Chapter-8
Image Compression

The Need for Compression

- Image: 6 mega pixel camera, 3000x2000, RGB
  - 6x3 → 18 MB per uncompressed image
    → Only 56 uncompressed pictures per 1GB SD card
- Video: DVD Disc 4.7 GB
  - Video 720x480, RGB, 30 frames/s → 31.1MB/sec
  - 2-hour uncompressed movie: $31.1 \times 60^2 \times 2$ → 224 GB
- Send video from cell phone:

  352*240, RGB, 15 frames / second
  - 3.8 MB/sec, with no compression

Why do we need compression?

- In uncompressed form, Digital images/videos have considerable storage and transmission bandwidth requirements
- **Goal**: Reduce amount of data that needs to be stored to represent image/video

Main Idea:

- Reduce the amount of redundant data needed to reconstruct image/video
- Transform image data into a form that reduces statistical correlation and reduces information redundancy

Image Compression

- Image compression addresses the problem of reducing the amount of data required to represent a digital image, so that it can be stored or transmitted more efficiently

- Compression techniques fall into two categories:
  - Lossless (Information Preserving)
    - Allows images to be compressed and decompressed without loss of information
  - Lossy
    - Provides higher level of data reduction but results in less than perfect reproduction of the original image

8.1. Relative Data Redundancy

- Data are the means by which information is conveyed.
- Various quantities of data can be used to convey the same amount of information.
- If more data is used than strictly necessary, then we have data redundancy

8.1. Relative Data Redundancy

- Let b and b' denote the number of bits in two representations of the same information, the relative data redundancy R is

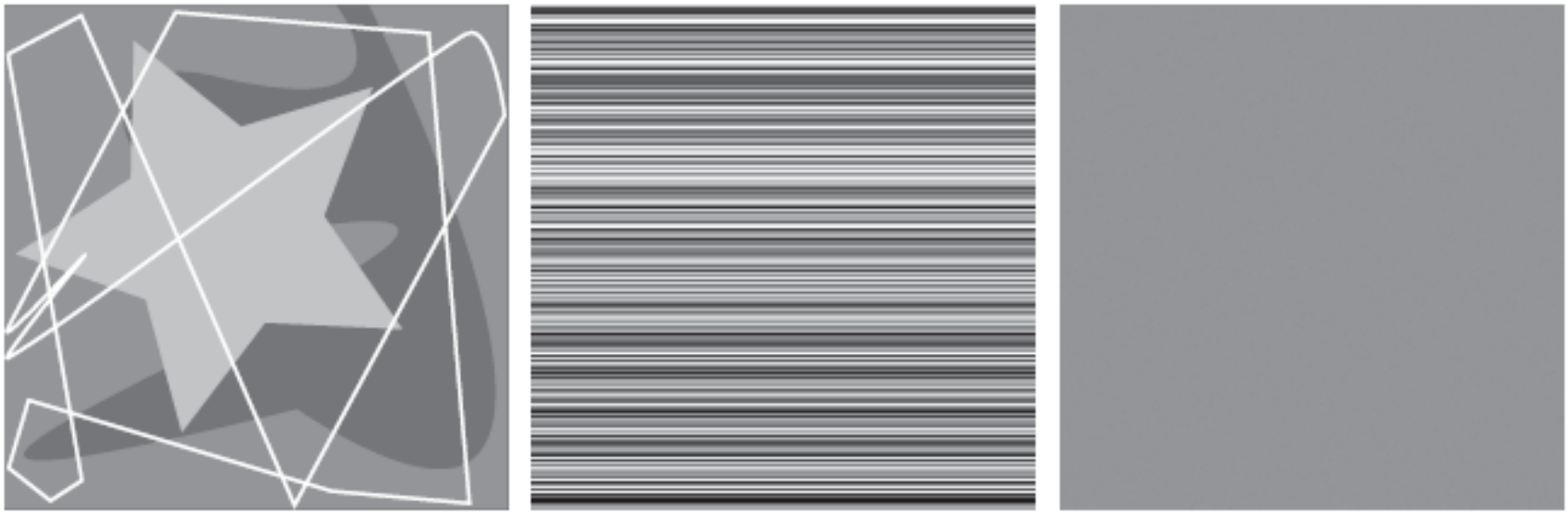
$$R = 1\text{-}1/C$$

- C is called the compression ratio, $C = b/b'$
- e.g., for $C = 10$, the corresponding relative data redundancy of the larger representation, $R = 0.9$, indicating that 90% of its data is redundant.

Types of Redundancy

- **Coding Redundancy (# of bits?)**
  - Most 2-D intensity arrays contain more bits than are needed to represent the intensities
- **Spatial and Temporal Redundancy (Correlated?)**
  - Pixels of most 2-D intensity arrays are correlated spatially
  - Video sequences are often correlated temporally between successive frames
- **Irrelevant Information (Important to human?)**
  - Most 2-D intensity arrays contain information that is ignored by the human visual system

Examples of Redundancy



a b c

FIGURE 8.1 Computer generated $256 \times 256 \times 8$ bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy, but may exhibit others as well.)

8.1.1 Coding Redundancy

- Gray levels in an image can be viewed as random variables
- Histogram shows the probability that each gray level appears in an image
- In most images, certain gray levels are more probable than the others
- By adjusting code length based on probability of gray levels, one can reduce coding redundancy

Coding Redundancy

- **Fundamentals:**
  - The grey level histogram of an image can reveal a great deal of information about the image, e.g. probability (frequency) of occurrence of grey level r_k is $p_r(r_k)$,

$$p_r(r_k) = \frac{n_k}{n} = \frac{n_k}{MN} \qquad k = 0, 2, \ldots, L-1 \quad (L = 256 \text{ for } m = 8\text{-bit image})$$

  - If the number of bits used to represent each value of r_k is $l(r_k)$, the average number of bits required to represent each pixel is

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

  - Coding of an M x N image requires MNL_{avg} bits

Variable Length Coding

- If m-bit natural binary code is used to represent the grey levels, then

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

- However, if some pixel values are more common than others, then →
- Possible cure: Assign fewer bits to the more probable grey levels than to less probable ones

Variable Length Coding - Example

- Consider the intensity distribution of the computer generated image in Fig 8.1 (a)

- Observation: Only 4 intensities exist

TABLE 8.1
Example of variable-length coding.

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

- $L_{avg} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81$ bits
- $C = 8/1.81 = 4.42$ and $R = 1 - 1/(4.42) = 0.774$

77.4% Redundancy

Variable Length Coding

- For Code-1, average number of bits required to code image is 8 bits

- For Code-2, only 1.81 bits are needed on average

Conclusion:

➢ Code-1 has high coding redundancy as it requires more bits than necessary to code

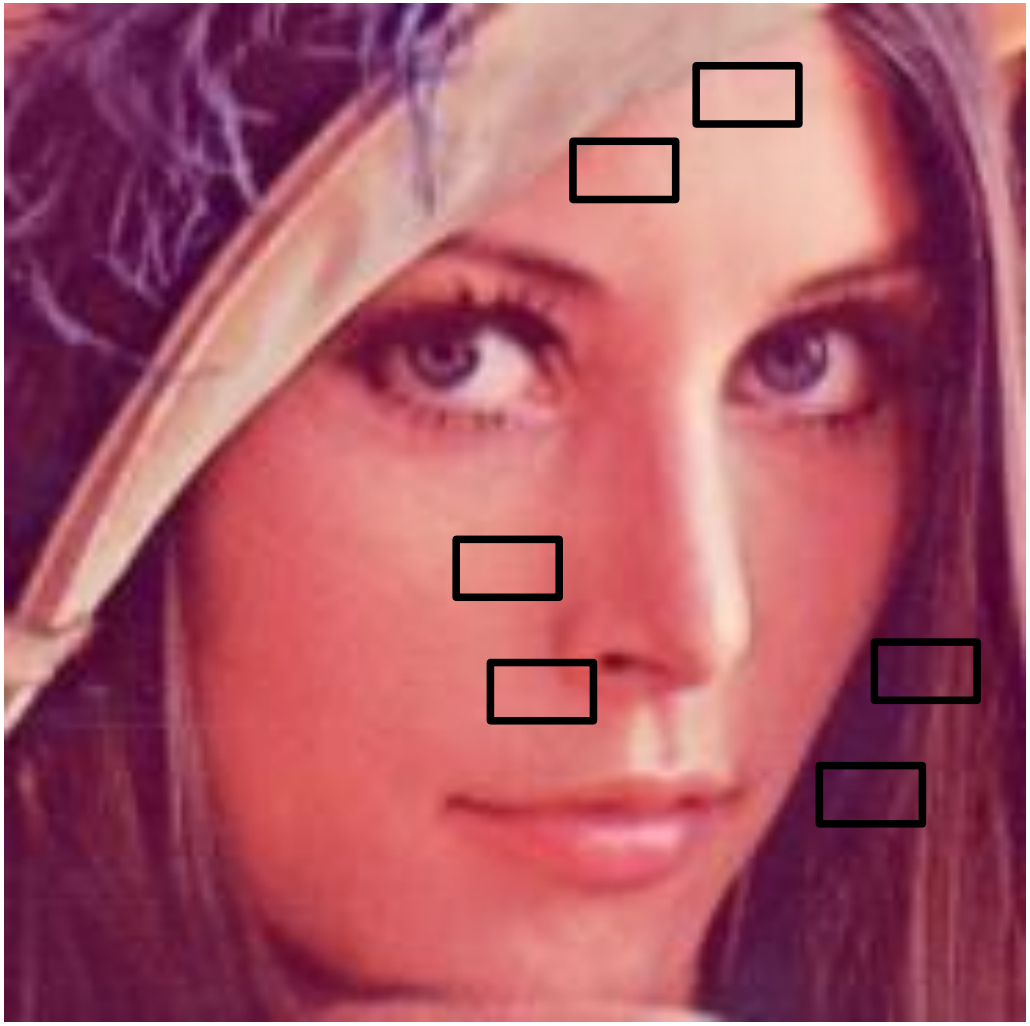
8.1.2 Spatial Redundancy

- Most images possess a lot of structural and intensity self-similarity

- Therefore, value of a given pixel or region can be reasonably predicted by neighboring pixels or regions

- One can take advantage of this self-similar nature of images to reduce the amount of information that needs to store/transmit.

Example of Spatial Redundancy

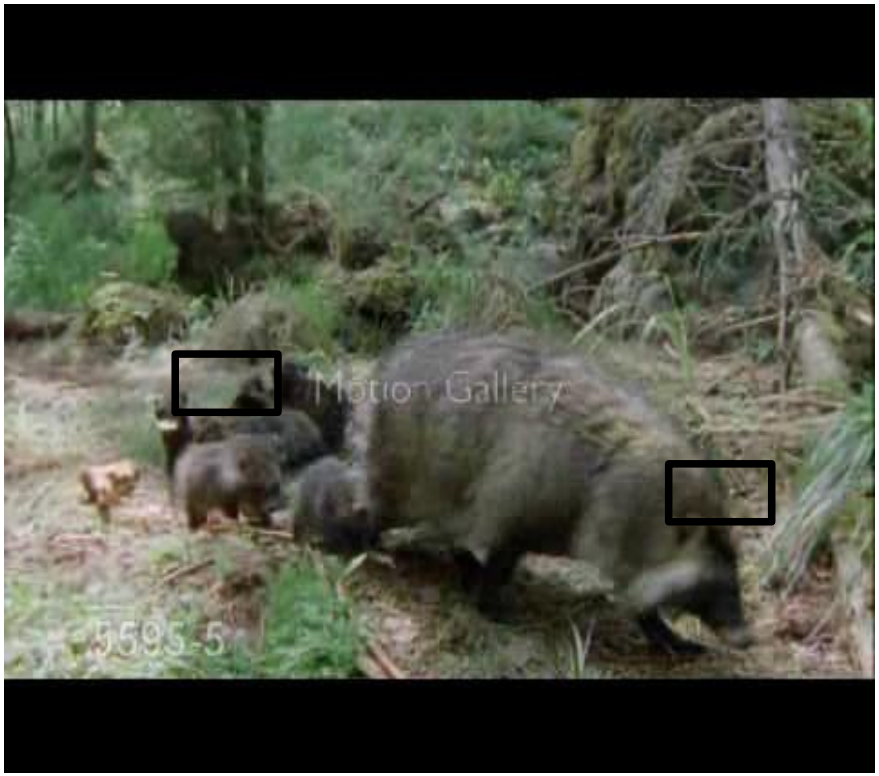


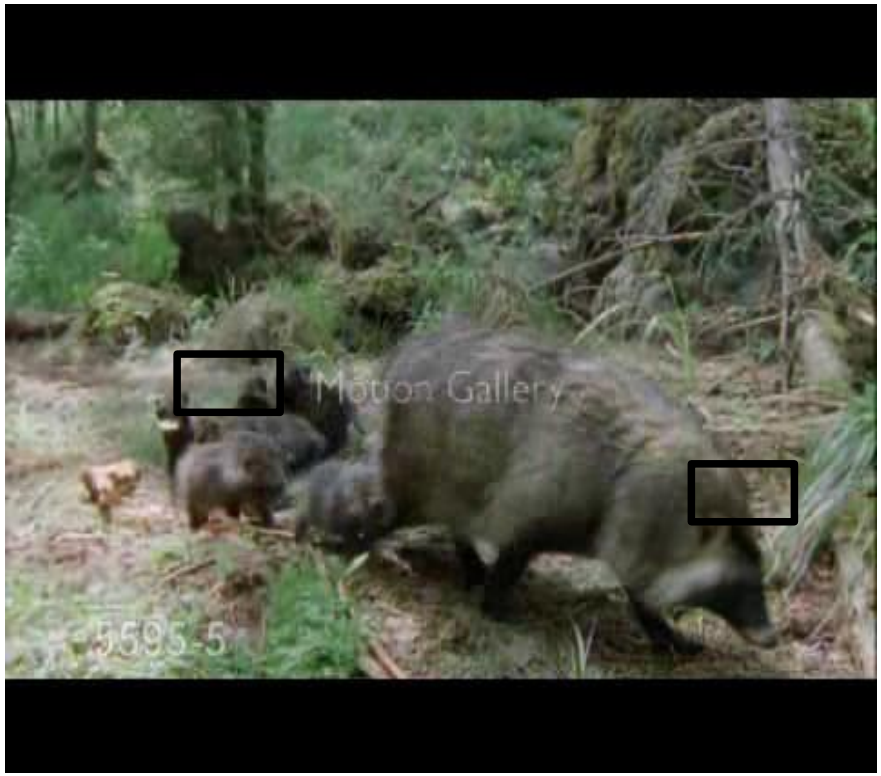
Temporal Redundancy

- In a video sequence, pixels and/or regions within each frame are similar to or dependent on pixels and/or regions from adjacent frames.

- Therefore, they can be reasonably predicted by temporally neighboring pixels or regions to reduce the amount of information that needs to be stored.

Example of Temporal Redundancy





Spatial and Temporal Redundancy

1. All 256 intensities are equally probable.
2. The pixels along each line are identical.
3. The intensity of each line was selected randomly.

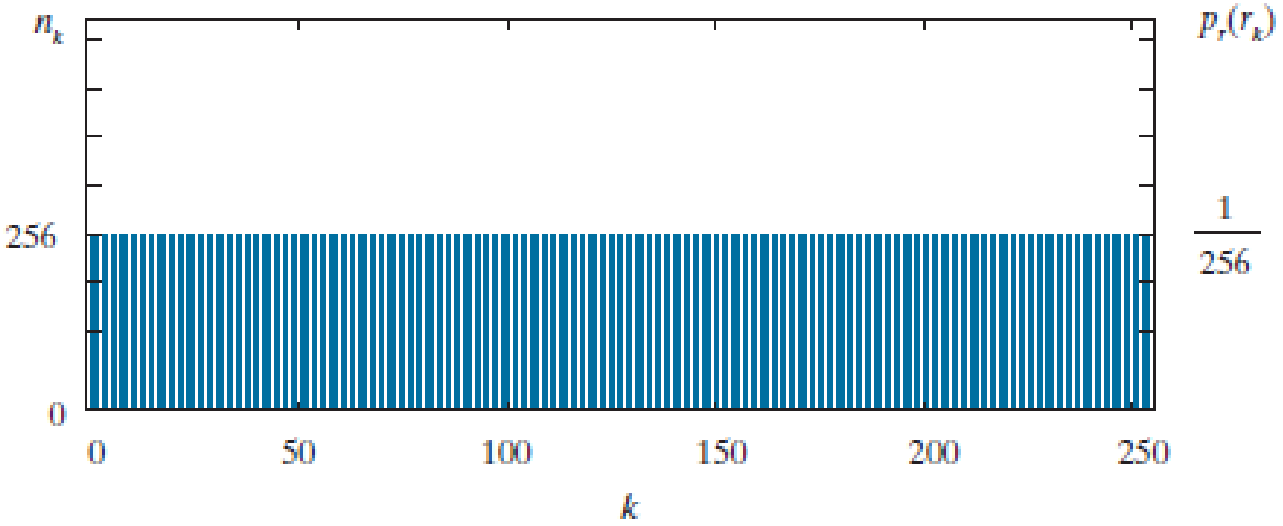
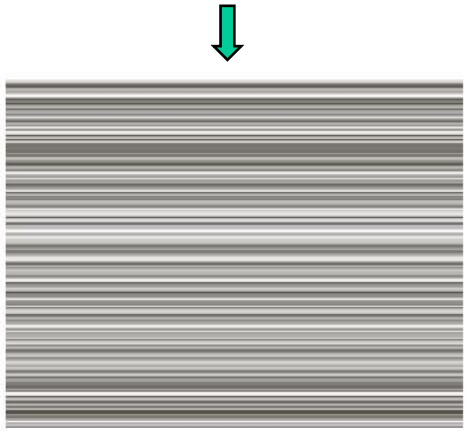


FIGURE 8.2 The intensity histogram of the image in Fig. 8.1(b).



Spatial and Temporal Redundancy

1. All 256 intensities are equally probable.
2. The pixels along each line are identical.
3. The intensity of each line was selected randomly.

- Run-length pair specifies the start of a new intensity and the number of consecutive pixels that have the same intensity.
- Each 256-pixel line of the original representation is replaced by a single 8-bit intensity value and length 256 in the run-length representation.

The compression ratio is

$$\frac{256 \times 256 \times 8}{(256 + 256) \times 8} = 128:1$$

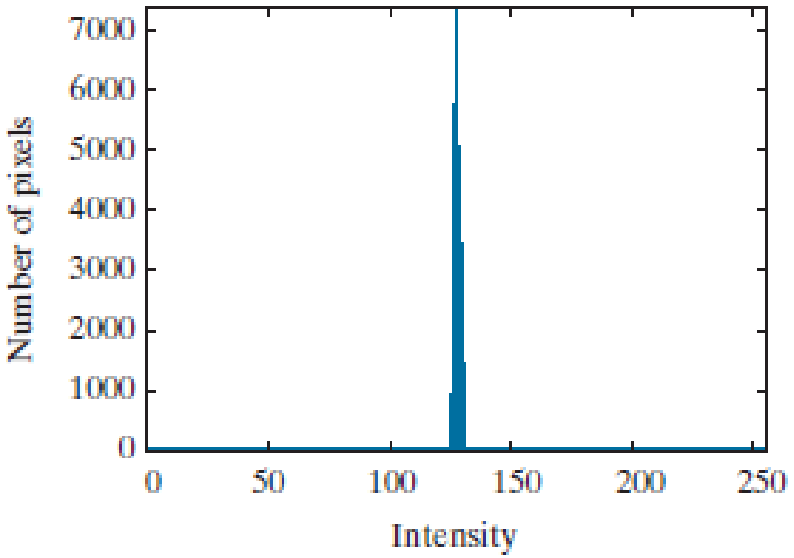
Lengths

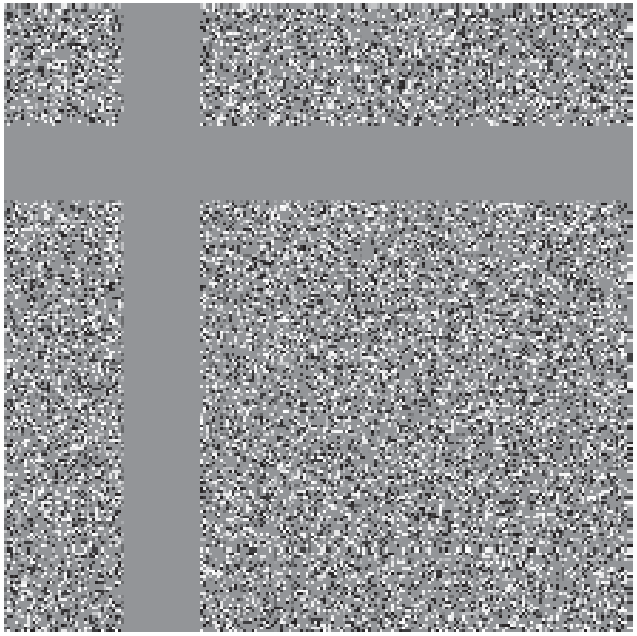
Intensity Values

8.1.3 Irrelevant Information

a b

FIGURE 8.3
(a) Histogram of the image in Fig. 8.1(c) and (b) a histogram equalized version of the image.





- Using "average intensity" i.e., a single 8-bit value

$$256 \times 256 \times 8 / 8 = 65536 : 1$$

8.1.4 Measuring Image Information

- Question: What is the fewest bits needed to represent the information in an image?
- i.e., Is there a minimum amount of data that is sufficient to describe the image without losing any information?

- Answer: Use Information Theory

- A random event E with probability $P(E)$ contain

$$I(E) = \log_2 \frac{1}{P(E)} = -\log_2 P(E)$$ units of information.

Measuring Image Information

- Consider a source of statistically independent random events from a discrete set of possible events $\{a_1, a_2, \ ..., \ a_J\}$ with associated probabilities $\{P(a_1), \ P(a_2), \ ..., \ P(a_J)\}$.
- Average information per source output, called the ENTROPY of the source

Entropy: $$H = -\sum_{j=1}^{J} P(a_j) \log_2 P(a_j)$$

a_j is called source symbols.

- Because they are statistically independent, the source is called *zero – memory source.*

Measuring Image Information

- An image is considered to be the output of an imaginary zero – memory intensity source
- We can use the histogram of the observed image to estimate the symbol probabilities of the source
- The intensity source's entropy becomes

$$\tilde{H} = -\sum_{k=0}^{L-1} p_{\mathrm{r}}(r_k) \log p_{\mathrm{r}}(r_k)$$

- $p_{\mathrm{r}}(r_k)$: The normalized histogram

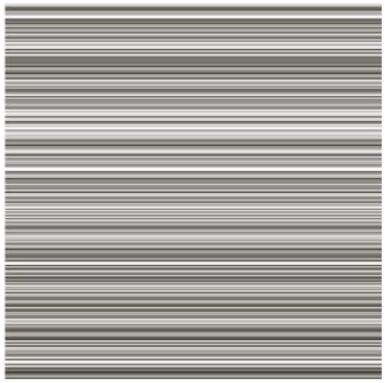
Measuring Image Information

For Fig.8.1(a), Entropy:

$$\tilde{H} = -[0.25\log_2 0.25 + 0.47\log_2 0.47 + 0.25\log_2 0.25 + 0.03\log_2 0.03]$$

$$\approx 1.6614 \text{ bits/pixel}$$

Recall Code-2: L_{avg} = 1.81 bits (Close)



For Fig.8.1(b),

$\tilde{H} = 8$ bits/pixel

For Fig.8.1(c):

$\tilde{H} = 1.566$ bits/pixel

8.1.5 Fidelity Criteria

- Objective: Need a means of quantifying information loss – two choices:
  - Root Mean Square Error (RMSE or, e_{rms})
  - Signal to Noise Ratio (SNR).

Let $f(x, y)$ be an input image and $\hat{f}(x, y)$ be an approximation of $f(x, y)$. The images are of size $M \times N$.

The *root - mean - square error* is

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

Fidelity Criteria

The *mean-square signal-to-noise ratio* of the output image, denoted SNR_{ms}

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

Fidelity Criteria

- **Subjective (Done by analysts/experts)**
  - Most decompressed images are viewed by human beings.
  - Subjective evaluations of compressed image quality by human observers (Analyists) are often more appropriate

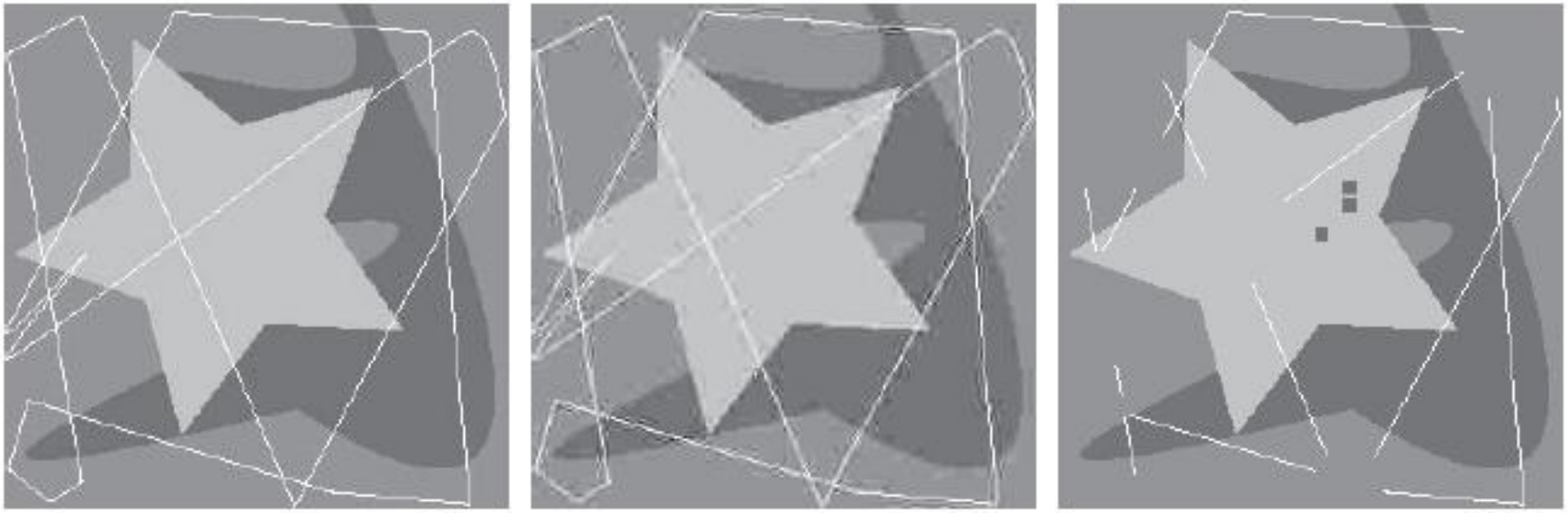
TABLE 8.2 Rating scale of the Television Allocations Study Organization. (Frendendall and Behrend.)

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.

Fidelity Criteria

(a), (b) After compression and Reconstruction (c) Computer generated distortion

RMSE = 5.17 RMSE = 15.67 RMSE = 14.17



a b c

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

- According to Subjective Criterion: (a) Excellent, (b) passable but (c) is inferior/unusable, although (c) has lower RMSE than (b)

Shannon's First Theorem

- Shannon's first theorem for a zero-memory source

$$H \le \frac{L_{avg,n}}{n} < H + \frac{1}{n}$$

$$H = \lim_{n \to \infty} \left[\frac{L_{avg,n}}{n} \right]$$

- $L_{avg,n}$: Average number of code symbols to represent all n-symbol groups.
- It is possible to make $L_{avg,n}/n$ arbitrarily close to H by coding infinitely long extensions of the source
- Efficiency = entropy/ L_{avg}= H/ L_{avg}

8.1.6 Image Compression Models

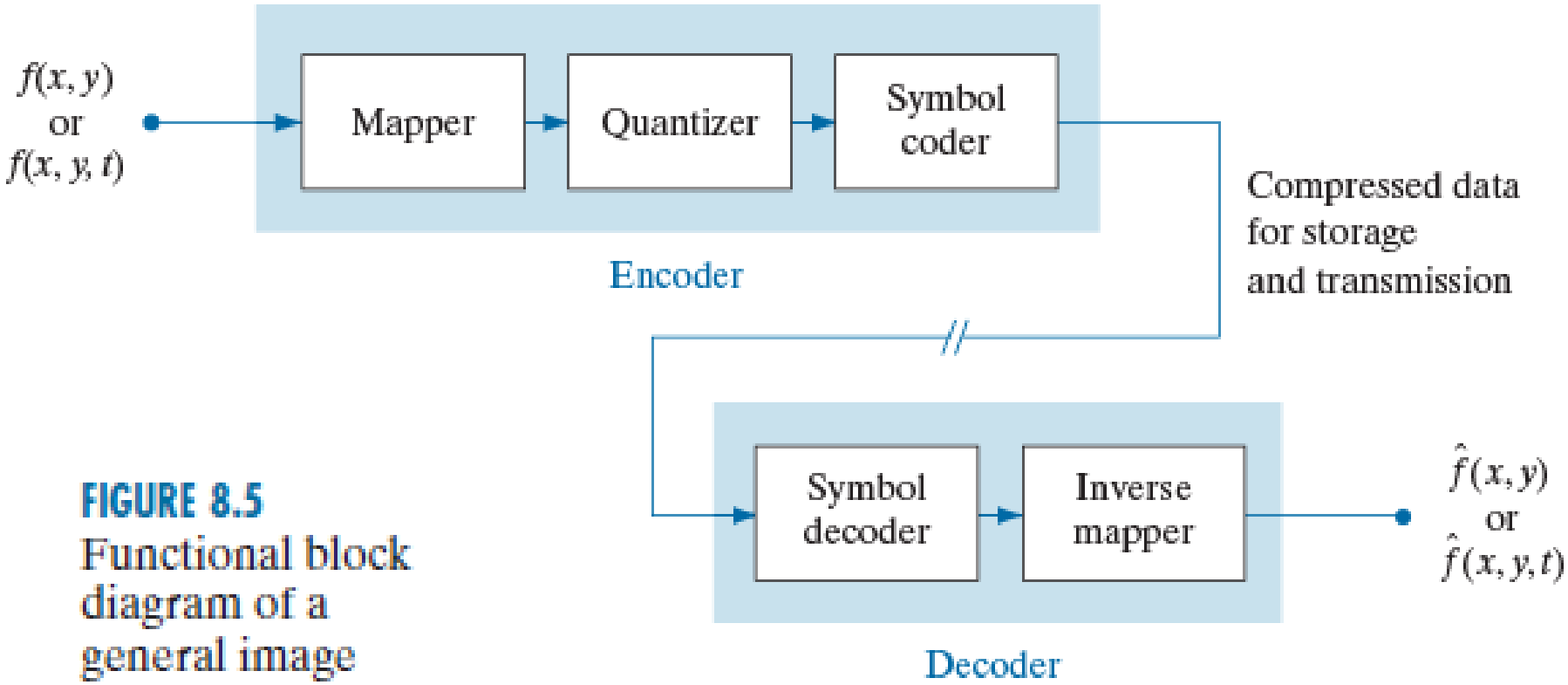


FIGURE 8.5 Functional block diagram of a general image compression system.

Image Compression Model

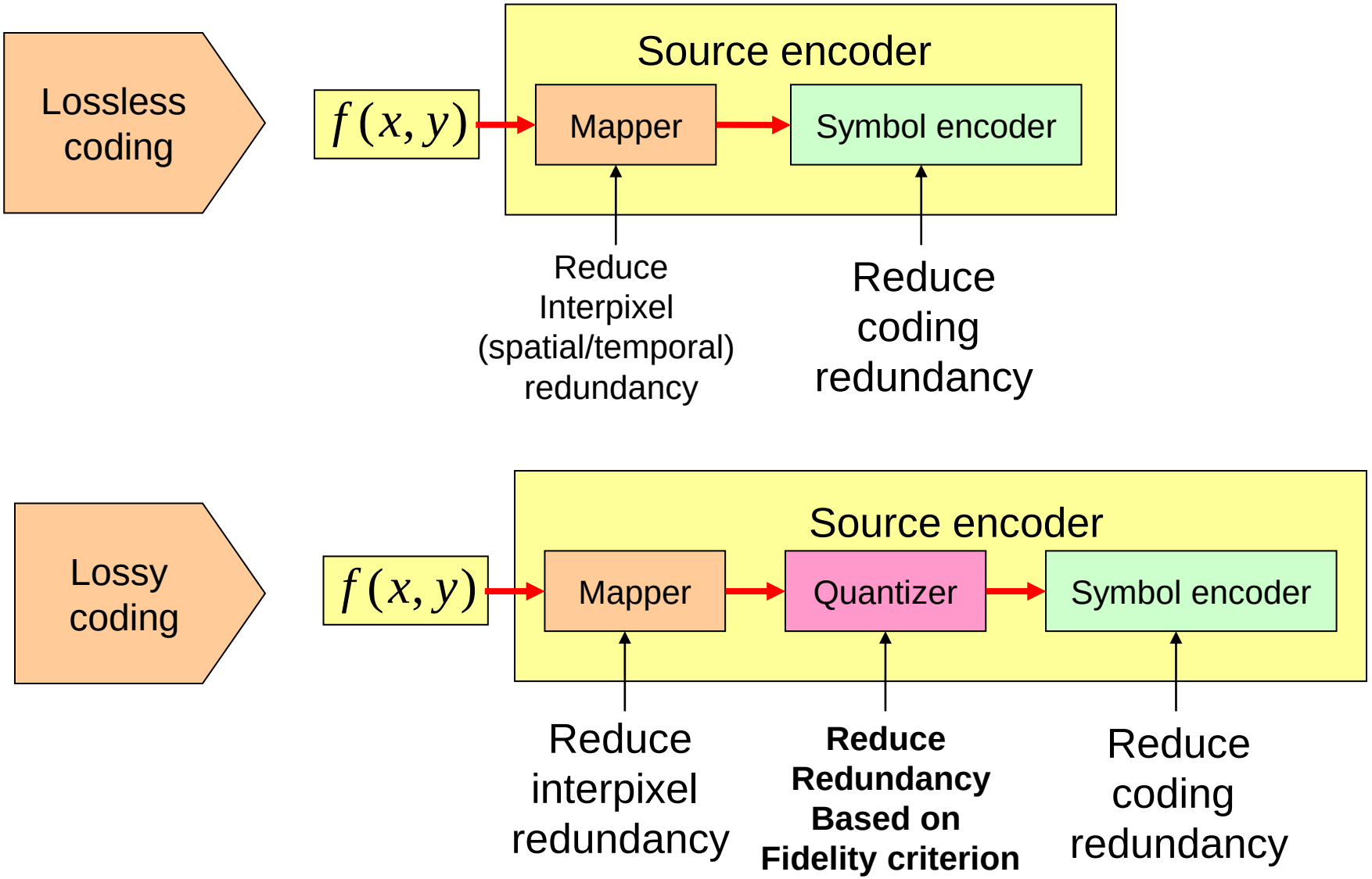


Image Compression Standards

FIGURE 8.6
Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in blue; all others are in black.

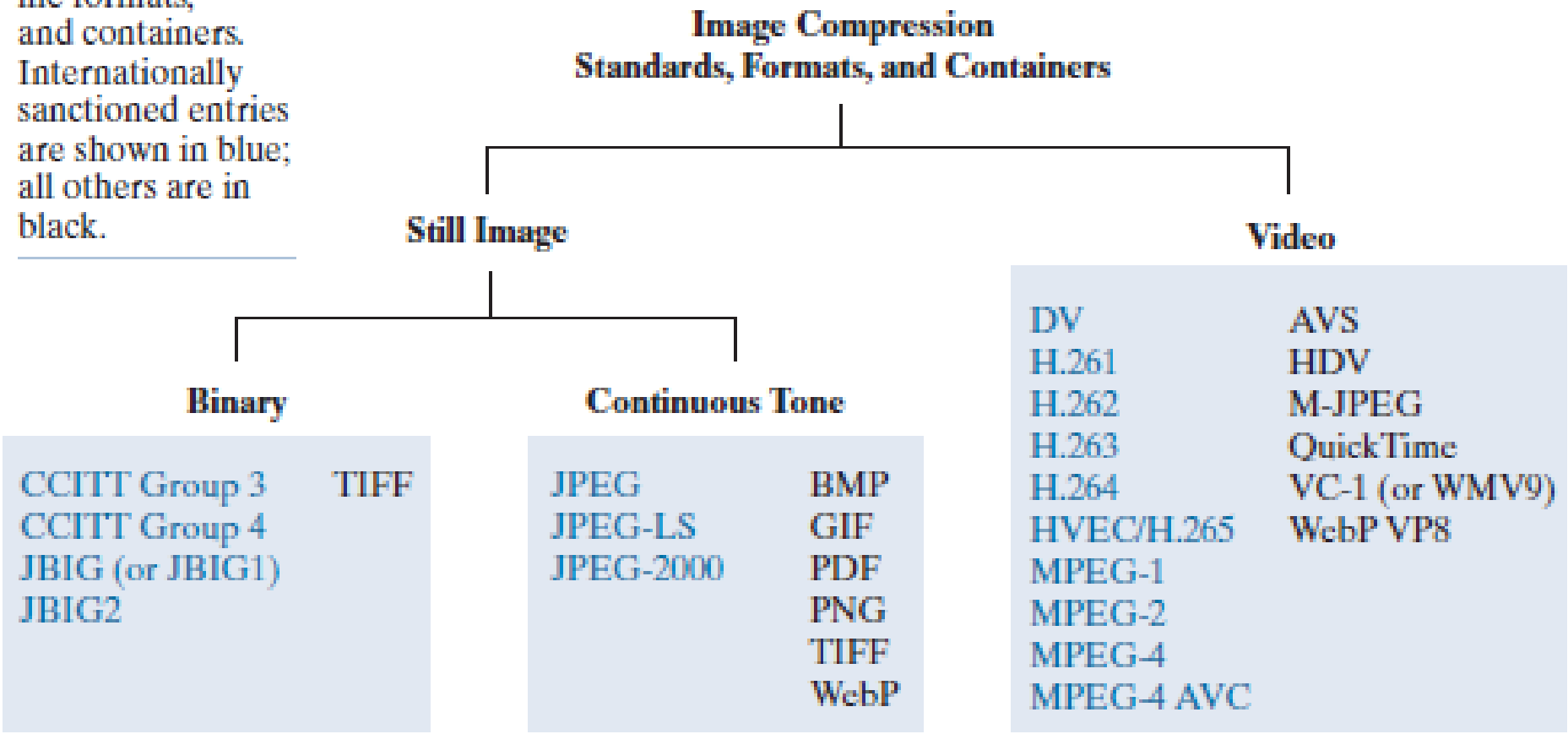


TABLE 8.3
Internationally sanctioned image compression standards. The numbers in brackets refer to sections in this chapter.

Name	Organization	Description
Bi-Level Still Images		
CCITT Group 3	ITU-T	Designed as a facsimile (FAX) method for transmitting binary documents over telephone lines. Supports 1-D and 2-D run-length [8.6] and Huffman [8.2] coding.
CCITT Group 4	ITU-T	A simplified and streamlined version of the CCITT Group 3 standard supporting 2-D run-length coding only.
JBIG or JBIG1	ISO/IEC/ ITU-T	A *Joint Bi-level Image Experts Group* standard for progressive, lossless compression of bi-level images. Continuous-tone images of up to 6 bits/pixel can be coded on a bit-plane basis [8.8]. Context-sensitive arithmetic coding [8.4] is used and an initial low-resolution version of the image can be gradually enhanced with additional compressed data.
JBIG2	ISO/IEC/ ITU-T	A follow-on to JBIG1 for bi-level images in desktop, Internet, and FAX applications. The compression method used is content based, with dictionary-based methods [8.7] for text and halftone regions, and Huffman [8.2] or arithmetic coding [8.4] for other image content. It can be lossy or lossless.
Continuous-Tone Still Images		
JPEG	ISO/IEC/ ITU-T	A *Joint Photographic Experts Group* standard for images of photographic quality. Its lossy baseline coding system (most commonly implemented) uses quantized discrete cosine transforms (DCT) on image blocks [8.9], Huffman [8.2], and run-length [8.6] coding. It is one of the most popular methods for compressing images on the Internet.
JPEG-LS	ISO/IEC/ ITU-T	A lossless to near-lossless standard for continuous-tone images based on adaptive prediction [8.10], context modeling [8.4], and Golomb coding [8.3].
JPEG-2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.4] and quantized discrete wavelet transforms (DWT) [8.11] are used. The compression can be lossy or lossless.

TABLE 8.4
Internationally sanctioned video compresssion standards. The numbers in brackets refer to sections in this chapter.

Name	Organization	Description
DV	IEC	*Digital Video*. A video standard tailored to home and semiprofessional video production applications and equipment, such as electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.9] similar to JPEG.
H.261	ITU-T	A two-way videoconferencing standard for ISDN (*integrated services digital network*) lines. It supports non-interlaced 352×288 and 176×144 resolution images, called CIF (*Common Intermediate Format*) and QCIF (*Quarter CIF*), respectively. A DCT-based compression approach [8.9] similar to JPEG is used, with frame-to-frame prediction differencing [8.10] to reduce temporal redundancy. A block-based technique is used to compensate for motion between frames.
H.262	ITU-T	See MPEG-2 below.
H.263	ITU-T	An enhanced version of H.261 designed for ordinary telephone modems (i.e., 28.8 Kb/s) with additional resolutions: SQCIF (*Sub-Quarter* CIF 128×96), 4CIF (704×576) and 16CIF (1408×512).
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, streaming, and television. It supports prediction differences within frames [8.10], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.4].
H.265 MPEG-H HEVC	ISO/IEC ITU-T	*High Efficiency Video Coding* (HVEC). An extension of H.264 that includes support for macroblock sizes up to 64×64 and additional intraframe prediction modes, both useful in 4K video applications.
MPEG-1	ISO/IEC	A *Motion Pictures Expert Group* standard for CD-ROM applications with non-interlaced video at up to 1.5 Mb/s. It is similar to H.261 but frame predictions can be based on the previous frame, next frame, or an interpolation of both. It is supported by almost all computers and DVD players.
MPEG-2	ISO/IEC	An extension of MPEG-1 designed for DVDs with transfer rates at up to 15 Mb/s. Supports interlaced video and HDTV. It is the most successful video standard to date.
MPEG-4	ISO/IEC	An extension of MPEG-2 that supports variable block sizes and prediction differencing [8.10] within frames.
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 *Advanced Video Coding* (AVC). Identical to H.264.

TABLE 8.5
Popular image and video compression standards, file formats, and containers not included in Tables 8.3 and 8.4. The numbers in brackets refer to sections in this chapter.

Name	Organization	Description
Continuous-Tone Still Images		
BMP	Microsoft	*Windows Bitmap*. A file format used mainly for simple uncompressed images.
GIF	CompuServe	*Graphic Interchange Format*. A file format that uses lossless LZW coding [8.5] for 1- through 8-bit images. It is frequently used to make small animations and short low-resolution films for the Internet.
PDF	Adobe Systems	*Portable Document Format*. A format for representing 2-D documents in a device and resolution independent way. It can function as a container for JPEG, JPEG-2000, CCITT, and other compressed images. Some PDF versions have become ISO standards.
PNG	*World Wide Web Consortium* (W3C)	*Portable Network Graphics*. A file format that losslessly compresses full color images with transparency (up to 48 bits/pixel) by coding the difference between each pixel's value and a predicted value based on past pixels [8.10].
TIFF	Aldus	*Tagged Image File Format*. A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000, JBIG2, and others.
WebP	Google	*WebP* supports lossy compression via WebP VP8 intraframe video compression (see below) and lossless compression using spatial prediction [8.10] and a variant of LZW backward referencing [8.5] and Huffman entropy coding [8.2]. Transparency is also supported.
Video		
AVS	MII	*Audio-Video Standard*. Similar to H.264 but uses exponential Golomb coding [8.3]. Developed in China.
HDV	Company consortium	*High Definition Video*. An extension of DV for HD television that uses compression similar to MPEG-2, including temporal redundancy removal by prediction differencing [8.10].
M-JPEG	Various companies	*Motion JPEG*. A compression format in which each frame is compressed independently using JPEG.
QuickTime	Apple Computer	A media container supporting DV, H.261, H.262, H.264, MPEG-1, MPEG-2, MPEG-4, and other video compression formats.
VC-1 WMV9	SMPTE Microsoft	The most used video format on the Internet. Adopted for HD and *Blu-ray* high-definition DVDs. It is similar to H.264/AVC, using an integer DCT with varying block sizes [8.9 and 8.10] and context-dependent variable-length code tables [8.2], but no predictions within frames.
WebP VP8	Google	A file format based on block transform coding [8.9] prediction differences within frames and between frames [8.10]. The differences are entropy encoded using an adaptive arithmetic coder [8.4].

8.2 Error-Free Compression: Huffman Coding

Huffman coding: Gives the smallest possible number of code symbols per source symbol.

- Optimal in Shannon sense if source symbols are coded one at a time (i.e., not block coding)

With reference to Tables 8.3-8.5, Huffman codes are used in

- CCITT
- JBIG2
- JPEG
- MPEG-1,2,4
- H.261, H.262, H.263, H.264

and other compression standards.

Step 1: Huffman Source reduction

FIGURE 8.7 Huffman source reductions.

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				

Error-Free Compression: Huffman Coding

Step 2: Huffman Code assignment procedure

FIGURE 8.8 Huffman code assignment procedure.

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								

The code is instantaneous & uniquely decodable without referencing succeeding symbols.

The average length of this code is,

$$L_{avg} = 0.4*1+0.3*2+0.1*3+0.1*4+0.06*5+0.04*5$$

$$= 2.2 \text{ bits/pixel}$$

Entropy, H =2.14 bits/pixel $\rightarrow$ Very close

8.3 Golomb Coding

- For coding nonnegative integer inputs with Exponentially decaying (e.g., geometric) probability distribution
- Can be optimally coded (in Shannon sense)
- Computationally simpler than Huffman Coding of the same set of symbols
- Notation used $\lfloor \bullet \rfloor \rightarrow$ Floor

  $\lceil \bullet \rceil \rightarrow$ Ceiling

With reference to Tables 8.3-8.5, Golomb codes are used in

- JPEG-LS
- AVS

compression.

8.2.2 Golomb Coding

Given a nonnegative integer n and a positive integer divisor $m > 0$, the Golomb code of n with respect to m, denoted by $G_m(n)$, is constructed as follows:

Step 1. Form the unary code of quotient, $q = \lfloor n/m \rfloor$

(The unary code of integer q is defined as q # of 1's followed by a 0)

Step 2. Let $k = \lceil \log_2 m \rceil$, $c = 2^k - m$, $r = n \bmod m$, and compute truncated remainder r' such that

$$r' = \begin{cases} r \text{ truncated to } k\text{-}1 \text{ bits} & 0 \le r < c \\ r + c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases}$$

Step 3. Concatenate the results of steps 1 and 2.

With reference to Tables 8.3-8.5, Golomb codes are used in

- JPEG-LS
- AVS

compression.

Golomb Coding Steps: Example-1

$G_4(9): \quad n = 9, m = 4$

Step 1. Form the unary code of quotient $q = \lfloor n/m \rfloor$

(The unary code of integer q is defined as q 1s followed by a 0)

$q = \lfloor 9/4 \rfloor = 2,$

the unary code is 110

Step 2. Let $k = \lceil \log_2 m \rceil,\ c = 2^k - m,\ r = n \bmod m,$ and compute truncated remainder r' such that

$k = \lceil \log_2 4 \rceil = 2, c = 2^2 - 4 = 0,$

$r = 9 \bmod 4 = 1001 \bmod 0100$

$= 0001 > c; \quad r = 0001$

$$r' = \begin{cases} r \text{ truncated to } k\text{-1 bits} & 0 \le r < c \\ r + c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases}$$

$r' = r + c = 0001$

truncated to k=2-bits $\rightarrow$ 01

Step 3. Concatenate the results of steps 1 and 2.

$G_4(9) = 11001$

Golomb Coding: Example-2

$G_4(7): \quad n = 7, m = 4$

Step 1. Form the unary code of quotient $q = \lfloor n/m \rfloor$

(The unary code of integer q is defined as q 1s followed by a 0)

$q = \lfloor 7/4 \rfloor = 1,$

the unary code is 10

Step 2. Let $k = \lceil \log_2 m \rceil, c = 2^k - m, r = n \bmod m,$ and compute truncated remainder r' such that

$k = \lceil \log_2 4 \rceil = 2, c = 2^2 - 4 = 0,$

$r = 7 \bmod 4 = 3 > c; \quad r = 0011$

$$r' = \begin{cases} r \text{ truncated to } k\text{-1 bits} & 0 \le r < c \\ r + c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases}$$

$r' = r + c = 0011$

truncated to $k = 2$-bits $\rightarrow 11$

Step 3. Concatenate the results of steps 1 and 2.

$G_4(7) = 1011$

Golomb Coding

- Golomb Codes for several integers

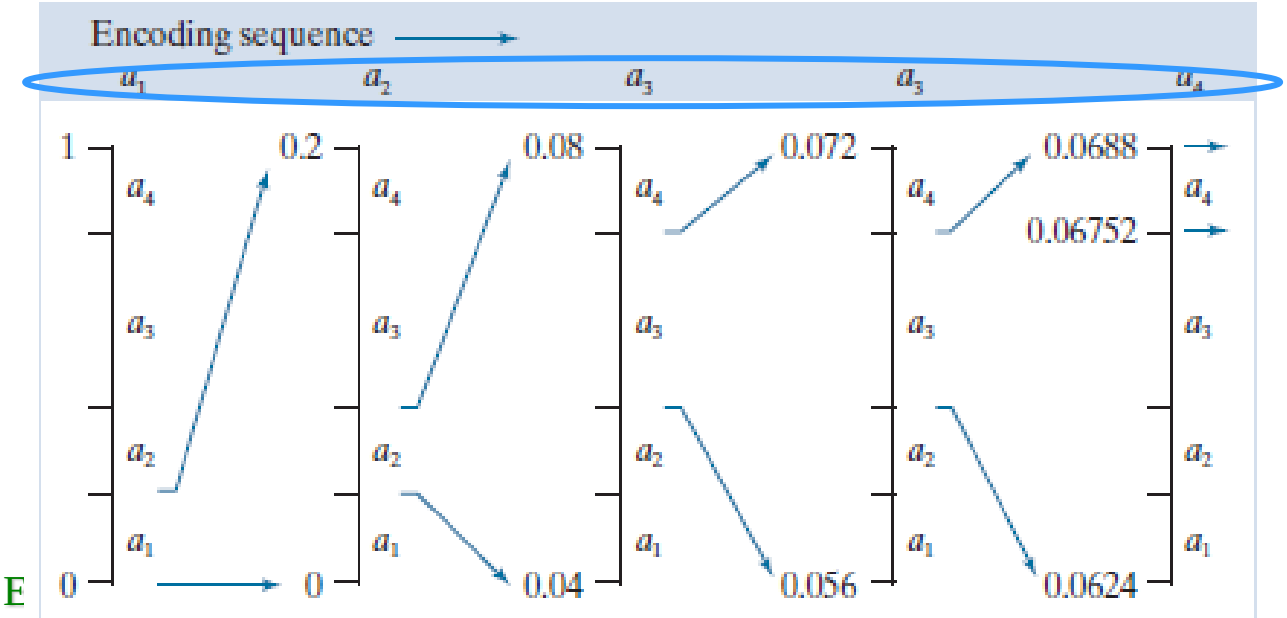
TABLE 8.6 Several Golomb codes for the integers 0–9.

n	$G_1(n)$	$G_2(n)$	$G_4(n)$	$G^0_{exp}(n)$
0	0	00	000	0
1	10	01	001	100
2	110	100	010	101
3	1110	101	011	11000
4	11110	1100	1000	11001
5	111110	1101	1001	11010
6	1111110	11100	1010	11011
7	11111110	11101	1011	1110000
8	111111110	111100	11000	1110001
9	1111111110	111101	11001	1110010

- Optimal if integers are geometrically distributed
- Seldom used for coding intensities
- Used mainly for coding Intensity Differences
- Is a variable length code

8.4 Arithmetic Coding (1963)

- Nonblock code: One-to-one correspondence between source symbols. Code words do not exist.
- Concept: Entire sequences of source symbols is assigned a single arithmetic code word in the form of a real number in an interval between 0 and 1.
- 5-symbol sequence using 4-symbols



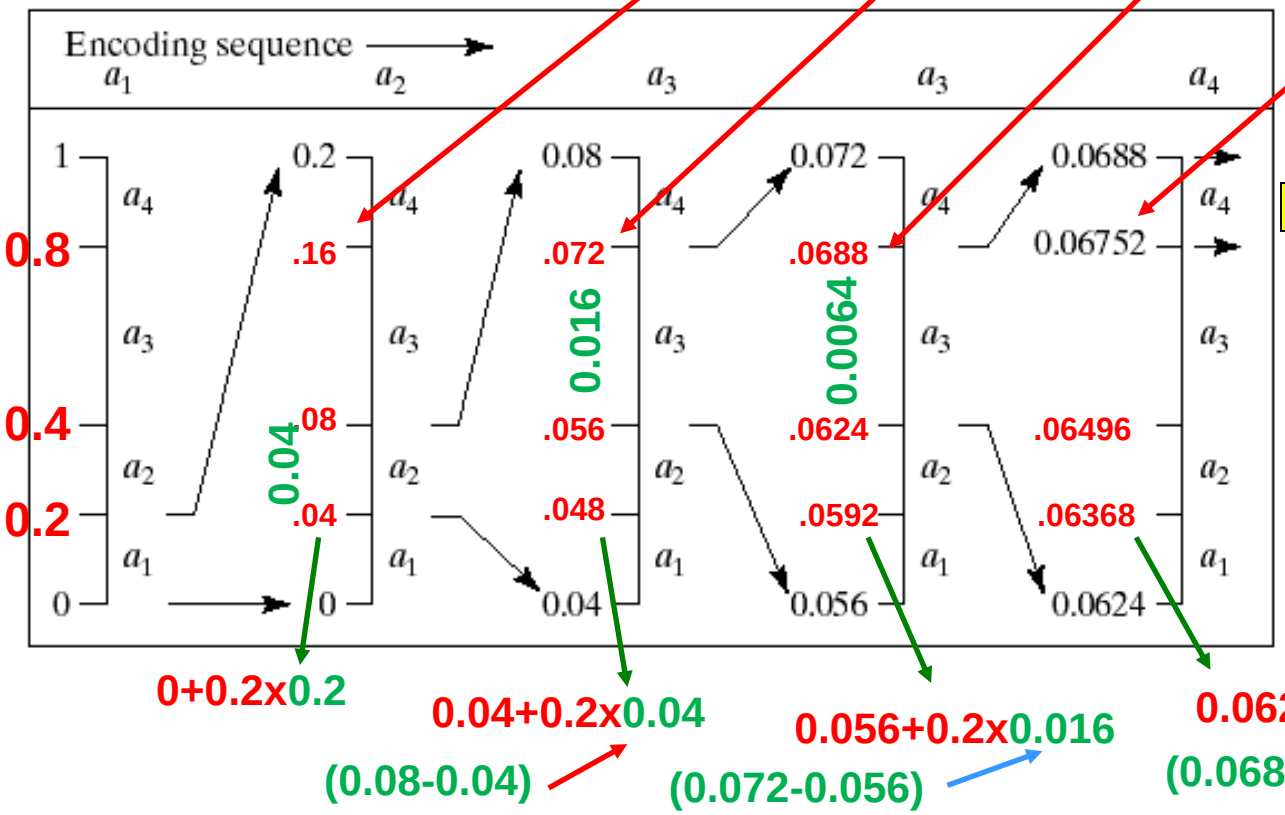
With reference to Tables 8.3-8.5, Arithmetic coding is used in

- JBIG1
- JBIG2
- JPEG-2000
- H.264
- MPEG-4 AVC

and other compression standards.

FIGURE 8.12
Arithmetic coding procedure.

Arithmetic Coding Example



Any number between 0.06752 and 0.0688 can be used to represent the entire sequence $a_1\ a_2\ a_3\ a_3\ a_4$

TABLE 8.7
Arithmetic coding 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)

8.5 LZW (Dictionary coding)

- **LZW (Lempel-Ziv-Welch) coding** [1984]

➢ Assigns fixed-length code words to variable length sequences of source symbols

➢ Requires no *a priori* knowledge of the probability of the source symbols. (No histogram)

- LZW is used in:
  - ➢ Tagged Image file format (TIFF)
  - ➢ Graphic interchange format (GIF)
  - ➢ Portable document format (PDF)

With reference to Tables 8.3-8.5, LZW coding is used in

- GIF
- TIFF
- PDF

formats, but not in any of the internationally sanctioned compression standards.

LZW Coding Algorithm

- A codebook or "dictionary" containing the source symbols is constructed.
- For 8-bit monochrome images, the first 256 words of the dictionary are assigned to each gray level 0-255
- Remaining part of the dictionary is filled with *sequences* of the gray levels in the image itself (not predeterminded)

LZW Coding Algorithm - Details

0. Initialize a dictionary by all possible gray values (0-255)
1. Input current pixel
2. If the current pixel combined with previous pixels form one of existing dictionary entries

Then → No new Dictionary Entry

2.1 Move to the next pixel and repeat Step 1

Else → Need new Dictionary Entry

2.2 Output the dictionary location of the currently recognized sequence (not including the current pixel)

2.3 Create a new dictionary entry by appending the currently recognized sequence in 2.2 with current pixel

2.4 Move to the next pixel and repeat Step 1

EXAMPLE 8.7: LZW coding.

Dictionary Location	Entry
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

LZW Coding

Example: Consider a 4 x 4 8-bit image

39	**39**	**126**	**126**
39	**39**	**126**	**126**
39	**39**	**126**	**126**
39	**39**	**126**	**126**

LZW Coding

- Lempel-Ziv-Welch coding : Assign fixed length code words to variable length sequences of source symbols.

TABLE 8.8 LZW Coding example.

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

24 Bits

9 Bits

Important features of LZW

1. The dictionary is created while the data are being encoded. So encoding can be done "on the fly" !!!
2. Key Idea: The dictionary need not be transmitted. The dictionary will be built up in the decoding
3. If the dictionary "overflows" then we have to reinitialize the dictionary and add a bit to each one of the code words.
4. Choosing a large dictionary size avoids overflow, but spoils compressions

Decoding LZW

- Let the bit stream received be:

39 39 126 126 256 258 260 259 257 126

- In LZW, the dictionary which was used for encoding need not be sent with the image.
- A separate dictionary is built by the decoder, "on the fly", as it reads the received code words.

Decoding LZW

Received/ Recognized	Encoded value	pixels	New Dic. address	New Dic. entry
	39	39		
39	39	39	256	39-39
39	126	126	257	39-126
126	126	126	258	126-126
126	256	39-39	259	126-39
256	258	126-126	260	39-39-126
258	260	39-39-126	261	126-126-39
260	259	126-39	262	39-39-126-126
259	257	39-126	263	126-39-39
257	126	126	264	39-126-126

WRIGHT STATE UNIVERSITY

8.6 Run-Length Coding

1. Run-length Encoding, or **RLE** is a technique used to **reduce the size of a repeating string of characters.**

With reference to Tables 8.3-8.5, the coding of run-lengths is used in

- CCITT
- JBIG2
- JPEG
- M-JPEG
- MPEG-1,2,4
- BMP

and other compression standards and file formats.

1. This repeating string is called a *run*, typically RLE encodes a run of symbols into two bytes , a **count** and a **symbol**.

2. RLE can compress any type of data

3. RLE cannot achieve high compression ratios compared to other compression methods

Run-Length Coding

5. However, it is easy to implement

6. Quick to execute.

7. Run-length encoding is supported by most bitmap file formats such as TIFF, BMP and PCX

Some Basic Compression Methods: Run-Length Coding

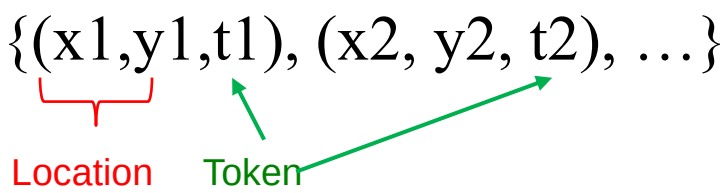
WWWWWWWWWWWWBWWWWWWWWWWWWBB
BWWWWWWWWWWWWWWWWWWWWWWWWBW
WWWWWWWWWWWWW

RLE coding:

12W1B12W3B24W1B14W

8.7 Symbol-Based Coding

- In symbol- or token-based coding, an image is represented as a collection of frequently occurring sub-images, called symbols.
- Each symbol is stored in a symbol dictionary
- Image is coded as a set of triplets

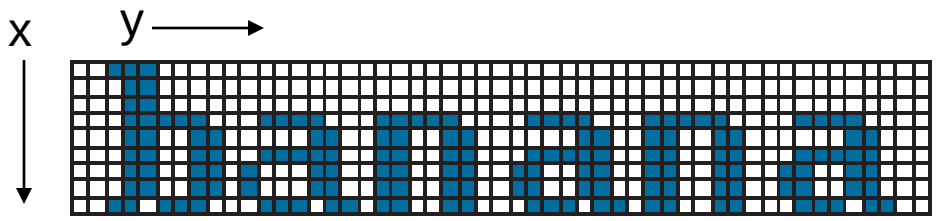


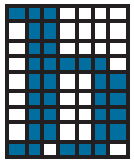
With reference to Tables 8.3-8.5, symbol-based coding is used in

- JBIG2

compression.

Some Basic Compression Methods: Symbol-Based Coding



Token	Symbol
0	
1	
2	

Triplet
(0, 2, 0)
(3,10, 1)
(3, 18, 2)
(3, 26, 1)
(3, 34, 2)
(3, 42, 1)

a b c

FIGURE 8.17
(a) A bi-level document, (b) symbol dictionary, and (c) the triplets used to locate the symbols in the document.

8.8 Bit-Plane Coding

- An m-bit gray scale image can be converted into m binary images by bit-plane slicing.

bit-plane coding is used in the

- JBIG1
- JPEG-2000

- These individual images are then encoded using run-length coding.
- Code the bit-planes separately, using RLE (flatten each plane row-wise into an $1D$ array), Golomb coding, or any other lossless compression technique.
- Let I be an image where every pixel value is n-bit long
- Express every pixel in binary using n bits
- Form n binary matrices (called bitplanes), where the i-th matrix consists of the i-th bits of the pixels of I.

Bit-Plane Coding

Example: Let I be the following $2x2$ image where the pixels are 3 bits long

$$\begin{matrix} 101 & 110 \\ 111 & 011 \end{matrix}$$

- The corresponding 3 bitplanes are:

1	1
1	0

0	1
1	1

1	0
1	1

Bit-Plane Coding

- Disadvantage: A small difference in the gray level of adjacent pixels can cause a disruption of the run of zeroes or ones.
- e.g., Let us say one pixel has a gray level of 127 and the next pixel has a gray level of 128.
- In binary: 127 = 01111111

  & 128 = 10000000
- Therefore a small change in gray level has decreased the run-lengths in all the bit-planes!

Bit-Plane Coding: GRAY CODE

1. Gray coded images are free of this problem which affects images which are in binary format.

2. In Gray code the representation of adjacent gray levels will differ only in one bit (unlike binary format where all the bits can change.

Bit-Plane Coding: GRAY CODE

- Let $g_{m-1}.....g_1g_0$ represent the gray code of a binary number, $a_{m-1}.....a_1a_0$
- Then:

$$g_i = a_i \oplus a_{i+1} \qquad 0 \le i \le m-2$$

$$g_{m-1} = a_{m-1}$$

XOR

- In Gray code:

127 = 01000000

128 = 11000000

XOR Truth Table		
Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Gray Coding

- To convert a binary number $b_1 b_2 b_3 .. b_{n-1} b_n$ to its corresponding binary reflected Gray code.

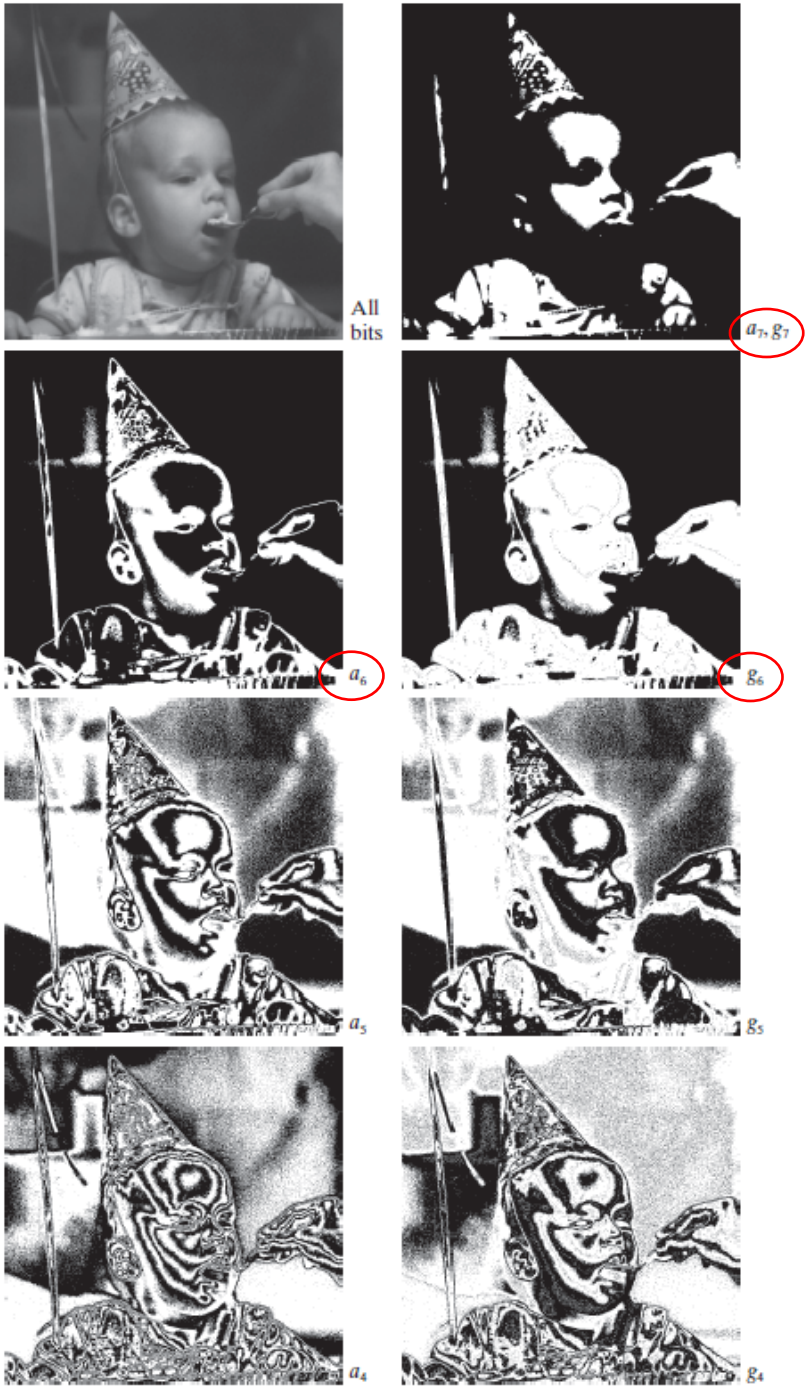
- Start at the right with the digit b_n. If the b_{n-1} is 1, replace b_n by 1-b_n ; otherwise, leave it unchanged. Then proceed to b_{n-1} .

- Continue up to the first digit b_1, which is kept the same since it is assumed to be a $b_0 = 0$.

- The resulting number is the reflected binary Gray code.

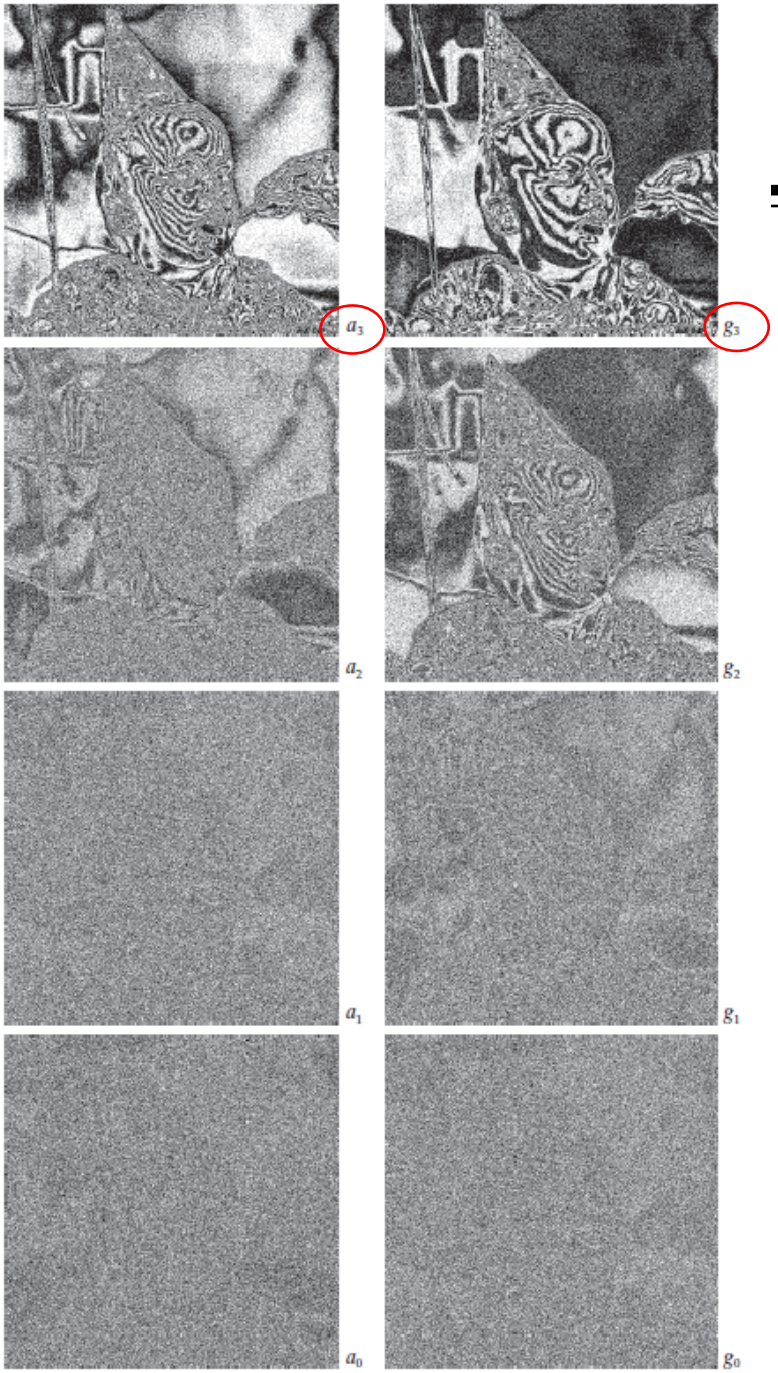
Examples: Gray Coding

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



a b
c d
e f
g h

FIGURE 8.19
(a) A 256-bit monochrome image.
(b)–(h) The four most significant binary and Gray-coded bit planes of the image in (a).



a b
c d
e f
g h

FIGURE 8.20
(a)–(h) The four least significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.19(a).

Examples: Gray Coding

- Decoding a gray coded image
- The MSB is retained as such, i.e.,

$$b_i = g_i \oplus b_{i+1} \qquad 0 \le i \le m-2$$

$$b_{m-1} = g_{m-1}$$

XOR

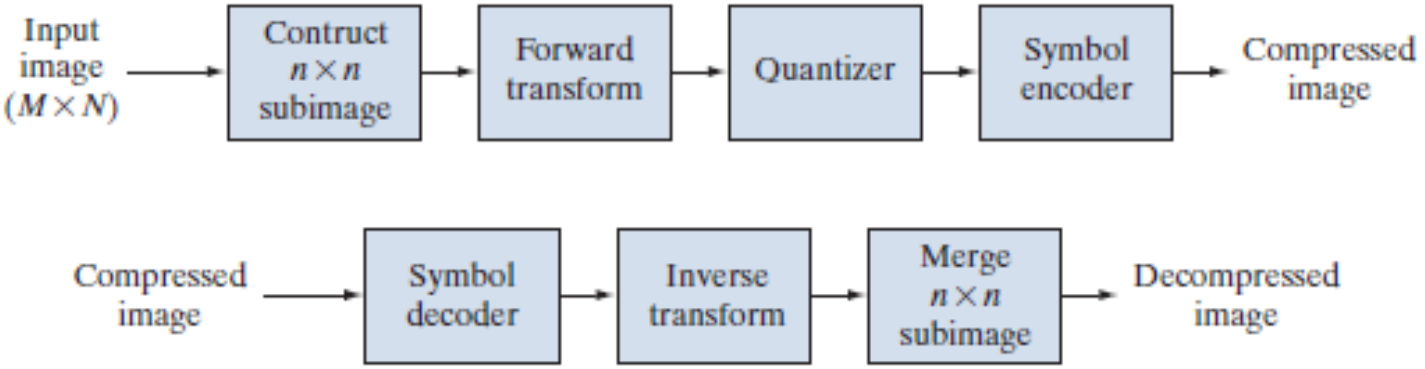
Transform Coding - Motivation

- At the heart of the majority of video coding system and standards.
- Spatial image data is inherently difficult to compress
- Neighboring samples are highly correlated (interrelated) and the energy tends to be evenly distributed across an image
- Makes it difficult to discard data or reduce the precision of data without adversely affecting image quality.

Block Transform Coding

- **Block Transform Coding**: Divides an image into small non- overlapping blocks of equal size (e.g. 8 x 8) and processes the blocks independently using a 2-D transform.
- A reversible, linear transform (such as Fourier transform) is used to map each block or subimage into a set of transform coefficients, which are then quantized and coded.
- Popular transforms commonly used:
  - Discrete Fourier Transform (DFT)
  - Discrete Cosine Transform (DCT)
  - Walsh-Hadamard Transform (WHT)
  - Discrete Wavelet Transform (DWT)

8.2.8 Block Transform Coding



a
b

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

- The transformation process acts to decorrelate the pixels of each subimage
- Pack as much information as possible into the smallest number of transform coefficients
- Compression is achieved during quantization of the transformed coefficients.

Transform Coding

- Compact the energy in the image
  - Concentrate the energy into a small number of significant coefficients
- Decorrelate the data → Use Orthogonal Transforms
  - Minimize the interdependencies between coefficients
  - Discarding insignificant data has a minimal effect on image quality

Forward Transform

For a subimage $g(x, y)$ of size $n \times n$ whose forward discrete transform $T(u, v)$ is expressed as

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

for $u, v = 0, 1, 2, ..., n\text{-}1$.

Inverse Transform

Given $T(u,v)$, the subimage $g(x,y)$ similarly can be obtained using the inverse discrete transform

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

for $x, y = 0,1,2,...,n\text{-}1$.

Typical Image Transforms

Two main types:

- **Orthogonal Transform**:

  e.g. Walsh-Hadamard, DCT, DFT

- **Subband Transform**:

  e.g. Wavelet transform

Orthogonal Transform

Orthogonal matrix: W

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix}$$

➔ C=W $\underline{d}$

- Reduce redundancy
- Isolate frequencies

Block Transform Coding: Walsh-Hadamard Transform (WHT)

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

Generated Recursively

$$H_1 = [1] \qquad H_2 = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Block Transform Coding
Discrete Cosine Transform (DCT)

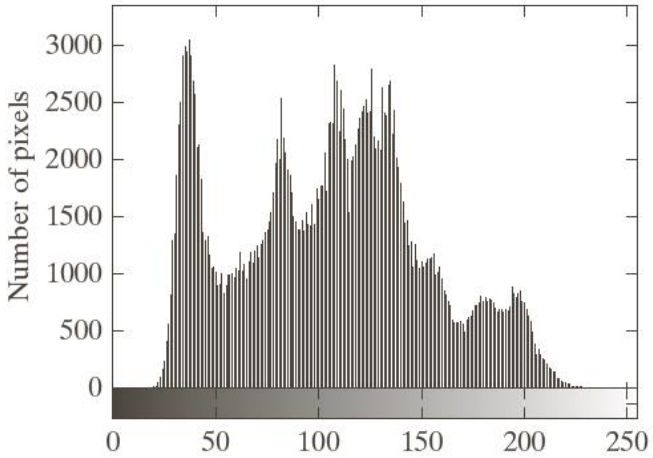
Same & Separable

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

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

$$\text{where, } \alpha(u \text{ or } v) = \begin{cases} \sqrt{\frac{1}{n}} & \text{for } u,v = 0 \\ \sqrt{\frac{2}{n}} & \text{for } u,v = 1,2,\ldots,n-1 \end{cases}$$

Example



a b

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

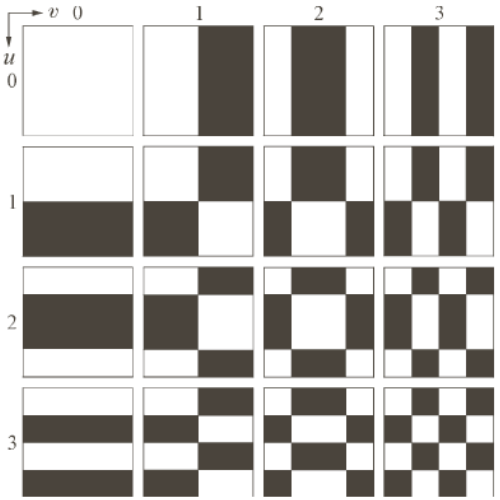


FIGURE 8.22 Walsh-Hadamard basis functions for $n = 4$. The origin of each block is at its top left.

32 coefficients with maximum magnitudes shown

Walsh-Hadamard

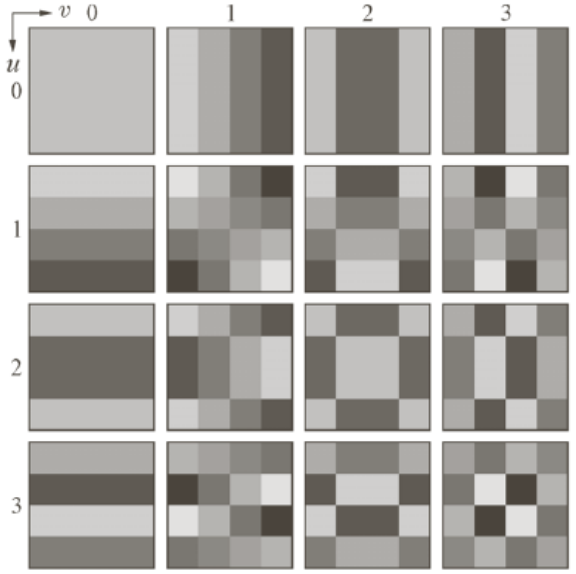


FIGURE 8.23 Discrete-cosine basis functions for $n = 4$. The origin of each block is at its top left.

DCT

Which Transform to use?

- DFT vs. DCT vs. WHT ??
- The choice of a particular transform in a given application depends on the amount of reconstruction error that can be tolerated and computational resources available.
- All three transforms have basis images that are fixed or input independent → Can be easily pre-computed prior to the compression process (unlike KLT – data dependent)
  - A good property computation-wise

DFT vs. DCT vs. WHT

- In an experiment with all 3 transforms on the Lena image, dividing the image into subimages of size 8x8, and truncating 50% of the resulting coefficients, then taking the inverse transform to reconstruct the original image:
  - All three methods show little visual impact on the quality of the reconstructed image
  - The RMS (root mean square) error are 2.32, 1.78 and 1.13 intensities for DFT, WHT and DCT, respectively.
- Hence, DCT is quantitatively proven to be the most optimal transform (the least error) for use in block transform coding

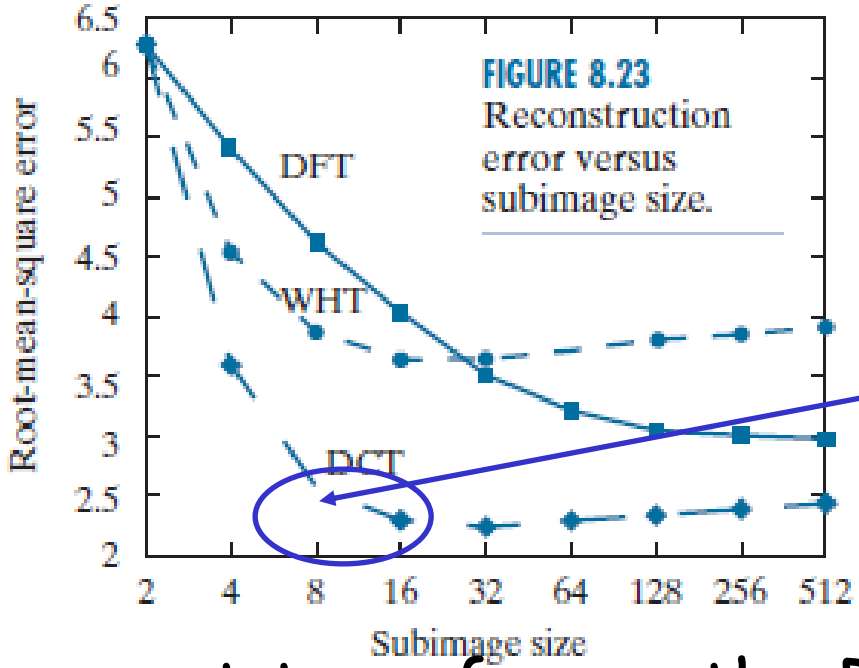
Example

- In each case, 50% of the resulting coefficients were truncated and taking the inverse transform of the truncated coefficients arrays.

a b c
d e f

FIGURE 8.22 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).

Subimage Size Selection



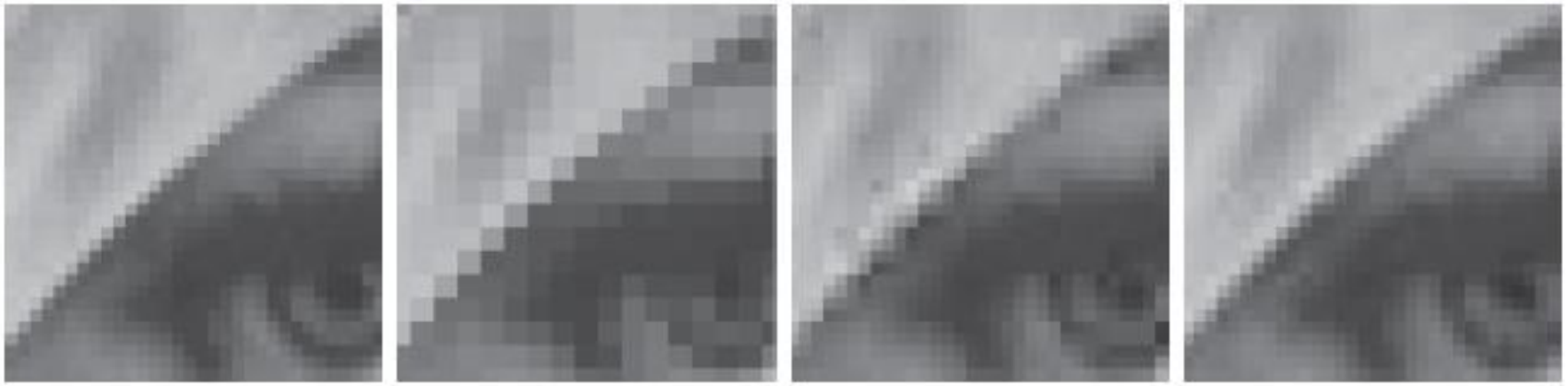
- Quantization made by truncating 75% of transform coefficients

Size 8x8 is enough

DCT is the best

- For most transforms, the RMS error decreases when larger subimage sizes are used, but they will reach their respective optimum points
- The DCT is superior over the other transforms at any given subimage size!

Subimage Size Selection



a b c d

FIGURE 8.24 Approximations of Fig. 8.24(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).

DCT as Choice of Transform

DCT, in comparison with the other transforms

- Has superior information packing ability than DFT and WHT
- Minimizes the amount of visible blocking artifacts compared to other transforms
- Provides a good compromise or balance between information packing ability and computational complexity

DCT vs. DFT Coding

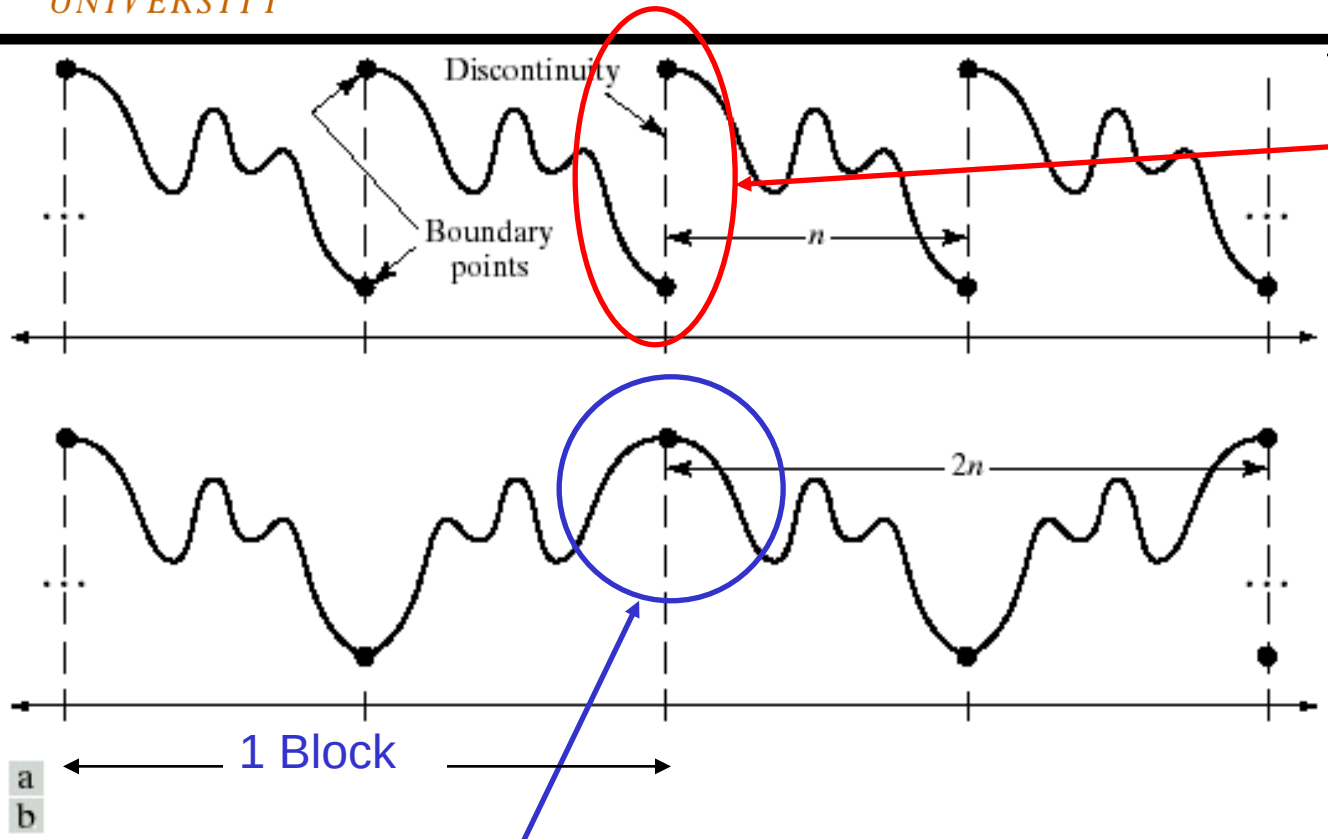


FIGURE 8.32 The periodicity implicit in the 1-D (a) DFT and (b) DCT.

DFT coefficients have abrupt changes at boundaries of blocks

- **Advantage of DCT over DFT:** DCT coefficients are more continuous at boundaries of blocks.

Discrete Fourier Transform (DFT)

- **Use** $e^{j\theta} = \cos\theta + j\sin\theta$ as its basis functions
- Fast Fourier Transform (FFT) - $O(n \log n)$
- Not so popular in image compression because
  - Performance is not good enough
  - Computational load for complex number is heavy

Discrete Cosine Transform (DCT)

- Uses cosine function as its basis function
- Performance approaches KLT
- Fast algorithm exists
  - Can be implemented by $2n$ points FFT (by making data symmetric)
- The periodicity implied by DCT implies that it causes less blocking effect than DFT
- Most popular in image compression application
- Used in most popular standards:
  - JPEG, H.26x, MPEG

Bit Allocation - Quantizer

- Transform of image efficiently puts most of the information into relatively few coefficients
- The overall process of truncating, quantizing, and coding the coefficients of a transformed subimage is commonly called **bit allocation**
- **Quantizer: Bit allocation**
  - Determines the number of bits for coding each coefficient
  - More bits used for lower-frequency components

Quantization Process: Bit Allocation

- To assign different numbers of bits to represent transform coefficients based on importance of each coefficient:

  - More important coefficients
    → Assign larger number of bits

  - Less important coefficients
    → Assign a small number of bits or not assign at all

Bit Allocation (cont.)

- Two ways of truncating coefficients
  - **Zonal coding** - Retained coefficients are selected on the basis of maximum variance
  - **Threshold coding** - Select according to the maximum magnitude of the coefficients

Zonal coding

- **Principle**: According to information theory, the transform coefficients with maximum variance carry most image information and should be retained in the coding process.

- The process can be viewed as multiplying each transformed coefficient by the corresponding element in a zonal mask

Zonal Coding Example

a b

FIGURE 8.26
A typical (a) zonal mask, (b) zonal bit allocation,

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

(a) Zonal mask

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

(b) Zonal bit allocation

Coefficients of maximum variance usually occurs at the origin of the image transform

Threshold Coding

- **Principle:** The transform coefficients with **largest magnitude** contribute most to reconstructed subimage quality
- Zonal Coding uses fixed mask
- Threshold Coding depends on image → Adaptive
  - Most often used in adaptive transform coding approach
  - Used in JPEG - due to its simplicity

Threshold coding: Three Methods

1. **A single global threshold for all subimages**
   - Compression ratio differs from image to image
2. **A different threshold for each subimage**
   - Called N-largest coding
   - Same number of coefficients is discarded
   - The coding rate is constant & known beforehand
3. **Threshold varies as a function of the location of each coefficient within the subimage** – results in a variable code rate and replace $T(u, v)$ with

$$\hat{T}(u,v) = round\left[\frac{T(u,v)}{Z(u,v)}\right]$$

$Z(u, v)$: An element of a normalization array Z

Threshold Coding

$$\hat{T}(u,v) = round\left[\frac{T(u,v)}{Z(u,v)}\right]$$

Normalization array

$$Z = \begin{bmatrix} Z(0,0) & Z(0,1) & \dots & Z(0,n-1) \\ Z(1,0) & \dots & \dots & \dots \\ \dots. & \dots & \dots & \dots \\ Z(n-1,0) & \dots & \dots & Z(n-1,n-1) \end{bmatrix}$$

Denormalization (Approximation)

$$\dot{T}(u,v) = \hat{T}(u,v)Z(u,v);$$

WRIGHT STATE UNIVERSITY

Threshold Coding Example

c d

FIGURE 8.26

(c) threshold mask, and (d) thresholded coefficient ordering sequence. Shading highlights the coefficients that are retained.

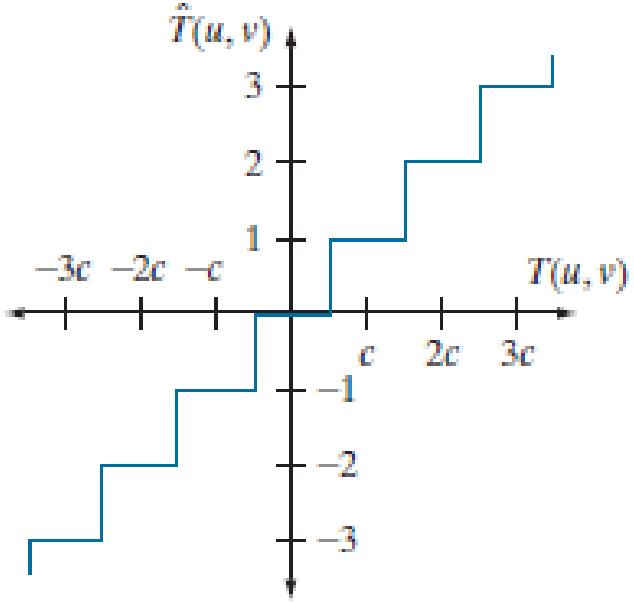
1	1	0	1	1	0	0	0
1	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c) Threshold mask

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

(d) Thresholded coefficient Ordering → Zig-zag scan

Threshold Coding (JPEG)



Z-Matrix

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 8.27 (a) A threshold coding quantization curve [see Eq. (8-28)]. (b) A typical normalization matrix.

A typical normalization matrix $Z(u, v)$ used by JPEG:
"Heuristically determined based on perceptual or psychological Importance"

Threshold Coding & Zonal Coding

RMSE = 4.5

RMSE = 6.5

a b
c d

FIGURE 8.25 Approximations of Fig. 8.9(a) using 12.5% of the DCT coefficients: (a)–(b) threshold coding results; (c)–(d) zonal coding results. The difference images are scaled by 4.

Threshold Coding

19:1

12:1

30:1

49:1

182:1

85:1

a b c
d e f

FIGURE 8.28 Approximations of Fig. 8.9(a) using the DCT and normalization array of Fig. 8.27(b): (a) **Z**, (b) 2**Z**, (c) 4**Z**, (d) 8**Z**, (e) 16**Z**, and (f) 32**Z**.

Facts about JPEG Compression

- JPEG → Joint Photographic Experts Group
- Used on 24-bit color files
- Works well on photographic images
- Although it is a lossy compression technique, it produces excellent quality image with high compression rates

Facts about JPEG Compression

- It defines three different coding systems:

1. A lossy baseline coding system, adequate for most compression applications

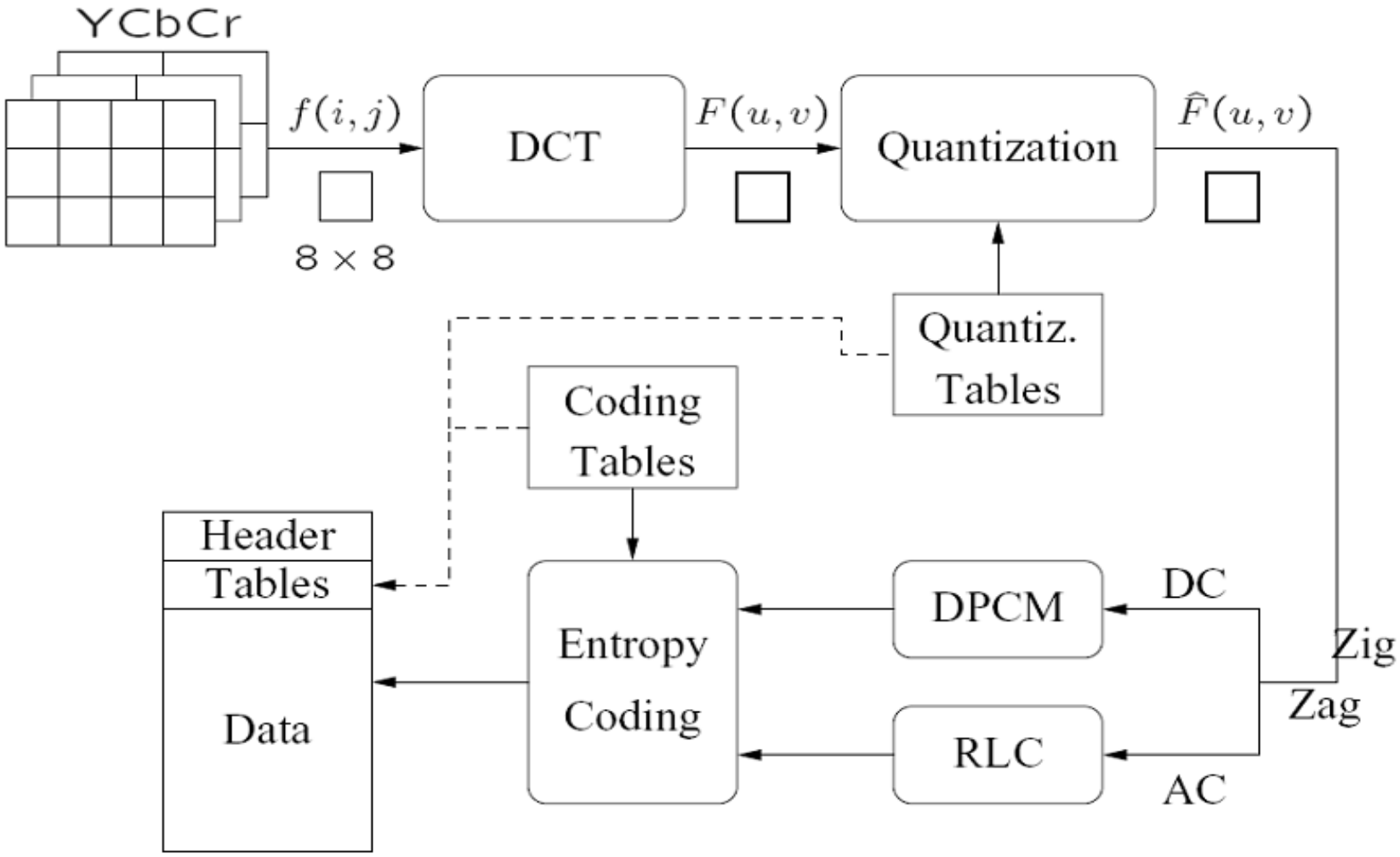
2. An extended coding system for greater compression, higher precision, or progressive reconstruction applications

3. A lossless independent coding system for reversible compression

WRIGHT STATE UNIVERSITY

JPEG Block Diagram

Y': luma component; C_B, C_R: blue-difference and red-difference chroma components.



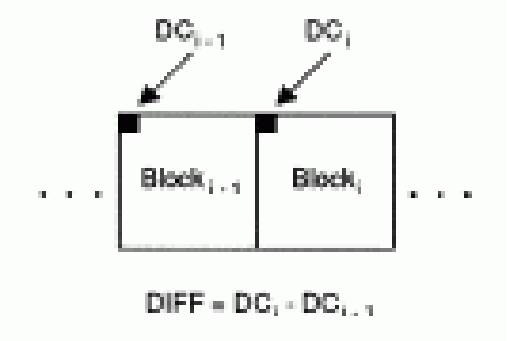
JPEG Sequence Example…

1. The image is subdivided into pixel blocks of size 8×8 , left to right, top to bottom
2. Level shift by subtracting the quantity 2^{n-1}, where 2^n is the maximum number of gray levels
   - Will contain negative values
3. Compute the 2D DCT of block
4. Quantize with the appropriate quantization table
5. Zigzag scan

JPEG Steps

6. Encoding

- DC coefficient encoding: (Differential PCM)
  - DC component is typically the largest → many bits
- Compute difference between current DC coefficient and that of the previously encoded sub-image



- AC coefficient encoding:
  = 'the number of zero-valued coefficients preceding the nonzero coefficient to be coded'
  +
  'the magnitude category of the nonzero coefficient'

JPEG Example

WRIGHT STATE UNIVERSITY

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

Subtracting - 128

2^{n-1} Shift Down

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

$x \rightarrow$ $y \downarrow$

8X8 block of pixel values taken from original image

DC Component

2D DCT

low frequency

high frequency

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

low frequency

Scalar quantisation

high frequency

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

Example: Rounding –415 (the DC coefficient) to the nearest integer

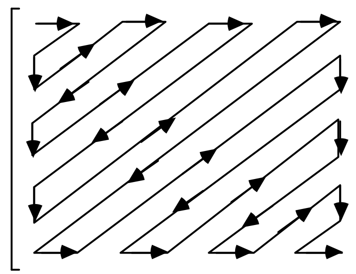
$$\text{round}\left(\frac{-415}{16}\right) = \text{round}\left(-25.9375\right) = -26$$

(16 is the value of the first pixel from the quantization matrix **Z – see slide-98**)

−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

- Assume the DC coefficient of the transformed and quantized 8x8 subimage to the immediate left is 17
- DPCM difference is [-26 – (-17)] = -9

zig-zag scan



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

- Truncate 0s → i.e., discard unimportant high frequency components → Compression!
- How many zeros truncated depends on each sub-image → Adaptive

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

Huffman coding (uses Table look-up): coded output – 92 bits

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

JPEG Decompression (Decoding)

Denormalization (multiply by Z matrix)

−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

−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

IDCT

−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

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

2^{n-1} Shifting Up

Discrete Cosine Transform

- The DCT transforms the data from the spatial domain to the frequency domain.
- The spatial domain shows the amplitude of the color as you move through space
- The frequency domain shows how quickly the amplitude of the color is changing from one pixel to the next in an image file.

DCT

- The frequency domain is a better representation for the data because it makes it possible to separate out – and throw away – information that is not very important to human perception

- The human eye is not very sensitive to high frequency changes – especially in photographic images

- So, the high frequency data can, to some extent, be discarded

DCT

- The color amplitude information can be thought of as a wave (in two dimensions)
- Essentially, we are decomposing the wave into its component frequencies.
- For the 8 x 8 matrix of color data, there is an 8 x 8 matrix of coefficients for the frequency components.

Quantize the Coefficients Computed by the DCT

- The DCT is lossless in that the reverse DCT gives back exactly the initial information (ignoring the rounding error that results from using floating point numbers.)
- The values from the DCT are initially floating-point.
- They are changed to integers by quantization.

Quantization

- Quantization: Divide each coefficient by an integer between 1 and 255 and rounding off.
- Quantization Table: Chosen to reduce the precision of each coefficient to no more than necessary.
- The quantization table is carried along with the compressed file.

Arrange in "zigzag" order

- This is done so that the coefficients are in order of increasing frequency.
- The higher frequency coefficients are more likely to be 0 (zero-valued) after quantization.
- This improves the compression of run-length encoding.
- Use run-length encoding and Huffman coding

EXAMPLE 8.17: JPEG baseline coding and decoding.

- Consider compression and reconstruction of an 8 x 8 Subimage
- Original image is 8-bits, i.e., 256 possible intensity values

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

JPEG Coding Step-1: Level-Shifting Subtract by 2^7=-128

EXAMPLE 8.17:
JPEG baseline coding and decoding.

−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 Step-2: Apply DCT for n=8

- This is the DCT-Transformed image

EXAMPLE 8.17: JPEG baseline coding and decoding.

$$T(u,v) = \begin{bmatrix} -415 & -29 & -62 & 25 & 55 & -20 & -1 & 3 \\ 7 & -21 & -62 & 9 & 11 & -7 & -6 & 6 \\ -46 & 8 & 77 & -25 & -30 & 10 & 7 & -5 \\ -50 & 13 & 35 & -15 & -9 & 6 & 0 & 3 \\ 11 & -8 & -13 & -2 & -1 & 1 & -4 & 1 \\ -10 & 1 & 3 & -3 & -1 & 0 & 2 & -1 \\ -4 & -1 & 2 & -1 & 2 & -3 & 1 & -2 \\ -1 & -1 & -1 & -2 & -1 & -1 & 0 & -1 \end{bmatrix}$$

Step-3: Quantize using Normalization Array and rounding

$$Z(u,v) =$$

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

$$\hat{T}(0,0) = \text{round}\left[\frac{T(0,0)}{Z(0,0)}\right] = \text{round}\left[\frac{-415}{16}\right] = -26$$

Notice all the zero coefficients → Don't need to send!

$$\hat{T}(u,v) =$$

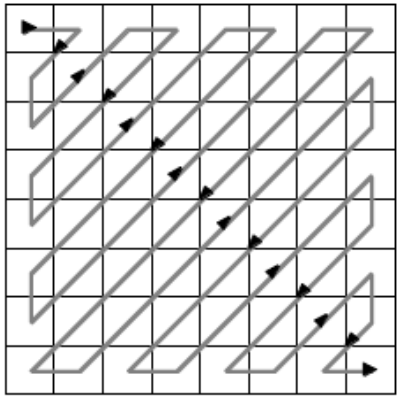
−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	1	2	−1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

WRIGHT STATE UNIVERSITY

Step-4: Reorder using zig-zag ordering pattern

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

Like this →



[−26 −3 1 −3 −2 −6 2 −4 1 −4 1 1 5 0 2 0 0 −1 2 0 0 0 0 0 −1 −1 EOB]

- Assuming the DC coefficient of the transformed and quantized subimage to its immediate left was -17.
- Use tables A.3, A.4 & A.5 for DC and AC Codes
- DPCM difference is [-26 – (-17)] = -9.

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

Table A.3

TABLE 8.17
JPEG coefficient coding categories.

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, …, −4, 4, …, 7	3	3
−15, …, −8, 8, …, 15	4	4
−31, …, −16, 16, …, 31	5	5
−63, …, −32, 32, …, 63	6	6
−127, …, −64, 64, …, 127	7	7
−255, …, −128, 128, …, 255	8	8
−511, …, −256, 256, …, 511	9	9
−1023, …, −512, 512, …, 1023	A	A
−2047, …, −1024, 1024, …, 2047	B	B
−4095, …, −2048, 2048, …, 4095	C	C
−8191, …, −4096, 4096, …, 8191	D	D
−16383, …, −8192, 8192, …, 16383	E	E
−32767, …, −16384, 16384, …, 32767	F	N/A

DC value -9 belongs to this category

Table A.4

Huffman code is used

TABLE 8.18 JPEG default DC code (luminance).

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Remaining 4 bits from LSBs of the negative difference value minus 1 → See textbook (page-647 of 4th Ed. or page-582 of 3rd Ed.)

Table A.5

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

TABLE 8.19
JPEG default AC code (luminance) *(continues on next page).*

Huffman code is used

3rd Edition

Table A.5 (cont.)

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

3rd Edition

Decoding Step-1:Table Look-up for Huffman Code

−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

Decoding Step-2: Denormalize

$$\dot{T}(u,v) = \hat{T}(u,v)Z(u,v); \qquad \dot{T}(0,0) = \hat{T}(0,0)Z(0,0) = (-26)(16) = -416, \text{ etc.}$$

−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

Decoding Step-3: Inverse DCT

−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

Decoding Step-4: Level-Shifting Add by 2^7=+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

JPEG Errors: -14 to 11

RMS Error due to Compression, Quantization and Reconstruction = 5.8 intensity levels

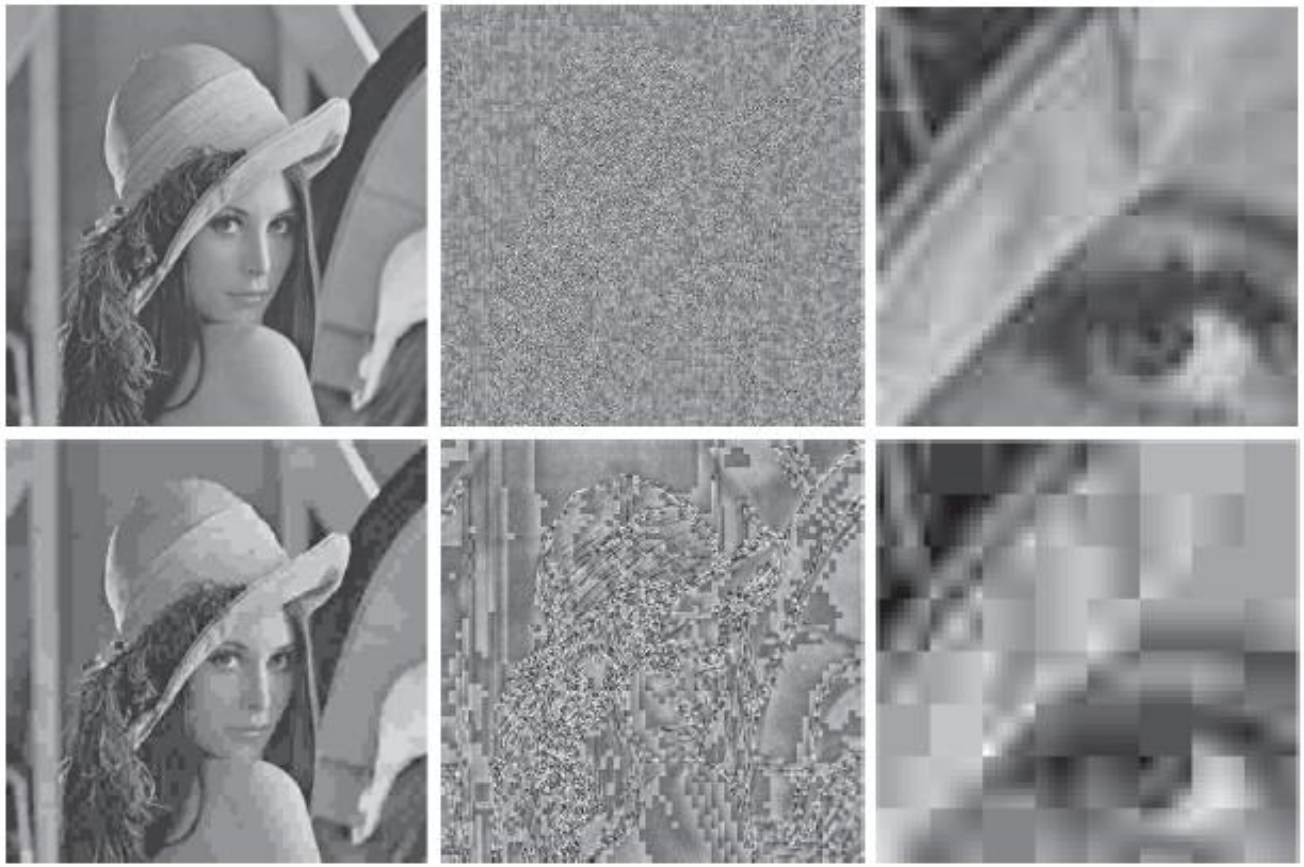
−6	−9	−6	2	11	−1	−6	−5
7	4	−1	1	11	−3	−5	3
2	9	−2	−6	−3	−12	−14	9
−6	7	0	−4	−5	−9	−7	1
−7	8	4	−1	6	4	3	−2
3	8	4	−4	2	6	1	1
2	2	5	−1	−6	0	−2	5
−6	−2	2	6	−4	−4	−6	10

JPEG Approximation Results

Compression Ratio
RMS Errors

25:1
5.4

52:1
10.7



a b c
d e f

FIGURE 8.29 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.

8.10 Predictive Coding

- Images and videos contain a large amount of spatial and temporal redundancy

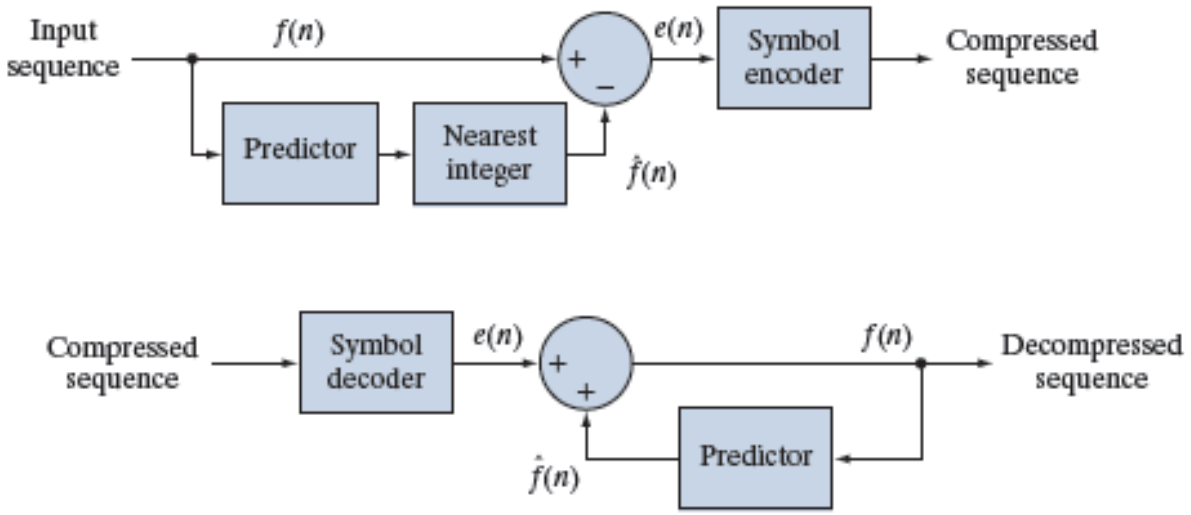
- Pixels in an image or video frame should be reasonably predictable by other pixels in
  - The same image (**intra-frame prediction**)
  - Adjacent frames (**inter-frame prediction**)

Predictive Coding

- Predict new pixel value based on the value of the m previous pixel values
- Code only the difference between prediction and actual value → known as prediction error

a
b

FIGURE 8.30
A lossless predictive coding model: (a) encoder; (b) decoder.



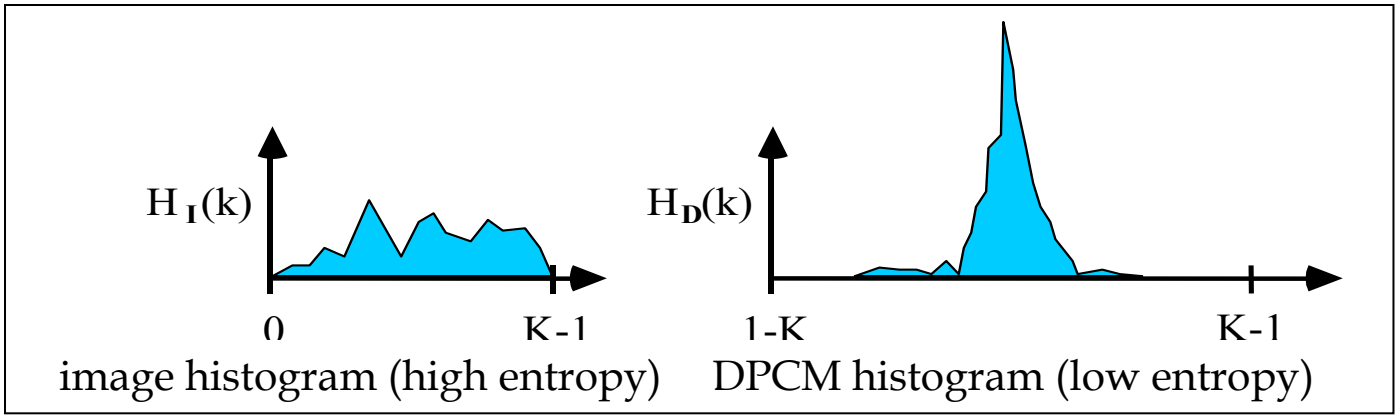
$$f_{n,\,pred} = round\left(a_1 f_{n-1} + a_2 f_{n-2} + \ldots + a_m f_{n-m}\right)$$

$$e_n = f_n - f_{n,\,pred}$$

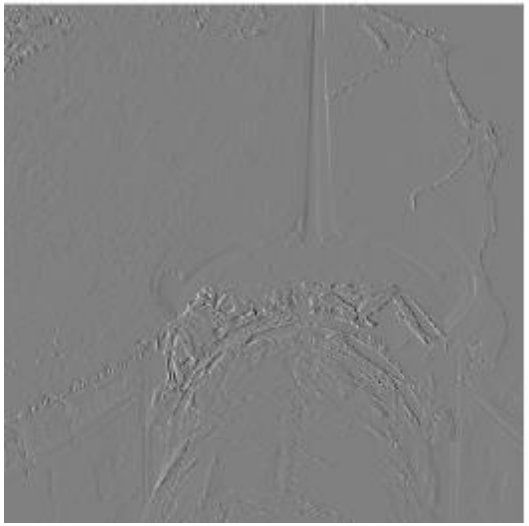
Predictive Coding & DPCM

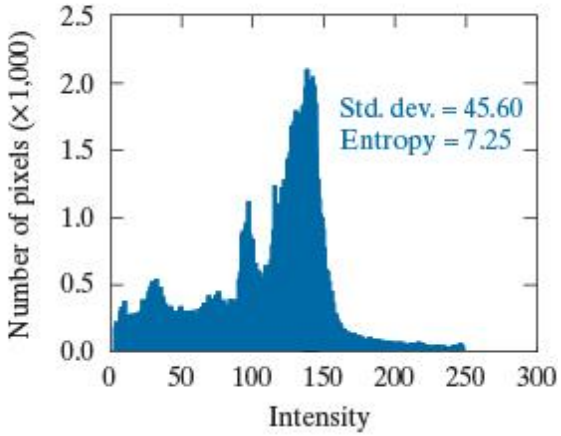
- Exploit Spatial/Interpixel Redundancy
  - Image pixels are highly correlated (dependent)
  - Predict the image pixels to be coded from those already coded
    - Differential Pulse-Code Modulation (DPCM)
    - Simplest form: Code the difference between pixels

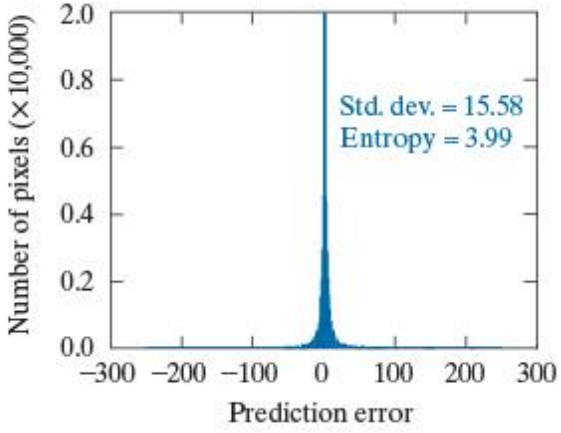
Original pixels:	DPCM:
82, 83, 86, 88, 56, 55, 56, 60, 58, 55, 50, ……	82, 1, 3, 2, -32, -1, 1, 4, -2, -3, -5, ……



Example: Predictive Coding







- Single pixel Prediction

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

a b
c d

FIGURE 8.31
(a) A view of the Earth from an orbiting space shuttle. (b) The intensity histogram of (a). (c) The prediction error image resulting from Eq. (8-33). (d) A histogram of the prediction error. (Original image courtesy of NASA.)

Inter-frame Prediction Coding (Motion Compensation)

- Inter-Frame: Prediction performed between frames
- Similar to intra-frame coding, but instead of within the same image, the prediction coding is performed between frames
- P-Frame: Based on previous (Predictive) frame
- B-Frame: Based on subsequent (backward Predictive) frame

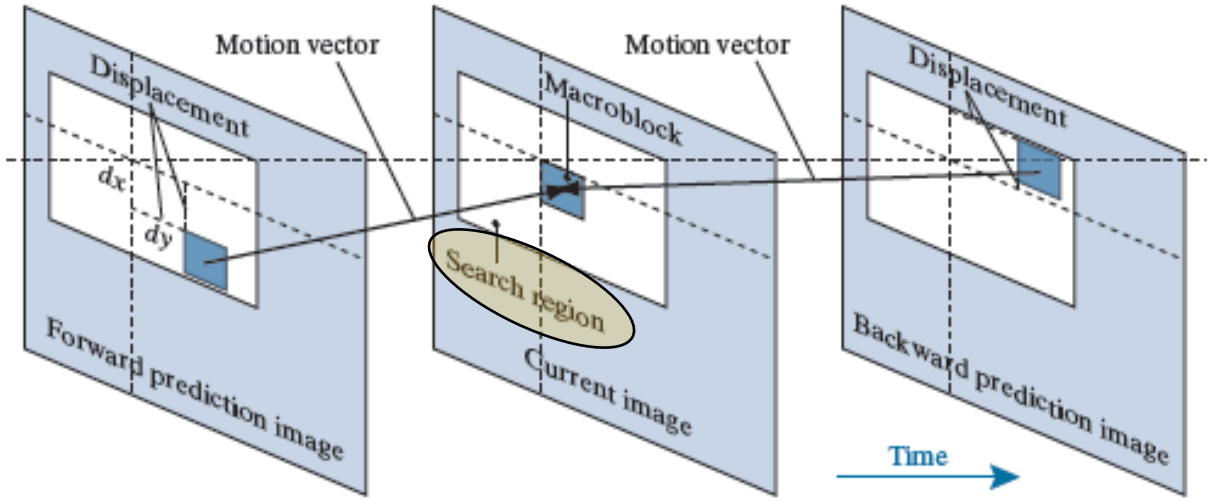


FIGURE 8.33 Macroblock motion specification.

Motion Compensation

- For a sub-image f, find the sub-image p that is most similar to f (**block matching** $m \times n$)
- Error residual between the two sub-images

$$e(x,y) = f(x+i, y+j) - p(x+i+dx, x+j+dy)$$

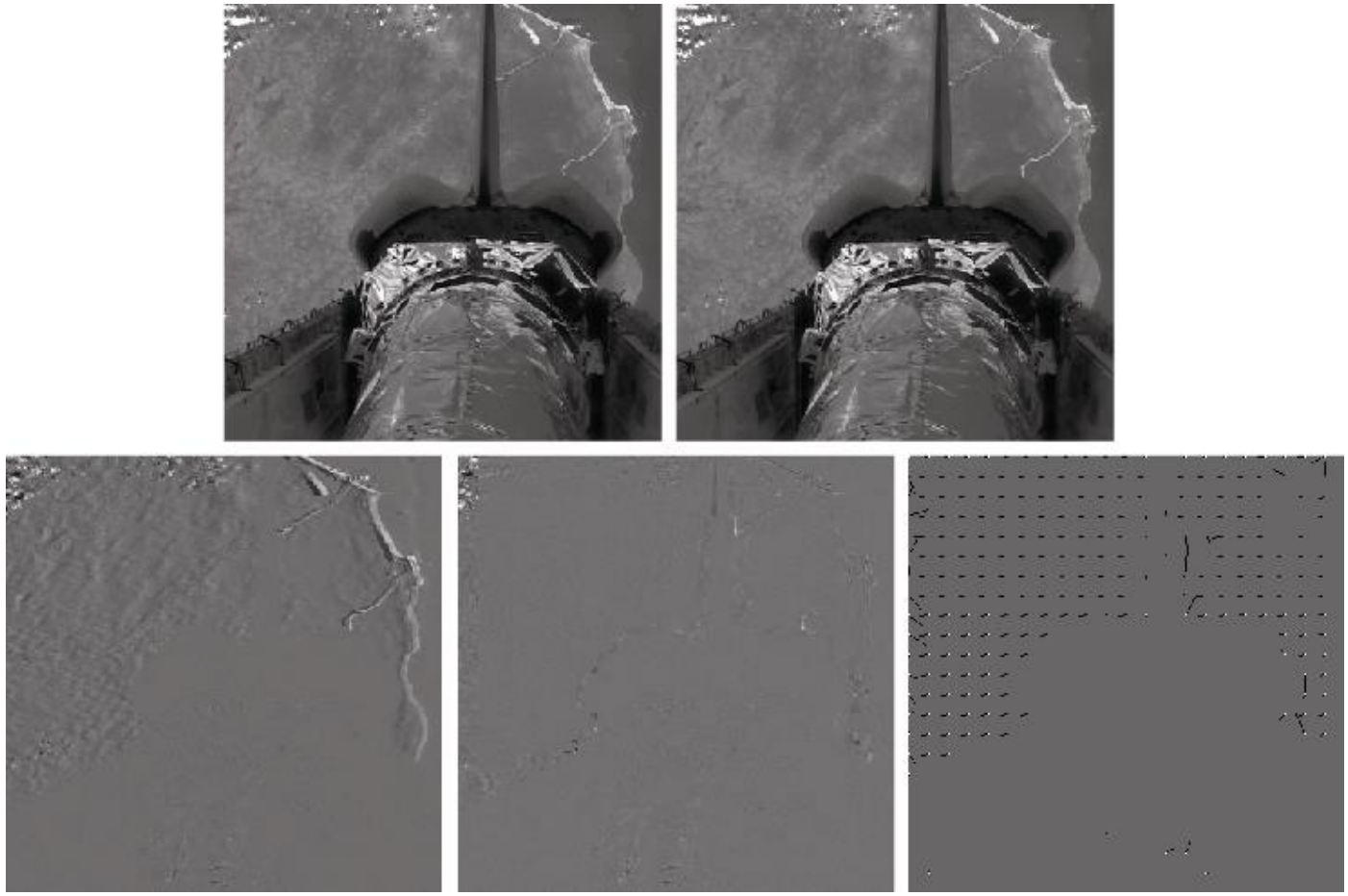
- Find the sub-image that minimizes the Mean Absolute Distortion (MAD) i.e., closest or minimum of

$$MAD(x,y) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \left| f(x+i, y+j) - p(x+i+dx, x+j+dy) \right|$$

Motion-Compensated Predictive Coding

- Usually performed on the Intensity channel
- Movement is encoded in a "Motion Vector"
- Encode & store vector (dx, dy) +/- 8 to +/- 64
- Transform prediction error residual (highly correlated) with image transform
- Quantize
- Subpixel motion compensation for further improvement → Added computation

Results using Inter-frame Prediction Coding

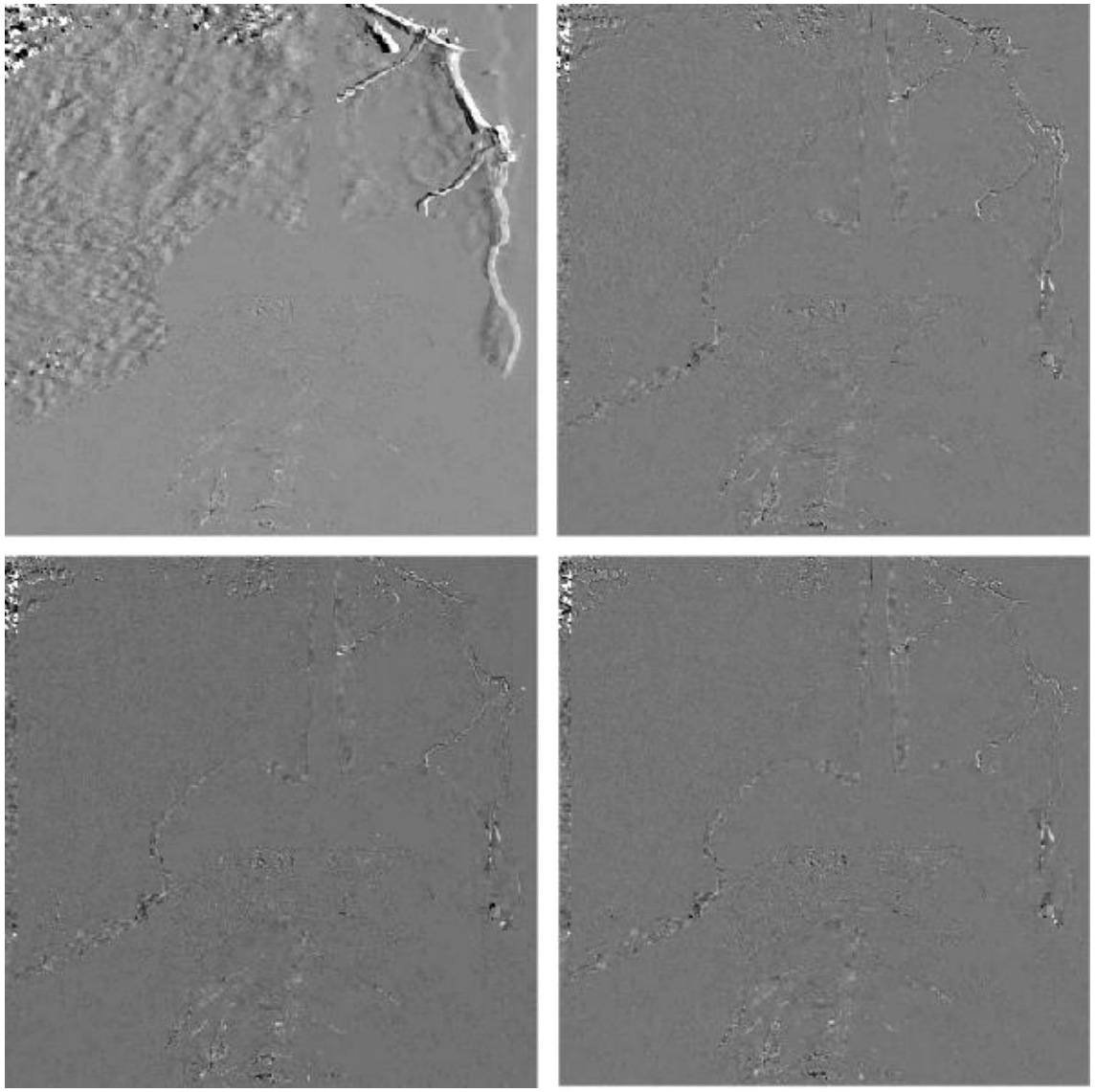


a b
c d e

FIGURE 8.34 (a) and (b) Two views of Earth that are thirteen frames apart in an orbiting space shuttle video. (c) A prediction error image without motion compensation. (d) The prediction residual with motion compensation. (e) The motion vectors associated with (d). The white dots in (e) represent the arrow heads of the motion vectors that are depicted. (Original images courtesy of NASA.)

Sub-Pixel Motion Compensation Prediction

- Subpixel images are created by interpolation
- Adds to cost



a b
c d

FIGURE 8.35
Sub-pixel motion compensated prediction residuals: (a) without motion compensation; (b) single pixel precision; (c) ½ pixel precision; and (d) ¼ pixel precision. (All prediction errors have been scaled to the full intensity range and then multiplied by 2 to increase their visibility.)

Video = Pictures in Motion

- Capturing video
  - Frame by frame → Image sequence
  - Image sequence: A 3-D signal
    - 2 spatial dimensions & time dimension
    - Continuous $I(x, y, t)$ → Discrete $I(m, n, t_k)$
- Encode digital video
  - Simplest way → Compress each frame image individually
    - e.g., "motion-JPEG (MPEG)"
    - Only spatial redundancy is explored and reduced
  - How about temporal redundancy? Is differential coding good?
    - Pixel-by-pixel difference could still be large due to motion

→ Need better prediction

Video Coding in Video Compression Standards

TABLE 8.12
Predictive coding in video compression standards.

Parameter	H.261	MPEG-1	H.262 MPEG-2	H.263	MPEG-4	VC-1 WMV-9	H.264 MPEG-4 AVC
Motion vector precision	1	½	½	½	¼	¼	¼
Macroblock sizes	16×16	16×16	16×16 16×8	16×16 8×8	16×16 8×8	16×16 8×8	16×16 16×8 8×8 8×4 4×8 4×4
Transform	8×8 DCT	8×8 DCT	8×8 DCT	8×8 DCT	8×8 DCT	8×8 8×4 4×8 4×4 Integer DCT	4×4 8×8 Integer
Interframe predictions	P	P, B	P, B	P, B	P, B	P, B	P, B
I-frame intra-predictions	No	No	No	No	No	No	Yes

"B" (bidirectional) and "P" (Predictive) frames

Motion-Compensated Video Coding System

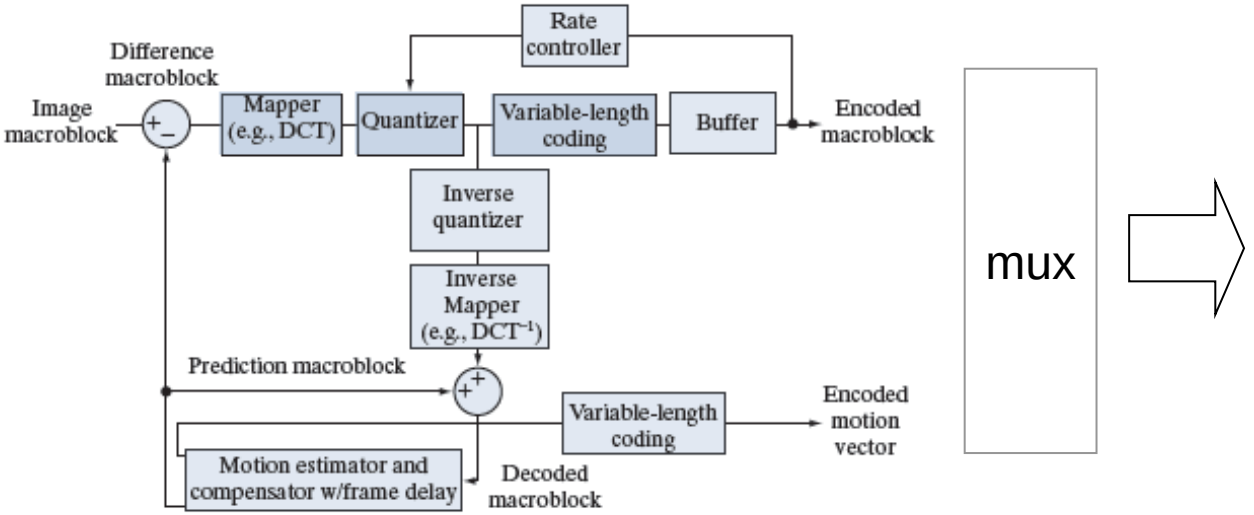
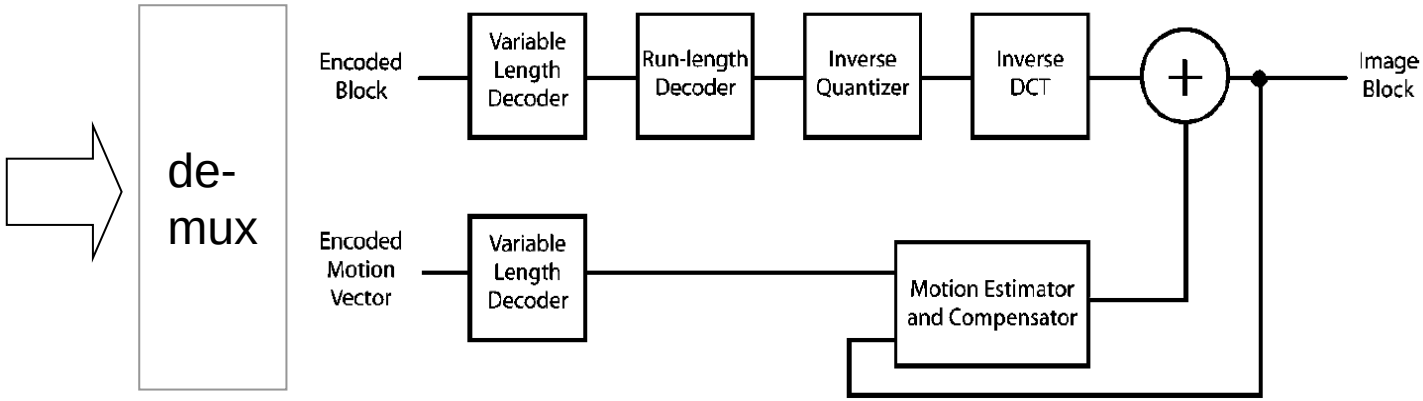
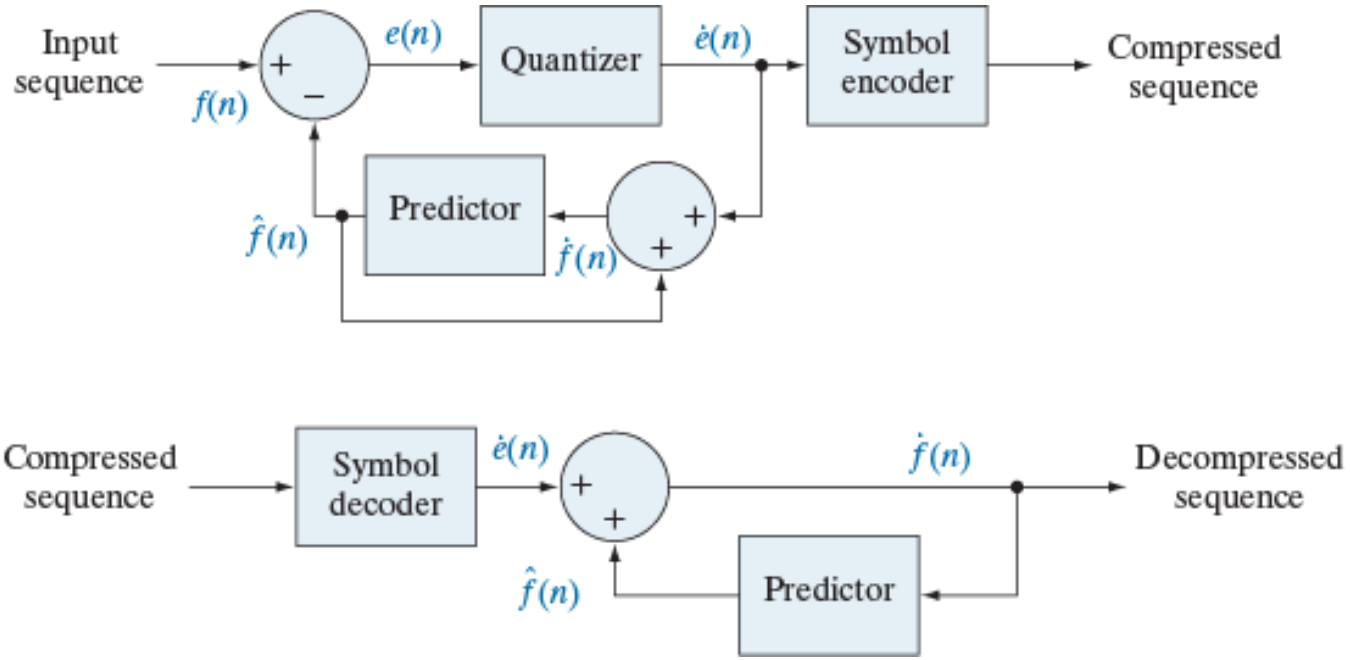


FIGURE 8.36
A typical motion compensated video encoder.



Lossy Predictive Coding Model



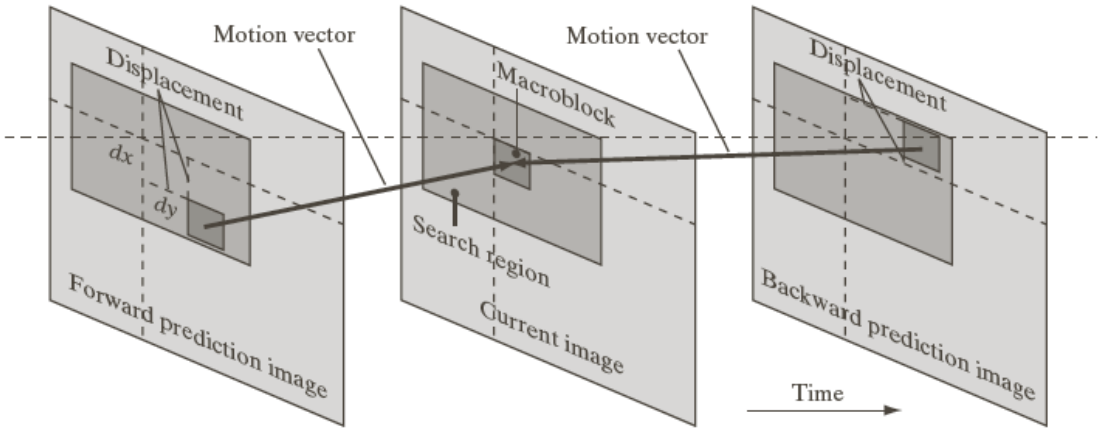
a
b

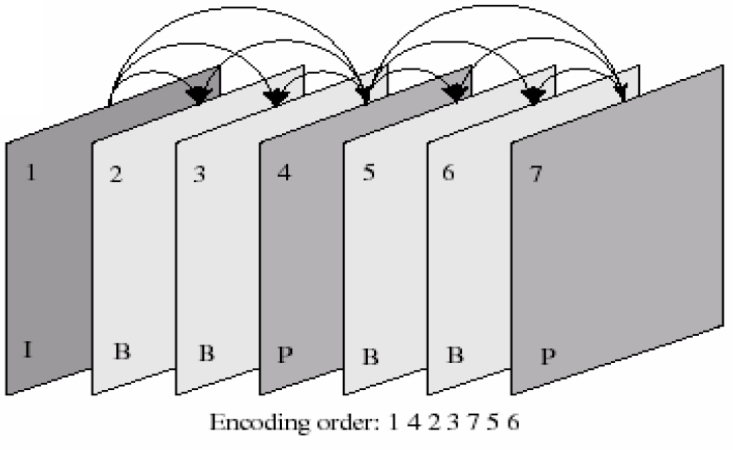
FIGURE 8.38
A lossy predictive coding model:
(a) encoder;
(b) decoder.

- The Quantizer introduces Loss

Ideas in Video Coding Systems

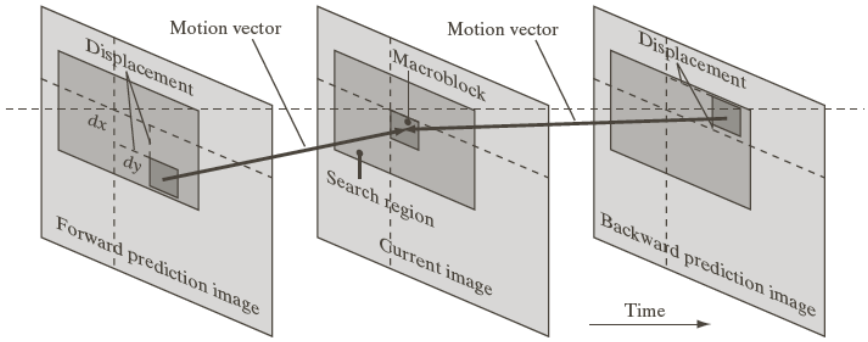
- Work on each macroblock (MB) (16x16 pixels) independently for reduced complexity
  - Motion compensation done at the MB level
  - DCT coding at block level (8x8 pixels) on prediction-error MB

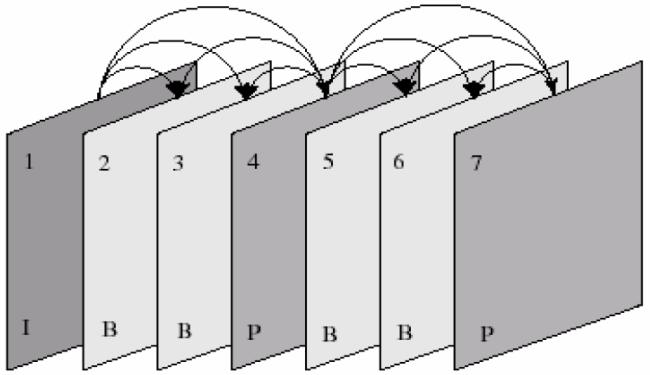




Representing Motion

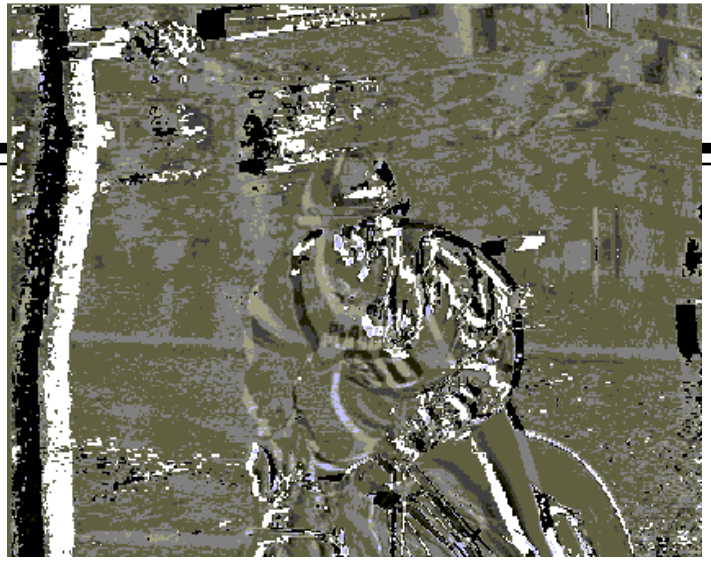
- Predict a new frame from a previous frame and only code the prediction error -- *Inter* prediction on "B" (bidirectional) and "P" (Predictive) frames
- Predict a current block from previously coded blocks in the same frame --- *Intra* prediction (introduced in the latest standard H.264)
- Prediction errors have smaller energy than the original pixel values and can be coded with fewer bits
  - DCT on the prediction errors
- Those regions that cannot be predicted well are coded directly using DCT --- Intra coding without intra-prediction





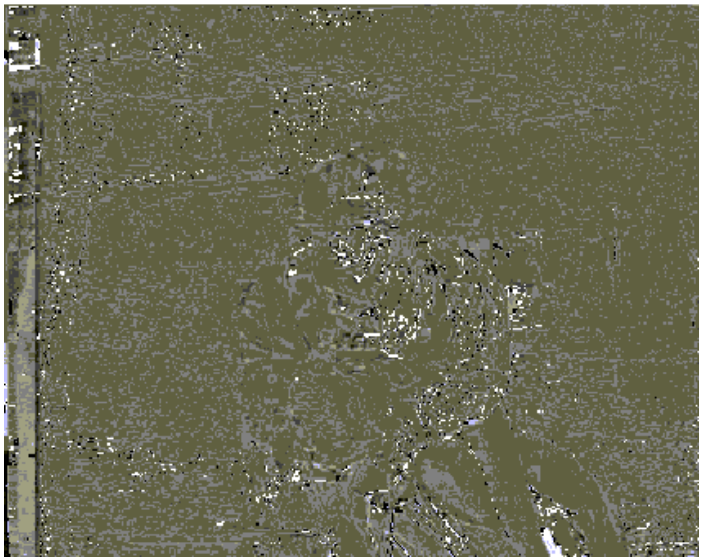
Encoding order: 1 4 2 3 7 5 6

"Horse ride"



Pixel-wise difference w/o motion compensation

Motion estimation



Residue after motion compensation

Motion Compensation

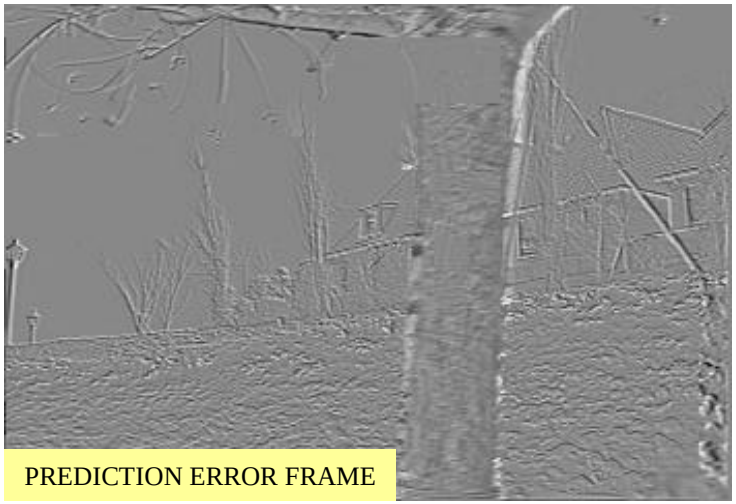
- Help reduce temporal redundancy of video

Revised from R.Liu Seminar Course '00 @ UMD

PREVIOUS FRAME

CURRENT FRAME

PREDICTED FRAME



PREDICTION ERROR FRAME

Motion Estimation

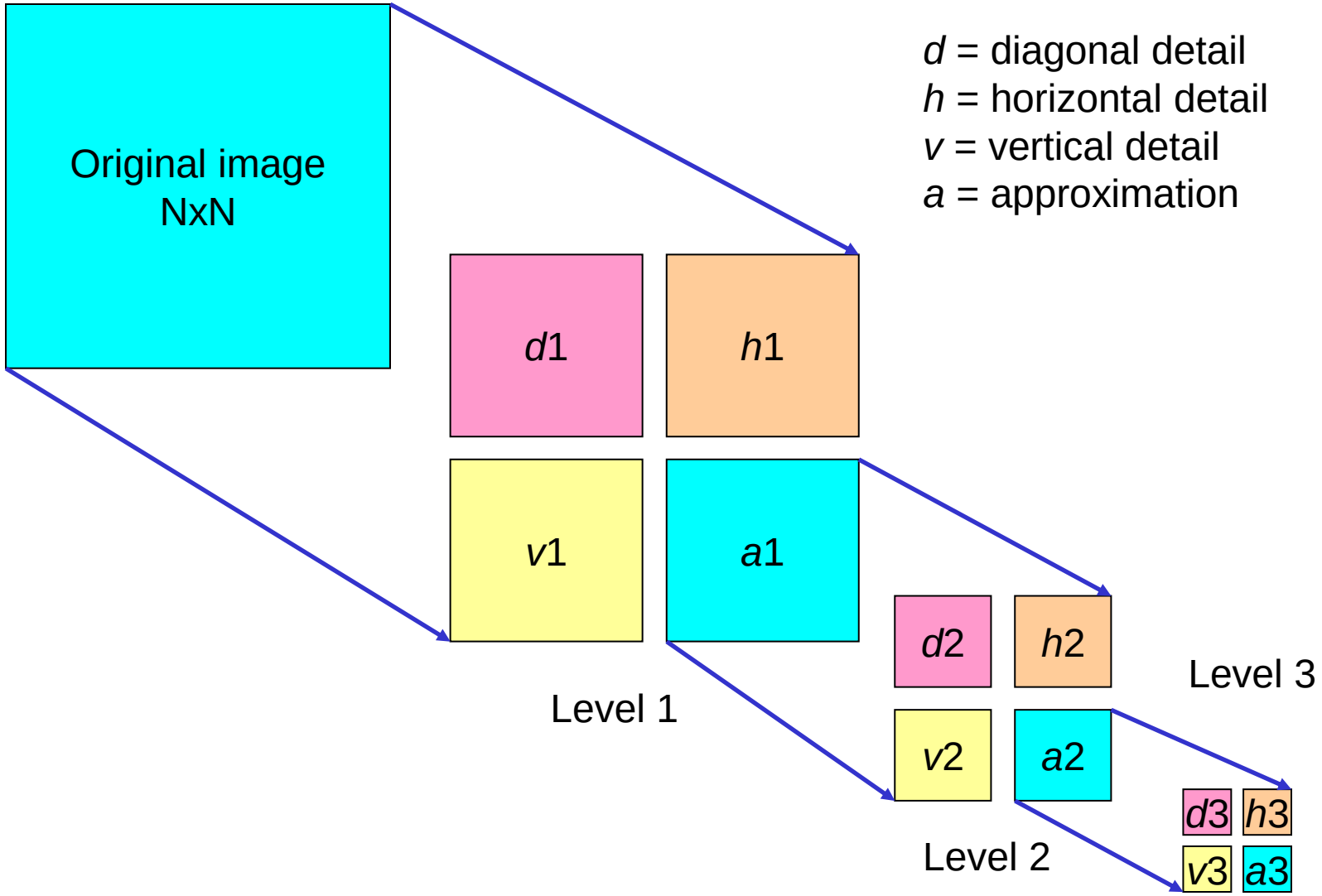
- Help understanding the content of image sequence
- Help exploit temporal redundancy of video
  - For better compression
- Stabilizes video by detecting and removing small, noisy global motions
  - Used for automatic stabilizer in camcorder
- A hard problem in general!

JPEG 2000

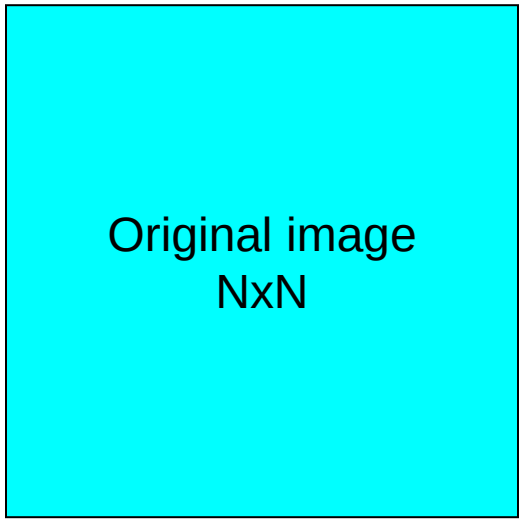
- Better image quality/coding efficiency, especially for low bit-rate compression performance
  - DWT – Discrete Wavelet Transform
  - Bit-plane coding (EBCOT-Embedded Block-Coding with Optimal Truncation)
  - Flexible block sizes
  - Etc.

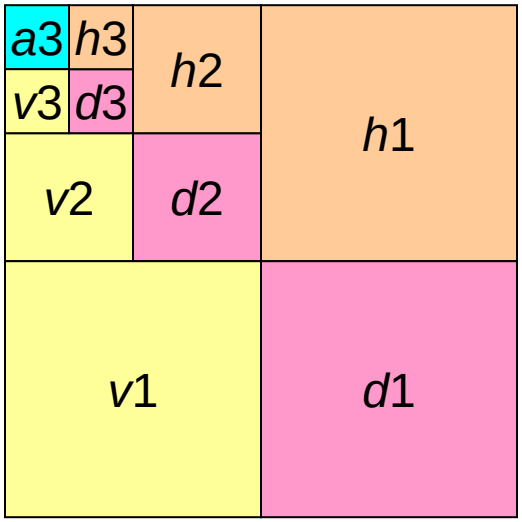
- More functionality
  - Support larger images
  - Progressive transmission by quality, resolution, component, or spatial locality
  - Lossy and Lossless compression
  - Random access to the bitstream
  - Region of Interest coding
  - Robustness to bit errors

2D Discrete Wavelet Transformation



2D Discrete Wavelet Transformation





Wavelet coefficients
*N*x*N*

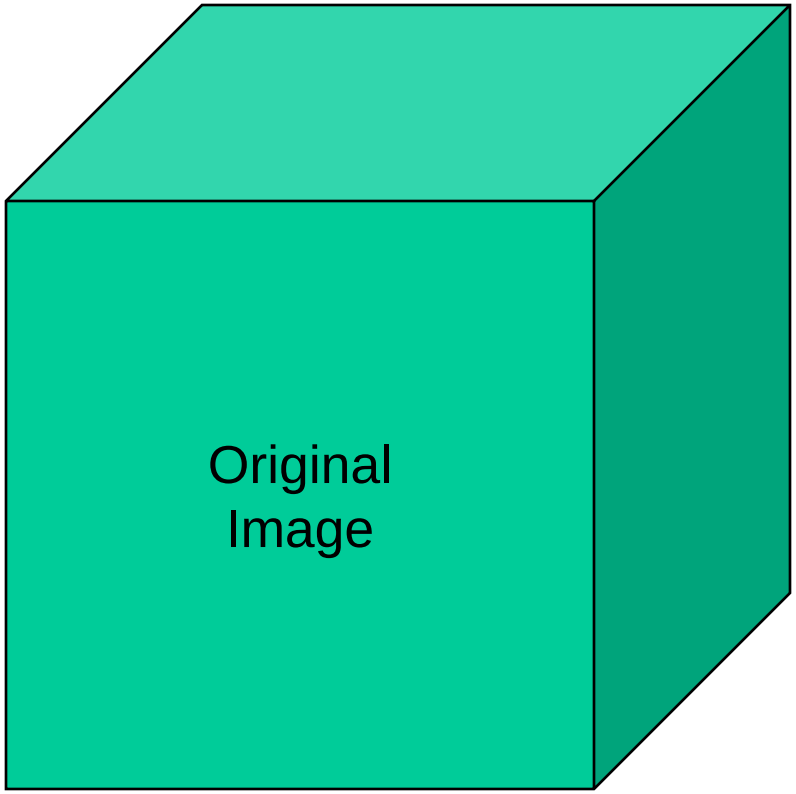
d = Diagonal detail: filtering in both x and y directions using $h_{\psi}(n)$

h = Horizontal detail: filtering in x-direction using $h_{\psi}(n)$ and in y direction using $h_{\varphi}(n)$

v = Vertical detail: filtering in x-direction using $h_{\varphi}(n)$ and in y direction using $h_{\psi}(n)$

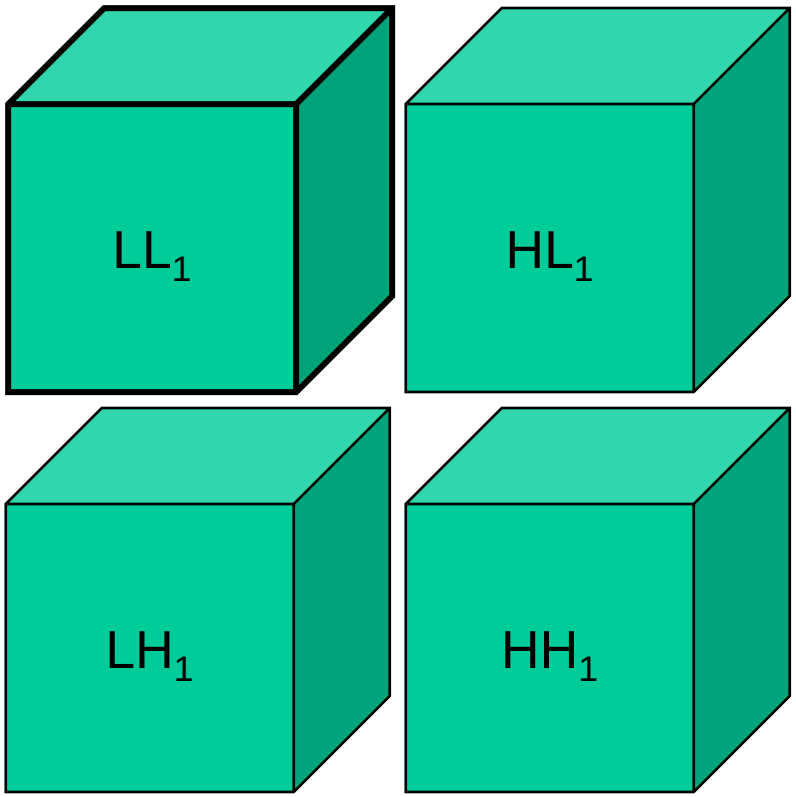
a = Approximation: filtering in both x and y directions using $h_{\varphi}(n)$

Example of 2D Wavelet Transformation



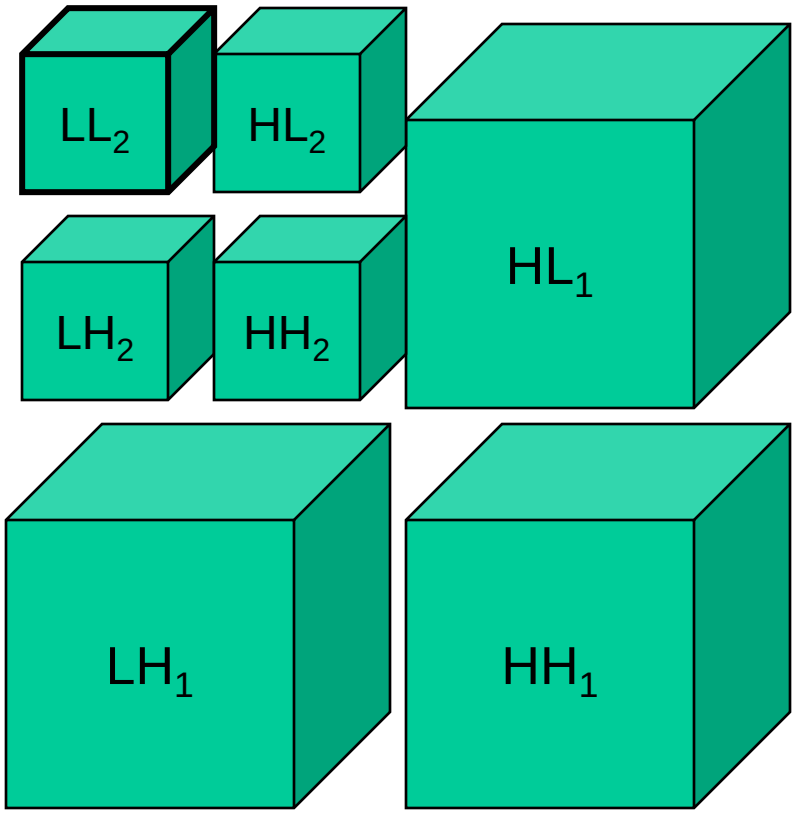
Original image

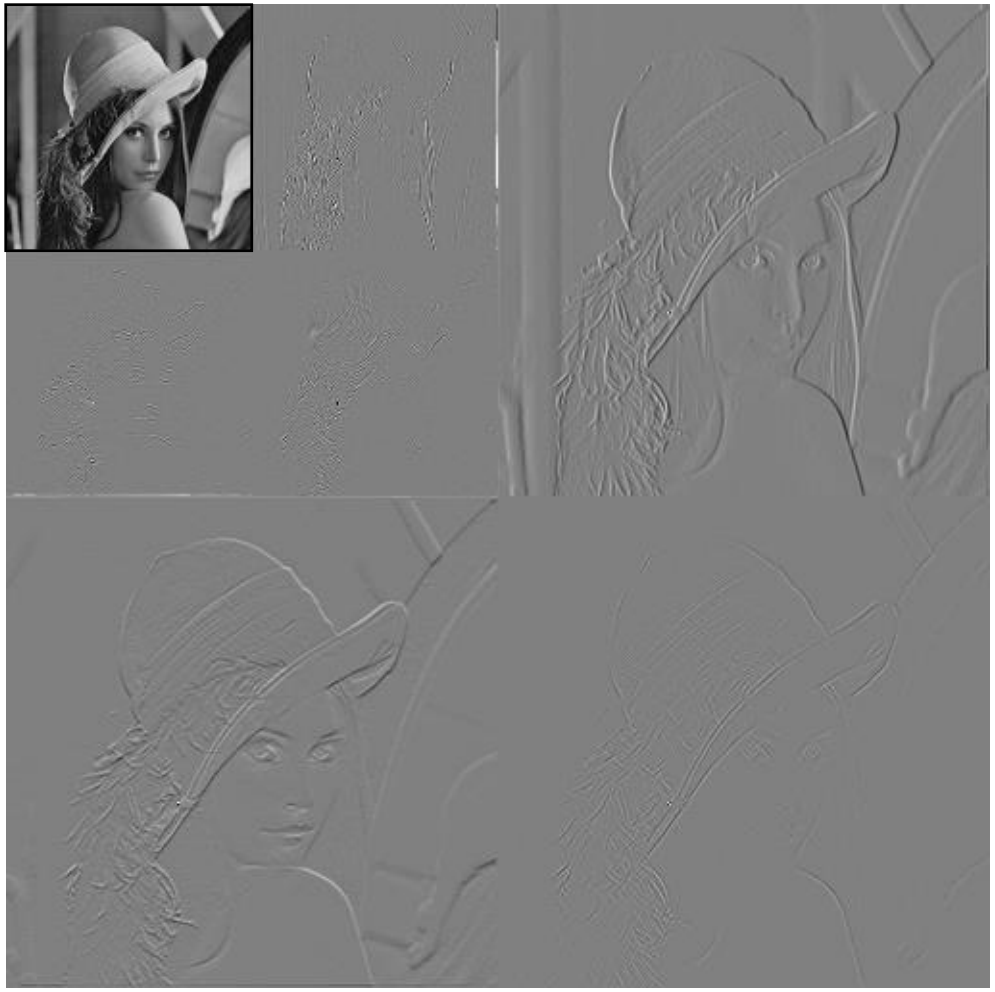
Example of 2D Wavelet Transformation



The first level wavelet decomposition

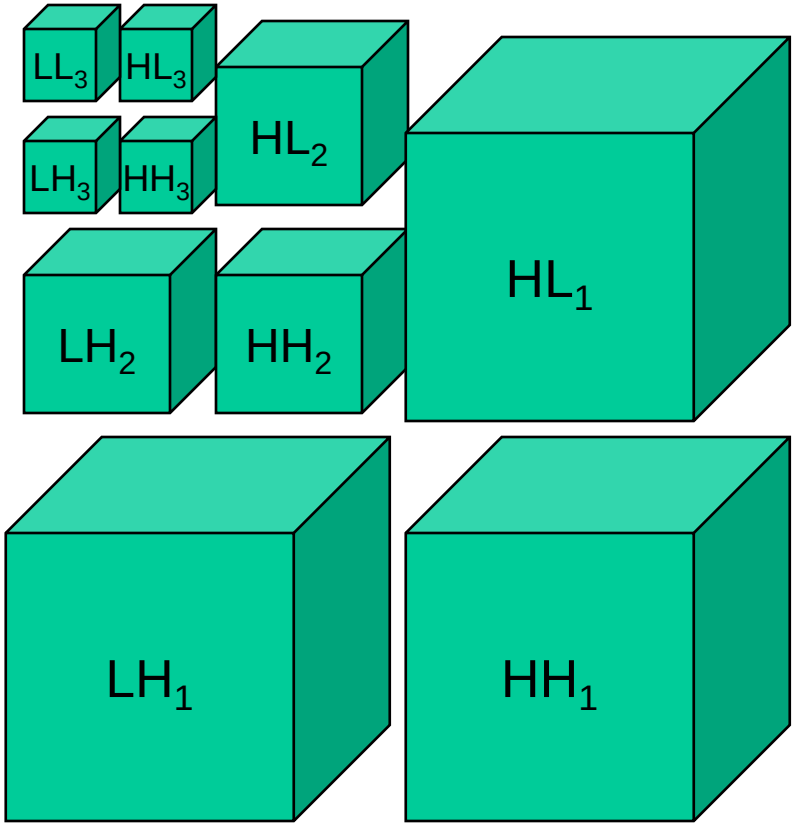
Example of 2D Wavelet Transformation

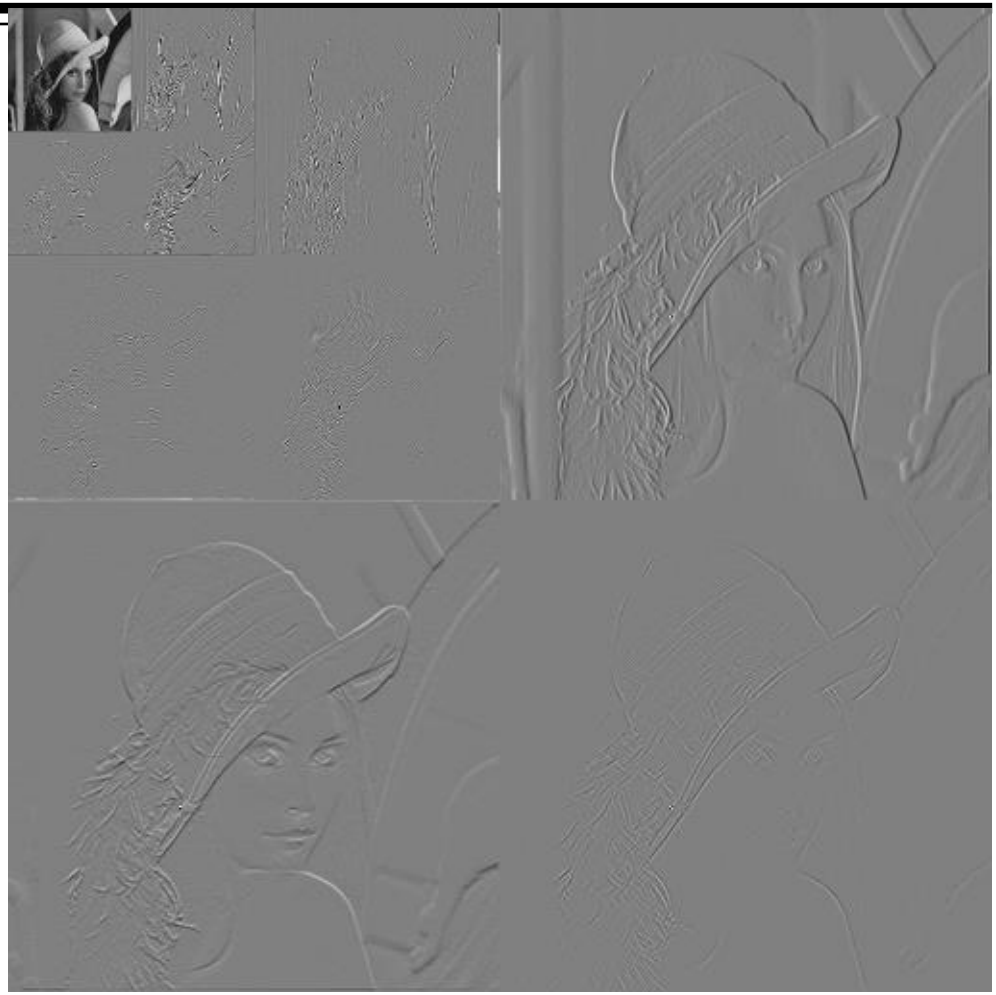




The second level wavelet decomposition

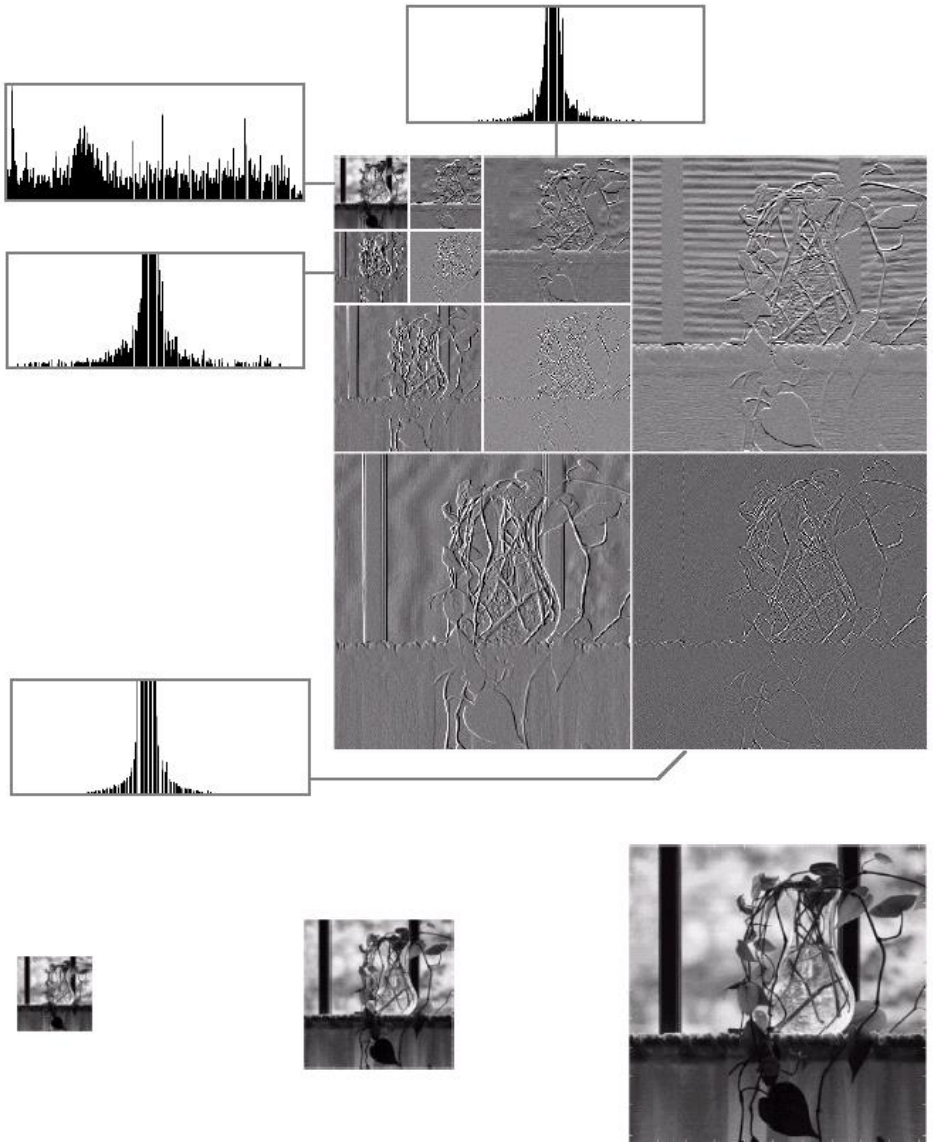
Example of 2D Wavelet Transformation





The third level wavelet decomposition

Example of 2D Wavelet Transformation



a
b c d

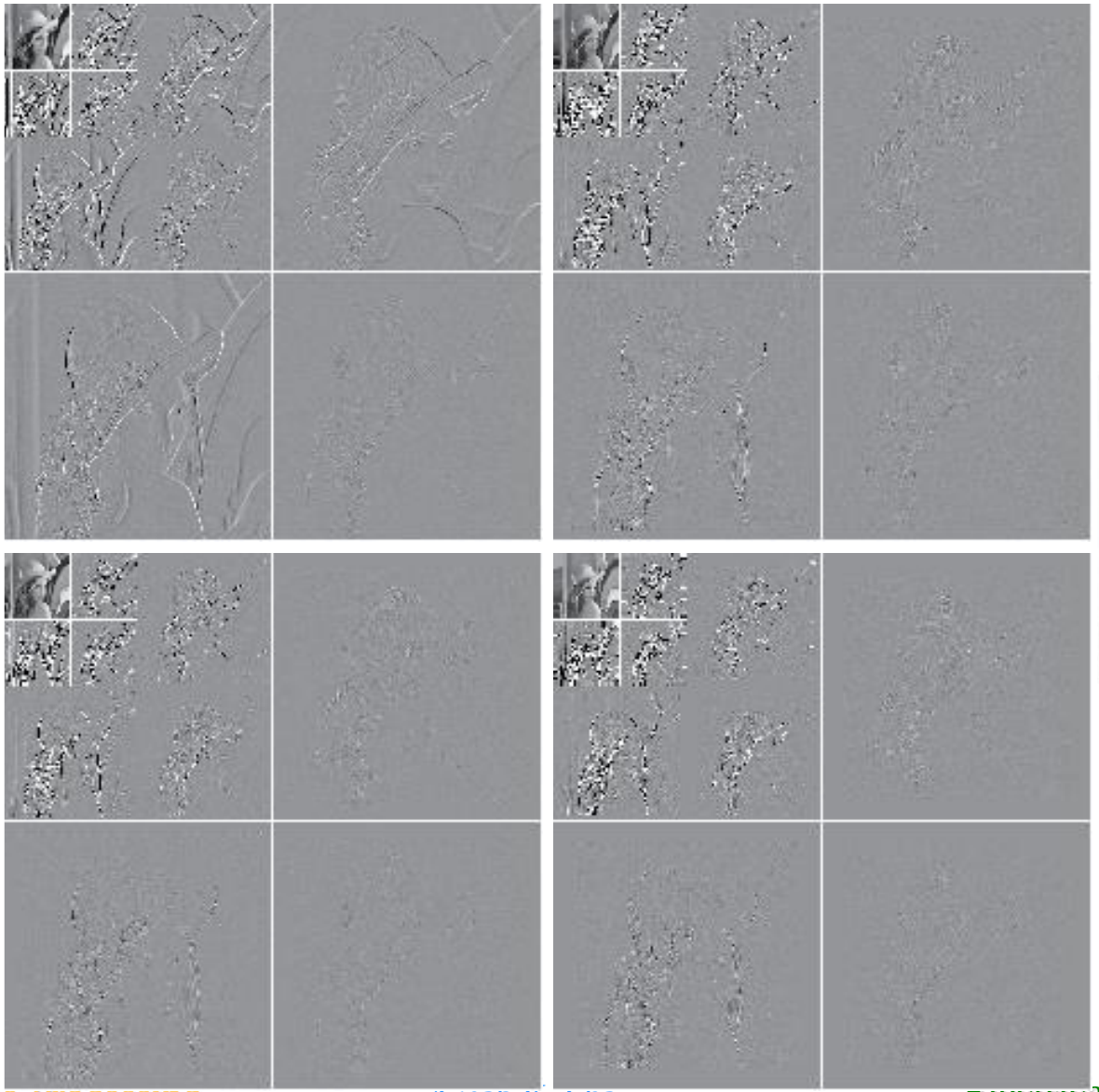
FIGURE 7.8 (a) A discrete wavelet transform using Haar basis functions. Its local histogram variations are also shown; (b)–(d) Several different approximations (64 × 64, 128 × 128, and 256 × 256) that can be obtained from (a).

3rd edition

Examples: Types of Wavelet Transform

Haar wavelets

Daubechies wavelets



a b
c d

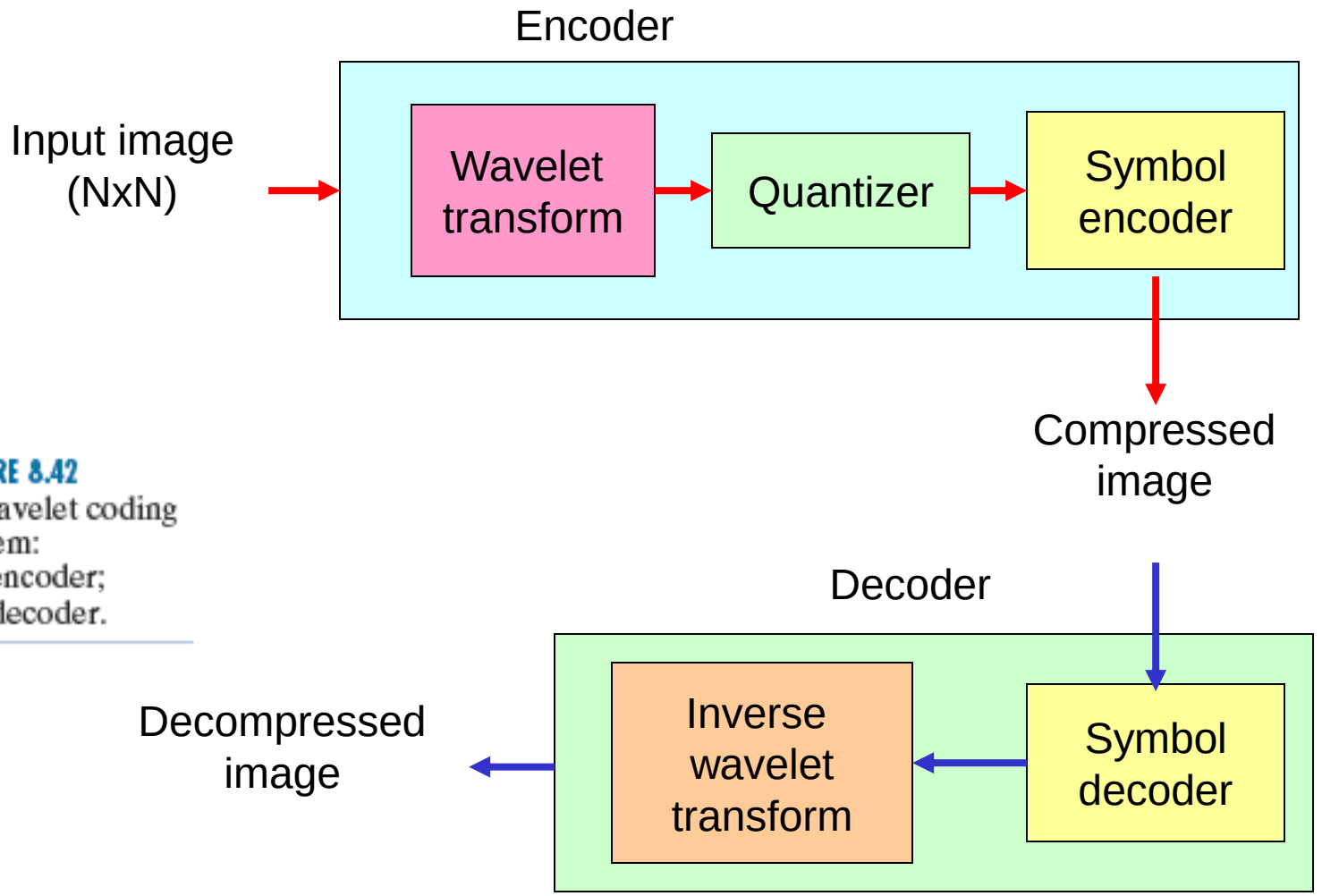
FIGURE 8.43
Three-scale wavelet transforms of Fig. 8.9(a) with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.

Original Image

Symlets

Biorthogonal wavelets

Wavelet Transform Coding for Image Compression



a
b

FIGURE 8.42
A wavelet coding system:
(a) encoder;
(b) decoder.

Unlike DFT and DCT, Wavelet transform is a multiresolution transform.

Examples: Types of Wavelet Transform

TABLE 8.14 Wavelet transform filter taps and zeroed coefficients when truncating the transforms in Fig. 8.43 below 1.5.

Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar	2 + 2	33.3%
Daubechies	8 + 8	40.9%
Symlet	8 + 8	41.2%
Biorthogonal	17 + 11	42.1%

TABLE 8.15 Decomposition level impact on wavelet coding the 512 × 512 image of Fig. 8.9(a).

Decomposition Level (Scales or Filter Bank Iterations)	Approximation Coefficient Image	Truncated Coefficients (%)	Reconstruction Error (rms)
1	256 × 256	74.7%	3.27
2	128 × 128	91.7%	4.23
3	64 × 64	95.1%	4.54
4	32 × 32	95.6%	4.61
5	16 × 16	95.5%	4.63

Impact on Dead Zone Interval on Wavelet Coding

Dead-Zone: Larger quantization interval around zero
→ Improved Quantization Performance

Eliminating every detail coefficient
→ Higher RMSE

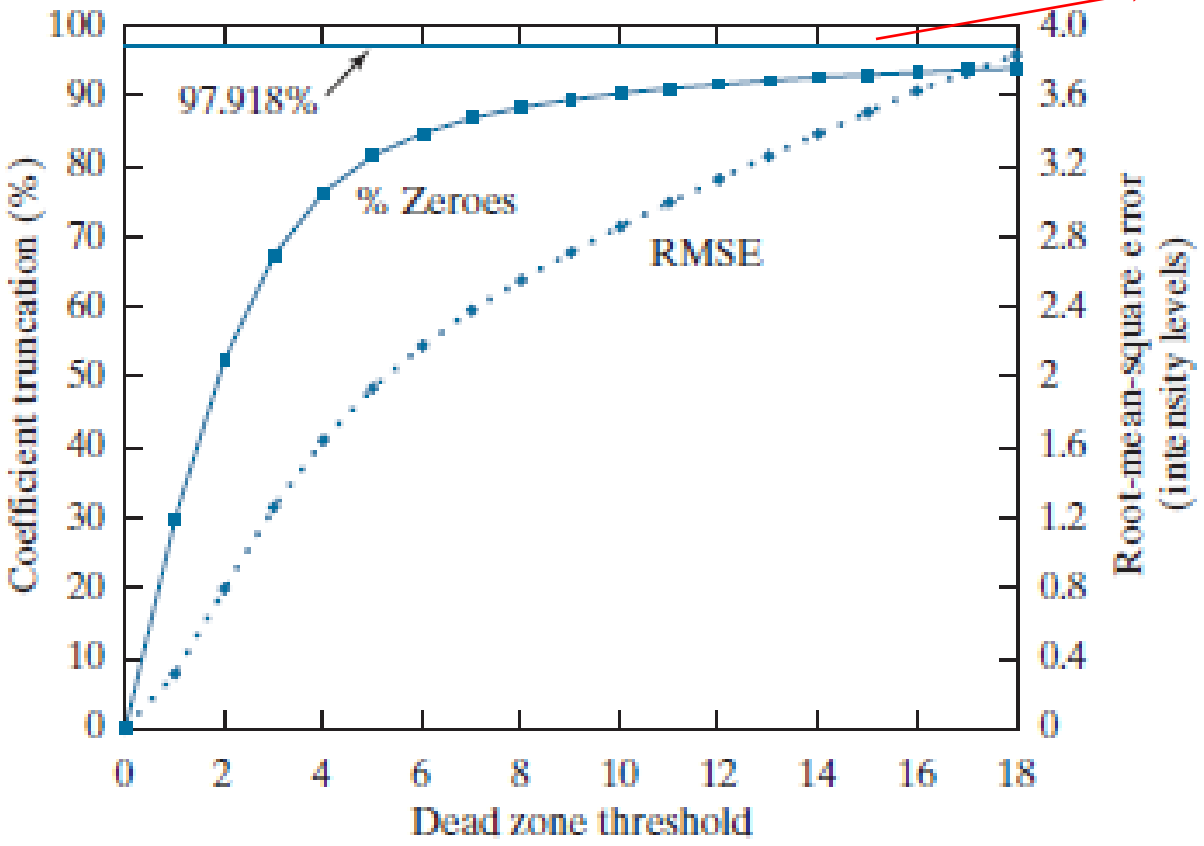


FIGURE 8.44
The impact of dead zone interval selection on wavelet coding.

JPEG 2000: 2-Scale Wavelet Transform

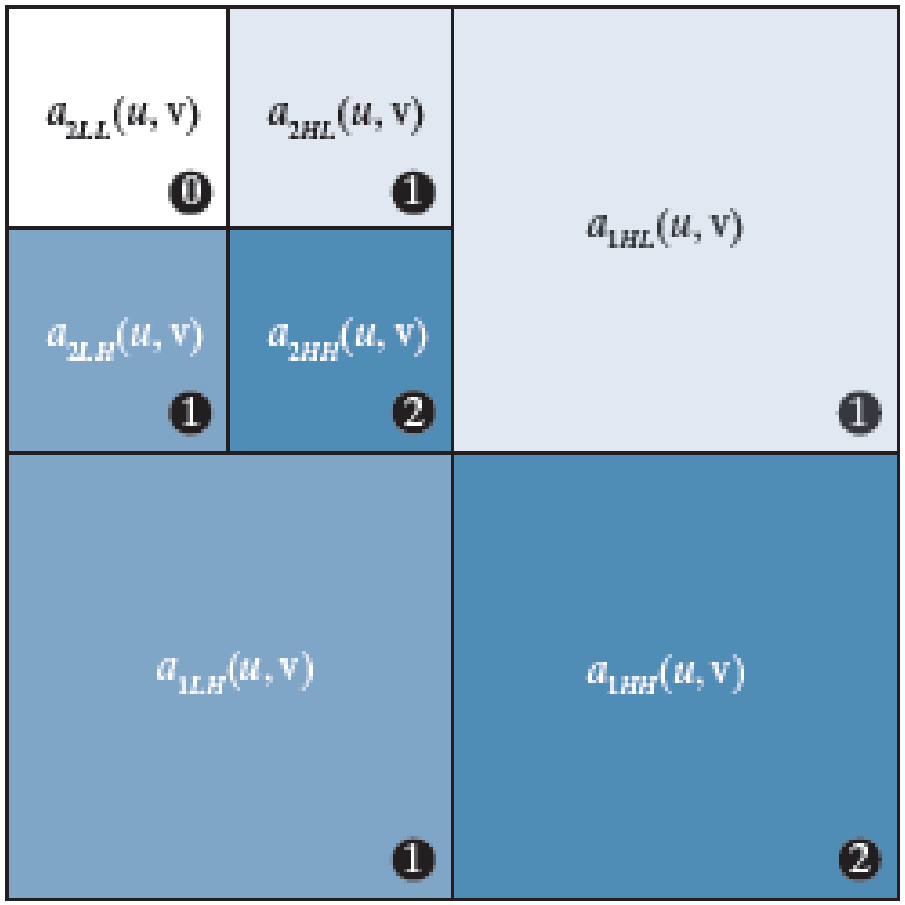


FIGURE 8.45
JPEG 2000 two-scale wavelet transform tile-component coefficient notation and analysis gain.

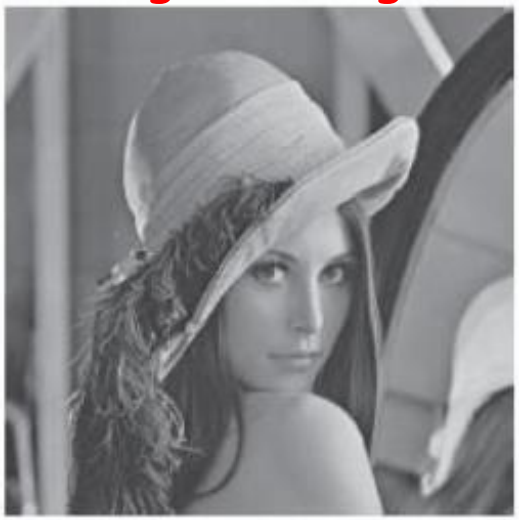
Compression Levels →

C = 25

JPEG 2000 approximation of Fig 8.9(a)

C=52

Original Image



C=75

C=105

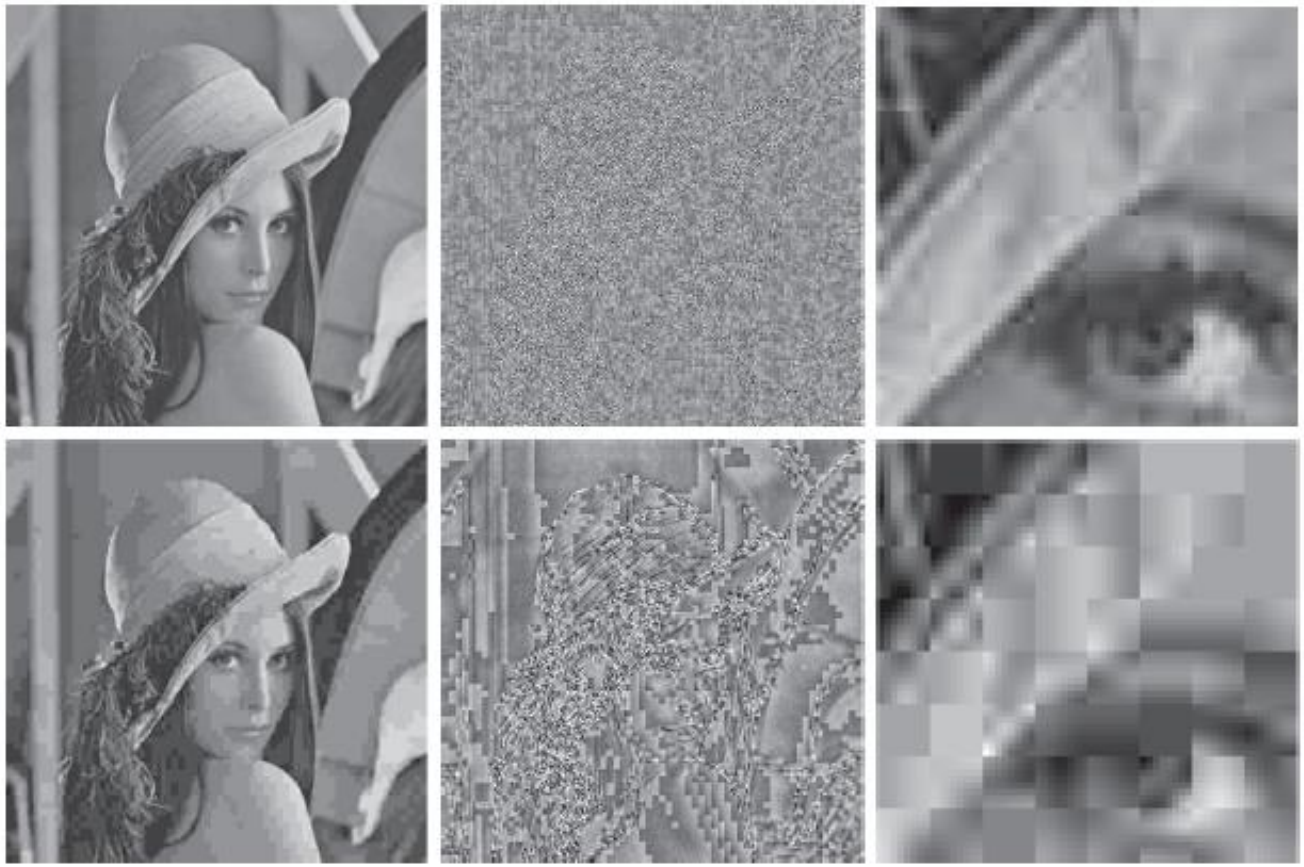
FIGURE 8.46 Four JPEG-2000 approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image. (Compare the results in rows 1 and 2 with the JPEG results in Fig. 8.29.). Slide-129 in next slide

JPEG Approximation Results

Compression Ratio
RMS Errors

25:1
5.4

52:1
10.7

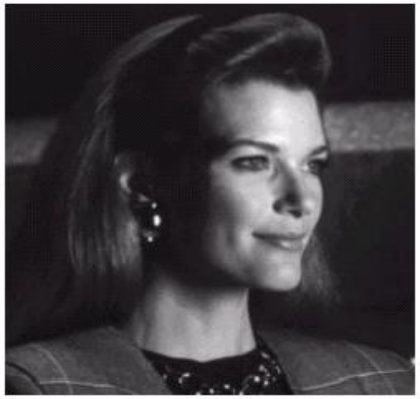


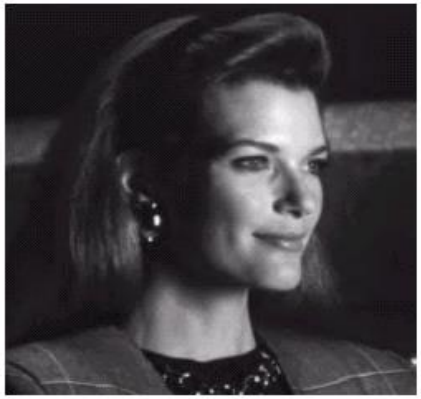
a b c
d e f

FIGURE 8.29 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.

Wavelet Transform Coding Example

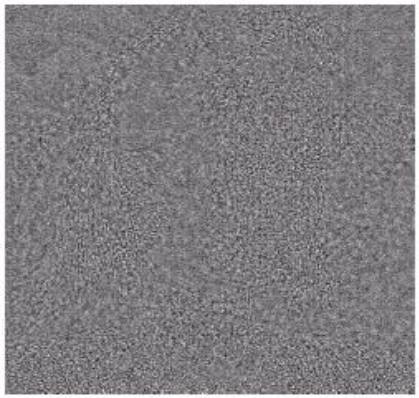
(C_R = 38:1)

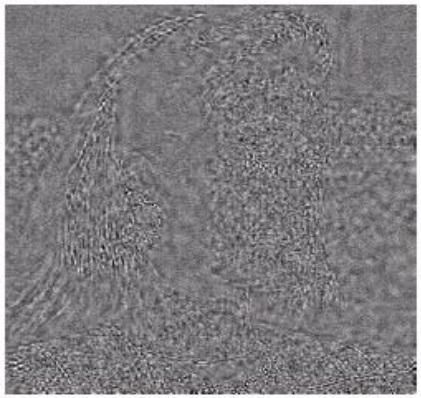




(C_R = 67:1)

Error Image
RMS Error = 2.29





Error Image
RMS Error = 2.96

Zoom details

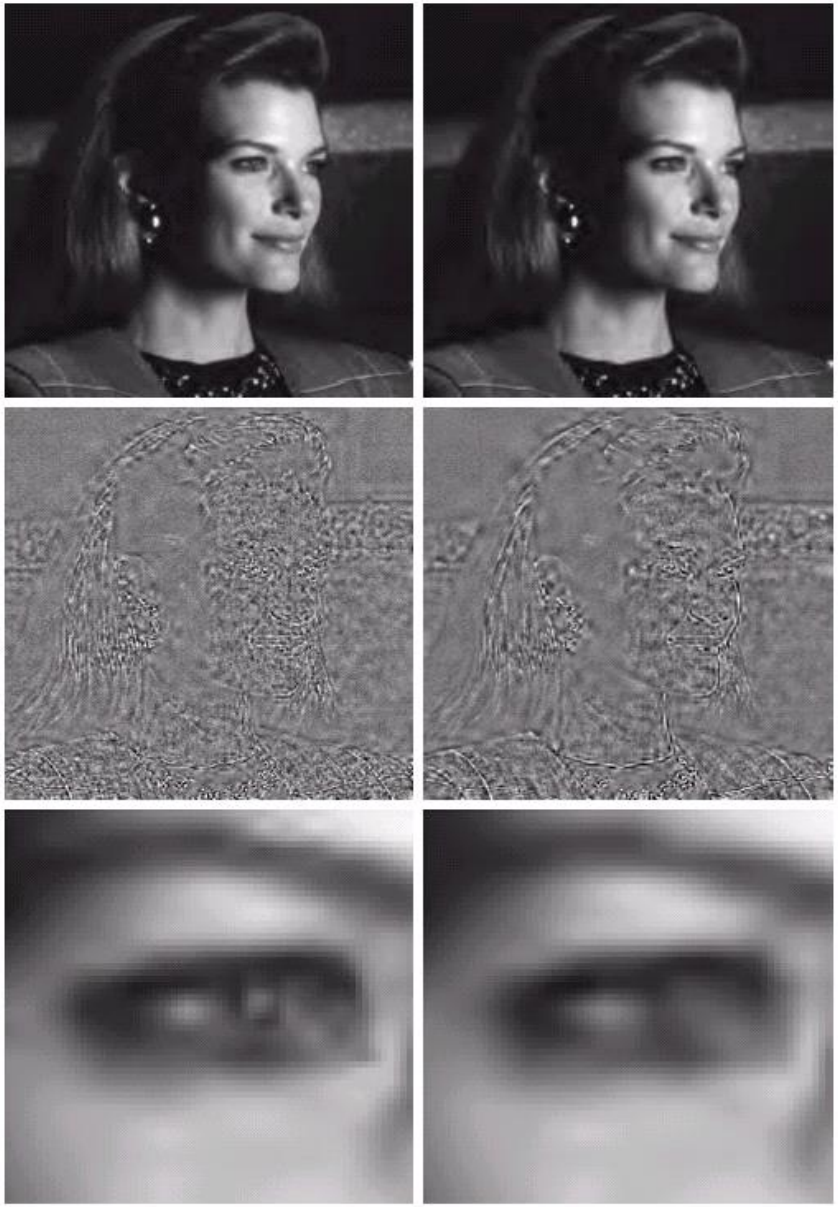
No blocking Artifact

2nd Edition

Wavelet Transform Coding Example

(C_R = 108:1)

(C_R = 167:1)



Error image
RMS Error = 3.72

Error image
RMS Error = 4.73

Zoom details

Wavelet Transform Coding vs. DCT Coding

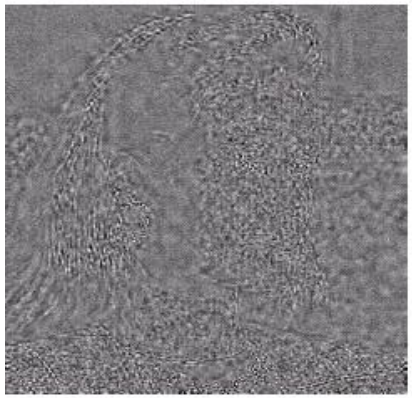
Wavelet

(C_R = 67:1)

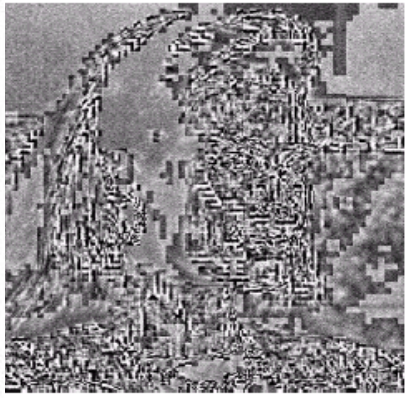
DCT 8x8

(C_R = 67:1)

Error image
RMS Error = 2.96

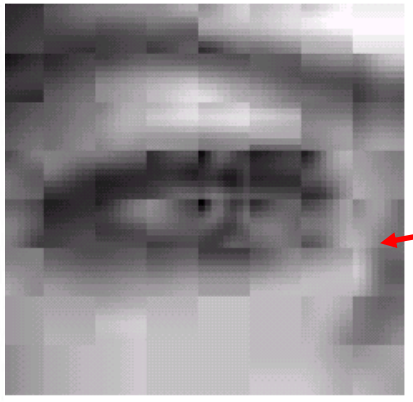


Error image
RMS Error = 6.33



No blocking
Artifact

Zoom details



blocking
Artifact

2nd Edition

JPEG at 0.125 bpp (enlarged)

C. Christopoulos, A. Skodras, T. Ebrahimi, JPEG2000 (online tutorial)

JPEG2000 at 0.125 bpp

C. Christopoulos, A. Skodras, T. Ebrahimi, JPEG2000 (online tutorial)

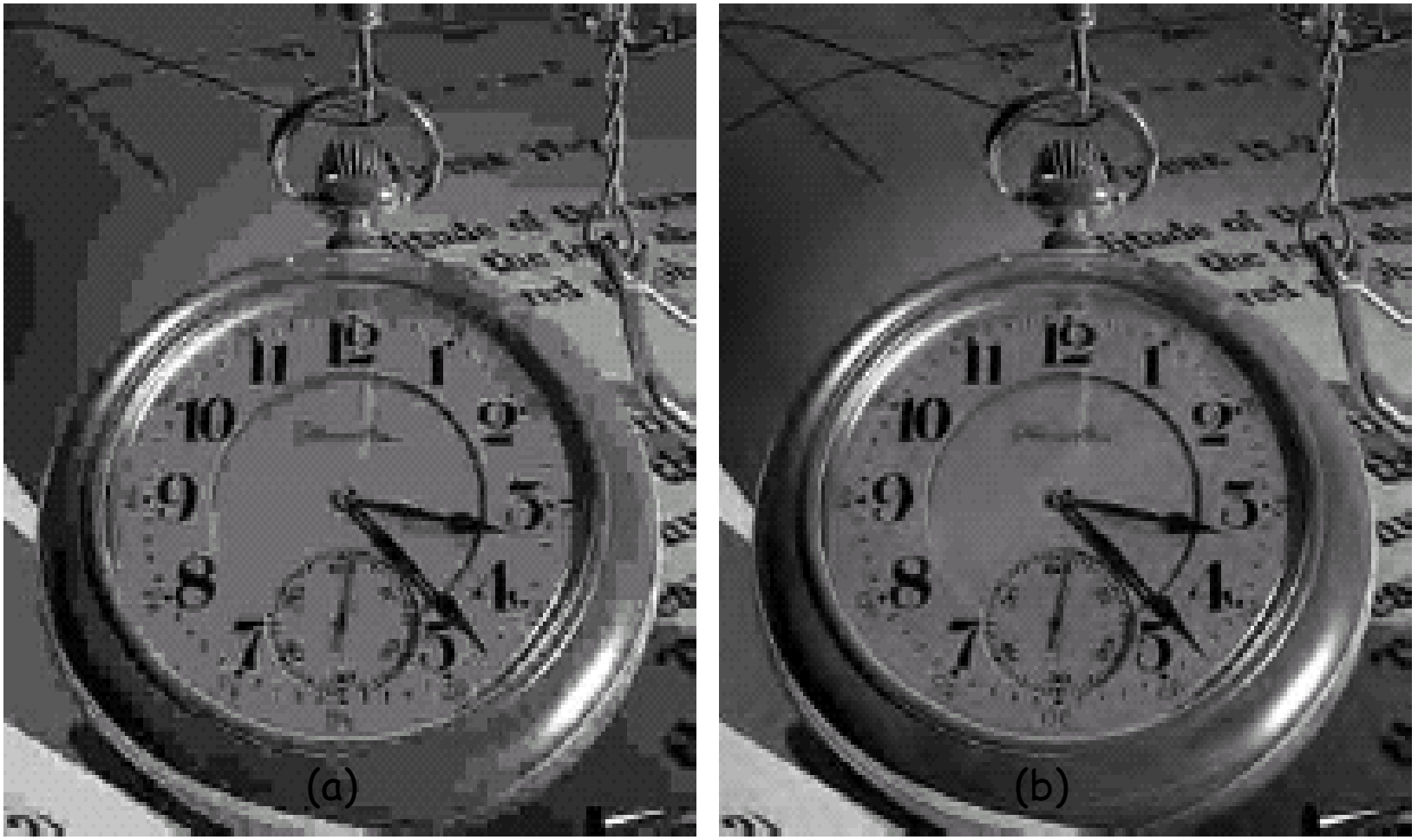
JPEG-2000 V.S. JPEG

(a)

(b)

Compression at 0.25 b/p by means of (a) JPEG (b) JPEG-2000

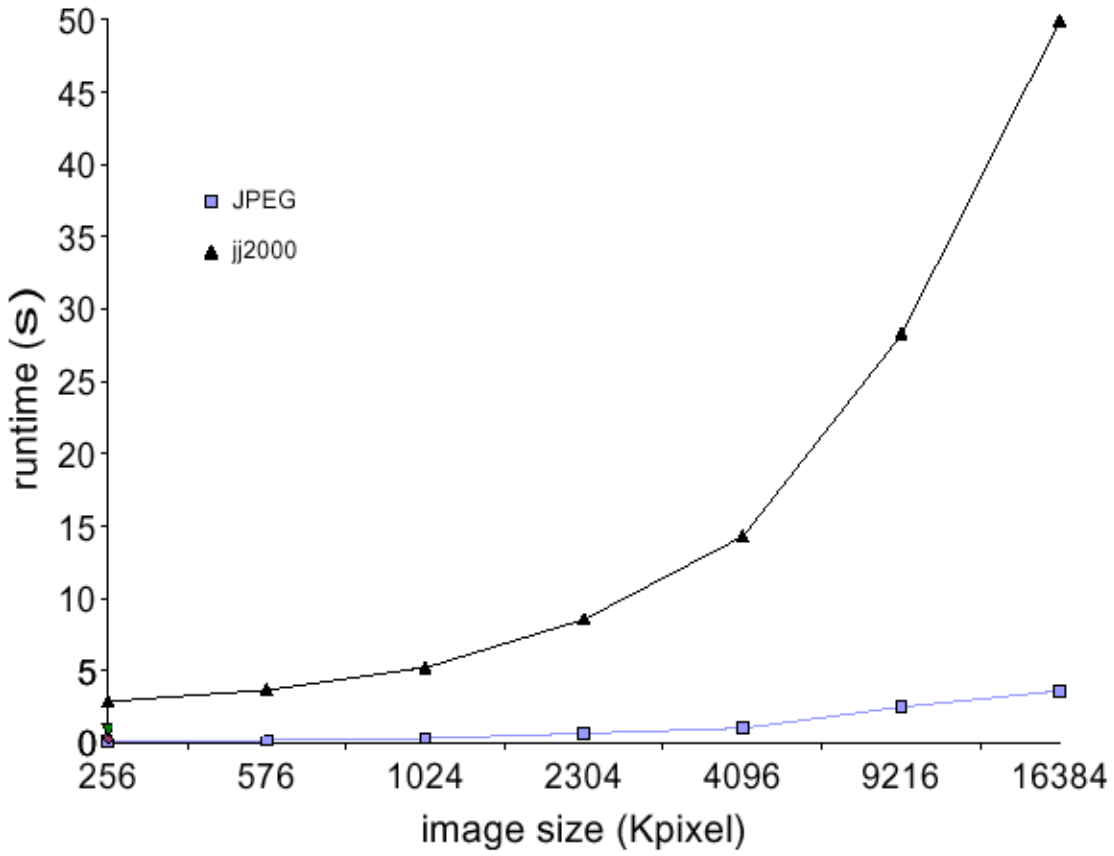
JPEG-2000 V.S. JPEG



Compression at 0.2 b/p by means of (a) JPEG (b) JPEG-2000

JPEG vs. JPEG-2000 Trade-off

- Price→ Cost: JPEG2000 has a much higher computational complexity than JPEG, especially for larger pictures.



WRIGHT STATE UNIVERSITY

Acknowlegements

The slides are primarily based on the figures and images in the Digital Image Processing textbook by Gonzalez and Woods:

- http://www.imageprocessingplace.com/DIP-3E/dip3e_book_images_downloads.htm

In addition, slides have been adopted and modified from the following sources:

- https://cs.nmt.edu/~ip/index.html
- http://www.cs.uoi.gr/~cnikou/Courses/Digital_Image_Processing
- http://www.comp.dit.ie/bmacnamee/gaip.htm
- http://baggins.nottingham.edu.my/~hsooihock/G52IIP/
- http://gear.kku.ac.th/~nawapak/178353.html