model: after quantization, AC components for photographs have lots of zeros, particularly in lower right triangle.
- Coding scheme:
  - use Zig-Zag Scan - group non-zero low frequency coefficients
  - use Run Length Encoding (RLE) - (run, value) pairs



(0,0) is end-of-block value

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

⇒ 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 Table13.10)

1100: code for 12 from Size-Value table (textbook Table13.9)

0-0s, 10: (0/4) 10 ➔ 10111010

1011: code for 0/4 from AC code table

1010: 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

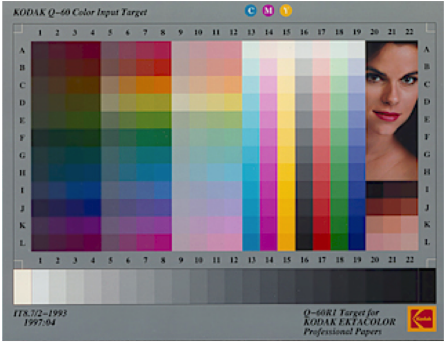
56-0s: (0,0) ➔ 1010 (special code for all 0’s until EOB)

JPEG default AC code for luminance

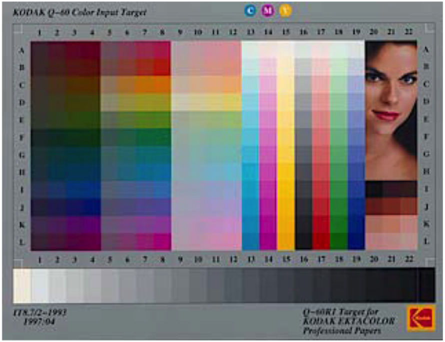
Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110010010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26

JPEG compression results

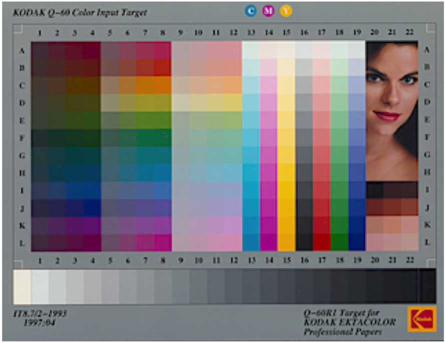


(a) 231KB original 320 X 240 X 24bit

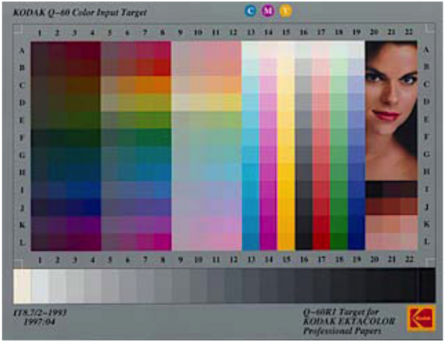


(b) 74KB 3.24 : 1 compression

JPEG compression results

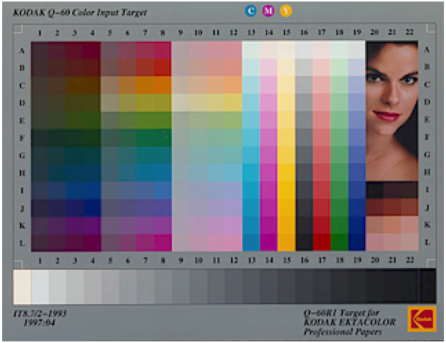


(c) 231KB original 320 X 240 X 24bit



(d) 38KB 6.08 : 1 compression

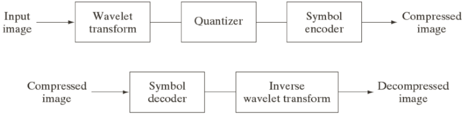
JPEG compression results



(e) 231KB original 320 X 240 X 24bit

(f) 11KB 21 : 1 compression

Wavelet Coding



a
b

FIGURE 8.45 A wavelet coding system: (a) encoder; (b) decoder.

- Difference between the wavelet image coding and transform coding is the omission of subimage processing.
- This eliminates the blocking artifact that characterizes DCT-based approximations at high compression ratios.

Wavelet Selection

- The most widely used expansion functions for wavelet-based compression are the Daubechies wavelets and biorthogonal wavelets.
- The latter allow
  - useful *analysis properties*, like number of zero moments, to be incorporated into the decomposition filters,
  - while important *synthesis properties*, like smoothness of reconstruction, are built into the reconstruction filters.

a	b
c	d

FIGURE 8.46 Three-scale wavelet transforms of Fig. 8.9(a) with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies Feauveau biorthogonal wavelets.

JPEG 2000

- JPEG 2000 extends the initial JPEG standard to provide increased flexibility in both the compression of continuous tone still images and access to the compressed data.
- portions of a JPEG 2000 compressed image can be extracted for retransmission, storage, display, and/or editing.

JPEG 2000

Filter Tap	Highpass Wavelet Coefficient	Lowpass Scaling Coefficient
0	−1.115087052456994	0.6029490182363579
±1	0.5912717631142470	0.2668641184428723
±2	0.05754352622849957	−0.07822326652898785
±3	−0.09127176311424948	−0.01686411844287495
±4	0	0.02674875741080976

TABLE 8.15 Impulse responses of the low- and highpass analysis filters for an irreversible 9-7 wavelet transform.

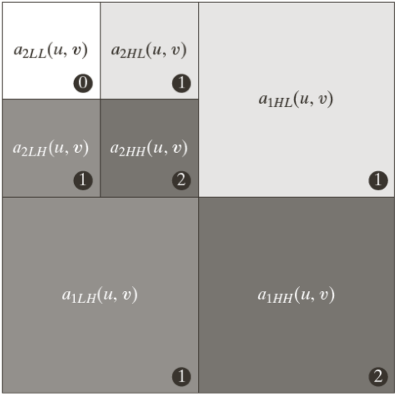
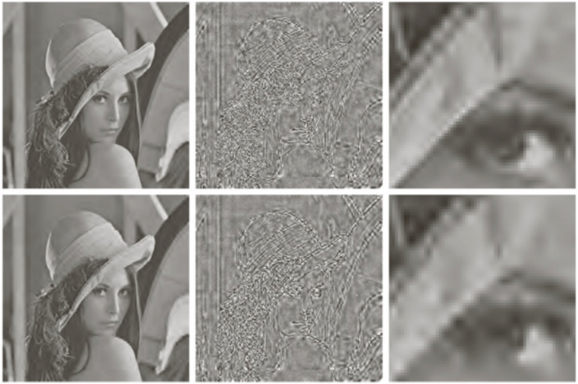


FIGURE 8.48 JPEG 2000 two-scale wavelet transform tile-component coefficient notation and analysis gain.

25:1

52:1

FIGURE 8.49 Four JPEG-2000 approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image. (Compare the results in rows 1 and 2 with the JPEG results in Fig. 8.32.)

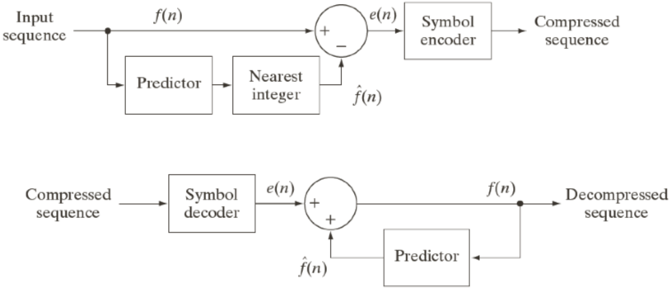


75:1

105:1

FIGURE 8.49 Four JPEG-2000 approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image. (Compare the results in rows 1 and 2 with the JPEG results in Fig. 8.32.)

Video Compression Standards



a
b

FIGURE 8.33
A lossless predictive coding model:
(a) encoder;
(b) decoder.