Digital Image Processing

Chap 8: Image Compression

清華大學電機系林嘉文
cwlin@ee.nthu.edu.tw
Tel: 03-5731152

Image Compression

- Fundamental
- Basic Compression Techniques
  - Lossless coding
    - Huffman
    - Arithmetic
    - LZW
    - Bit-plane coding
  - Lossy Coding
    - Predictive Coding
    - Temporal Predictive Coding
    - Transform Coding
    - JPEG/JPEG 2000
    - H.261/H.263/H.264
    - MPEG1/MPEG2/MPEG4

Image Compression - Fundamentals

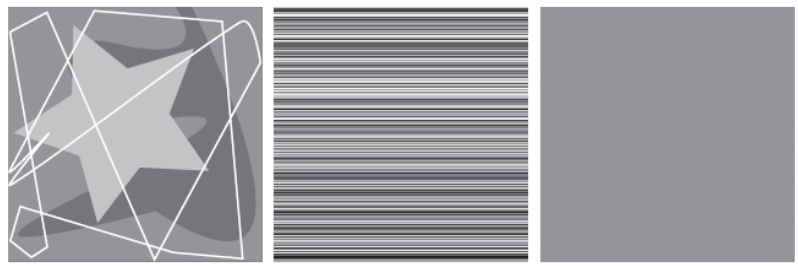
- Image compression addresses the problem of reducing the amount of data required to represent a digital image.
- Removal of **redundant data**.
- Transform 2-D pixel array into a statistically uncorrelated data set.
- Reduce video transmission bandwidth.
- Three basic **redundancy** can be exploited for image compression:

  Coding (statistical) redundancy,

  Spatial and temporal redundancy,

  Irrelevant information

Image Compression - Fundamentals



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.)

Image Compression - Fundamentals

- **Data compression** removes **data redundancy**
- Let b and b' denote the numbers of information carrying units (bits) in two (larger and smaller) representations of the same information.
- The **relative data redundancy** R_D is

  $$R_D = 1 - 1/C_R$$

  where $C_R = b/b'$ is the **compression ratio**
- $b = b'$, $R_D = 0$, and $C_R = 1$, no data redundancy
- $b \gg b'$ and $C_R \gg 1$, $R_D \cong 1$ highly redundant data.

Coding Redundancy

- **Coding (statistical) redundancy:** Codes assigned to a set of events (gray-level values) have not been selected to take full advantage of the probabilities of the events.
- A discrete random variable r_k in the interval [0,1] represents the gray levels of an image ($M \times N$) and that each r_k occurs with probability $p_r(r_k) = n_k/n$, $n = MN$, $k = 0,1,\ldots,L-1$, where L is the number of gray-levels.
- If the number of bits required to represent r_k is $l(r_k)$, then the average number of bits required to represent a pixel is

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

Coding Redundancy

$$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k)\, p_r(r_k)$$
$$= 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

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	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k = 87, 128, 186, 255$	0	—	8	—	0

Compression ratio: $C = 8/1.81 \approx 4.42$,

Redundancy: $R = 1 - 1/4.42 = 0.774$

Spatial and Temporal Redundancy

- **Spatial and Temporal Redundancy**:
- In Fig. 8.1(b) the pixels are independent of each other in the vertical direction, but maximally correlated to each other in the **horizontal** direction

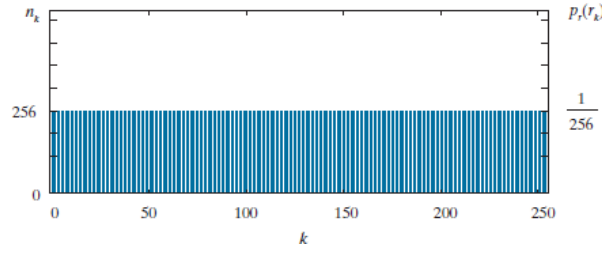
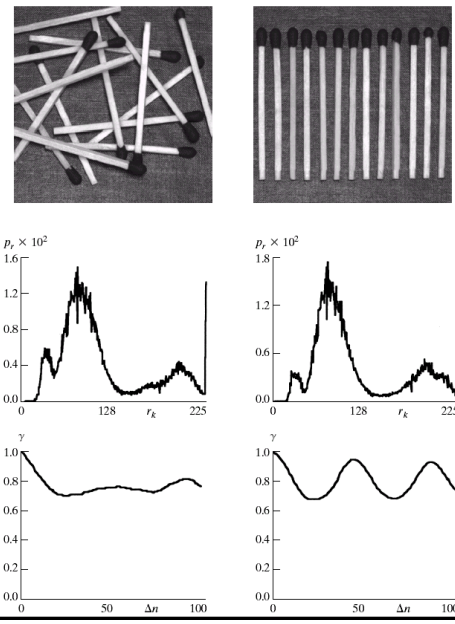


FIGURE 8.2 The intensity histogram of the image in Fig. 8.1(b).

Spatial and Temporal Redundancy

- Spatial Redundancy:

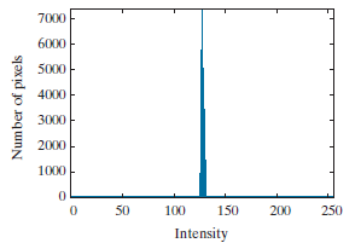


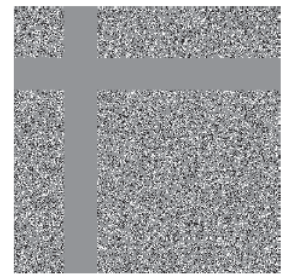
Irrelevant Information (Psychovisual redundancy)

- Fig 8.1(c) appears to be a homogeneous field of gray levels
  → represented by a single 8-bit value
- However, its histogram is shown in Fig. 8.3(a)
- After **histogram equalization** (image enhancement) →Fig 8.3(b).

a b

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





Irrelevant Information (Psychovisual redundancy)

- **Psychovisual redundancy** is reduced by quantization which maps a broad range of input value to a limited number of output values
- Certain information simply has less importance for human vision, It can be eliminated without significantly impairing the quality of image perception.
- Elimination of psychovisual redundancy results in information loss which is not recoverable, it is an **irreversible operation.**
- **Quantization** will induce the **false contouring**.

Measuring Image Information

- The generation of **information** can be modeled as a probabilistic process that can be measured in a manner that agrees with intuition.
- **A random event** E that occurs with probability $P(E)$ is said to contain $I(E) = \log\left(\frac{1}{P(E)}\right) = -\log P(E)$ units of information.
- *I*(*E*) is called the **self-information** of *E*.

  If $P(E) = 1$ then $I(E) = 0$ bit

  If $P(E) = 1/2$ then $I(E) = 1$ bit

Measuring Image Information: Entropy

- The **average information** per source output is

$$H(\mathbf{z}) = -\sum_{j=1}^{J} P(a_j)\log P(a_j)$$

$H(\mathbf{z})$ is the **uncertainty** or **entropy** of the source

- Let image be considered as the output of an imaginary zero-memory "intensity source" we can use the histogram of the observed image to estimate the symbol probability of the source.
- The entropy of an image is $H = -\sum_{k=1}^{L-1} P_r(r_k)\log_2 P_r(r_k)$
- Fig 8.1(b) H = 8 bits/pixel
- Fig 8.1(c) H = 1.6614 bits/pixel (Table 8.1)

Measuring Image Information: Entropy

- **Shannon's first theorem — noiseless coding theorem**
- He looked at representing groups of n **consecutive source symbols** with a single codeword and showed that

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

- where $L_{\text{avg},n}$ is the average number of code symbols required to represent all n-symbol groups.

Measuring Image Information: Entropy

- If the value of A is **equally likely**, then
  $I(A) = K$ bits/pel $\Rightarrow P(a) = 2^{-K} \Rightarrow H(z) = K$ bits/pel
- As $\{P(a)\}$ becomes more **highly concentrated**, the **entropy** becomes smaller.

P(0)	*P*(1)	*P*(2)	*P*(3)	*P*(4)	*P*(5)	*P*(6)	*P*(7)	Entropy (bits/pel)
1.0	0	0	0	0	0	0	0	0.00
0	0	0.5	0.5	0	0	0	0	1.00
0	0	0.25	0.25	0.25	0.25	0	0	2.00
0.06	0.23	0.30	0.15	0.08	0.06	0.06	0.06	2.68
0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	3.00

Entropy

- 8-bits image as

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

First order entropy = 1.81 bits/pixel

Second order entropy = 2.5/2 =1.25 bits/pixel

Gray level	count	probability
21	12	3/8
95	4	1/8
169	4	1/8
243	12	3/8

Gray-level pair	count	probability
21, 21	8	¼
21, 95	4	1/8
95, 169	4	1/8
169, 243	4	1/8
243, 243	8	¼
243, 21	4	1/8

Entropy

- Difference images

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

First order entropy = 1.41 bits/pixel

Gray level	count	Probability
0	12	½
21	4	1/8
74	12	3/8

Fidelity Criteria

Objective or Subjective Fidelity Criteria

- $f(x,y)$ is the original image, $\hat{f}(x,y)$ is the decompressed image
- The error is defined as $e(x,y) = \hat{f}(x,y) - f(x,y)$
- Total error between two images (size $M \times N$):

$$\sum_x \sum_y \left[\hat{f}(x,y) - f(x,y)\right]$$

- The **root-mean squared error** e_{rms}:

$$e_{\text{rms}} = \left[\frac{1}{MN}\left\{\sum_x \sum_y \left[\hat{f}(x,y) - f(x,y)\right]^2\right\}\right]^{1/2}$$

- The **mean-square signal to noise ratio**: SNR_{ms}

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

Fidelity Criteria

Subjective evaluation by human observers: The evaluation can be made by an absolute rating scale or by means of side-by-side comparison of $f(x, y)$ and $f'(x, y)$

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 c

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

Subjective evaluation ranking: (a) (b) (c)

rms evaluation ranking: (a) (c) (b)

Fidelity Criteria

Random "Analog" Noise

MSE = 27.10

Blocky "Digital" Noise

MSE = 21.26

Measures like MSE suitable for Analog noise no longer work for Digital noise

Image Compression Model

Image Compression Model:

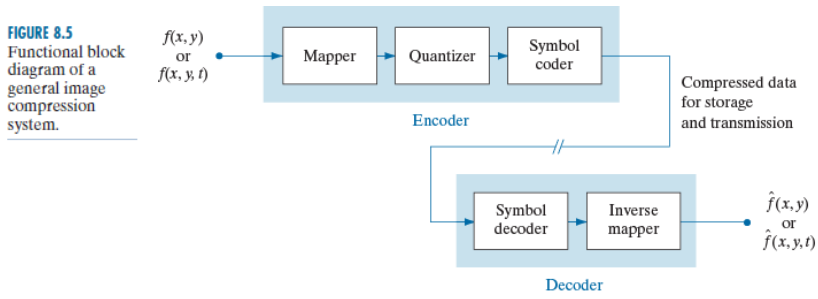


FIGURE 8.5 Functional block diagram of a general image compression system.

Image Compression Model

- $f(x,y)$ equals to $\hat{f}(x,y)$
  $\rightarrow$ error-free, lossless coding
- $f(x,y)$ does not equal to $\hat{f}(x,y)$
  $\rightarrow$ lossy coding

Mapper $\rightarrow$ reduce **Spatial and Temporal redundancy**

Quantizer $\rightarrow$ reduce **Psychovisual redundancy**

Symbol Coder $\rightarrow$ reduce **Coding redundancy**

Image Compression Standards

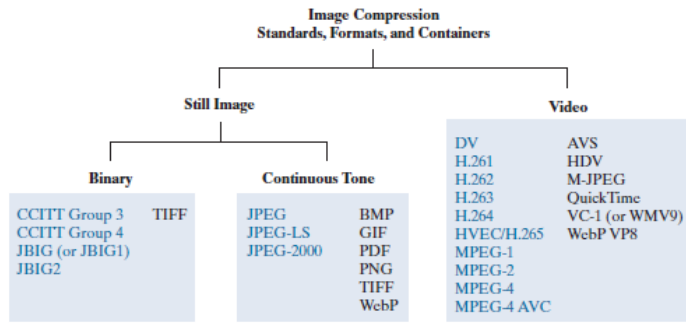


FIGURE 8.6 Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in blue; all others are in black.

Image Compression Standards

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.

Image Compression Standards

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

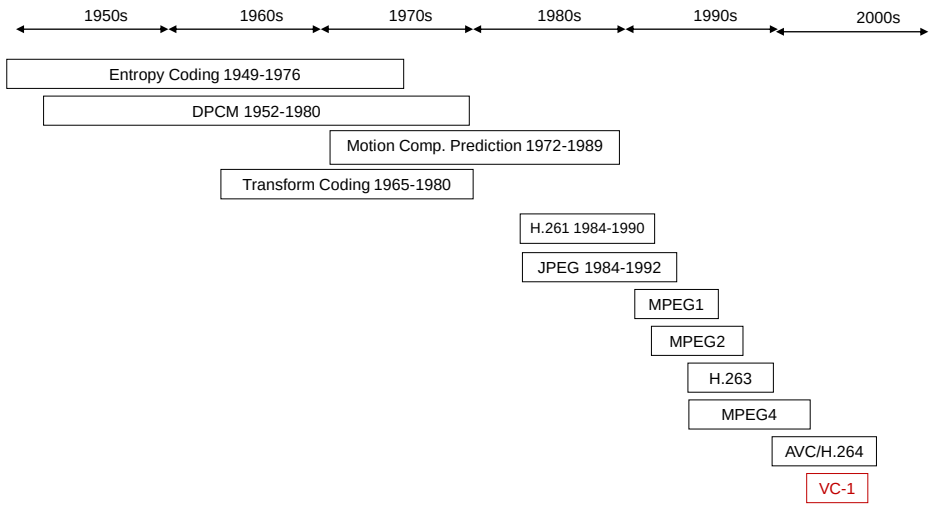
Name	Organization	Description
DV	IEC	*Digital Video*. A video standard tailored to home and semiprofessional video production applications and equipment, such as electronic news gathering and camcorders. Frames are compressed independently for uncomplicated editing using a DCT-based approach [8.9] similar 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.

Image Compression Standards

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.
Quick-Time	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].

Image/Video Compression Standards



Basic Compression Methods

- Entropy Coding (Lossless coding)
  - Huffman coding
  - Arithmetic coding
  - Lampel-Ziv-Welch (LZW) coding
- Run-length Coding
- Symbol-based Coding
- Bit-plane Coding
- Predictive Coding (spatial)
  - Lossless predictive coding
  - Lossy predictive coding
- Transform Coding (JPEG)
- Wavelet Coding (JPEG2000)
- Temporal Predictive Coding

Basic Compression Methods

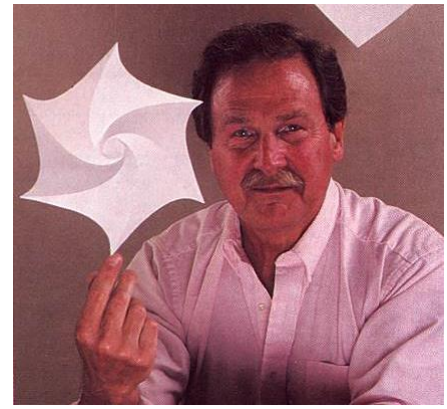
- Instead of assigning ***K*-bit** words to each of the possible 2^K luminance levels, we assign words of **longer** length to levels with **lower probability** and words of **shorter** length to levels with **higher probability**
- **Variable word-length Coding → Entropy Coding**
- Symbol b with probability $P(b)$ is assigned with code word length $L(b)$ bits, then the average codeword length is

$$\bar{L} = \sum_{b=1}^{2^K} L(b)P(b) \text{ bits/symbol}$$

Huffman Coding

David A. Huffman

Born	August 9, 1925
Died	October 7, 1999 (aged 74)
Residence	USA
Fields	Information theory, Coding theory
Alma mater	Ohio State University (BS 1944, MS 1949), MIT (PhD 1953)
Affiliation	MIT (1953~1967) UC Santa Cruz (1967~1994)
Known for	Huffman code



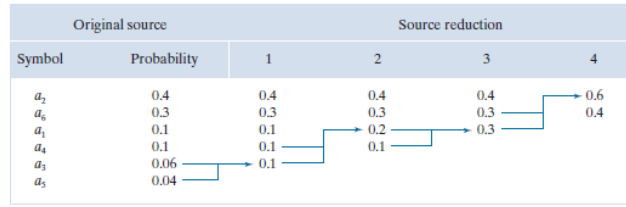
D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., pp 1098-1102, Sept. 1952.

Huffman Coding

- **Input**: Symbols (characters) and their frequency of occurrence.
- **Output**: Huffman code tree
  - Binary tree
  - Root node
  - Branches are assigned the value of 0 or 1
  - Branch node
  - Leaf node is the point where the branch end
    - To which the symbols being encoded are assigned
- An unbalanced tree
  - Some branches are shorter than the others
  - The degree of imbalance is a function of **relative frequency** of occurrence of the characters: the larger the spread, the more unbalanced the tree is

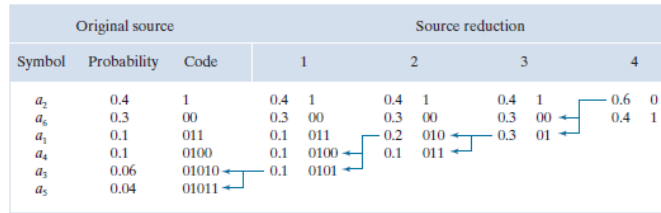
Huffman Coding

FIGURE 8.7 Huffman source reductions.



Huffman Coding

FIGURE 8.8 Huffman code assignment procedure.



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

Entropy = 2.14 bits/symbol

Bit string 010100111100 → $a_3\ a_1\ a_2\ a_2\ a_6$

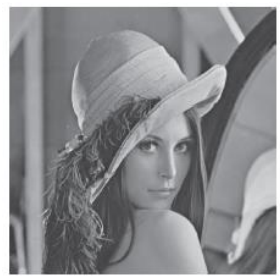
Huffman Coding

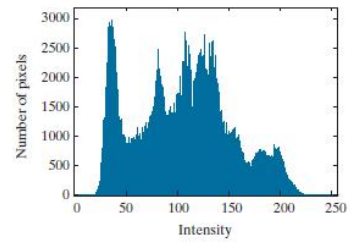
- Huffman code itself is an **instantaneous, uniquely decodable block code**.
- **Block code**: each source symbol is mapped into a **fixed sequence** of code symbols.
- **Instantaneous**: each codeword can be decoded without referencing the succeeding symbols.
- **Uniquely decodable**: code string can be decoded in only one way.

Huffman Coding

a b

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





Huffman coding compression (including Huffman Table)

$\rightarrow$ 7.428 bits/pixel

Compression ratio: $C = 8/7.428 = 1.077$

Data Redundancy $R = 1 - (1/0.1077) = 0.0715$

Vector Huffman Coding

- Huffman coding can only use an integer number of bits to encode a symbol
- It achieves the entropy bound only when each symbols' probability is a power of 1/2
- For a source alphabet S with highly skewed probabilities

Assign one codeword for each symbol

Letters	$P(a_i)$	Codeword
a_1	0.99	0
a_2	0.01	1

$$H = 0.08 \text{ bits/symbol}$$
$$\bar{l} = 1 \text{ bits/symbol}$$

Assign one codeword for every **two** symbols

Letters	$P(a_i)$	Codeword
$a_1 a_1$	0.9801	0
$a_1 a_2$	0.0099	11
$a_2 a_1$	0.0099	100
$a_2 a_2$	0.0001	101

$$\bar{l}^{(2)} = 1.0299 \text{ bits/block}$$
$$\bar{l} = \frac{\bar{l}^{(2)}}{2} = 0.51495 \text{ bits/symbol}$$

Vector Huffman Coding

- For a source alphabet S with highly skewed probabilities

Assign one codeword for every *three* symbols

Letters	$P(a_i)$	Codeword
$a_1 a_1 a_1$	0.970299	0
$a_1 a_1 a_2$	0.009801	100
$a_1 a_2 a_1$	0.009801	101
$a_1 a_2 a_2$	0.000099	11100
$a_2 a_1 a_1$	0.009801	110
$a_2 a_1 a_2$	0.000099	11101
$a_2 a_2 a_1$	0.000099	11110
$a_2 a_2 a_2$	0.0001	11111

$$H = 0.08 \text{ bits/symbol}$$
$$\bar{l}^{(3)} = 1.05998 \text{ bits/block}$$
$$\bar{l} = \frac{\bar{l}^{(3)}}{3} = 0.3533 \text{ bits/symbol}$$

To "block" several symbols together at a time

- per-symbol inefficiency is now spread over the whole block
- the size of the codebook increases exponentially
  - e.g., $A = \{a_1, a_2, \ldots, a_m\} \Rightarrow m^n$ codewords are needed to generate one codeword for every n symbols

Arithmetic Coding

- Arithmetic coding is better than Huffman coding in achieving the **Shannon value**.
- A **non-block codes**→ an **entire sequence of source symbols** is assigned a single arithmetic codeword.
- Divide the numeric range from 0 to 1 into a number of different characters present in the message to be sent.
- The **size** of each segment is determined by the probability of the related character.
- Each subsequence in the string subdivides the range into progressively smaller segments.
- The **codeword** for the complete string is **any number within the range**.

Arithmetic Coding

• Arithmetic Code

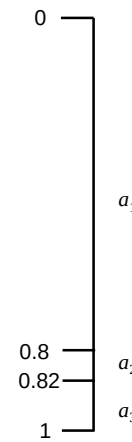
~~- X assign codewords to individual symbols~~

- assign one (normally long) code to the entire input stream

⇒ A source message is represented by an interval of real number in [0,1)

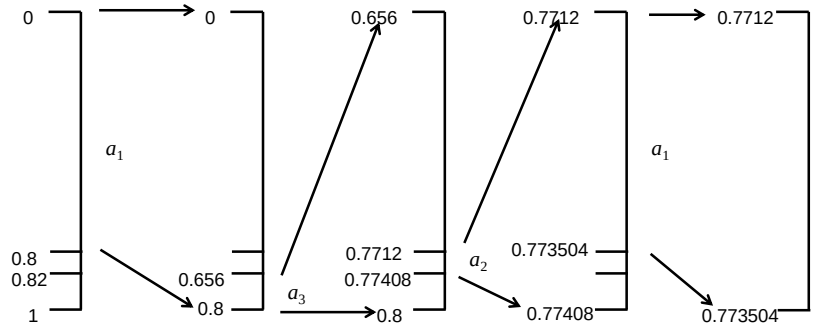
• Example:

Letters	$P(a_i)$	cdf $F_x(i)$	range
a_1	0.8	0.8	[0,0.8)
a_2	0.02	0.82	[0.8,0.82)
a_3	0.18	1	[0.82,1)



Arithmetic Coding

- Input sequence: $X = (x_1 x_2 \ldots x_n) = (a_1 a_3 a_2 a_1)$



$$\begin{cases} l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_x(x_{n-1}) \\ u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_x(x_n) \end{cases}$$

Arithmetic Coding

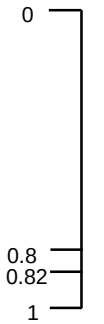
Code = 0.772352

$\begin{cases} l^{(0)} = 0 \\ u^{(0)} = 1 \end{cases}$ ⇒ $x_1 = a_1$, $\begin{cases} l^{(1)} = 0 \\ u^{(1)} = 0.8 \end{cases}$

$code^* = \frac{0.772352 - 0.0}{0.8 - 0.0} = 0.96544$, $x_2 = a_3$, $\begin{cases} l^{(2)} = 0.656 \\ u^{(2)} = 0.8 \end{cases}$

$code^* = \frac{0.772352 - 0.656}{0.8 - 0.656} = 0.808$, $x_3 = a_2$, $\begin{cases} l^{(3)} = 0.7712 \\ u^{(3)} = 0.77408 \end{cases}$

$code^* = \frac{0.772352 - 0.7712}{0.77408 - 0.7712 - 0.0} = 0.4$, $x_4 = a_1$



Decoding steps:

(1) Initialization

$$\begin{cases} l^{(0)} = 0 \\ u^{(0)} = 1 \end{cases}$$

(2) For each *k*

$$code^* = \frac{code - l^{(k-1)}}{u^{(k-1)} - l^{(k-1)}} = \frac{code - lower\_limit}{range}$$

(3) Find x_k s.t.

$$F_X(x_k - 1) \le code^* < F_X(x_k)$$

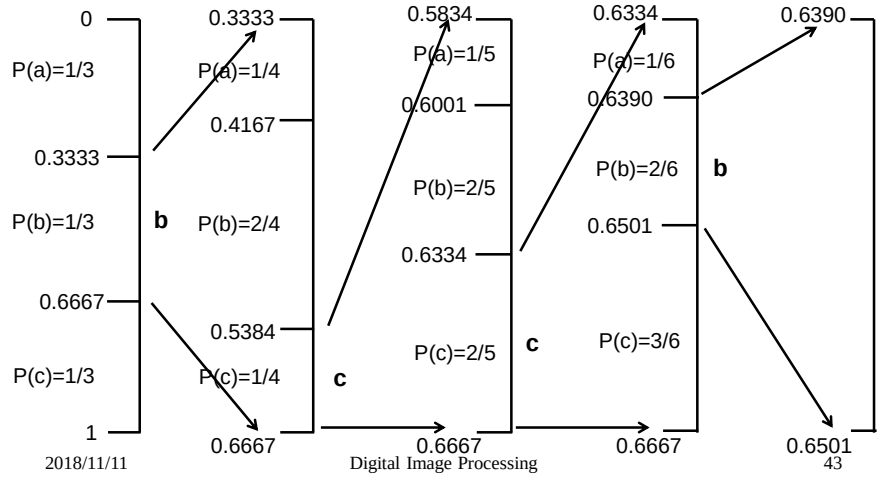
(4) Update $l^{(k)}$ & $u^{(k)}$

(5) Continue until the entire sequence has been decoded

Adaptive Arithmetic Coding

Example

- for a source alphabet {a,b,c}
- input sequence = b c c b

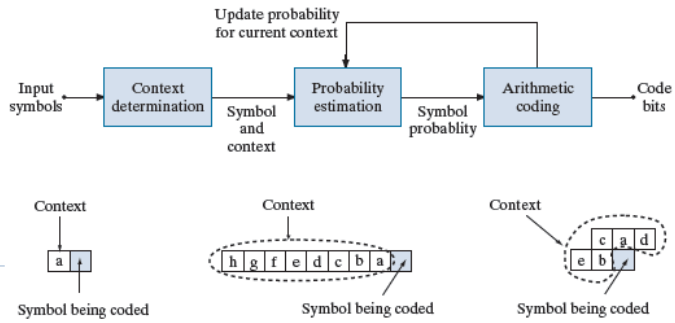


Adaptive Context-Based Arithmetic Coding

Adaptive context-based arithmetic coding (MPEG-4, H.264)

a
b c d

FIGURE 8.13
(a) An adaptive, context-based arithmetic coding approach (often used for binary source symbols). (b)–(d) Three possible context models.

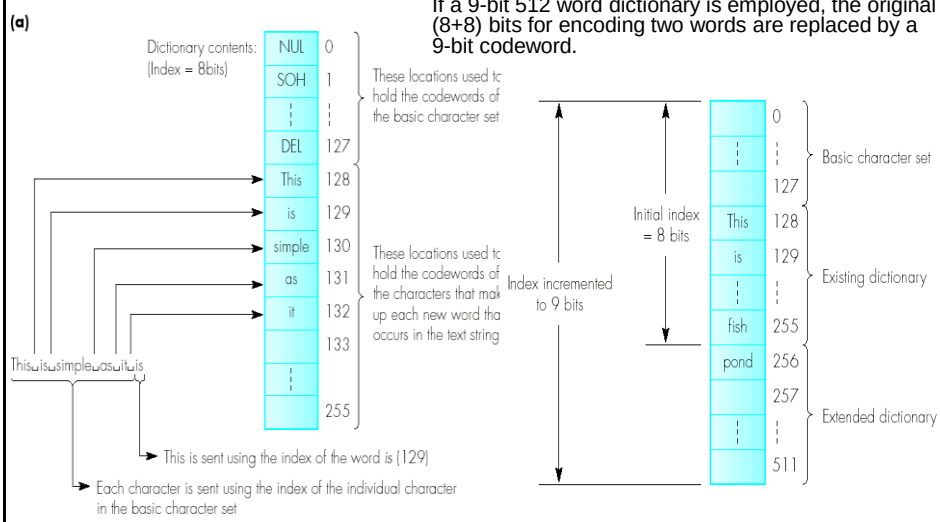


LZW Coding

- An error-free compression coding addresses the **spatial redundancy**
- **Lampel-Ziv-Welch** (**LZW**) coding: assign a fixed-length code to variable length sequence of source symbols.
- Applied for image file formats: GIF, TIFF, and PDF
- The encoder/decoder builds the **dictionary** dynamically as the text is being transferred.
- The more frequently the words stored in the dictionary occur in the text, the higher the level of compression
- Prior to sending each word in the form of single character, the encoder first checks to determine if the word is currently stored in the dictionary. If it is yes, then send only the **index** of the word stored in the dictionary.
- On detecting **insufficient locations** in the dictionary, both the decoder and encoder may double the size of the dictionary.

LZW Coding

If a 9-bit 512 word dictionary is employed, the original (8+8) bits for encoding two words are replaced by a 9-bit codeword.



LZW Coding

- **Example** : The LZW applied for encoding **images**.
- Consider 4 x 4 image of a vertical edge

  39 39 126 126

  39 39 126 126

  39 39 126 126

  39 39 126 126

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

- Each successive gray-level is concatenated with a variable (column 1 in Table 8.7) as "**currently recognized sequence**".
- The dictionary (Table 8.7) is searched for each concatenated sequence and if found, as was the case in the **1st row** of the table, it is replaced by the newly concatenated and recognized (located in the dictionary) sequence.

LZW Coding

- It is done in the column 1 row 2. No output codes are generated, nor the dictionary is altered.
- If the concatenated sequence is not found, however, the address of the **current recognized sequence** is output as the next encoded value, the concatenated but unrecognized sequence is added to the dictionary, and the **currently recognized sequence** is initialized to the **current pixel value**.
- In Table 8.7, *9 additional code* words are added.
- Reduce the original 128 bits (16 x 8) image to 90 bits (10 x 9) image

LZW Coding

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		

Initially found

not found

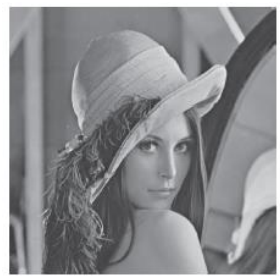
found

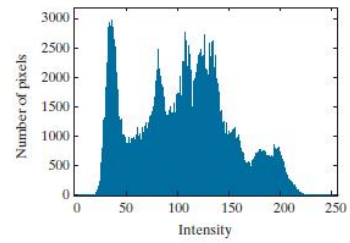
not found

LZW Coding

a b

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





Adobe Photoshop TIFF's LZW resulting a file size of 224,420 bytes

The compression ratio C = 1.28 > 1.077 (Huffman coding)

Run-length Coding

Binary image compression standard

- Image can be compressed by representing runs of identical intensities as **run-length pairs** (start of a new intensity and number of consecutive pixels) — **run-length encoding** (**RLE**) or Facsimile (FAX) coding.
- When few (or no) runs of identical pixels→ **RLE** results in data expansion
- CCITT Group 3 and 4 standards for ***binary image compression***, originally design for FAX.
- Group 3 and 4 are non-adaptive and sometimes results in data expansion (*i.e.*, with **half-tone images**).
- CCITT jointly with ISO proposed **JBIG**, an adaptive arithmetic compression.

Run-length Coding

Binary image compression standards

- In CCITT Group 3, each line of an image is encoded as a series of **variable-length code words** that represent the run lengths of the alternative white and black runs.
- If the run length ≤ 63, a **terminating code** is used.
- If the run length > 63, the largest possible **make-up code** is used.
- The **make-up code** in conjunction with the **terminating code** that represent the variable length code word.

Run-length Coding

- For document image, each **scan line** is composed of either a stream of white pixels or black pixels.
- The black and white run lengths can be coded separately using variable length coding (Huffman coding).
- Let a_j be a black run length of length j, then the **entropy** of this **black run-length source** is denoted as H_0 and the **entropy** for the **white runs** is H_1
- The approximate **run-length entropy** of the image is

$$H_{RL} = (H_0 + H_1)/(L_0 + L_1)$$

where L_0 and L_1 denote the **average lengths** of **black run** and **white run**, respectively.

Run-length Coding

1-D CCITT Group 3 compression standard

- **Modified Huffman** (**MH**) **Codes**
  - Tables of code words were produced based on the relative frequency of occurrence of the number of contiguous white and black pixels found in the scanned line.
- **Termination codes**
  - For white and black run length from 0 to 63 steps in step of 1 pel.
- **Make-up codes**
  - For run length in multiple of 64 pels.
- **Over-scanning**
  - All lines start with a minimum of one white pel.
  - First code word is always related to white pixel.
- **Examples**:
  - A run length of 12 **white pels**: 001000
  - A run length of 140 **black pels**: 128 + 12 black pels, it is encoded as 000011001000+0000111

Run-length Coding

(a) termination-codes

(a)

White run-length	Code-word	Black run-length	Code-word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000001100101
27	0100100	27	000001100111
28	0011000	28	000001100100
29	00000010	29	000001100101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	0010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101011	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111

Run-length Coding

(b) make-up codes.

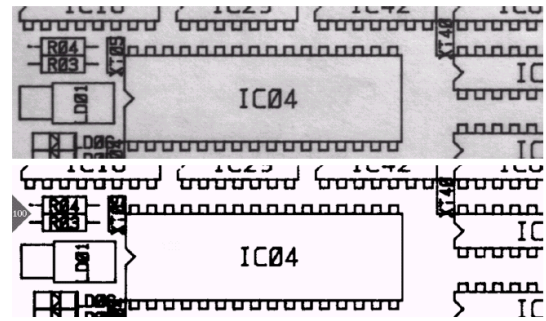
(a) cont.

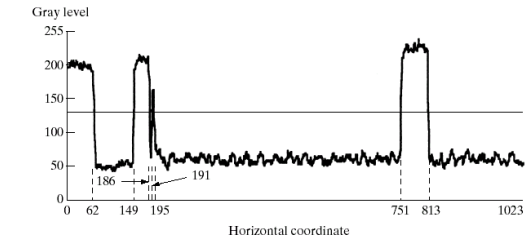
White run-length	Code-word	Black run-length	Code-word
56	01011001	56	000000101000
57	01011010	57	000000101100
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

(b)

White run-length	Code-word	Black run-length	Code-word
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000000110100
576	01101000	576	0000000110101
640	01100111	640	0000001001010
704	011001100	704	0000001001011
768	011001101	768	0000001001100
832	011010010	832	0000001001101
896	011010011	896	0000001110010
960	011010100	960	0000001110011
1024	011010101	1024	0000001110100
1088	011010110	1088	0000001110101
1152	011010111	1152	0000001110110
1216	011011000	1216	0000001110111
1280	011011001	1280	0000001010010
1344	011011010	1344	0000001010011
1408	011011011	1408	0000001010100
1472	010011000	1472	0000001010101
1536	010011001	1536	0000001011010
1600	010011010	1600	0000001011011
1664	011000	1664	0000001100100
1728	010011011	1728	0000001100101
1792	00000001000	1792	00000001000
1856	00000001100	1856	00000001100
1920	00000001101	1920	00000001101
1984	000000010010	1984	000000010010
2048	000000010011	2048	000000010011
2112	000000010100	2112	000000010100
2176	000000010101	2176	000000010101
2240	000000010110	2240	000000010110
2304	000000010111	2304	000000010111
2368	000000011100	2368	000000011100
2432	000000011101	2432	000000011101
2496	000000011110	2496	000000011110
2560	000000011111	2560	000000011111
EOL	00000000001	EOL	00000000001

Run-length Coding





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.

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

Bit-plane Coding

Bit-plane decomposition: Decompose the image into a collection of binary images

- m-bit grey-level image: $a_{m-1}a_{m-2}....a_1a_0$

  $a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + ... + a_1 2^1 + a_0 2^0$

- **Disadvantage:** small changes in grey-level can have a significant impact on the complexity of the bit-plane.

  Grey-levels: 127 = 01111111 and 128 = 1000000

- An alternative decomposition approach to reduce the effect of small grey-level variations is to represent the image by *m*-bit **Gray code**.

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

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

  $a_i \rightarrow [a_i \oplus a_{i+1}] \rightarrow g_i$

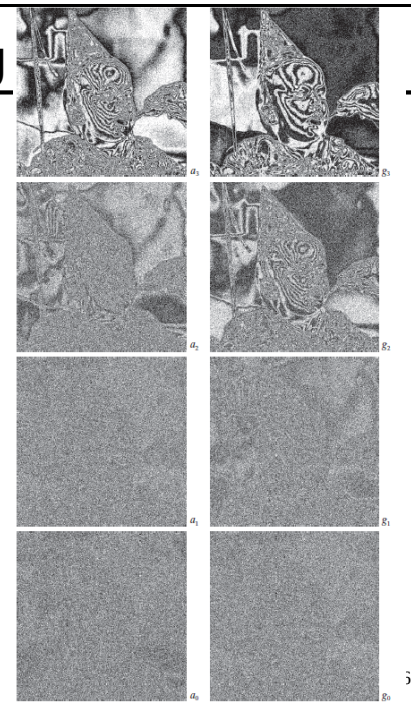
- The Gray code for 128=11000000 and 127=0100000

Bit-plane Coding

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).

Bit-plane Coding



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).

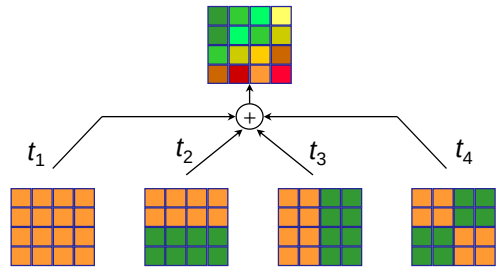
Bit-plane Coding

TABLE 8.11 JBIG2 lossless coding results for the binary and Gray-coded bit planes of Fig. 8.19(a). These results include the overhead of each bit plane's PDF representation.

Coefficient m	Binary Code (PDF bits)	Gray Code (PDF bits)	Compression Ratio
7	6,999	6,999	1.00
6	12,791	11,024	1.16
5	40,104	36,914	1.09
4	55,911	47,415	1.18
3	78,915	67,787	1.16
2	101,535	92,630	1.10
1	107,909	105,286	1.03
0	99,753	107,909	0.92

Block Transform Coding

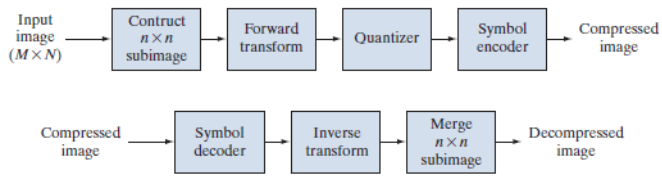
- Motivation:
  - Represent a vector (e.g. a block of image samples) as the superposition of some typical vectors (block patterns)
  - Quantize and code the coefficients
  - Can be thought of as a constrained vector quantizer



Block Transform Coding

a
b

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



Block Transform Coding

- The image $g(x,y)$ with size $N \times N$ whose **forward transform** $T(u,v)$ *is*

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

- The **inverse transform** is

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

- The transformation kernel is separable if

$$r(x,y,u,v) = r_1(x,u) r_2(y,v)$$

- The kernels for **Fourier transform** are **separable** as

$$r(x,y,u,v) = e^{-j2\pi(ux+vy)/N}/N^2 = r_1(x,u) r_2(y,v)$$

$$s(x,y,u,v) = e^{j2\pi(ux+vy)/N} = s_1(x,u) s_2(y,v)$$

Block Transform Coding

- **Walsh-Hadamard transform** (WHT) is derived from the identical kernels as

$$r(x,y,u,v) = s(x,y,u,v) = \frac{1}{N}(-1)^{\sum_{i=0}^{m-1}[b_i(x)p_i(u)+b_i(y)p_i(v)]}$$

where $N = 2^m$, the summation is performed in modulo 2 arithmetic and $b_k(z)$ is the kth bit in the binary representation of z.

- If $m = 3$, $z = 6$ (110), then $b_0(z) = 0$, $b_1(z) = 1$ and $b_2(z) = 1$
- $p_i(u)$ are defined as follows:

$p_0(u) = b_{m-1}(u)$, $p_1(u) = b_{m-1}(u) + b_{m-2}(u)$,
$p_2(u) = b_{m-2}(u) + b_{m-3}(u)$, …, $p_{m-1}(u) = b_1(u) + b_0(u)$
where the sums are performed in modulo 2 arithmetic.

Walsh-Hadamard Transform

- 1-D WHT transformation $\boldsymbol{H}$ ($\boldsymbol{H} = [g_1(x,u)]$ or $\boldsymbol{H} = [g_2(y,v)]$) is real, symmetry and orthogonal with the property $H = H^* = H^T = H^{-1}$

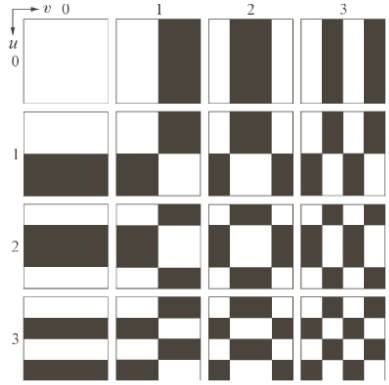
$$H_n = H_{n-1} \otimes H_1 = H_1 \otimes H_{n-1} = \frac{1}{\sqrt{2}}\begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}$$

$$H_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$H_2 = [\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]$
vector $\mathbf{u}_i$ is 1-D basis

Walsh-Hadamard Transform



Block Transform Coding

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

  or $\mathbf{G}=\sum_{u=0}^{n-1}\sum_{v=0}^{n-1}T(u,v)\mathbf{H}_{uv}$

- The image $\mathbf{G}=[g(x,y)]$ contains $n\times n$ pixels, and

$$\mathbf{S}_{uv}=\begin{bmatrix} s(0,0,0,0) & s(0,1,u,v) & \dots & s(0,n-1,u,v) \\ s(1,0,u,v) & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ s(n-1,0,u,v) & s(n-1,1,u,v) & \dots & s(n-1,n-1,u,v) \end{bmatrix}$$

- Transform coefficient masking function

$$\chi(u,v)=\begin{cases}0 & \text{if } T(u,v) \text{ satisfies a specified truncation criterion} \\ 1 & \text{otherwise}\end{cases}$$

Block Transform Coding

- An approximation of $\mathbf{G}$ can be obtained from the truncation as

$$\hat{\mathbf{G}} = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} \chi(u,v)T(u,v)\mathbf{S}_{uv}$$

- The mean square error between the $\mathbf{G}$ and approximation $\hat{\mathbf{G}}$ is

$$e_{\text{ms}} = E\left\{\left\|\mathbf{G}-\hat{\mathbf{G}}\right\|^2\right\} = E\left\{\left\|\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\mathbf{S}_{uv}\left[1-\chi(u,v)\right]\right\|^2\right\}$$

$$= \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} \sigma^2_{T(u,v)}\left[1-\chi(u,v)\right]$$

- Transformations that redistribute or pack the most information into the fewest coefficients provide the smallest reconstruction error

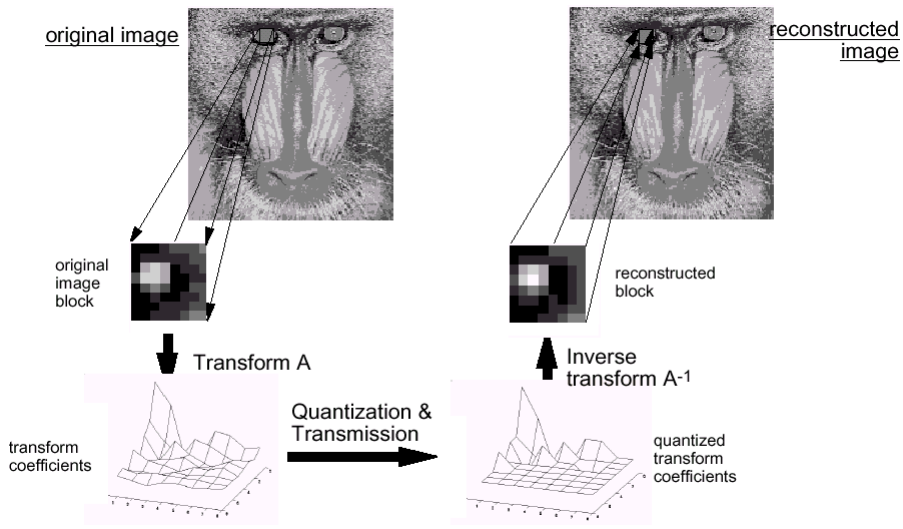
Block Transform Coding

- Transformation that redistributes or packs the most information into the fewest coefficients provide the best subimage approximations and the smallest e_{ms}.
- Figure 8.31 shows that the information packing ability of the DCT is superior than the DFT and the WHT.
- KLT *(*Karhunen-Loeve Transform*)* provides the optimal information packing capability.
- The **transformation kernels** of KLT are **data dependent,** which requires much more computation.
- DCT provides a good compromise between information packing and computation complexity.

Block Transform Coding

- The transform should achieve
  - De-correlation: minimizes the correlation among resulting coefficients, so that scalar quantization can be employed without losing too much in coding efficiency compared to vector quantization
  - Energy Compaction: compacts the energy into as few coefficients as possible
- Optimal transform
  - Karhunan Loeve Transform (KLT): signal statistics dependent
- Suboptimal transform
  - Discrete Cosine transform (DCT): nearly as good as KLT for common image signals

Block Transform Coding



Karhunen Loeve Transform (KLT)

- Also called: eigenvector, principal component, or Hotelling [1933] transforms.
- Consider a population of random vectors and the mean vector (*E* is the expectation operator):

$$x = [x_1, x_2, \ldots, x_n]^T, m_x = E\{x\}$$

- The autocovariance matrix is defined as

$$C_x = E\{(x - m_x)(x - m_x)^T\}$$

- Given M vector samples, sample average is used to replace the expectation operator.

Karhunen Loeve Transform (KLT)

$$m_x = \frac{1}{M}\sum_{k=1}^{M} x_{(k)}$$

$$C_x = \frac{1}{M}(\sum_{k=1}^{M} x_{(k)}x_{(k)}^T - m_x m_x^T)$$

- Eigen analysis of C_x, $C_x = A^T \Sigma A = \sum_{i=1}^{n} \lambda_i a_i a_i^T$
  where A is an orthonormal matrix whose rows are formed from eigenvectors of C_x, i.e., Σ is the diagonal matrix consists of the "**ordered**" eigenvalues (λ_j).

Karhunen Loeve Transform (KLT)

- Basis vectors $\{a_i\}$ are the eigenvectors of the covariance matrix of the source.
- KLT projects the input vector onto the "orthonormal principal components (PCs)". $y = A(x - m_x)$
- C_y is a diagonal matrix whose elements along the main diagonal are $\{\lambda_i\}$. C_y and C_x have the same eigenvalues.
- KLT is the optimum transform for de-correlation. It is optimal in the sense that it minimizes the mean square error between x and the approximations x' (keeping only k largest $\{\lambda_i\}$).
- **KLT basis vectors are signal dependent.**

From DFT to DCT

- DFT of any real and symmetric sequence contains only real coefficients corresponding to the cosine terms of the series.
- Construct a new symmetric sequence $y(n)$ of length $2N$ out of $x(n)$ of length N.

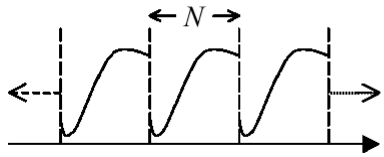
$$y(n) = x(n), 0 \le n \le N-1,$$

$$y(n) = x(2N-1-n), N \le n \le 2N-1.$$

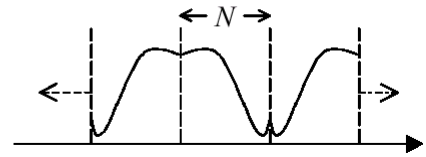
- $Y(n)$ is symmetrical about $n = N - (1/2)$.

From DFT to DCT

- DCT has a higher compression ration than DFT
  - DCT avoids the generation of spurious spectral components (Gibbs Phenomenon)



DFT periodicity



DCT periodicity

From DFT to DCT

$$
\begin{aligned}
Y(k) &= \sum_{n=0}^{2N-1} y(n) W_{2N}^{(n+\frac{1}{2})k}, \\
&= \sum_{n=0}^{N-1} y(n) W_{2N}^{(n+\frac{1}{2})k} + \sum_{n=N}^{2N-1} y(n) W_{2N}^{(n+\frac{1}{2})k} \\
&= \sum_{n=0}^{N-1} x(n) W_{2N}^{(n+\frac{1}{2})k} + \sum_{n=N}^{2N-1} x(2N-1-n) W_{2N}^{(n+\frac{1}{2})k} \\
&= \sum_{n=0}^{N-1} x(n) W_{2N}^{(n+\frac{1}{2})k} + \sum_{n=0}^{N-1} x(n) W_{2N}^{[2N-(n+\frac{1}{2})k]} \\
&= \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi(2n+1)k}{2N},
\end{aligned}
$$

$$
0 \le k \le 2N-1, and\ W_{2N} = e^{-j\frac{2\pi}{2N}}
$$

Discrete Cosine Transform

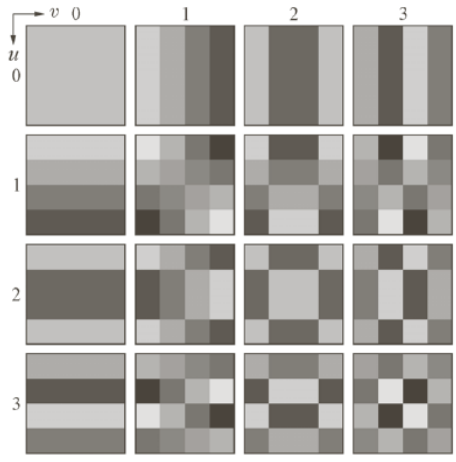
- **Discrete Cosine Transform** (DCT) is derived from the identical kernels as

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

where

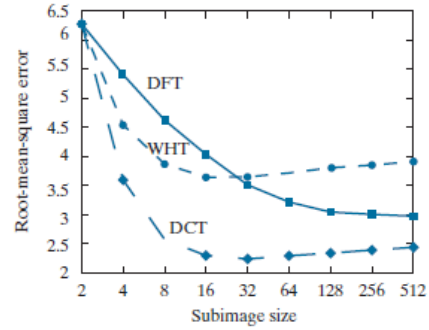
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{1}{N}} & \text{for } u = 1,2,\ldots N-1 \end{cases}$$

4x4 DCT Basis Function



Block Transform Coding

FIGURE 8.23 Reconstruction error versus subimage size.



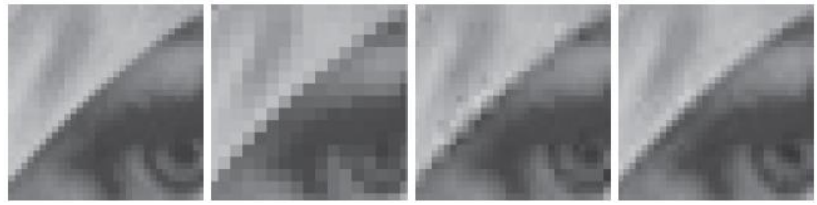
Images are subdivided so that the correlation (redundancy) between adjacent subimages is reduced to some acceptable level

Block Transform Coding

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).

Block Transform Coding



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).

Zonal Coding

- The transform coefficients of **maximum variance** carry the most image information and should be retained.
- The zonal sampling process is to multiply the $T(u, v)$ by the corresponding element in a **zonal mask**.
- The coefficients retained must be **quantized** and **coded**. The levels of quantization for each coefficients are different which is proportional to $\log_2 \sigma^2_{T(u,v)}$
- Based on the rate-distortion theory, a Gaussian random variable of variance σ^2 cannot be represented by less than $1/2 \log_2(\sigma^2/D)$ bits and reproduced with a mean-square error less than D.
- The information content of a Gaussian random variable is proportional to $\log_2(\sigma^2/D)$.

Zonal Coding

a b
c d

FIGURE 8.26
A typical
(a) zonal mask,
(b) zonal bit allocation,
(c) threshold mask, and
(d) thresholded coefficient ordering sequence.
Shading highlights the coefficients that are retained.

1	1	1	1	1	0	0	0
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	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

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

Multiply each $T(u, v)$ by the corresponding element in zonal mask

Threshold Coding

- The location of the transform coefficients retained for each subimage vary from one subimage to another. — **adaptive transform coding**
- For any subimage, the transform coefficients of **largest magnitude** make the most significant contribution to reconstructed subimage quality.
- Because the **locations** of **maximum coefficients** vary from one image to another, the element of $\gamma(u, v)T(u, v)$ normally are recorded to form a 1-D run-length sequence.
- These runs normally are **run-length coded**.

Threshold Coding

- Three ways to threshold the coefficients:
  (1) A single global threshold.
  The level of compression differs from image to image.
  (2) Different threshold used for each subimage.
  ***N*-largest coding**: the same number of coefficients is discarded for each subimage. **Constant coding rate**.
  (3) The threshold can be varied as a function of location of each coefficient. It results in **a variable code rate**.
  (4) The **thresholding** and **quantization** can be combined by

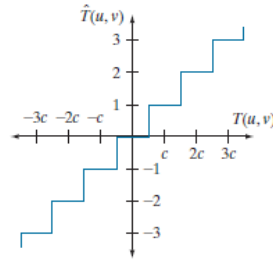
$$\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$$

where $\hat{T}(u,v)$ is a thresholded and quantized approximation of $T(u,v)$, and $Z(u,v)$ is an element of the **transformation normalization array.**

Threshold Coding

a b

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

Threshold Coding

- The de-normalization (decompression) results are
$$\dot{T}(u,v) = \hat{T}(u,v)Z(u,v)$$
- Assume $Z(u,v) = c$ and $\hat{T}(u,v) = k$ then
$$kc - c/2 \leq T(u,v) \leq kc + c/2$$
- If $Z(u,v) > 2T(u,v)$ then $T(u,v)$ is completely truncated and discarded.
- $\hat{T}(u,v)$ is coded by variable length coding (i.e., Huffman code), of which the code length increases with the magnitude of k.

Threshold Coding

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

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**.

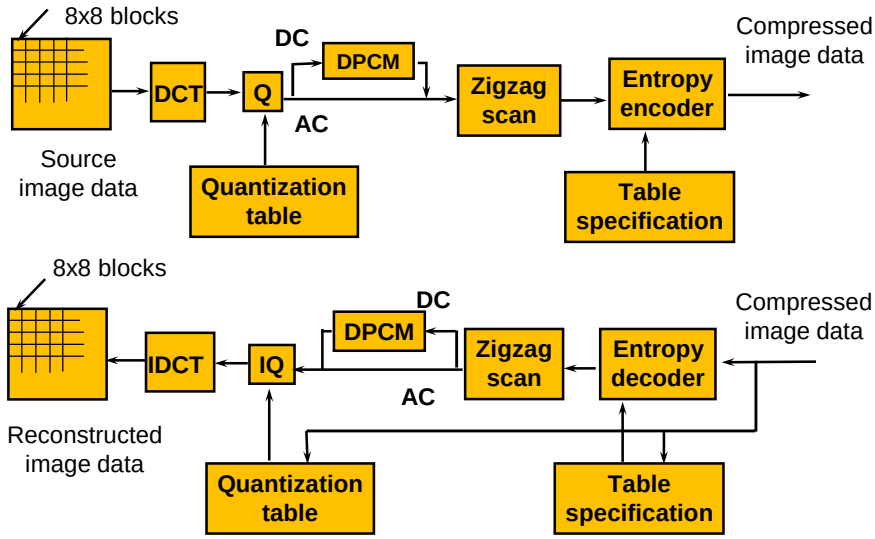
JPEG (Joint Picture Expert Group)

- JPEG = Joint Photographic Expert Group Joint standards committee of ITU-T and ISO
- Flexible standard for monochrome and color image compression
- Intraframe coding scheme, optimized for still image
- Flexible picture size
- Coding of color components separately, arbitrary color spaces possible, best compression for Y/R-Y/B-Y
- Variable compression ratio
- Compression 24:1 for ITU-R 601 images without loss of quality visually

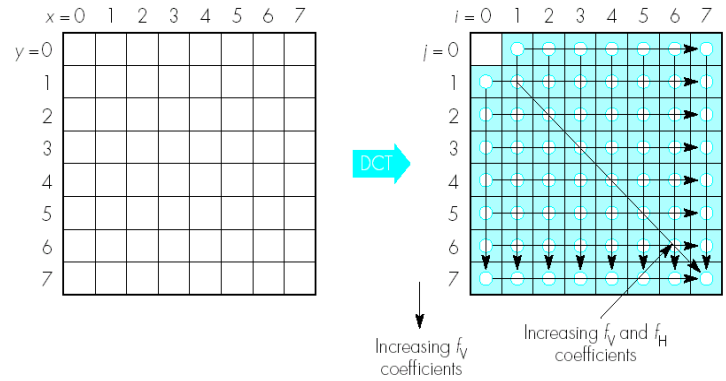
JPEG (Joint Picture Expert Group)

- Since 1986, ITU and ISO called a joint meeting to establish the standard for **multilevel color still image**. In 1987 IEC joined in.
- Schedule
  - 1988: select the DCT-based method
  - 1988-1990: simulating, testing, and documenting
  - 1991: draft
  - 1992: international standard

JPEG



JPEG



$P[x, y]$ = 8 × 8 matrix of pixel values

$F[i, j]$ = 8 × 8 matrix of transformed values/spatial frequency coefficients

In $F[i, j]$: □ = DC coefficient ■ = AC coefficients

f_H = horizontal spatial frequency coefficient

f_V = vertical spatial frequency coefficient

JPEG

DCT coefficients

120	60	40	30	4	3	0	0
70	48	32	3	4	1	0	0
50	36	4	4	2	0	0	0
40	4	5	1	1	0	0	0
5	4	0	0	0	0	0	0
3	2	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantizer

Quantized coefficients

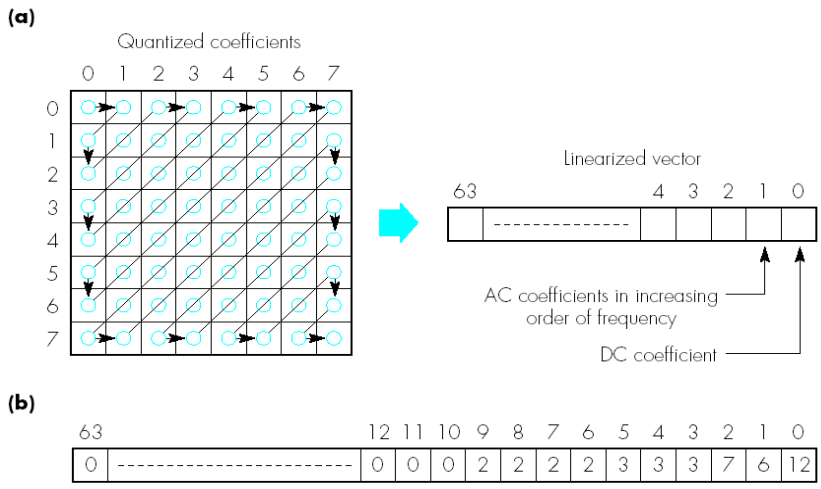
12	6	3	2	0	0	0	0
7	3	2	0	0	0	0	0
3	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

10	10	15	20	25	30	35	40
10	15	20	25	30	35	40	50
15	20	25	30	35	40	50	60
20	25	30	35	40	50	60	70
25	30	35	40	50	60	70	80
30	35	40	50	60	70	80	90
35	40	50	60	70	80	90	100
40	50	60	70	80	90	100	110

Quantization table

JPEG

JPEG-Entropy Encoding



JPEG

Example 8.17

An 8×8 sub image

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.

Level shift –128

−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

Example (continue)

DCT → T(u,v)

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

DCT coefficients Normalization

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

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

$$= \text{round}\left[\frac{-415}{16}\right] = -26$$

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

JPEG

Example (continue)

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

Zigzag ordering

[−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]

DC coefficient is DPCM **coded**

AC coefficients are **run-length coded**

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

The compression ratio = 512/92 = 5.61

JPEG

Example (continue)

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

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

$\dot{T}(0,0) = \hat{T}(0,0)Z(0,0)$

$= (-26)(16) = -416$

DCT coefficient De-normalization

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

JPEG

Example (continue)

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

IDCT

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

Level shift +128

JPEG

Original

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

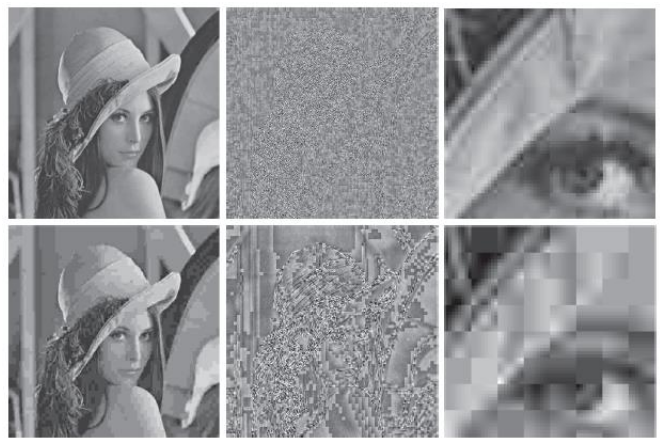
Reconstructed

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

Difference

−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



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.

JPEG

Original

JPEG Encoded

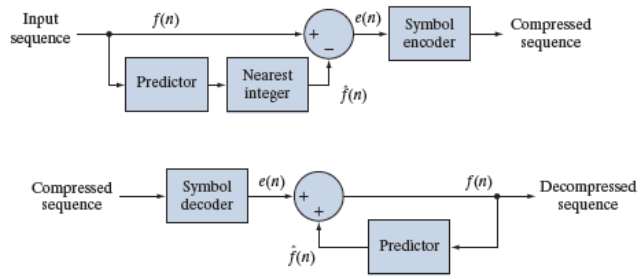
Predictive Coding

- Eliminate **inter-pixel redundancy** using **predictor**.
- The predictor generates the anticipated value of current pixel f_n based on past pixels.
- The output of the predictor is round to the nearest integer denoted as
- The **predictor error** is $e_n = f - \hat{f}_n$
- Various local, global, and adaptive methods can be used to generate $\hat{f}_n$
- The prediction is formed by a linear combination of *m* previous pixels as $\hat{f}_n = \text{round}\left(\sum_{i=1}^{m} \alpha_i f_{n-i}\right)$

Lossless Predictive Coding

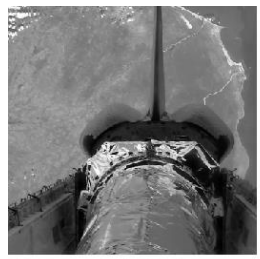
a
b

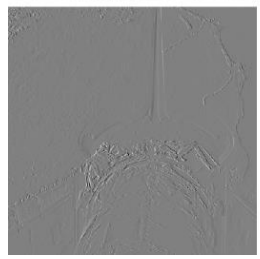
FIGURE 8.30
A lossless predictive coding model: (a) encoder; (b) decoder.

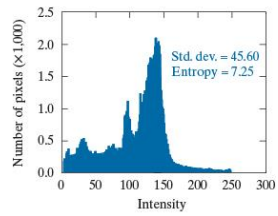


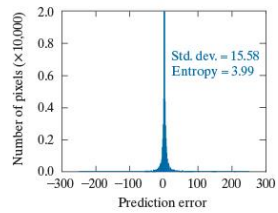
Lossless Predictive Coding

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









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.)

Predictive Coding in JPEG-Lossless

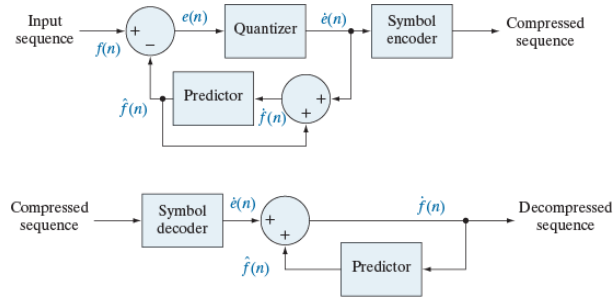
	C	B	
	A	X	

Selection-value	Prediction
0	no prediction
1	A
2	B
3	C
4	A + B - C
5	A + (B - C) / 2
6	B + (A - C) / 2
7	(A + B) / 2

*** JPEG lossless coding typically produce ~2:1 compression**

Lossy Predictive Coding

- **Lossy compression** techniques compromise the accuracy of the reconstructed image in exchange for increased compression.
- The distortion is tolerable (not visually apparent), the compression ratio may be significant.



a
b

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

Lossy Predictive Coding

- A quantizer is inserted.
- **The predictions generated by the encoder and the decoder must be equivalent**.
- Replace the lossy encoder's predictor within a feedback loop, where its input denoted as, $\dot{f}_n$, is generated as a function of past predictions and the corresponding quantized errors, i.e., $\dot{f}_n = \dot{e}_n + f_n$
- **Delta modulation**, the predictor is $\hat{f}_n = \alpha \dot{f}_n$

  and the quantizer is

$$\dot{f}_n = \begin{cases} +\zeta & \text{for } e_n > 0 \\ -\zeta & \text{otherwise} \end{cases}$$

  The output quantizer can be represented by a single bit.

Delta Modulation

- **Delta modulation**, the predictor is $\hat{f}_n = \alpha \dot{f}_n$
  and the quantizer is

$$\dot{f}_n = \begin{cases} +\zeta & \text{for } e_n > 0 \\ -\zeta & \text{otherwise} \end{cases}$$

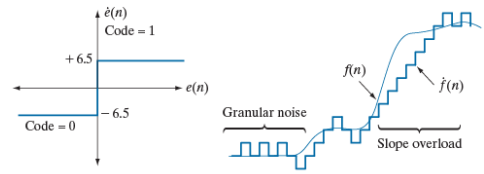
The output quantizer can be represented by a single bit.

- **Example**: Input sequence: {14, 15,14,15,13,15,15,14, 20, 26, 27, 28, 27, 27, 29, 37,47, 62, 75, 77, 78, 79, 80, 81, 81, 82, 82……}
- $\alpha = 1$ and ζ= 6.5
- When ζ is too small, the **slope overload** occurs, ζ is too large, the **granular noise** appears

Delta Modulation

a b
c

FIGURE 8.39
An example of delta modulation.

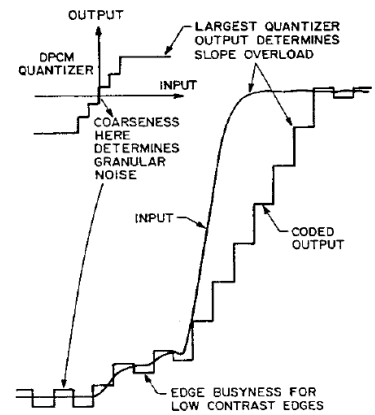


Input		Encoder				Decoder		Error
n	$f(n)$	$\hat{f}(n)$	$e(n)$	$\dot{e}(n)$	$\dot{f}(n)$	$\hat{f}(n)$	$\dot{f}(n)$	$f(n)-\dot{f}(n)$
0	14	–	–	–	14.0	–	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	− 5.5
2	14	20.5	− 6.5	− 6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	− 5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.5	17.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Delta Modulation

slope overload

Edge busyness ***Granular noise***



Optimal Predictors

- Optimal predictor minimizes the encoder's mean prediction error

$$E\{e_n^2\} = E\left\{\left[f_n - \hat{f}_n\right]^2\right\}$$

where

$\dot{f}_n = \dot{e}_n + \hat{f}_n \approx e_n + \hat{f}_n = f_n$ and $\hat{f}_n = \sum_{i=1}^{m} \alpha_i \dot{f}_{n-i} \cong \sum_{i=1}^{m} \alpha_i f_{n-i}$

- $\partial E\{e_n^2\}/\partial \alpha_i = 0$ where $E\{e_n^2\} = E\left\{\left[f_n - \sum_{i=1}^{m} \alpha_i f_{n-1}\right]^2\right\}$

Optimal Predictors

- $\boldsymbol{\alpha} = \mathbf{R}^{-1}\mathbf{r}$ where $\mathbf{R}^{-1}$ is the inverse of the $m \times m$ autocorrelation matrix

$$\mathbf{R} = \begin{bmatrix} E\{f_{n-1}f_{n-1}\} & E\{f_{n-1}f_{n-2}\} & \cdots & E\{f_{n-1}f_{n-m}\} \\ E\{f_{n-2}f_{n-1}\} & E\{f_{n-2}f_{n-2}\} & \cdots & E\{f_{n-2}f_{n-m}\} \\ \vdots & \vdots & \ddots & \vdots \\ E\{f_{n-m}f_{n-1}\} & E\{f_{n-m}f_{n-2}\} & \cdots & E\{f_{n-m}f_{n-m}\} \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} E\{f_n f_{n-1}\} \\ E\{f_n f_{n-2}\} \\ \vdots \\ E\{f_n f_{n-m}\} \end{bmatrix} \qquad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix}$$

Lossy Predictive Coding

- The variance of the prediction error is
  $\boldsymbol{\sigma}_e^2 = \boldsymbol{\sigma}^2 - \boldsymbol{\alpha}^T \mathbf{r} = \boldsymbol{\sigma}^2 - \sum_i E\{f_n f_{n-1}\}\alpha_i$
- The generalized fourth order prediction

$$\hat{f}(x,y) = \alpha_1 f(x,y-1) + \alpha_2 f(x-1,y-1) + \alpha_3 f(x-1,y) + \alpha_4 f(x+1,y-1)$$

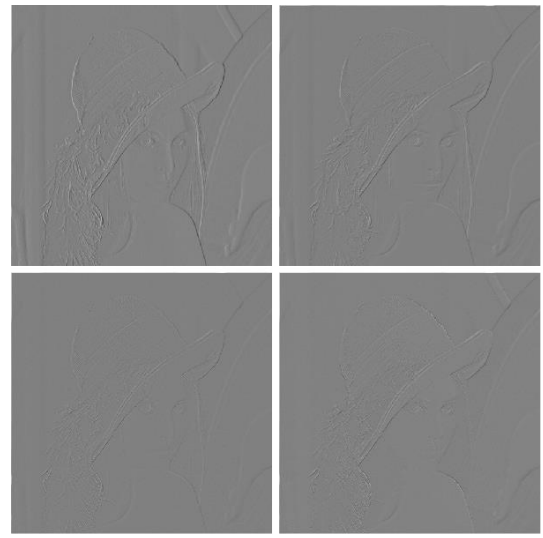
- $\alpha_1 = \rho_h$, $\alpha_2 = -\rho_v\rho_h$,$\alpha_3 = \rho_v$, $\alpha_4 = 0$
  where ρ_h and ρ_v are the horizontal and vertical correlation coefficients.

Lossy Predictive Coding

- ***Example*** : Consider four DPCM predictors
- 1 $\hat{f}(x,y) = 0.97 f(x,y-1)$
- 2 $\hat{f}(x,y) = 0.5 f(x,y-1) + 0.5 f(x-1,y)$
- 3 $\hat{f}(x,y) = 0.75 f(x,y-1) + 0.75 f(x-1,y) - 0.5 f(x-1,y-1)$
- 4 $\hat{f}(x,y) = \begin{cases} 0.97 f(x,y-1) & \text{if } \Delta h \le \Delta v \\ 0.97 f(x-1,y) & \text{otherwise} \end{cases}$

where $\Delta h = |f(x-1,y) - f(x-1,y-1)|$ horizontal gradients
$\Delta v = |f(x,y-1) - f(x-1,y-1)|$ vertical gradients

Lossy Predictive Coding



a b
c d
FIGURE 8.40
A comparison of four linear prediction techniques.

Optimal Quantization

$t = q(s)$ is an odd function

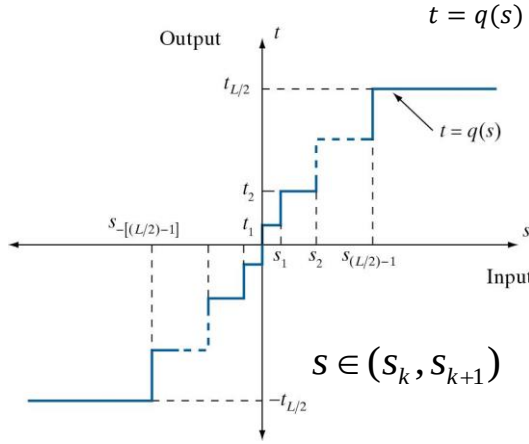


FIGURE 8.41
A typical quantization function.

$$s \in (s_k, s_{k+1}) \quad s \xrightarrow{is\ mapped} t_k$$

s_i: **decision level**, t_i: **reconstruction level**

Optimal Quantization

Example $s : 0.0 \sim 10.0 \rightarrow s^* = \{t_k : k = 1 \sim 256\}$

$$s_k = \frac{10(k-1)}{256}, k = 1,..., 257$$

$$t_k = s_k + 5/256, k = 1,..., 256$$

- Quantization interval $\theta = t_k - t_{k-1} = s_k - s_{k-1}$
- **Zero memory quantizer** : one input sampled at one time output value depends only on that input.

Optimal Quantization (Lloyd-Max Quantizer)

- Let s be a real random variable with continuous probability density function $P(s)$
- **Goal:** to find the decision levels s_k and reconstruction level t_k for an L-level quantizer such that the **MSE**.

$$\varepsilon = E[(s - s^*)^2] = \int_{s_1}^{s_{L+1}} (s - s^*)^2 P(s) ds$$

is minimized. So, we minimize

$$\varepsilon = \sum_{i=1}^{L} \int_{s_i}^{s_{i+1}} (s - t_i)^2 P(s) ds \quad \text{by setting} \quad \frac{\partial \varepsilon}{\partial t_k} = \frac{\partial \varepsilon}{\partial s_k} = 0$$

under the condition that

$$s_i = \begin{cases} 0 & i = 0 \\ \dfrac{t_i + t_{i+1}}{2} & i = 1,2,....\dfrac{L}{2} - 1 \\ \infty & i = L/2 \end{cases}$$

and $s_{-i} = -s_i, t_{-i} = -t_i$

Optimal Quantization (Lloyd-Max Quantizer)

TABLE 8.13 Lloyd-Max quantizers for a Laplacian probability density function of unit variance.

Levels	2		4		8	
	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	

Temporal Predictive Coding

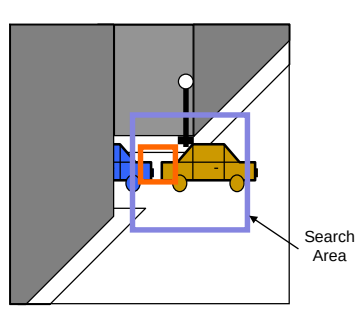
- Successive frames in a video sequence often are very similar —— **Temporal redundancy**.
- **Temporal predictive coding** needs to consider

  1) **Tracking object movement** (**motion vectors**) and 2) **compensating** it during the prediction and differencing process.

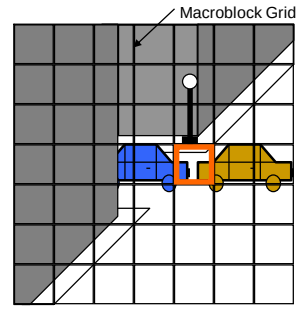
  Motion estimation: Find the movement of a small segment between two successive frames.

  Motion compensation: The difference between the predicted and actual positions of the moving segment involved need to be compensated (or encoded).

Motion Estimation

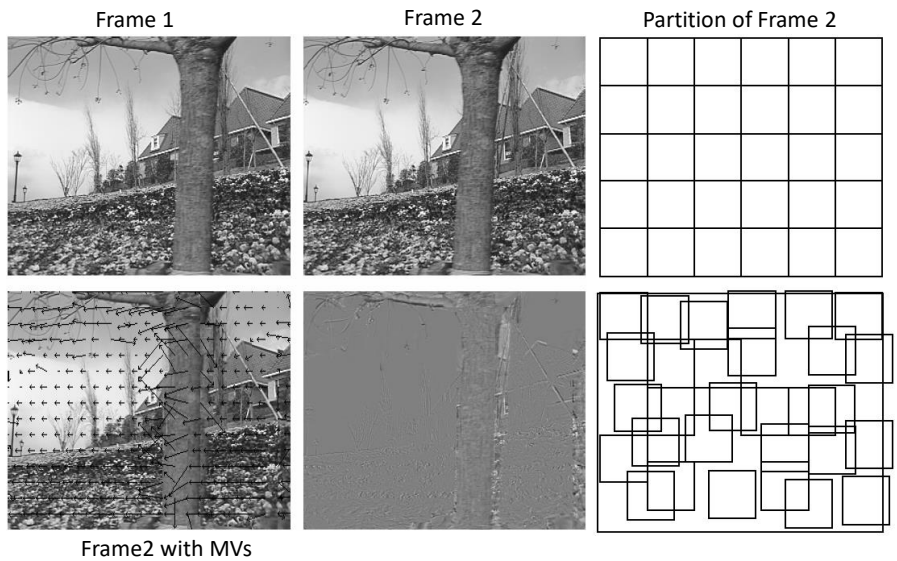


Previous Picture.
Within the search area, a good match is found for this moving object. Encoder sends appropriate forward motion vector.



Current Picture.
Current MB is shown with heavy outline. Since a match is found, this MB is intercoded.

Motion-Compensated Prediction

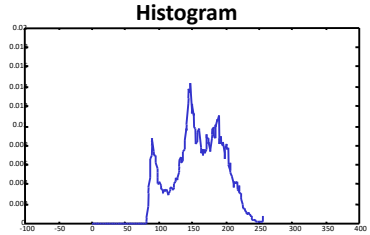


Motion-Compensated Prediction

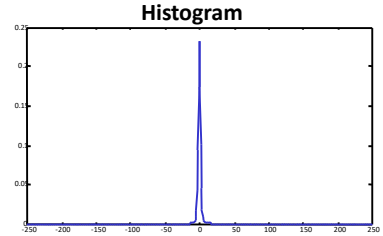
One Frame of Original Image Pair

Prediction Error

Histogram



Histogram

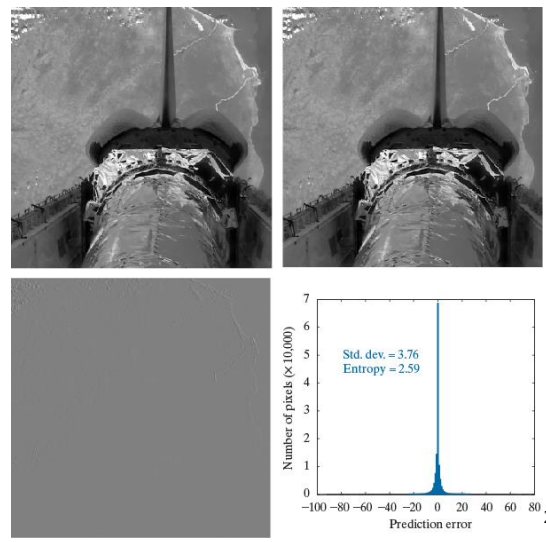


Temporal Predictive Coding

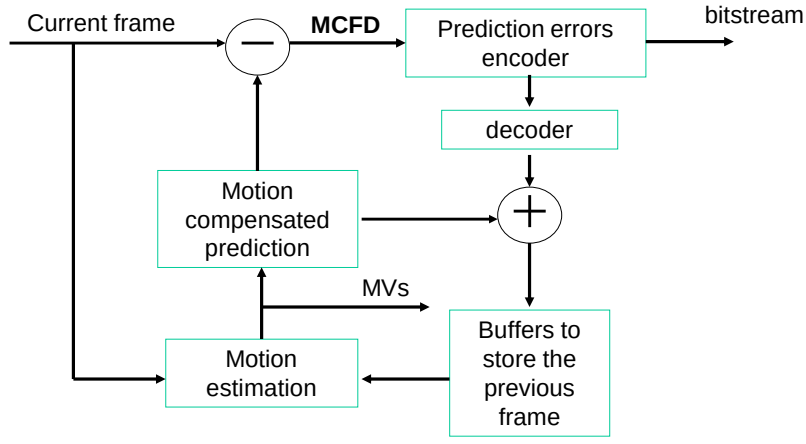
$$\hat{f}(x,y,t) = \text{round}[\alpha f(x,y,t-1)]$$

a b
c d

FIGURE 8.32
(a) and (b) Two views of Earth from an orbiting space shuttle video. (c) The prediction error image resulting from Eq. (8-35). (d) A histogram of the prediction error. (Original images courtesy of NASA.)



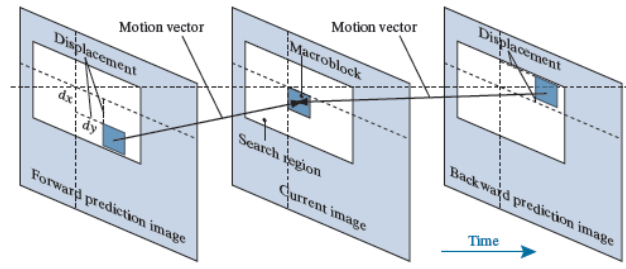
Motion Estimation & Compensation



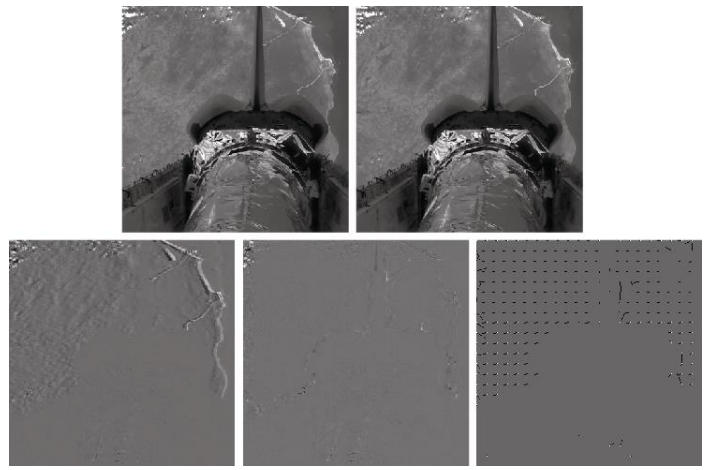
MCFD = motion compensated frame difference

Motion Estimation & Compensation

FIGURE 8.33 Macroblock motion specification.



Motion Estimation & Compensation



a b
c d e

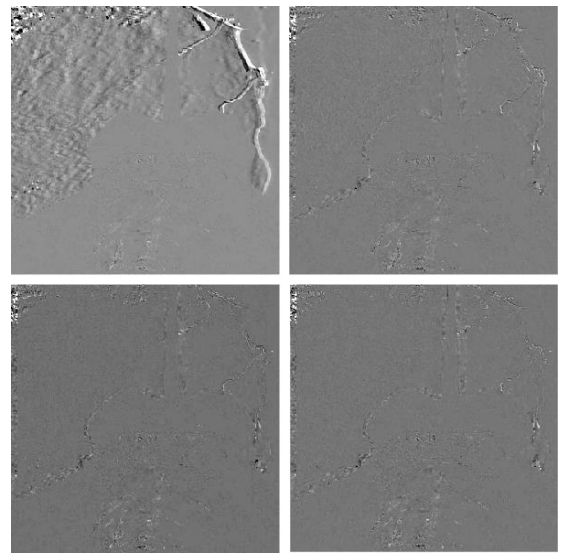
FIGURE 8.34 (a) and (b) Two views of Earth that are thirteen frames apart in an orbiting space shuttle video. (c) A prediction error image without motion compensation. (d) The prediction residual with motion compensation. (e) The motion vectors associated with (d). The white dots in (e) represent the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

131

Motion Estimation & Compensation

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.)



Motion Estimation & Compensation

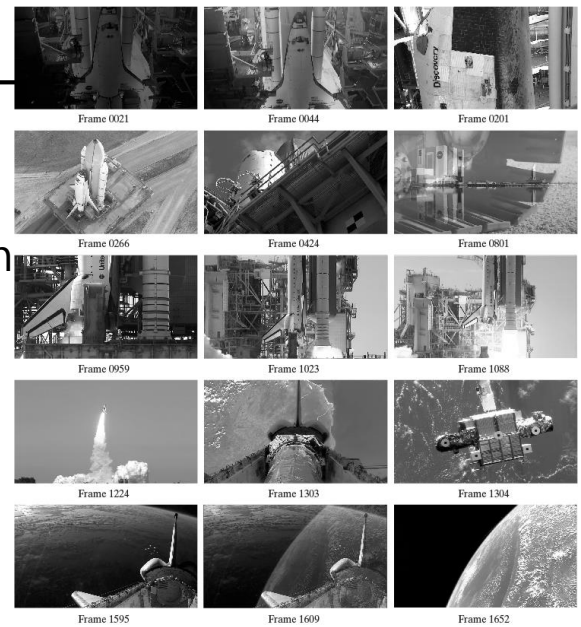


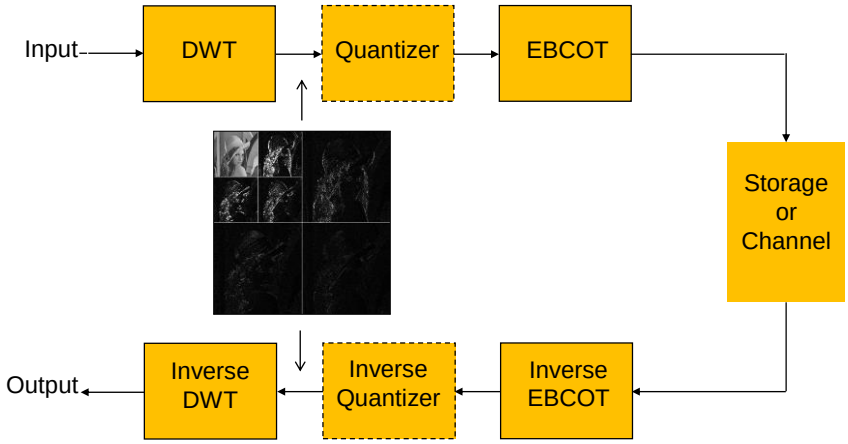
FIGURE 8.37 Fifteen frames from an 1829-frame, 1-minute NASA video. The original video is in HD full color. (Courtesy of NASA.)

JPEG-2000

- JPEG 2000 provides increased flexibility in both the compression of still image and access the compressed data.
- Portion of a JPEG 2000 compressed image can be extracted for retransmission.
- Coefficients quantization is adapted to individual scales and subbands.
- The quantized coefficients are arithmetically coded on a bit-plane basis.

JPEG-2000

JPEG-2000: Discrete Wavelet Transform (DWT) + Embedded Block Coding with Optimum Truncation (EBCOT)



JPEG-2000 vs. JPEG

0.125bpp, JPEG

0.125bpp, JPEG-2000

JPEG-2000 vs. JPEG

0.25bpp, JPEG

0.25bpp, JPEG-2000

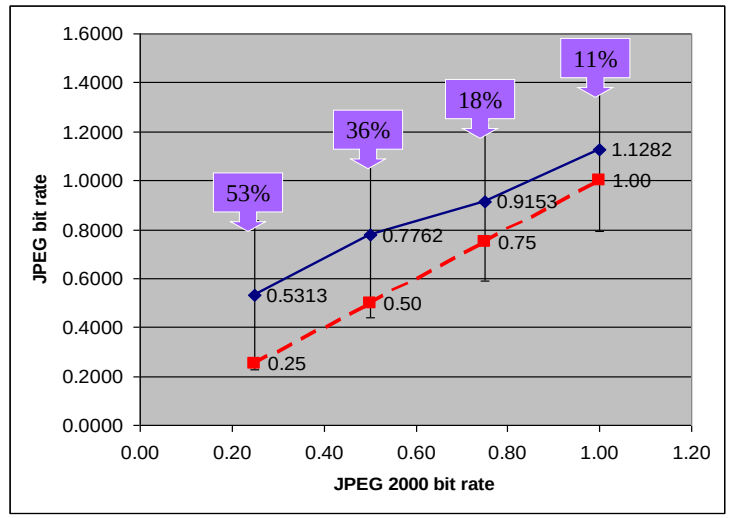
JPEG-2000 vs. JPEG

Compression Rate: 130:1

JPEG2000(7KB, 5922 bytes)

JPEG (7KB, 6220 bytes)

JPEG-2000 vs. JPEG



6 observers
6 color images
300 dpi