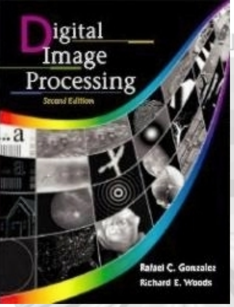


Chapter 8 Image Compression

- 8.1 Fundamental
- 8.2 Image compression method
- 8.3 Information Theory
- 8.4 Error-Free Compression
- 8.5 Lossy Compression
- 8.6 Image Compression Fundamental

8.1 Image Compression -Fundamental

- Image compression address the problem of reducing the amount of data required to represent a digital image.
- Removal 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 redundancy***, ***inter-pixel redundancy***, ***psychovisual redundancy***

8.1 Image Compression -Fundamental

- Data compression removes ***data redundancy***
- Let n_1 and n_2 denote the number of information carrying units in two data sets that represent the same information.
- The ***relative data redundancy*** R_D is

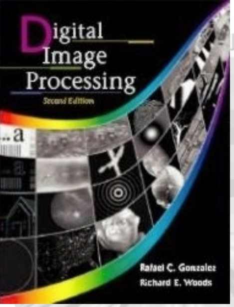
$$R_D = 1\text{-}1/C_R$$

  where C_R is the ***compression ratio*** $C_R = n_1/n_2$
- $n_1 = n_2$ $R_D = 0$, *and* $C_R = 1$, *no data redundancy*
- $n_1 >> n_2$ *and* $C_R >> 1$, $R_D \cong 1$ *highly redundant data.*

8.1 Image Compression -Fundamental

- ***Coding 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 and that each r_k occurs with the probability $p_r(r_k) = n_k/n$, $k=0,1.,,,,L-1$, where L is the number of gray-level.
- 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*

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



8.1 Image Compression -Fundamental

r_k	$p_r(r_k)$	**Code 1**	$l_1(r_k)$	**Code 2**	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

TABLE 8.1 Example of variable-length coding.

$$L_{avg} = \sum_{k=0}^{7} l_2(r_k) p_r(r_k)$$

$$= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + ... + 6(0.02)$$

$$= 2.7\ bits$$

8.1 Image Compression-Fundamental

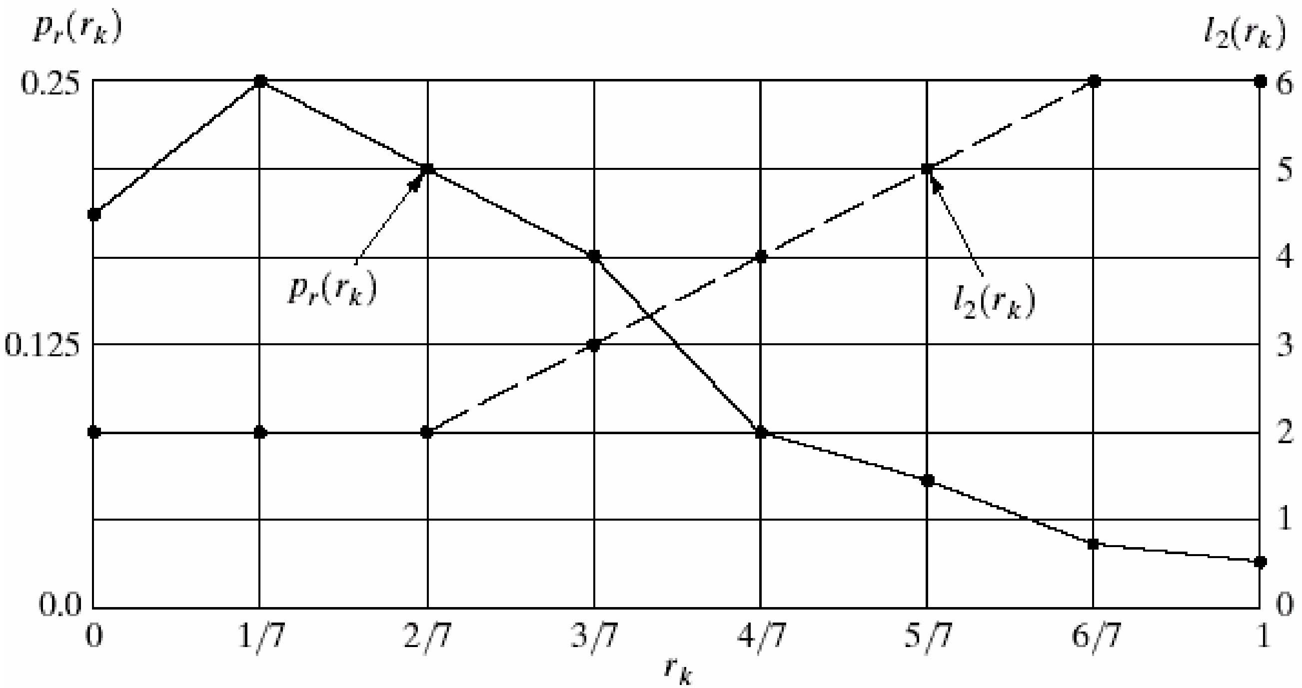


FIGURE 8.1 Graphic representation of the fundamental basis of data compression through variable-length coding.

$l_2(r_k)$ and $p_r(r_k)$ is inverse proportional

8.1 Image Compression-Fundamental

- ***Interpixel redundancy***

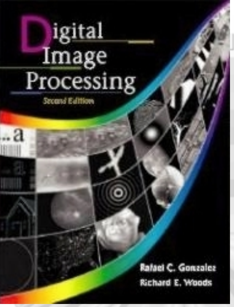
  Figures 8.2(e) and (f) show the respective ***autocorrelation coefficients*** as

$$\gamma(\Delta n)=A(\Delta n)/A(0)$$

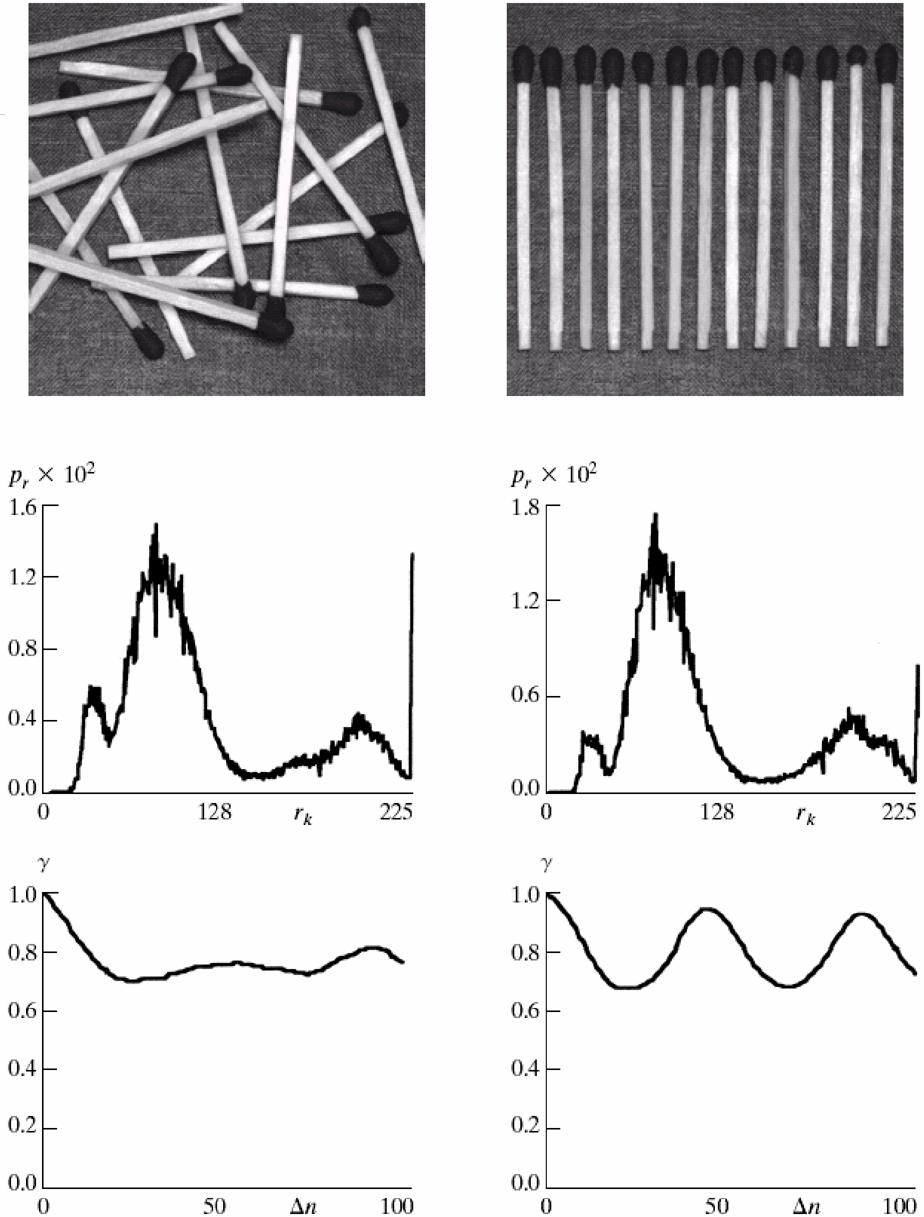
  where

$$A(\Delta n)=\frac{1}{N-\Delta n}\sum_{y=0}^{N-1-\Delta n} f(x,y)\,f(x,y+\Delta n)$$

- Spatial redundancy, inter-pixel redundancy
- The value of any given pixel can be predicted from the values of its neighbors.



8.1 Image Compression-Fundamental

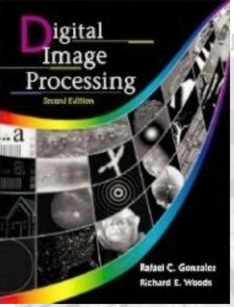


a	b
c	d
e	f

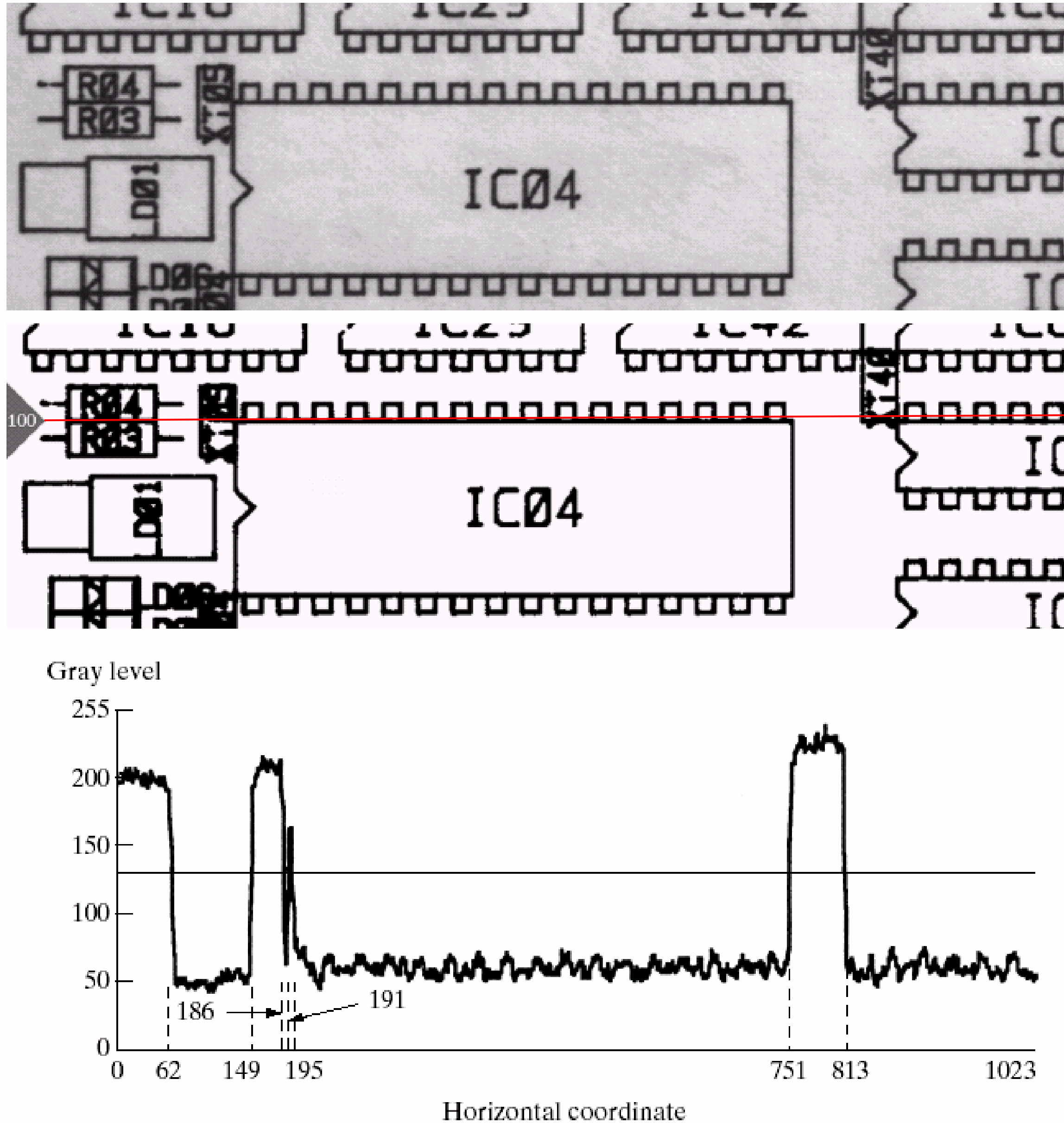
FIGURE 8.2 Two images and their gray-level histograms and normalized autocorrelation coefficients along one line.

8.1 Image Compression-Fundamental

- To reduce interpixel redundancy, 2-D pixel array is transformed into a more efficient format (less number of bits).
- This format can be reversible mapped back to the original 2-D pixel array --- ***reversible mapping.***



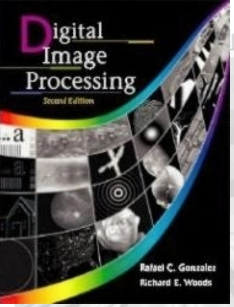
8.1 Image Compression-Fundamental



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

a
b
c
d

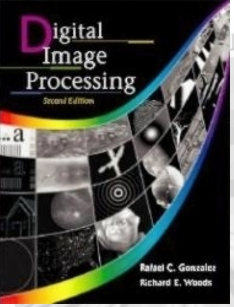
FIGURE 8.3 Illustration of run-length coding: (a) original image. (b) Binary image with line 100 marked. (c) Line profile and binarization threshold. (d) Run-length code.



8.1 Image Compression-Fundamental

- ***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.***
- ***IGS*** *(**Improved Gray-Scale Quantization**)*: adding each pixel a ***pseudo-random number***, which is generated from the low-order bits of neighboring pixels, before quantizing the result.

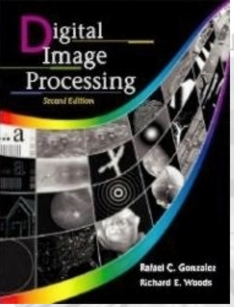


8.1 Image Compression-Fundamental

IGS: Improved Gray-Scale Quantization

a b c

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



8.1 Image Compression-Fundamental

Pixel	Gray Level	Sum	IGS Code
$i-1$	N/A	0000 0000	N/A
i	0110 1100	0110 1100	0110
$i+1$	1000 1011	1001 0111	1001
$i+2$	1000 0111	1000 1110	1000
$i+3$	1111 0100	1111 0100	1111

TABLE 8.2
IGS quantization procedure.

1) The sum (initially zero) is formed from the current 8-bit gray-level value and the ***four least significant bits*** of a previously generated sum.

2) The ***four most significant bits*** of the resulting sum are used as the coded pixel values

8.1 Image Compression-Fundamental

- ***Fidelity Criteria:***

  (a) Objective Fidelity Criteria

  (b) Subjective Fidelity Criteria
- Let $f(x, y)$ be the original image and $f'(x, y)$ be the decompressed image
- The error is defined as $e(x,y)=f'(x, y)- f(x, y)$
- Total error between two images (*size* $M \times N$):

$$\sum_x \sum_y [f'(x, y)- f(x, y)]$$

- The root-mean error e_{rms}:

$$e_{rms} =[1/MN\{\sum_x \sum_y [f'(x,y)- f(x,y)]^2\}]^{1/2}$$

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

8.1 Image Compression-Fundamental

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 8.3
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.

8.2 Image Compression Models

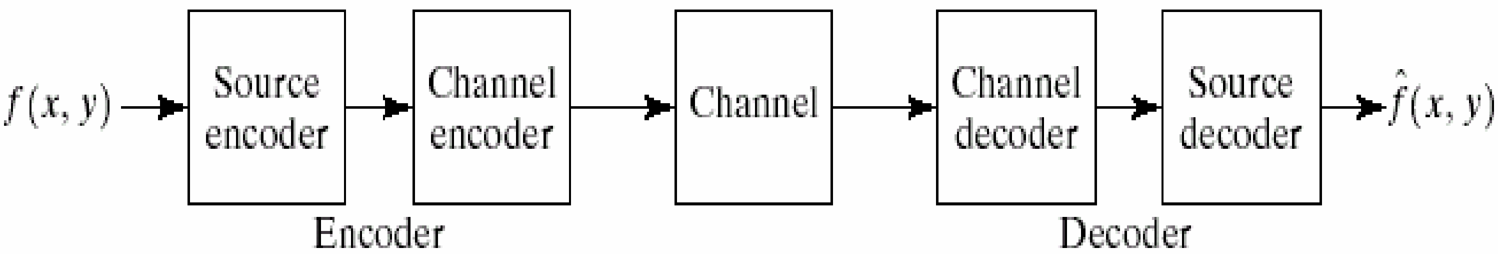


FIGURE 8.5 A general compression system model.

8.2 Image Compression Models

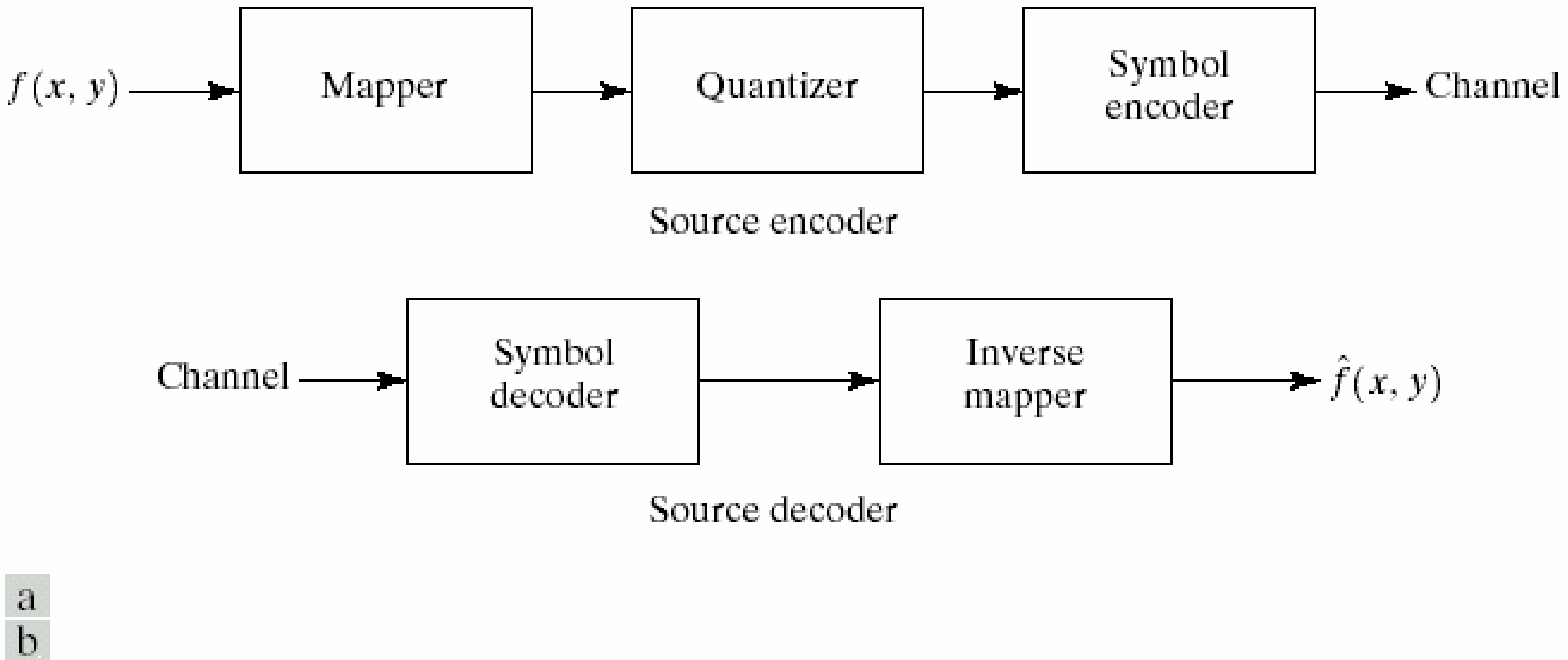


FIGURE 8.6 (a) Source encoder and (b) source decoder model.

8.2 Image Compression Models - channel coder and decoder

- They are designed to reduce the impact of channel noise by inserting a controlled form of ***redundancy*** into the source encoded data.
- ***Joint source channel coding*** (JSCC)

  Source coder: remove source redundancy

  Channel coder: add redundancy to coded data.

  JSCC : compromises the source/channel coder

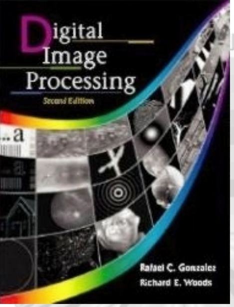
8.2 Image Compression Models - channel coder and decoder

- 3-bit of redundancy are added to a 4-bit word, so that the distance between any two valid code word is 3, all single-bit errors can be detected and corrected.
- 7-bit Hamming (7, 4) code word $h_1 h_2 \ldots h_6 h_7$ *associated with 4-binary number* $b_0 b_1 b_2 b_3$
- ***Even parity bits***: $h_1 = b_3 \oplus b_2 \oplus b_0$, $h_2 = b_3 \oplus b_1 \oplus b_0$, $h_4 = b_2 \oplus b_1 \oplus b_0$, $h_3 = b_3$, $h_5 = b_2$, $h_6 = b_1$, $h_7 = b_0$
- A single error is indicated by a nonzero parity word $c_1 c_2 c_4$, where $c_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7$, $c_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7$, $c_4 = h_4 \oplus h_5 \oplus h_6 \oplus h_7$
- If non-zero value is found, the decoder simply complements the ***code word bit position*** indicated by the parity word.

8.3 Element of Information Theory

- The generation of information can be modeled as a probabilistic process that can be measured in a manner that agree with intuition.
- A random event E that occurs with probability $P(E)$ is said to contain

  $I(E) = \log(1/P(E)) = -\log P(E)$ unit of information.
- The $I(E)$ 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



8.3 Element of Information Theory

- Information channel, a physical medium that links the source to the user.
- Assume that the source generates symbols $A=\{a_1, a_2, \ldots, a_J\}$, A is the source alphabet, and $\Sigma_j P(a_j)=1$
- *Let* $\mathbf{z}=[P(a_1), P(a_2), \ldots P(a_J)]$, the finite ***ensemble*** $(A, \mathbf{z})$ describes the information source.
- If k symbols are generated, for sufficient large k, symbol a_j will be output $kP(a_j)$ times.
- The average self information obtained from k outputs is

$$-kP(a_1)\log P(a_1) - kP(a_2)\log P(a_2) \ldots - kP(a_J)\log P(a_J)$$

or $$-k\sum_{j=1}^{J} P(a_j)\log P(a_j)$$

8.3 Element of Information Theory

- 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 $\mathbf{v}=[P(b_1), P(b_2),\ldots P(b_K)]$, and $B=\{b_1, b_2, ..,b_K\}$, B is the channel alphabet, and $\Sigma_j P(b_j)=1$
- The prob. of given channel output and the prob distribution of the source **z** are related as

$$p(b_k) = \sum_{j=1}^{J} p(b_k | a_j) p(a_j)$$

8.3 Element of Information Theory

- 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

8.3 Element of Information Theory

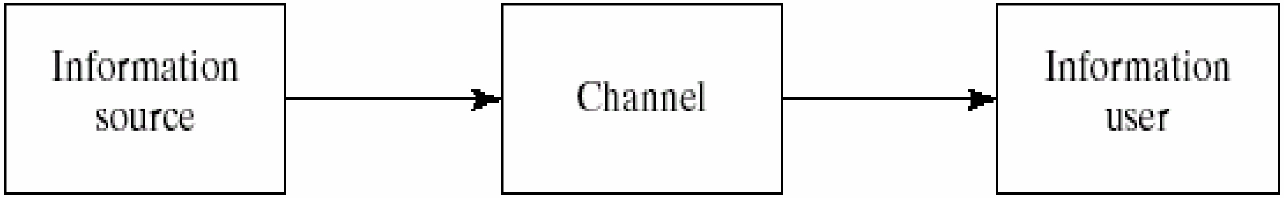


FIGURE 8.7 A simple information system.

Ensemble $(A, \mathbf{z})$

$A = \{a_j\}$

$\mathbf{z} = [P(a_1), P(a_2), \ldots, P(a_J)]^T$

$Q = [q_{kj}]$

Ensemble $(B, \mathbf{v})$

$B = \{b_k\}$

$\mathbf{v} = [P(b_1), P(b_2), \ldots, P(b_K)]^T$

8.3 Element of Information Theory

- Let $K\times J$ matrix $\mathbf{Q}$ (or forward channel transition matrix) as

$$\mathbf{Q}=\begin{bmatrix} P(b_1|a_1) & P(b_1|a_2) & & P(b_1|a_J) \\ P(b_2|a_1) & & & \\ & & & \\ & & & \\ P(b_K|a_1) & P(b_K|a_2) & & P(b_K|a_J) \end{bmatrix}$$

then $\mathbf{v}=\mathbf{Q}\cdot\mathbf{z}$

- The condition entropy is

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

8.3 Element of Information Theory

- The average (expected) value over all b_k is

$$H(\mathbf{z}|\mathbf{v}) = -\sum_{k=1}^{K} H(\mathbf{z}|b_k)p(b_k)$$

- $H(\mathbf{z})$ is the ***average information*** of one source symbol without any knowledge of output symbol, and $H(\mathbf{z}|\mathbf{v})$ is the ***equivocation*** of $\mathbf{z}$ with respect to $\mathbf{v}$.
- The difference between $H(\mathbf{z})$ and $H(\mathbf{z}|\mathbf{v})$ is average information received upon observing a single output symbol, which is called the ***mutual information*** of $\mathbf{z}$ and $\mathbf{v}$, *i.e.*, $I(\mathbf{z}, \mathbf{v}) = H(\mathbf{z}) - H(\mathbf{z}|\mathbf{v})$.
- Since $P(a_j) = P(a_j, b_1) + P(a_j, b_2) + \ldots + P(a_j, b_K)$ then

$$I(\mathbf{z},\mathbf{v}) = \sum_{j=1}^{J}\sum_{k=1}^{K} P(a_j,b_k)\log\frac{P(a_j,b_k)}{P(a_j)p(b_k)}$$

8.3 Element of Information Theory

- $I(\mathbf{z}, \mathbf{v})=0$ when $p(a_j, b_k) = p(a_j)p(b_k)$
- The maximum value of $I(\mathbf{z}, \mathbf{v})$ over all choices of source probabilities in vector **z** is the ***capacity*** C of the channel, *i.e.*, $C = \max_{\mathbf{z}}\{I(\mathbf{z}, \mathbf{v})\}$
- The ***capacity*** of the channel defines the maximum rate (*m-ary* information units per source symbol) at which information can be transmitted reliably through the channel.
- It does not depend on the input probabilities of the source (how channel is used) but is a function of the conditional probability defining the channel alone.

8.3 Element of Information Theory
Binary Symmetry Channel (BSC) example

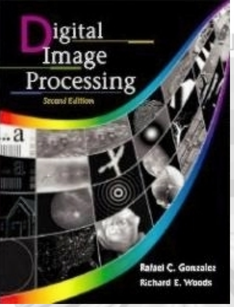
- ***Example***: Consider $A=\{a_1, a_2\}=\{0, 1\}$ and $P(a_1)=p_{\mathrm{bs}}$
  $P(a_2)=1-p_{\mathrm{bs}}=p'_{\mathrm{bs}}$, $\mathbf{z}=[P(a_1), P(a_2)]^T=[p_{\mathrm{bs}}, p'_{\mathrm{bs}}]^T$,
  $H(\mathbf{z})= - p_{bs}\log p_{bs}- p'_{bs}\log p'_{bs}$
- $H(\mathbf{z})$ depends on p_{bs} *only* and can be denoted as $H_{bs}(p_{bs})$.
- The ***binary entropy function*** $H_{bs}(t)$ is defined as

  $$H_{bs}(t)= - t \log t- t' \log t'$$

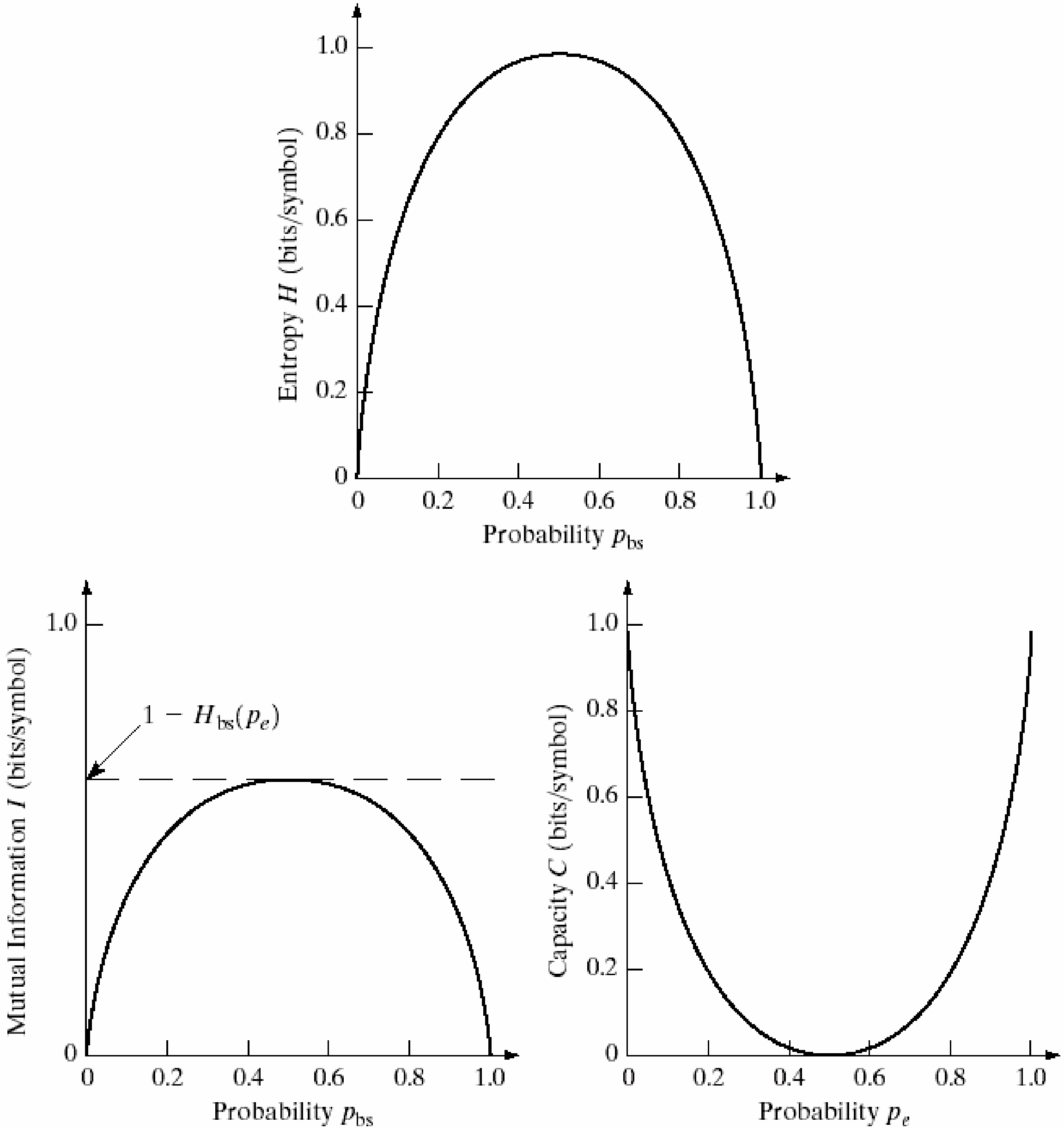
- The probability of error during transmission of any symbol is p_e.
- $\mathbf{Q}$ is defined as $\mathbf{Q}=\begin{bmatrix} 1-p_e & p_e \\ p_e & 1-p_e \end{bmatrix}=\begin{bmatrix} p'_e & p_e \\ p_e & p'_e \end{bmatrix}$

8.3 Element of Information Theory BSC example

- B= $\{b_1, b_2\}=\{0, 1\}$
- $\mathbf{v}= \mathbf{Q}\cdot\mathbf{z} = [P(b_1), P(b_2)]^T = \mathbf{Q}\cdot[p_{bs}, p'_{bs}]^T=[P(0), P(1)]^T$
  $=[p'_e\, p_{bs}+ p_e p'_{bs}, p_e\, p_{bs}+ p'_e p'_{bs}]$
- It is called a ***binary symmetric channel*** (***BSC***)
- $I(\mathbf{z}, \mathbf{v})=H_{bs}(p_{bs}p_e + p'_{bs}p'_e) - H_{bs}(p_e)$
- If $p_{bs}=0$ or 1 then $I(\mathbf{z},\mathbf{v})=0$
- $I(\mathbf{z}, \mathbf{v})$ is maximum (for any p_e) when $p_{bs}=1/2$
  $I(\mathbf{z}, \mathbf{v})=1-H_{bs}(p_e)$
- The channel capacity $C=1-H_{bs}(p_e)$
- $p_e=1$ or 0, then $C=1$ *bit/symbol*
- $p_e=0$, then $C=0$ no information can be transferred



8.3 Element of Information Theory- example



a
b c

FIGURE 8.8 Three binary information functions: (a) the binary entropy function; (b) the mutual information of a binary symmetric channel (BSC); (c) the capacity of the BSC.

8.3.3 Fundamental of Coding Theorems-the *noiseless coding theorem*

- The ***noiseless coding theorem*** defines the minimum average code words.(or ***Shannon's first theorem***)
- A source of information with ***finite ensemble*** (A, $\mathbf{z}$) and statistically independent source symbols is called ***zero-memory source***
- The output is an *n-tuple* of symbols from the source alphabet, the source output takes one of J^n possible values, denoted as α_i, from a set of all possible *n* element sequences $A' = \{\alpha_1, \alpha_2, \ldots \alpha_{J^n}\}$
- Each α_i is (called ***block random variable***) is composed of *n* symbols, i.e., $\alpha_i = a_{j1} a_{i2} .. a_{jn}$
- The *probability* of α_i is $P(\alpha_i) = P(a_{j1}) P(a_{i2}) .. P(a_{jn})$

8.3.3 Fundamental of Coding Theorems - *noiseless coding theorem*

- Let the vector **z**' indicates the block random variable, **z**'$=\{P(\alpha_1), P(\alpha_2),.. P(\alpha_{J^n})\}$
- The ***entropy*** of the source is $H(z') = -\sum_{i=1}^{J^n} P(\alpha_i)\log P(\alpha_i)$
- $H(\mathbf{z}')=nH(\mathbf{z})$, the entropy of the zero-memory information source is n times the entropy of the single symbol source.
- The ***self-information*** of source α_i is $I(\alpha_i)=log[1/P(\alpha_i)]$
- To encode α_i with code word of ***length*** $l(\alpha_i)$ *is*

$$log[1/P(\alpha_i)] \le l(\alpha_i) \le log[1/P(\alpha_i)]+1$$

- Multiply $P(\alpha_i)$ summing over all i, we have

$$\sum_{i=1}^{J^n} P(\alpha)\log\frac{1}{P(\alpha_i)} \le \sum_{i=1}^{J^n} P(\alpha) l(\alpha_i) \le \sum_{i=1}^{J^n} P(\alpha)\log\frac{1}{P(\alpha_i)} + 1$$

8.3.3 Fundamental of Coding Theorems - *noiseless coding theorem*

- It can be written as $H(\mathbf{z}') \le L'_{avg} \le H(\mathbf{z}')+1$
- L'_{avg} represents the ***average word length*** of the code corresponding to the *n*th extension source. *i.e.*, $L'_{avg}=\Sigma^{J^n}_{i=1} P(\alpha_i)l(\alpha_i)$
- Dividing by *n* as $H(\mathbf{z}) \le L'_{avg}/n \le H(\mathbf{z})+1/n$
- If $n \rightarrow infinite$ then $lim[L'_{avg}/n]=H(\mathbf{z})$
- *H*(z) *is the* ***lower bound***, *the efficiency of any coding strategy can be defined as*

$$\eta=H(z')/L'_{avg}=H(z)/[L'_{avg}/n]=nH(z)/L'_{avg}$$

8.3.3 Fundamental of Coding Theorems - *noiseless coding theorem*

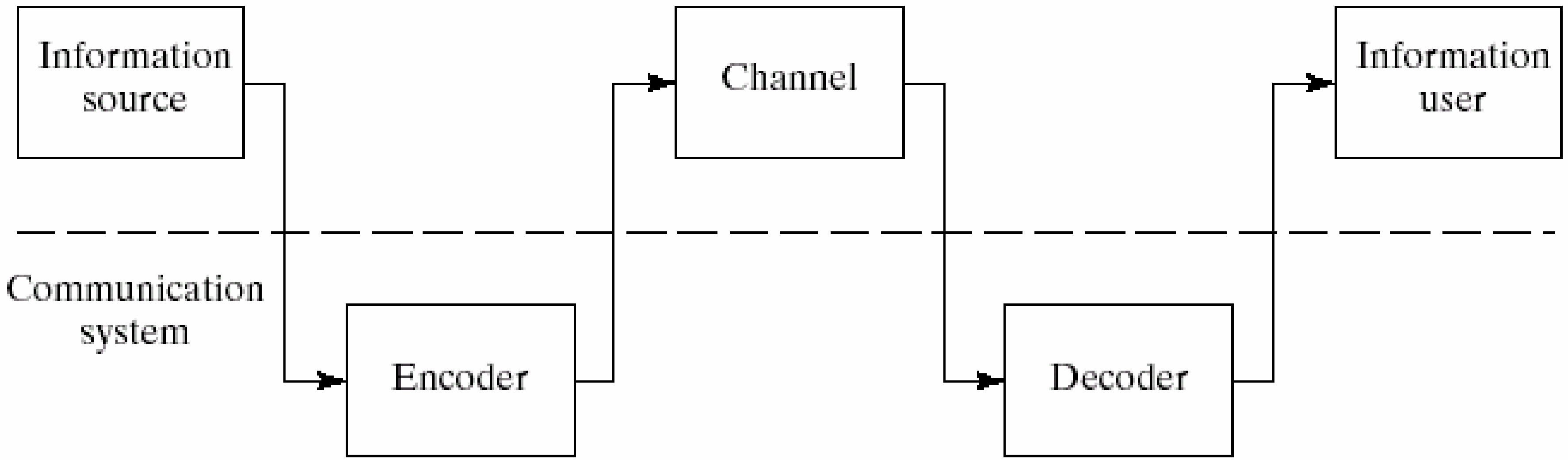


FIGURE 8.9 A communication system model.

8.3.3 Fundamental of Coding Theorems-example

A zero-memory information source with source alphabet A={a_1, a_2} has symbol probability $P(a_1)=2/3$,$P(a_2)=1/3$, *entropy* $H(\mathbf{z})=0.918$. If symbols a_1 and a_2 are represented by binary code words *0* and *1*, $L'_{avg}=1$ and the resulting ***code efficiency*** $\eta=0.918/1=0.918$. From ***Table 8.4***, the ***entropy*** of second extension=*1.83*, $L'_{avg}=1.89$, and the ***code efficiency*** $\eta=1.83/1.89=0.97$. The average number of code bits/symbol is reduced to *1.89/2=0.94bits*

TABLE 8.4
Extension coding example.

α_i	Source Symbols	$P(\alpha_i)$ Eq. (8.3-14)	$I(\alpha_i)$ Eq. (8.3-1)	$l(\alpha_i)$ Eq. (8.3-16)	Code Word	Code Length
First Extension						
α_1	a_1	2/3	0.59	1	0	1
α_2	a_2	1/3	1.58	2	1	1
Second Extension						
α_1	$a_1 a_1$	4/9	1.17	2	0	1
α_2	$a_1 a_2$	2/9	2.17	3	10	2
α_3	$a_2 a_1$	2/9	2.17	3	110	3
α_4	$a_2 a_2$	1/9	3.17	4	111	3

8.3.3 Fundamental of Coding Theorems- the ***noisy coding theorem***

- Suppose BSC has a probability of error $p_e=0.01$. A simple method for increase the reliability is to ***repeat each symbol several times***, *i.e.*, using 000 and 111.
- No error prob. is $(1-p_e)^3$ or $(p'_e)^3$, the single error prob. is 3 $p_e\, p'^2_e$,….
- If a nonvalid code word (not *000* nor *111)* is received, a majority vote of the three code bits determines the output.
- Prob. of ***incorrect decoding*** is the sum of the prob. of ***two-symbol error*** and ***three-symbol error*** or $p_e^3+3\, p_e^2 p'_e$.

  i.e., For $p_e=0.01$, $p_e^3+3\, p_e^2 p'_e=0.003$.

8.3.3 Fundamental of Coding Theorems- the ***noisy coding theorem***

- In general, we encoding the *nth* extension of the source using *K-ary* code sequences of length *r*, where $K^r \le J^n$.
- We select only φ of the K^r possible code sequences as valid codeword and devise a decision rule that optimizes the probability of correct decoding. ($\varphi=2$, $r=3$, $K^r=8$)

8.3.3 Fundamental of Coding Theorems- the ***noisy coding theorem***

- The maximum rate of coded information is $log(\varphi/r)$ when the φ valid code words are equal probable.
- A code of ***size*** φ and ***block length*** r is said to have a rate of $R = \log(\varphi/r)$.
- For any $R<C$, where C is the capacity of the zero-memory channel with matrix $\mathbf{Q}$, there exists a code block length r and rate R such that the probability of a block decoding error is less than or equal to ε, for any $\varepsilon>0$. – ***Shannon's second theorem (or noisy coding theorem).***

8.3.3 Fundamental of Coding Theorems- the ***rate-distortion theorem***

- Assume the channel is error-free, but the communication process is *lossy*, how to determine the smallest rate, subject to a given fidelity criterion.
- The information source and decoder outputs defined by $(A, \mathbf{z})$ and $(B, \mathbf{v})$, a ***channel matrix*** **Q** relating $\mathbf{z}$ to $\mathbf{v}$.
- Each time the source produce source symbol a_j (represented by a code symbol) that is then decoded to yield output symbol b_k with probability $q_{kj}.=p(b_k|a_j)$
- The ***distortion measure*** $\rho(a_j, b_k)$ defines the penalty associated with reproducing a_j with decoding output b_k.

8.3.3 Fundamental of Coding Theorems- the ***rate-distortion theorem***

- The ***average of distortion*** is
  $d(\mathbf{Q})= \sum_j\sum_k \rho(a_j, b_k)P(a_j, b_k)=\sum_j\sum_k \rho(a_j , b_k)P(a_j)q_{kj}$
- A encoding-decoding procedure is said to be ***D-admissible*** if and only of the average distortion associated with $\mathbf{Q} \leq D$.
- The ***set*** of D-admissible encoding-decoding procedure is $\mathbf{Q}_D=\{q_{kj} \mid d(\mathbf{Q})\leq D\}$
- Hence we define the ***rate-distortion function*** as

$$R(D)=\min_{\mathbf{Q}\in\mathbf{Q}_D}[I(\mathbf{z}, \mathbf{v})],$$

  where $I(\mathbf{z}, \boldsymbol{v})$ is a function of $\boldsymbol{z}$ and $\mathbf{Q}$
- To compute the rate, R(D), we ***minimize*** $I(\mathbf{z}, \boldsymbol{v})$ by appropriate choice of $\mathbf{Q}$ subject to the ***constraints***, $q_{ij} \geq 0$, $\sum_k q_{kj}=1$, and $d(\mathbf{Q})=D$.
- If D=0, then $R(D) \leq H(\mathbf{z})$, or $R(0) \leq H(\mathbf{z})$.

8.3.3 Fundamental of Coding Theorems-example

- ***Example***: A zero-memory ***binary source*** (***bs***) with simple distortion measure as $\rho(a_j, b_k)=1-\delta_{jk}$
  i.e., $\rho(a_j, b_k)=1$ if $a_j \neq b_k$, $\rho(a_j, b_k)=0$ otherwise
- Each encoding and decoding error is counted as one unit of distortion.
- Let μ_j, $j=1,..J+1$ is the Lagrange multipliers, we have
  $$J(\mathbf{Q}) = I(\mathbf{z}, \mathbf{v}) - \sum_j \mu_j \left(\sum_k q_{kj}\right) - \mu_{J+1} d(\mathbf{Q})$$
- Minimizing $J(\mathbf{Q})$ (*i.e.*, $dJ/dq_{kj}=0$), we find $\mathbf{Q}$ as

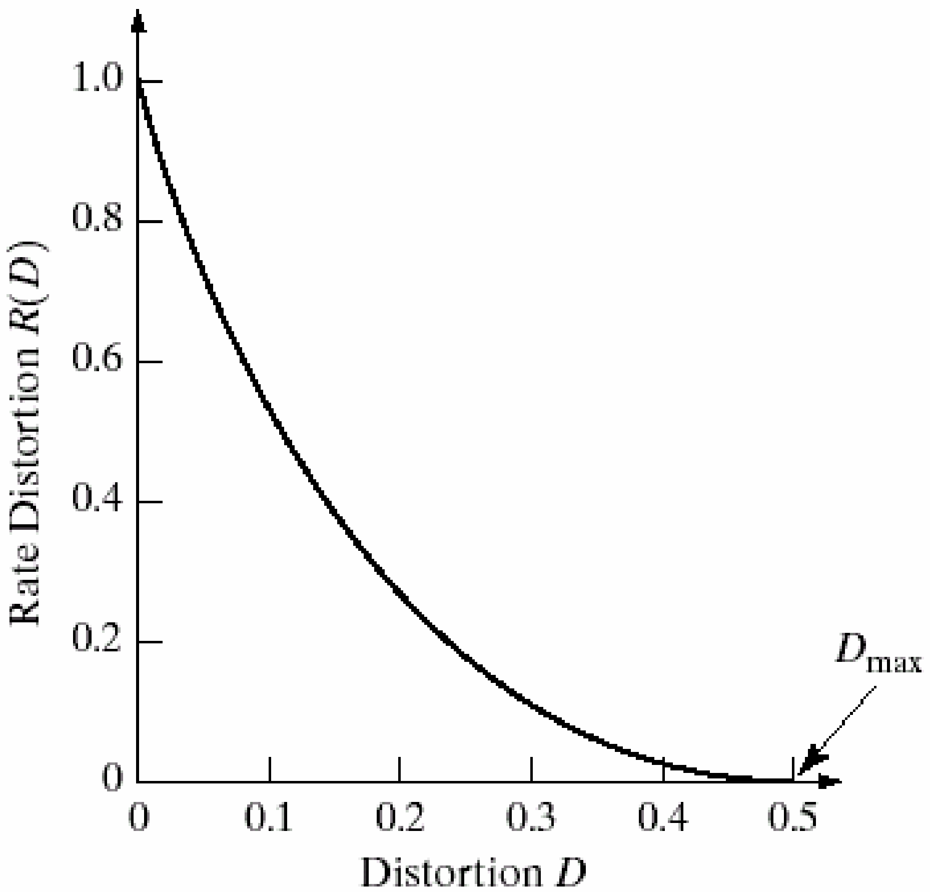
$$\mathbf{Q} = \begin{bmatrix} 1-D & D \\ D & 1-D \end{bmatrix}$$

$I(\mathbf{z}, \mathbf{v})=1-H_{bs}(D)$ with $p_{bs}=1/2$ and $p_e=D$

$R(D)=\min_{\mathbf{Q}\in\mathbf{Q}_D}[I(\mathbf{z}, \mathbf{v})]=1-H_{bs}(D)$

8.3.3 Element of Information Theory-example

FIGURE 8.10 The rate distortion function for a binary symmetric source.



8.3.4 Apply information theory on images

- 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

8.3.4 Apply information theory on images

- Difference images

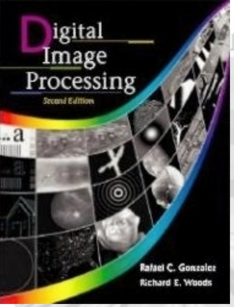
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

First order entropy
=1.41 bits/pixel

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

8.4 Error Free Compression-Source coding

- Huffman code
  - Encoding of a single character
- Arithmetic code
  - Encoding of a single character
- Lempel-Ziv code
  - Encoding variable-length strings of characters.



8.4 Variable length coding

- Instead of assigning K-bit words to each of the possible 2^K luminance levels, we assign words of longer length to levels having lower probability and words of shorter length to levels having 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 code-word length is

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

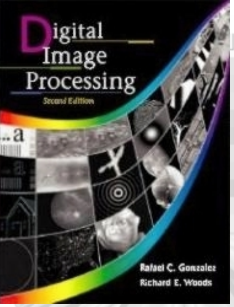
8.4.1 Huffman Code

- 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 is 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 is the tree.

8.4.1 Huffman coding

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

FIGURE 8.11 Huffman source reductions.



8.4.1 Huffman Coding

FIGURE 8.12 Huffman code assignment procedure.

Original source			Source reduction							
Sym.	Prob.	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								

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 → $a_3\ a_1\ a_2\ a_2\ a_6$

8.4.1 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 code word can be decoded without referencing succeeding symbols.
- ***Uniquely decodable***: code string can be decoded in only one way.

8.4.1 Huffman Coding

TABLE 8.5 Variable-length codes.

Source symbol	Probability	Binary Code	Huffman	Truncated Huffman	B_2-Code	Binary Shift	Huffman Shift
Block 1							
a_1	0.2	00000	10	11	C00	000	10
a_2	0.1	00001	110	011	C01	001	11
a_3	0.1	00010	111	0000	C10	010	110
a_4	0.06	00011	0101	0101	C11	011	100
a_5	0.05	00100	00000	00010	C00C00	100	101
a_6	0.05	00101	00001	00011	C00C01	101	1110
a_7	0.05	00110	00010	00100	C00C10	110	1111
Block 2							
a_8	0.04	00111	00011	00101	C00C11	111 000	00 10
a_9	0.04	01000	00110	00110	C01C00	111 001	00 11
a_{10}	0.04	01001	00111	00111	C01C01	111 010	00 110
a_{11}	0.04	01010	00100	01000	C01C10	111 011	00 100
a_{12}	0.03	01011	01001	01001	C01C11	111 100	00 101
a_{13}	0.03	01100	01110	10 0000	C10C00	111 101	00 1110
a_{14}	0.03	01101	01111	10 0001	C10C01	111 110	00 1111
Block 3							
a_{15}	0.03	01110	01100	10 0010	C10C10	111 111 000	00 00 10
a_{16}	0.02	01111	010000	10 0011	C10C11	111 111 001	00 00 11
a_{17}	0.02	10000	010001	10 0100	C11C00	111 111 010	00 00 110
a_{18}	0.02	10001	001010	10 0101	C11C01	111 111 011	00 00 100
a_{19}	0.02	10010	001011	10 0110	C11C10	111 111 100	00 00 101
a_{20}	0.02	10011	011010	10 0111	C11C11	111 111 101	00 00 1110
a_{21}	0.01	10100	011011	10 1000	C00C00C00	111 111 110	00 00 1111
Entropy	4.0						
Average length		5.0	4.05	4.24	4.65	4.59	4.13

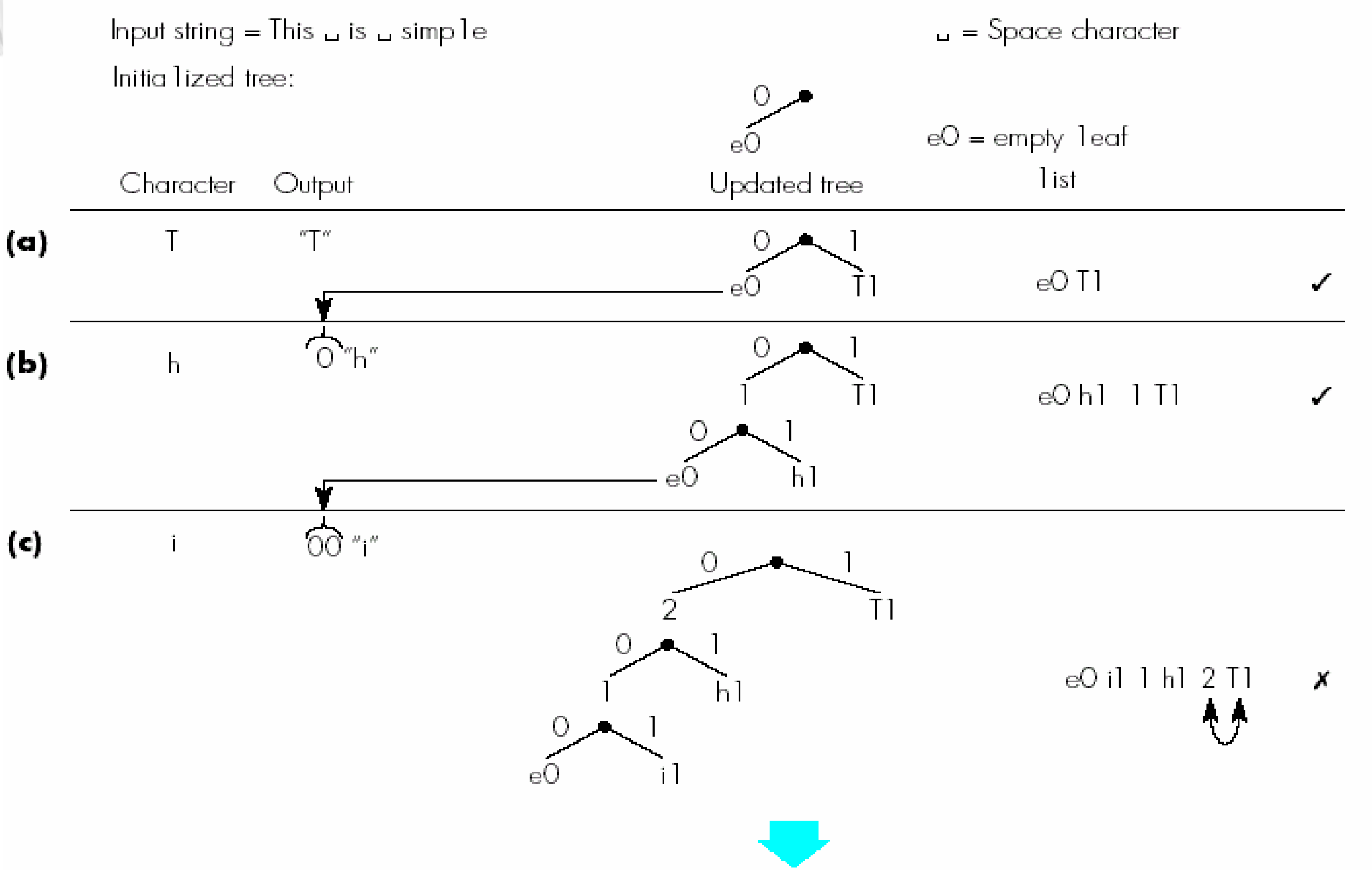
8.4.1.1-Dynamic Huffman Coding

- The codewords are not well-prepared before encoding or transmission.
  - Both of the transmitter and receiver modify the Huffman tree (Codeword table) dynamically as the characters are being transmitted and received.
- Known characters
  - If the character is currently present in the tree, then its codeword is determined and transmitted
- Unknown Characters
  - If it is in its first occurrence, then it is transmitted in its uncompressed form.
- The receiver has two jobs
  - Decoding the received codeword.
  - Carry the same modification of Huffman tree as the transmitter.

8.4.1.1 Dynamic Huffman Coding

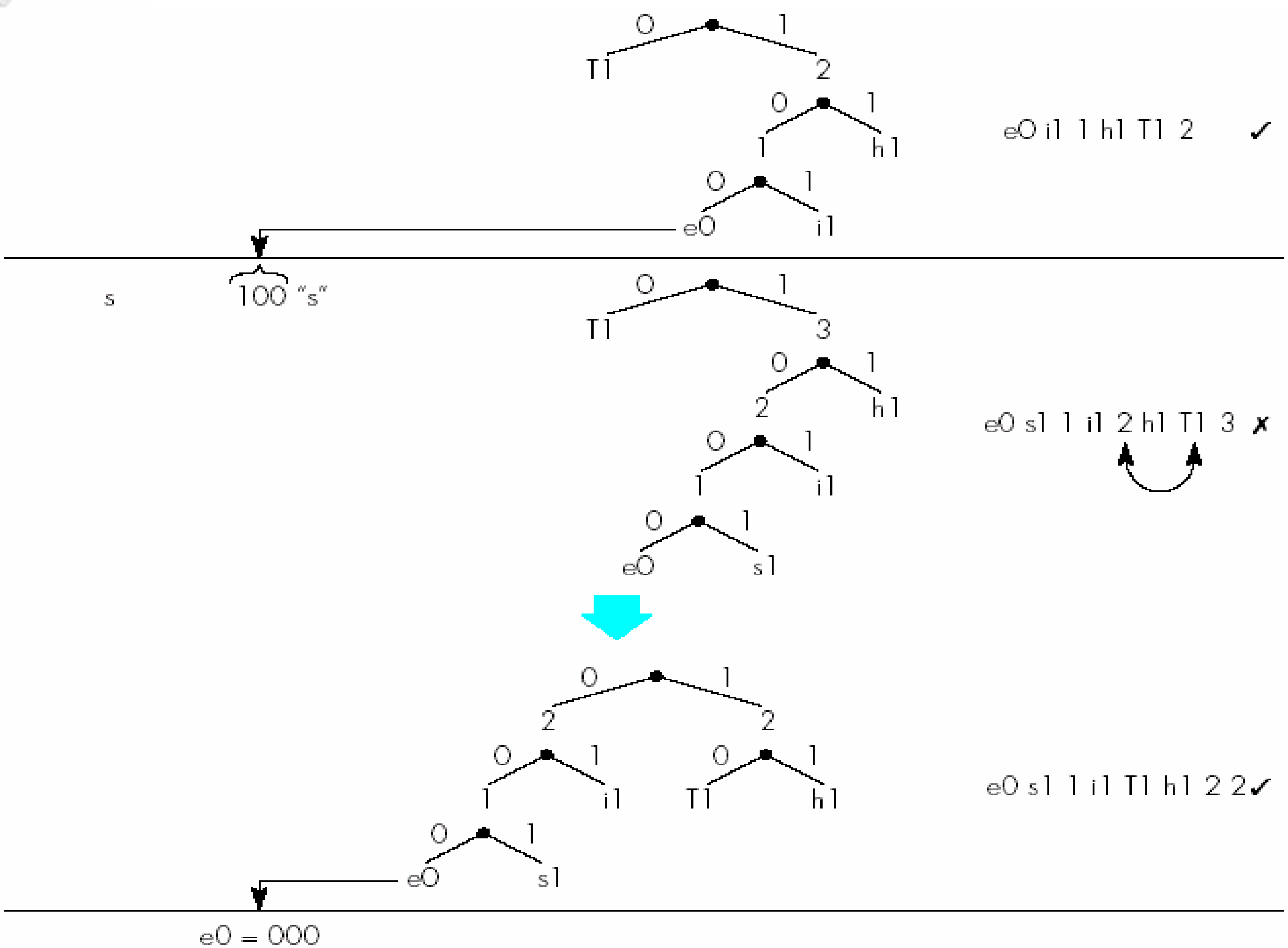
- For each subsequent character, the encode checks whether it is already in the tree:
  - If yes, send the current codeword
  - If not, send the current code word of the empty leaf, followed by the uncompressed codeword of the character.
- Each time the tree is updated either by adding a new character or by incrementing the frequency of occurrence of an existing character.
- The encoder and decoder both check if it is necessary to modify the tree.
- To make sure that they modify the tree in the same way, the criterion is to list the weights (frequency of occurrence) the leaf and branch nodes in the update tree ***from left to right*** and ***from bottom to top*** starting at the empty leaf.

8.4.1.1 Dynamic Huffman Coding

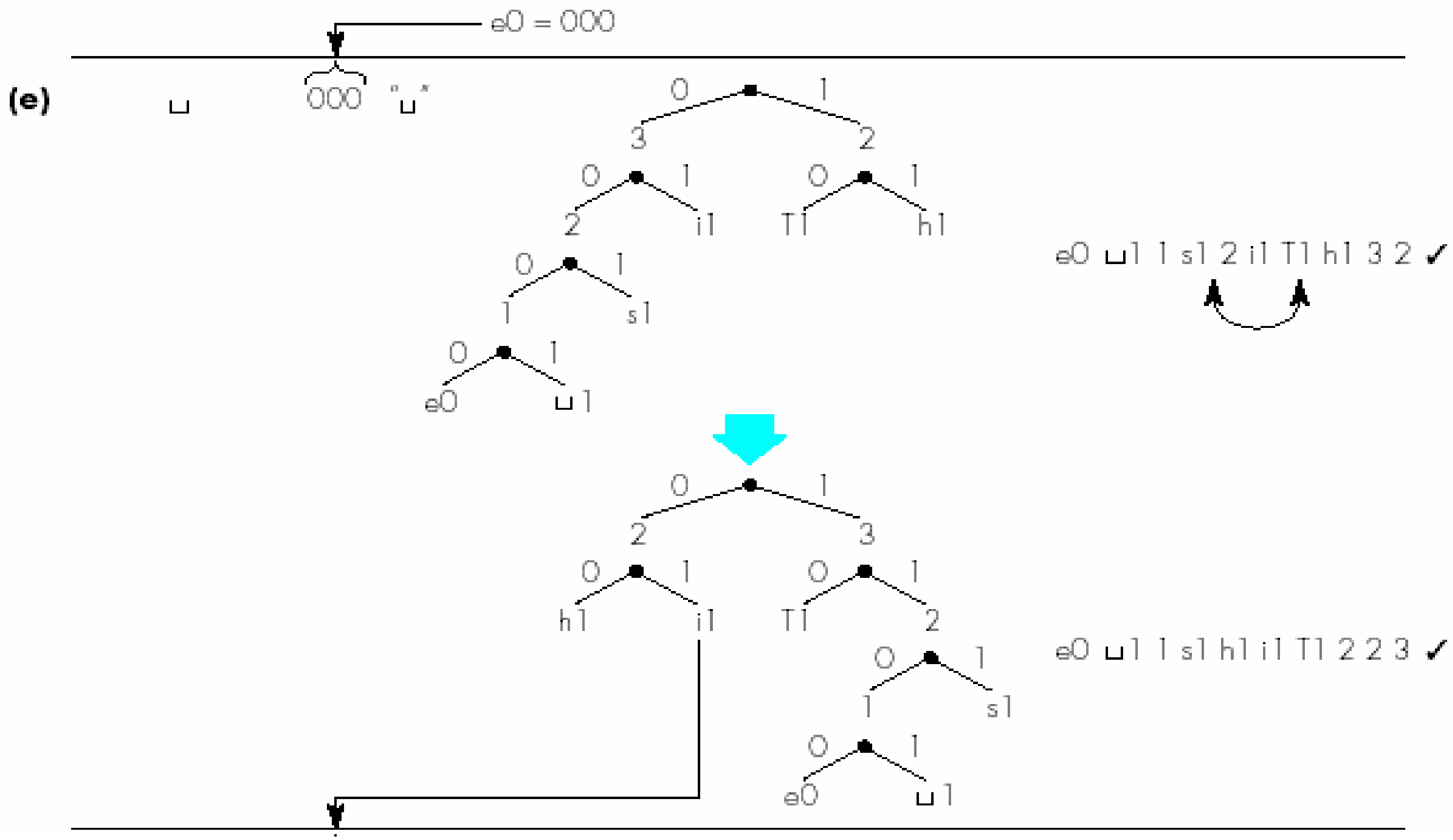


8.4.1.1-Dynamic Huffman Coding (continued 1)

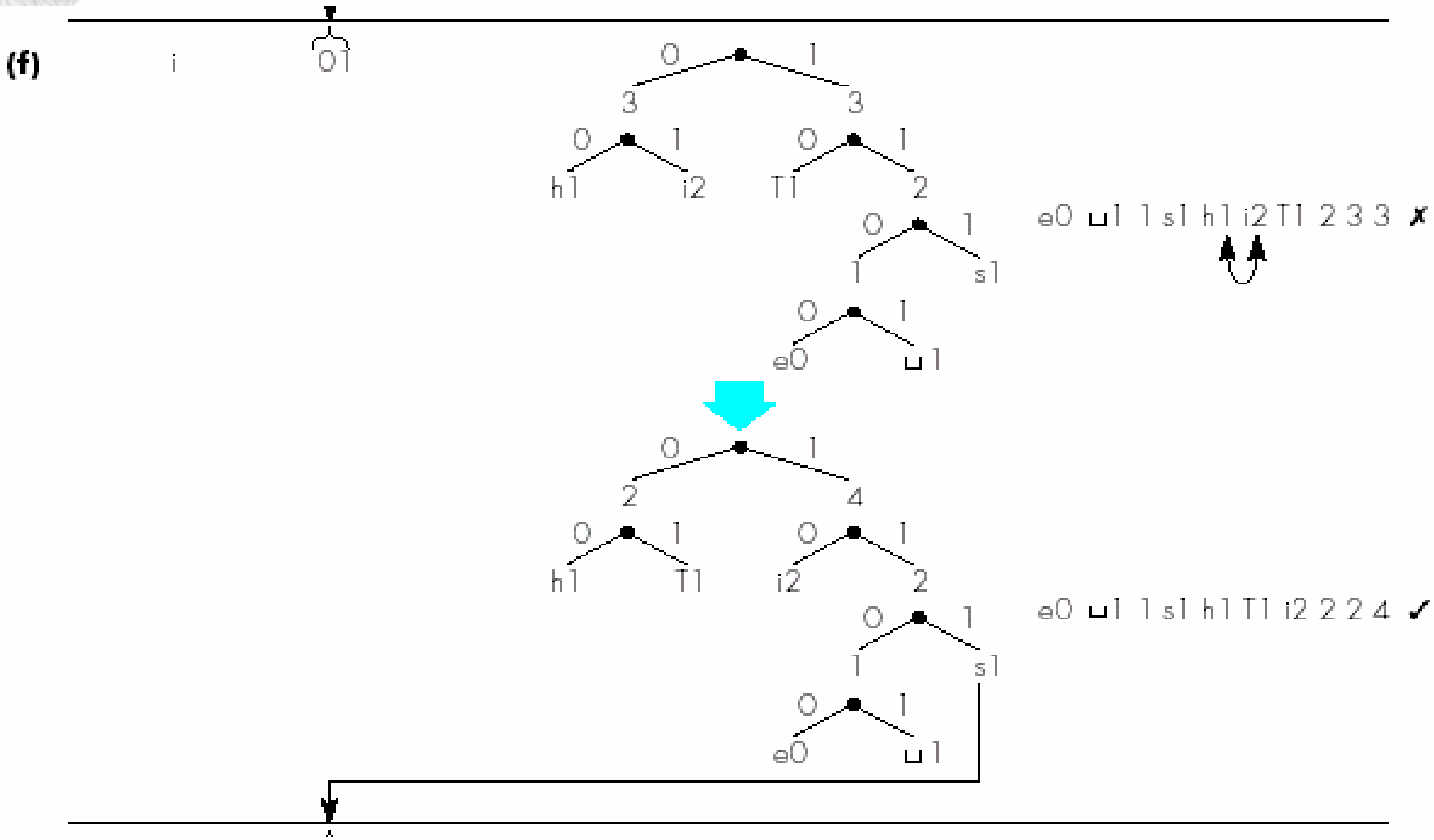
(d)



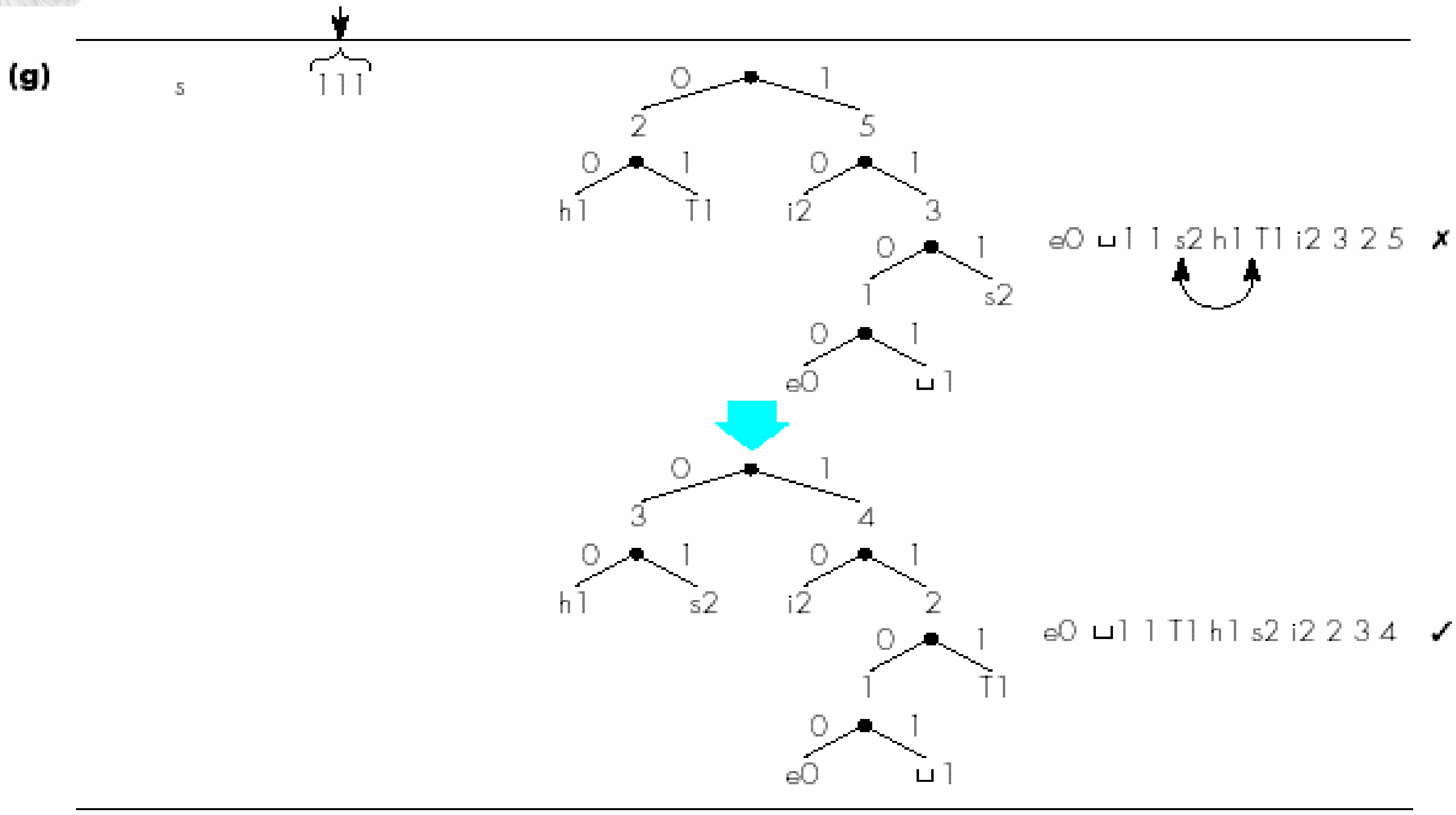
8.4.1.1 -Dynamic Huffman Coding (continued 2)



8.4.1.1 Dynamic Huffman Coding (continued 3)



8.4.1.1 Dynamic Huffman Coding (continued 4)



8.4.1.2 Arithmetic Coding

- Arithmetic coding is better than Huffman coding in achieve the ***Shannon value***.
- Huffman coding
  - A separate codeword for each character
- Arithmetic coding
  - A single codeword for each encoded string of characters.
  - 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.

8.4.1.2 Arithmetic Coding

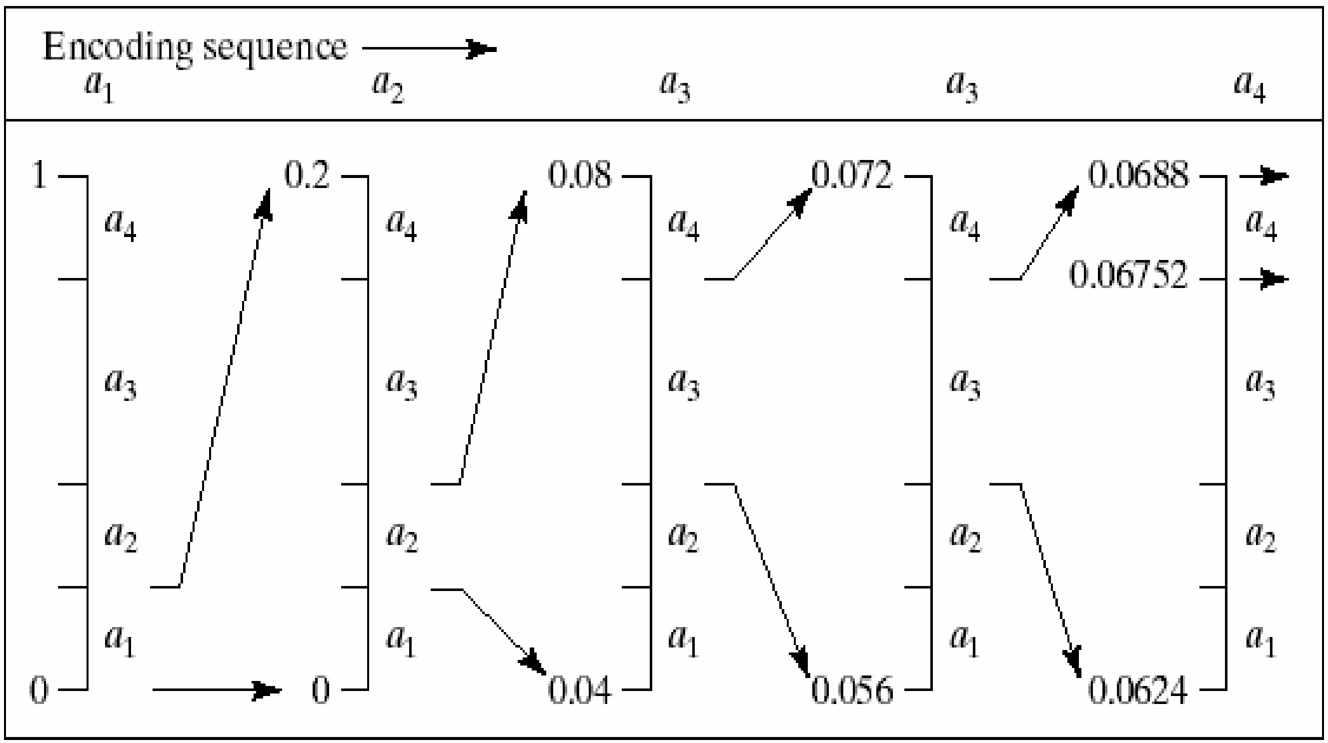


FIGURE 8.13 Arithmetic coding procedure.

8.4.1.3 Arithmetic Coding

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	$[0.8, 1.0)$

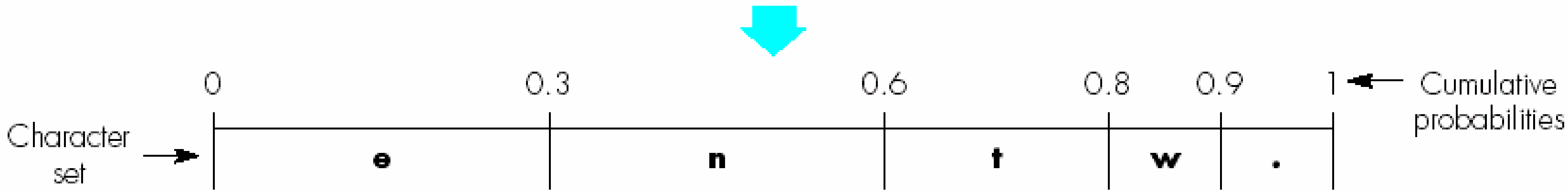
TABLE 8.6 Arithmetic coding example.

Arithmetic coding principles: (a) example character set and their range assignments; (b) encoding of the string

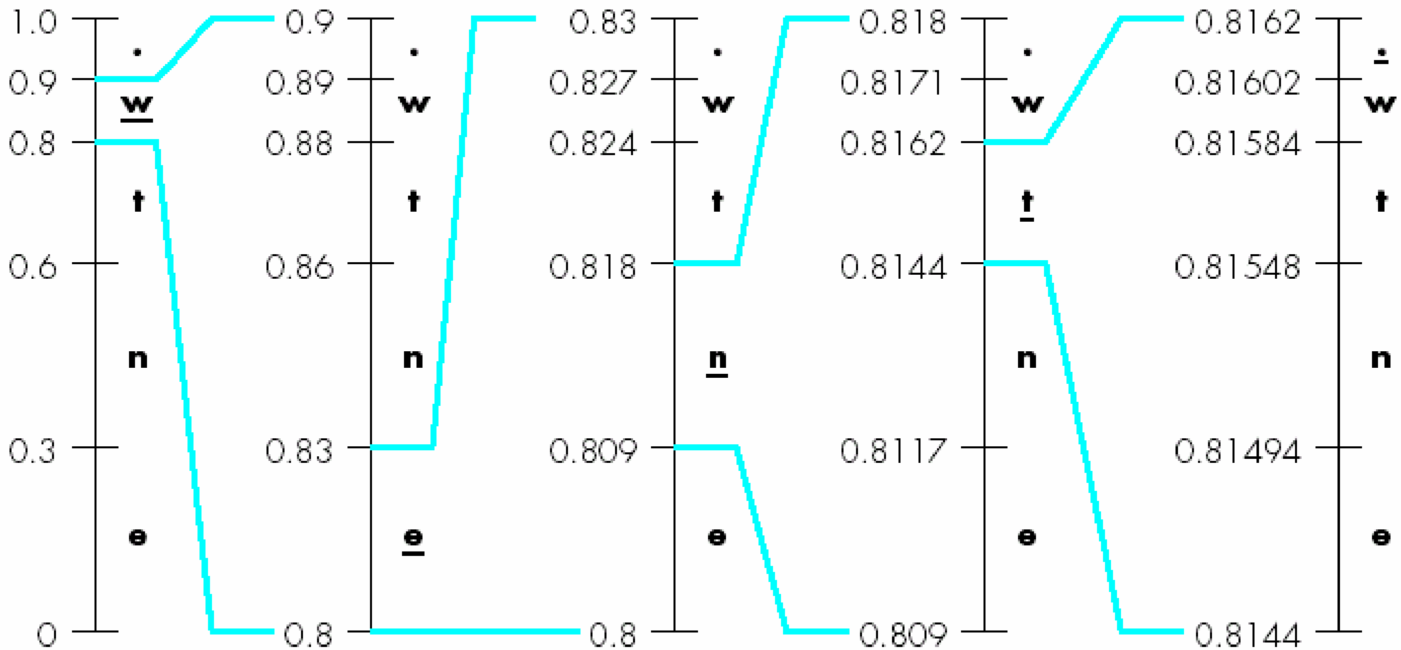
(a)

Example character set and their probabilities:

e = 0.3, **n** = 0.3, **t** = 0.2, **w** = 0.1, **.** = 0.1



(b)



Encoded version of the character string **went.** is a single codeword in the range 0.816 02 ⩽ codeword < 0.8162

Example (Huffman Code)

- Consider a four-symbol alphabet, for which the relative frequencies ½, ¼. 1/8, and 1/8.

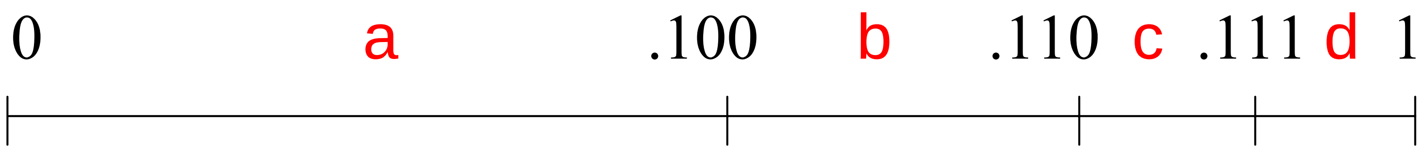
symbol	code word	probability (binary)	cumulative prob.
a	0	100	000
b	10	010	100
c	110	001	110
d	111	001	111

Example

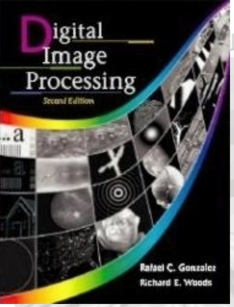
- The code for data string "a a b c" is 0010110
- Decoding input string : 0010110
- (1) remove 0 → decode as a.
- (2) remove 0 → decode as a.
- (3) remove 1 → not decodable → remove 10 → decode as b.
- (4) remove 1 → not decodable → remove 11 → not decodable → remove 110 → decode as c.
- From the above table, each codeword is a cumulative probability P.

Example (Arithmetic Coding)

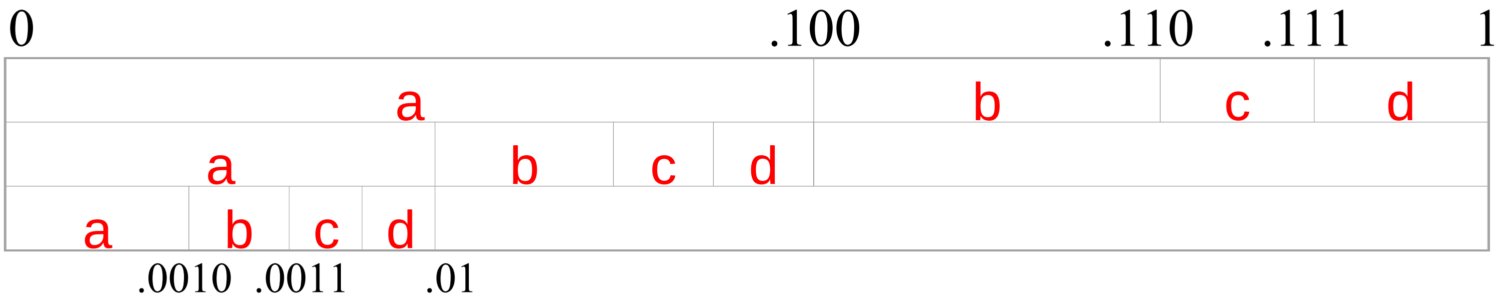
- We view codewords as points (or **code points**) on the number line from 0 to 1, or the unit interval such as



- Once "a" has been encoded to [0, .1), we next subdivide the interval into the same proportions as the original unit interval.
- The subinterval assigned to the second "a" is [0, .01).
- For the third symbol, we sub-divide [0, .01), and the subinterval belonging to the third symbol "b" is [.001, .0011).



Example (Arithmetic Coding)



Encoding :

- The recursion begins with the "current" values of ***code point C*** and ***available width A***, and uses the value of symbol encoded to determine "New" values of code point ***C*** and width ***A***.
- At the end of the current recursion, and before the next recursion, the "new" value of code point ***C*** and width ***A*** become the current value.

Example (Arithmetic Coding)

- New code point $New\ C = Current\,C + A \times P_i$
- New interval width $New\ A = Current\,A \times P_i$

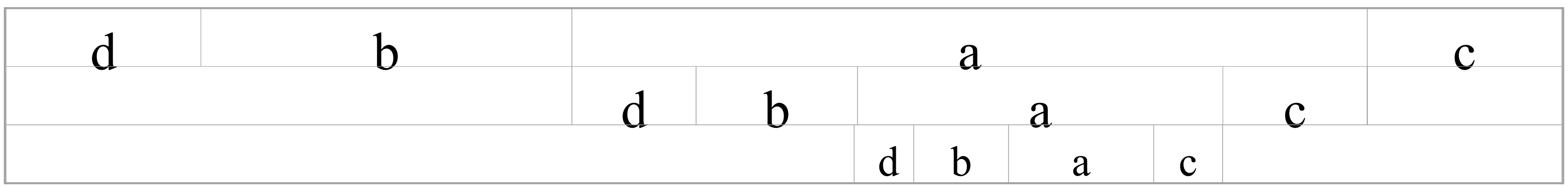


Table Arithmetic code example

Symbol	*Cumulative probability P*	*Symbol probability p*	*Length*
d	.000	.001	3
b	.001	.010	2
a	.011	.100	1
c	.111	.001	3

Example (Arithmetic Coding)

- Arithmetic coding of the string "a a b c"
- 1st symbol "a"
- C : New code point $C = 0 + 1 \times (.011) = .011$
- A : New interval width $A = 1 \times (.1) = .1$
- The first symbol yields $\rightarrow$ $\begin{cases} \text{code point } .011 \\ \text{interval } [.011, .111) \end{cases}$
- 2nd symbol "a"
- C : New code point $C = .011 + .1 \times (.011) = .1001$
- A : New interval width $A = .1 \times (.1) = .01$
- The second symbol yields $\rightarrow$ $\begin{cases} \text{code point } .1001 \\ \text{interval } [.1001, .1101) \end{cases}$

Example (Arithmetic Coding)

- 3rd symbol "b"

  New code point $C = .1001 + .01 \times (.001) = .10011$

  New interval width $A = .01 \times (.01) = .0001$

- 4th symbol "c"

  New code point $C = .10011 + .0001(.111) = .1010011$

  New interval width $A = .0001 \times (.001) = .0000001$

Example (Arithmetic Coding)

- ***Carry-Over Problem*** :

  The encoding of symbol "c" changes the value off the third coding-string bits.

  The first three bits changed from .100 to .101

- ***Code-string termination***

  Any value equal to or grater than .1010011, but less than .1010100 would survive to identify the interval.

Example (Arithmetic Coding)

- The coding is basically an addition of properly scaled cumulative probabilities P, called *augends*, to the coding string.

$$
\begin{array}{lr}
.011 & \text{"a"} \\
011 & \text{"a"} \\
001 & \text{"b"} \\
111 & \text{"c"} \\
\hline
.1010011 &
\end{array}
$$

Example (Arithmetic Coding)

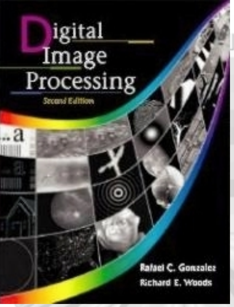
- Decoding the bit-string .1010011
- 1) Comparison. Examine the code string and determine the interval in which it lies. Decode the symbol corresponding to that interval. Since .1010011 lies in [.011, .110) which is as subinterval the first symbol must be "a"
- 2) Readjust. Subtract from the code string the angend value of the code point for the decoded symbol. We prepare to decode the second symbol by subtracting .011 from the code string .1010011 − .011=.0100011

Example (Arithmetic Coding)

- 3) Scaling. Rescale the code C for direct comparison with P by undoing the multiplication for the value A. Since the values for the second subinterval were adjusted by multiplying by .1 in the encoder.
- The decoder may "undo" that multiplication by multiplying the remaining value of the code string by 2. Our code string is now .100011

8.4.2 Lempel-Ziv-Welsh Coding

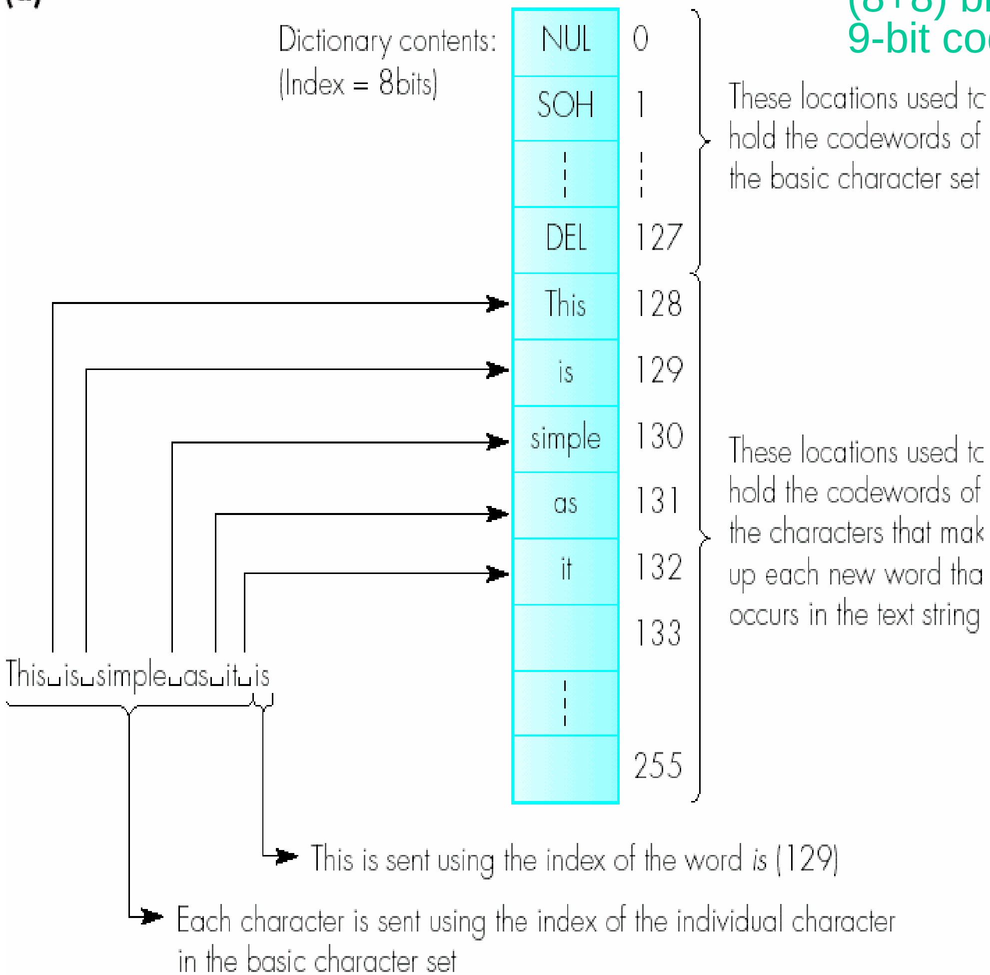
- Applied for image file formats: GIF, TIFF, and PDF
- The encoder/decoder build the ***dictionary*** dynamically as the text is being transferred.
- The more frequently the words stored in the dictionary occur in the text, the higher the level of compression
- Prior to sending each word in the form of single character, the encoder first checks to determine if the 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.

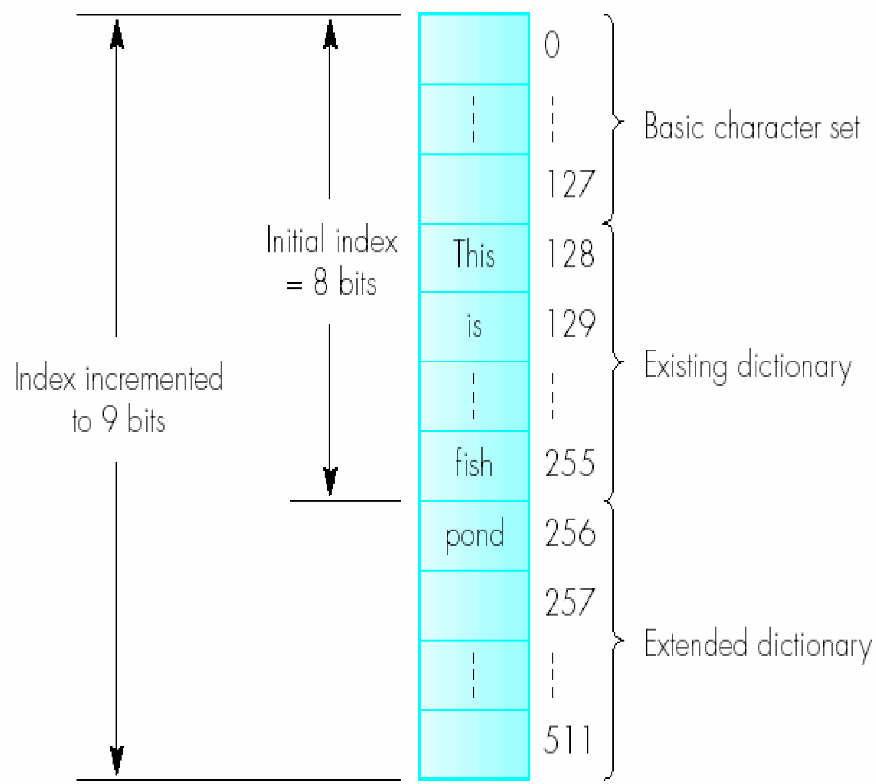


LZW compression algorithm: (a) basic operation; (b) dynamically extending the number of entries in the dictionary.

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

(a)





8.4.2 Lempel-Ziv-Welsh Coding -for images

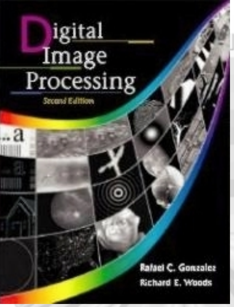
- The LZW can be applied for encoding ***images***.
- Consider 4 x 4 image of a vertical edge

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

- 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.

8.4.2 Lempel-Ziv-Welsh Coding-for images

- 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



Lempel-Ziv-Welsh Coding -for images

TABLE 8.7 LZW coding example.

Initially found

not found

found

not found

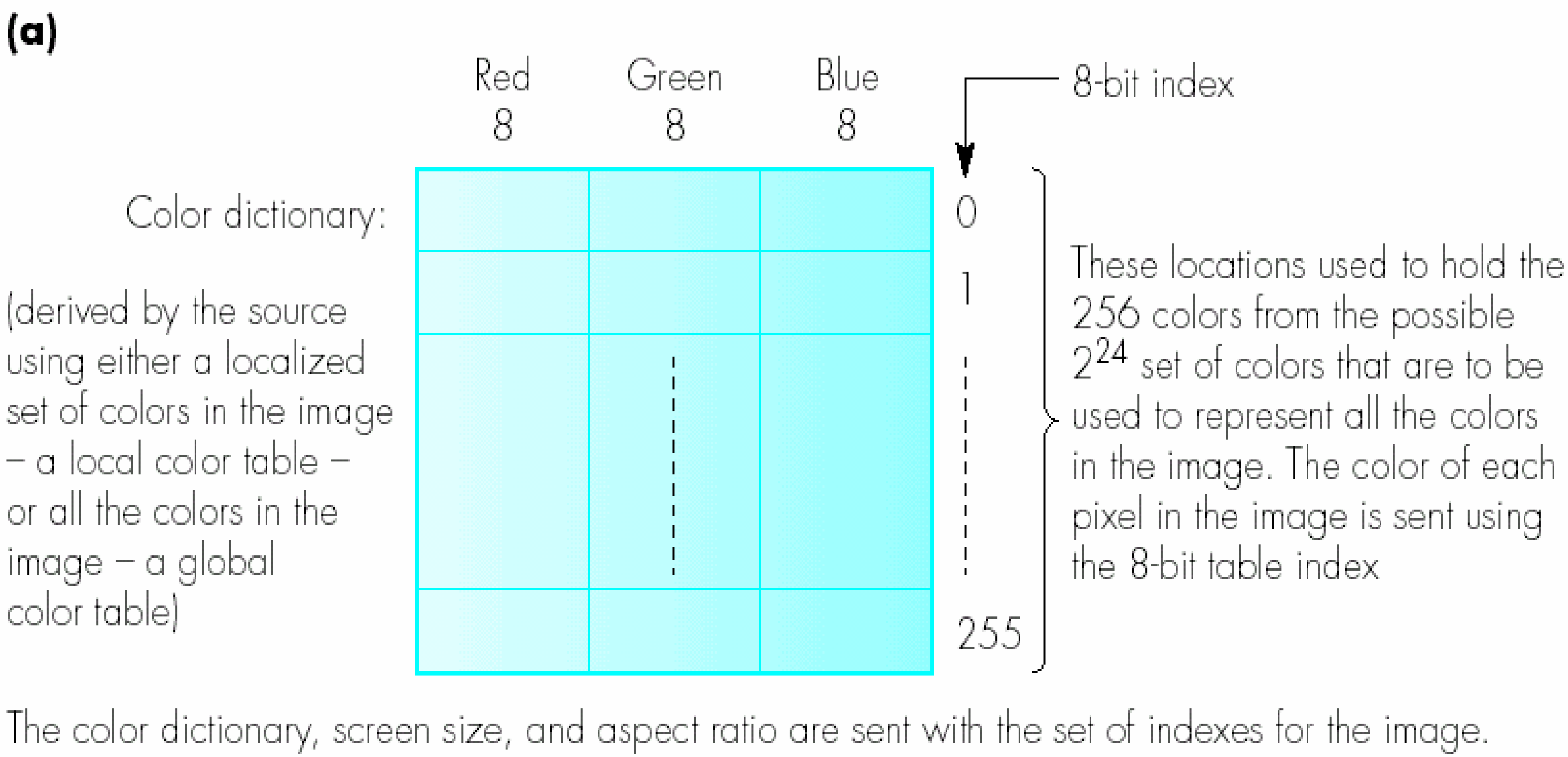
Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39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

Image Compression using LZW Coding - GIF

- GIF is used extensively with the Internet for the representation and compression of Graphical images.
- Real color: 24 bit for R, G, and B: Totally 2^{24} colors
- Color Table : 256 entries, each contain a 24-bit value.
- Reduce the total number of color from 2^{24} colors to 256 colors.

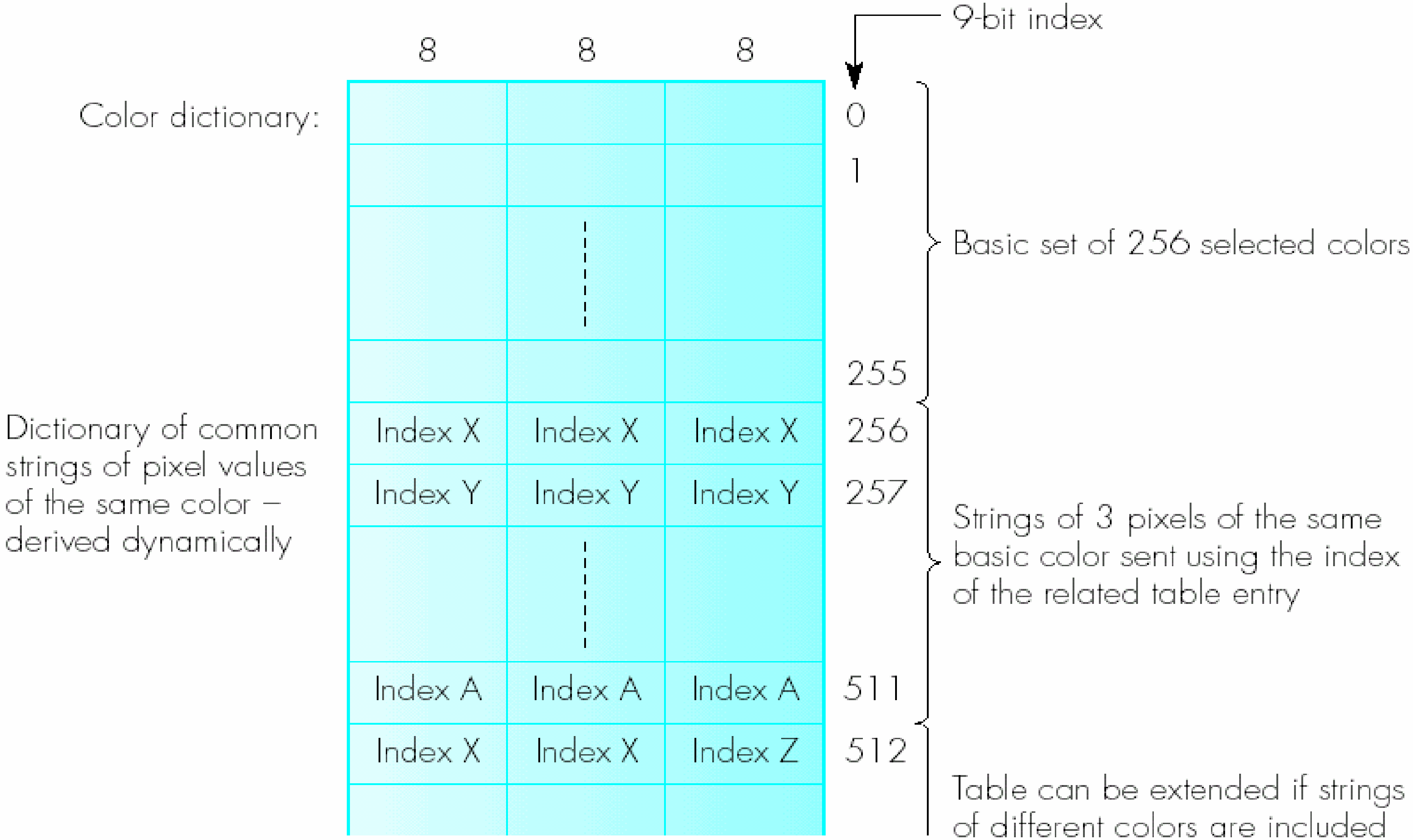
  Global color table, Local color table
- Apply LZW for further compression, the occurrence of common string of pixel values are detected and entered into the color table.
- ***Interlaced mode***: the compressed data is divided into four groups: 1/8, 1/8, ¼, and 1/2.
- Transmitted over IP with variable transmission rate.

GIF compression principles:
(a) basic operational mode;

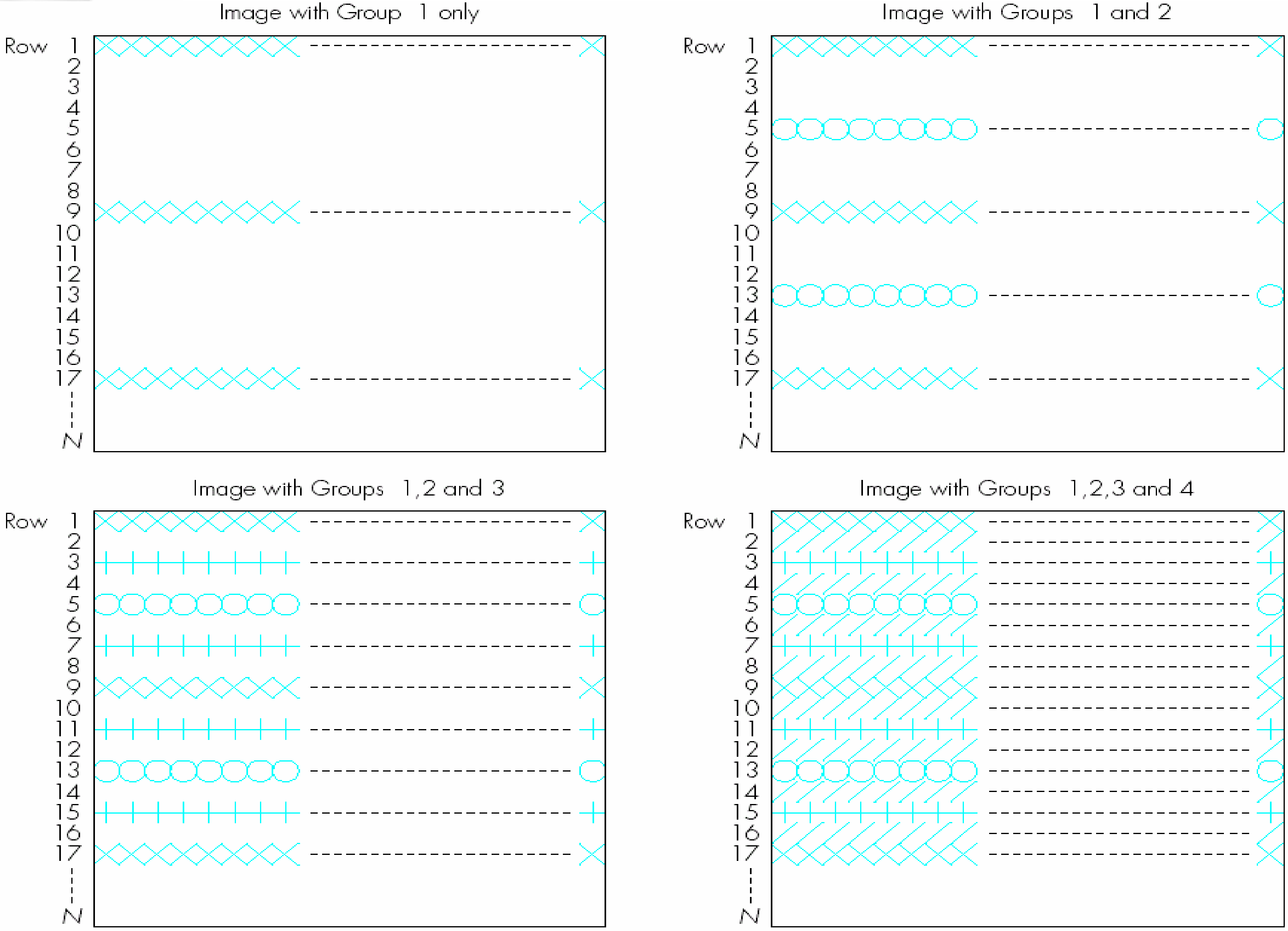


(b) dynamic mode using LZW coding.

(b)



GIF interlaced mode.



8.4.3 Bit-Plane Coding

Bit-plane decomposition

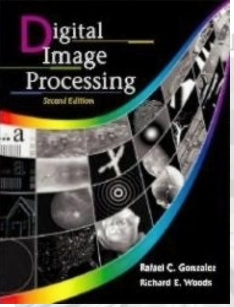
- *m*-bit gray-level image: $a_{m-1}a_{m-2}\ldots\ldots a_1a_0$

  $a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots\ldots + a_1 2^1 + a_0 2^0$
- ***Disadvantage:*** small changes in gray-level can have a significant impact on the complexity of the bit-plane. Gray-levels: 127=01111111 and 128=1000000
- An alternative decomposition approach to reduce the effect of small gray-level variations is to represent the image by *m*-bit ***Gray code***.

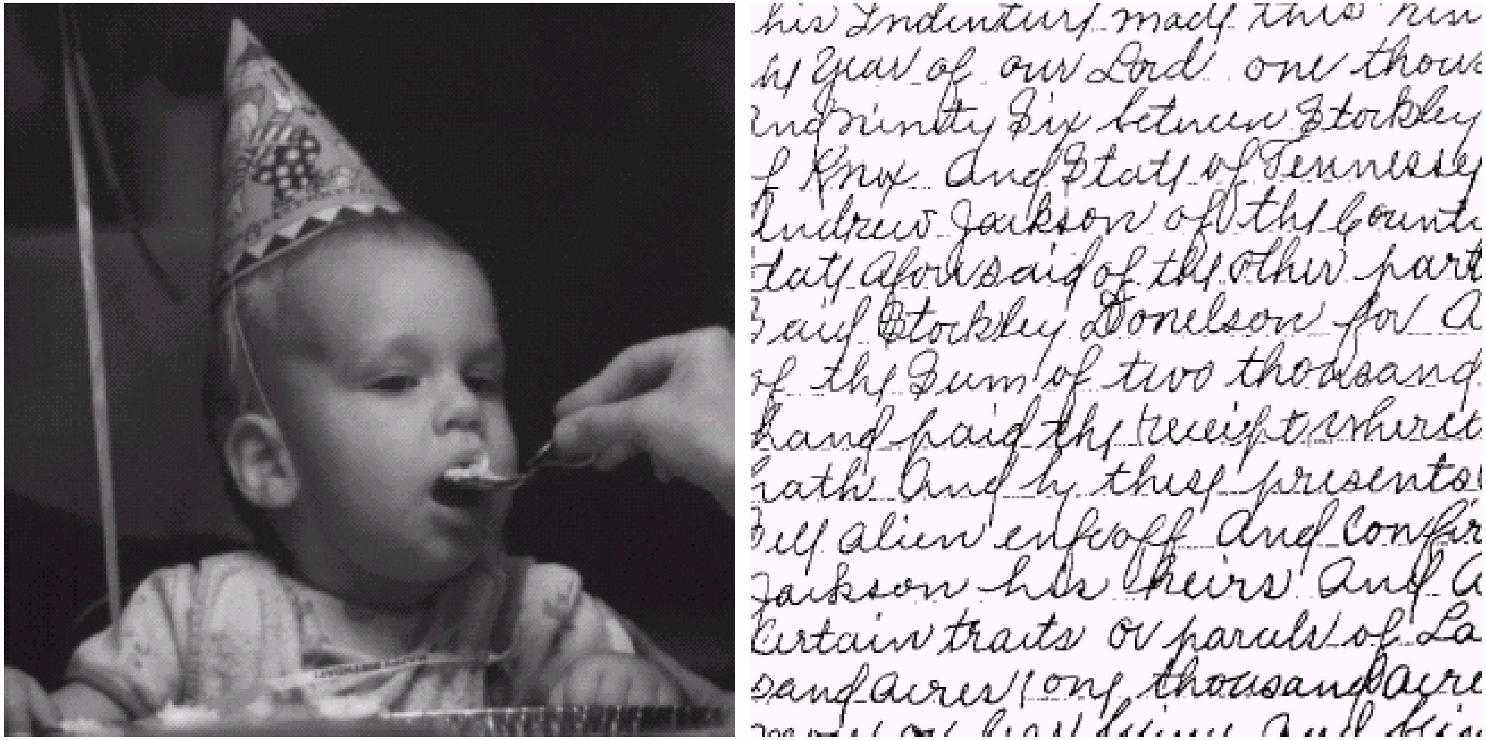
$$a_i \longrightarrow \boxed{a_i \oplus a_{i+1}} \longrightarrow g_i$$

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

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



8.4.3 Bit-Plane Coding



a b

FIGURE 8.14 A 1024 × 1024 (a) 8-bit monochrome image and (b) binary image.

8.4.3 Bit-Plane Coding

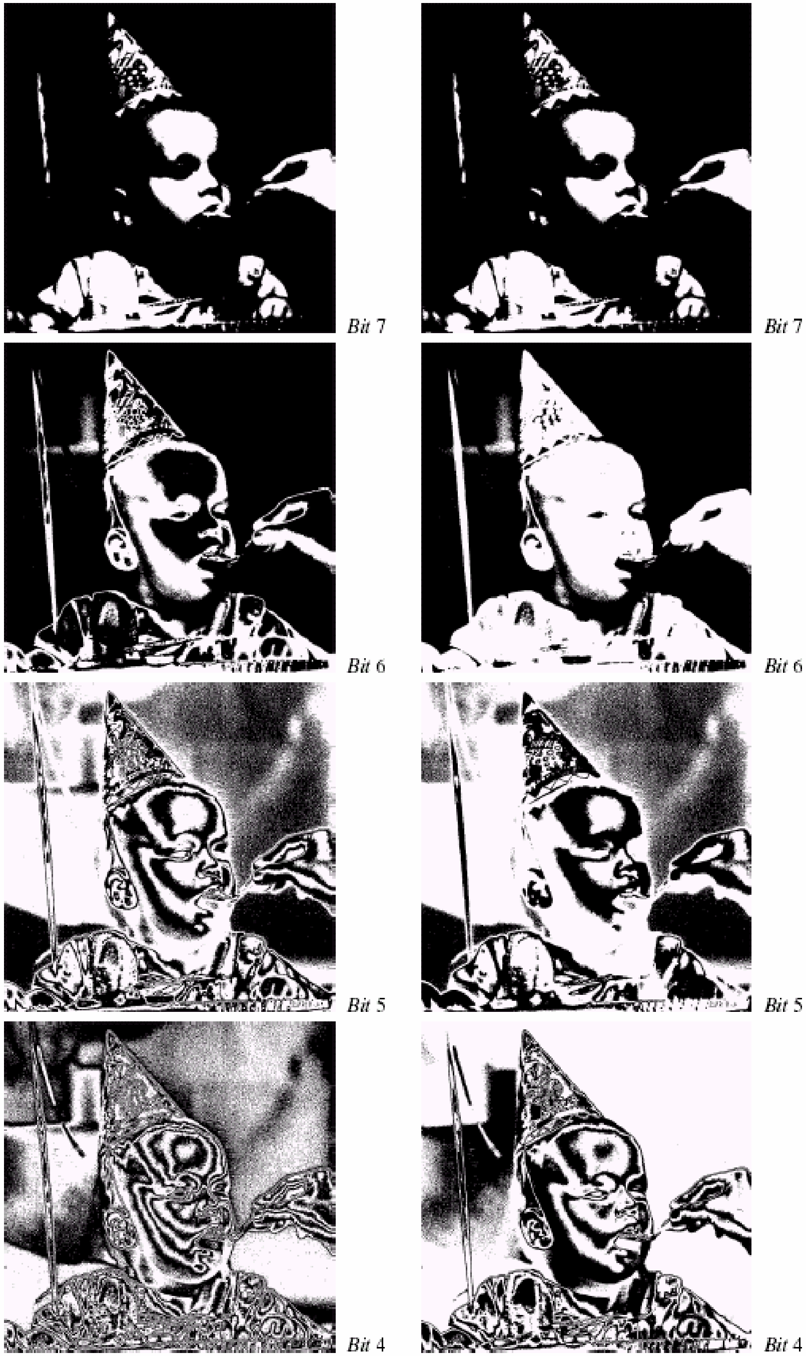


FIGURE 8.15 The four most significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.14(a).

8.4.3 Bit-Plane Coding

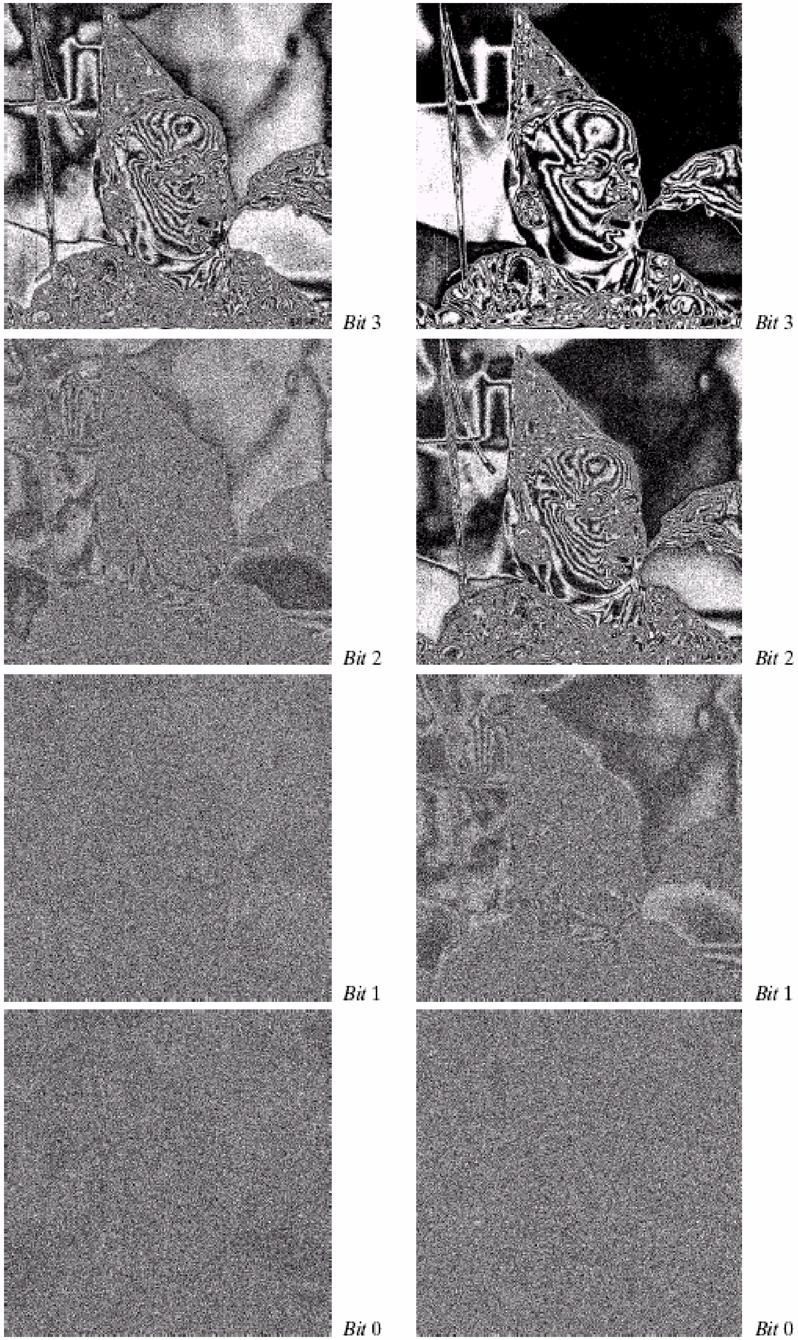
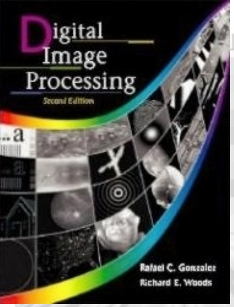


FIGURE 8.16 The four least significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.14(a).

8.4.3 Bit-Plane Coding- Constant area coding

- A special code for a large area of contiguous 1s or 0s.
- In ***Constant Area Coding*** (***CAC***), the image is divided into ***blocks*** of size $p \times q$ pixels, which is classified as ***all white***, ***all black***, or ***mixed***.
- ***White block skipping (WBS):*** for ***text documents***, code the ***solid white area*** (block size $1 \times q$) as ***0*** and other blocks (include the solid black blocks) by a ***1*** followed by the normal ***WBS code seqeunce***.
- Other iterative approach decompose into successively smaller and smaller subblocks.
- If the subblock is not solid white, the decomposition is repeated until a predefine subblock size is reached.



8.4.3 Bit-Plane Coding: 1-D and 2-D 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.

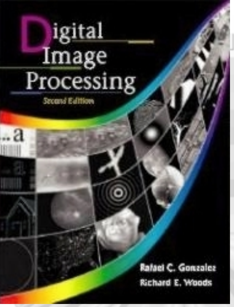
8.4.3 Bit-Plane Coding: 1-D and 2-D run-length coding

- Modified Huffman 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+000011

ITU–T Group 3 and 4 facsimile conversion codes: (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	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
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



(b) make-up codes.

(a) cont.

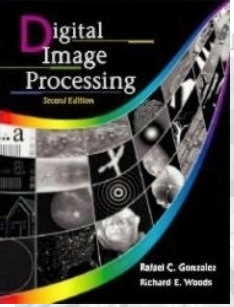
White run-length	Code-word	Black run-length	Code-word
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

(b)

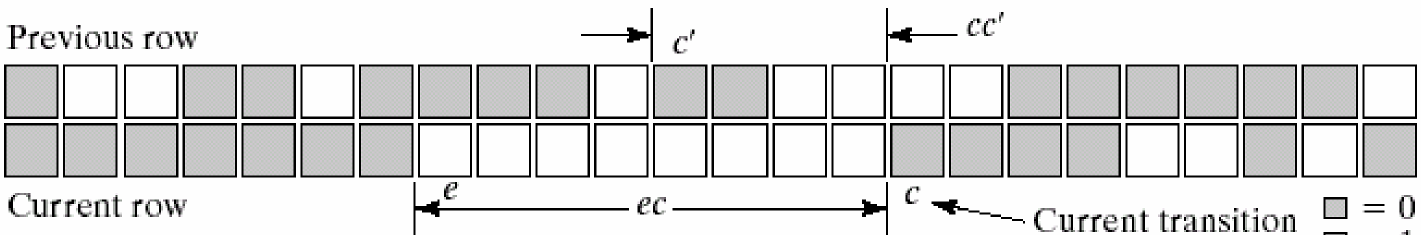
White run-length	Code-word	Black run-length	Code-word
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000001101100
576	01101000	576	0000001101101
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	000000000001	EOL	000000000001

8.4.3 Bit-Plane Coding- 1-D and 2-D run-length coding

- ***Relative address coding (RAC)*** based on the principal of tracking the ***binary transitions*** that begin and end each black and white run.
- *ec* is the distance from the current transition *c* to the last transition of the current line *e*.
- *cc'* is the distance from *c* to the first similar transition past *e* (denoted as *c'*).
- If $ec \leq cc'$, the RAC coded distance $d=ec$ else $d=cc'$
- As shown in Figure 8.17: ec=+8, cc'= +4 (c' to the left of c), d = +4, RAC code=1100011.
- If d=0, RAC code=0, c is directly below c'.
- If d=1, RAC code=100, the decoder has to determine the closest transition point (ec or cc')



8.4.3 Bit-Plane Coding- 1-D and 2-D run-length coding



Distance measured	Distance	Code	Distance range	Code $h(d)$
cc'	0	0	1 – 4	0 xx
ec or cc' (left)	1	100	5 – 20	10 xxxx
cc' (right)	1	101	21 – 84	110 xxxxxx
ec	$d(d>1)$	111 $h(d)$	85 – 340	1110 xxxxxxxx
cc' (c' to left)	$d(d>1)$	1100 $h(d)$	341 – 364	11110 xxxxxxxxxx
cc' (c' to right)	$d(d>1)$	1101 $h(d)$	1365 – 5460	111110 xxxxxxxxxxxx

a
b

FIGURE 8.17 A relative address coding (RAC) illustration.

8.4.3 Bit-Plane Coding- Contour tracing and coding

- Represent each contour by a set of boundary points or by a single boundary point and a set of directions, called ***direct contour tracing***.
- In ***predictive differential quantizing*** (***PDQ***), the front and back contours of each object of an image are traced simultaneously to generate a sequence of (Δ', Δ'') , where Δ' is the difference between the starting coordinates of the front contour on the adjacent lines, and Δ'' is the difference between the front-to-back contour lengths.
- Messages: ***the new start*** and ***the merge***.
- In ***double delta coding*** (***DDC***), we use Δ''' (the difference between the back contour coordinates of adjacent lines) to replace Δ''.
- Both ***PDQ*** and ***DDC*** coding represent Δ', Δ'' or Δ''' , and coordinates of the ***new starts*** and ***merge***s with a suitable ***VLC***

8.4.3 Bit-Plane Coding- Contour tracing and coding

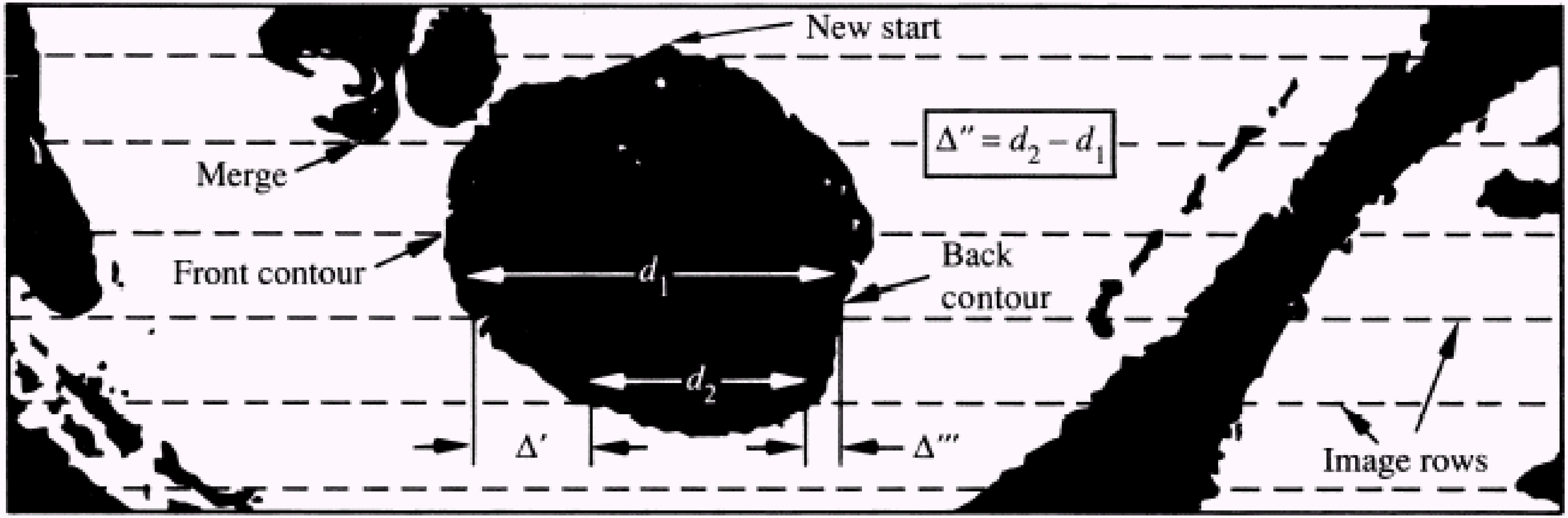
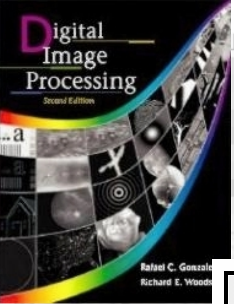


FIGURE 8.18 Parameters of the PDQ algorithm.



8.4.3 Bit-Plane Coding

Method	7	6	5	4	3	2	1	0	Code Rate	Compression Ratio
	Bit-plane code rate (bits/pixel)									
Binary Bit-Plane Coding										
CAC → CBC (4 × 4)	0.14	0.24	0.60	0.79	0.99	—	—	—	5.75	1.4:1
RLC	0.09	0.19	0.51	0.68	0.87	1.00	1.00	1.00	5.33	1.5:1
PDQ	0.07	0.18	0.79	—	—	—	—	—	6.04	1.3:1
DDC	0.07	0.18	0.79	—	—	—	—	—	6.03	1.3:1
RAC	0.06	0.15	0.62	0.91	—	—	—	—	5.17	1.4:1
Gray Bit-Plane Coding										
CAC → CBC (4 × 4)	0.14	0.18	0.48	0.40	0.61	0.98	—	—	4.80	1.7:1
RLC	0.09	0.13	0.40	0.33	0.51	0.85	1.00	1.00	4.29	1.9:1
PDQ	0.07	0.12	0.61	0.40	0.82	—	—	—	5.02	1.6:1
DDC	0.07	0.11	0.61	0.40	0.81	—	—	—	5.00	1.6:1
RAC	0.06	0.10	0.49	0.31	0.62	—	—	—	4.05	1.8:1

TABLE 8.8 Error-free bit-plane coding results for Fig. 8.14(a): $H \approx 6.82$ bits/pixel

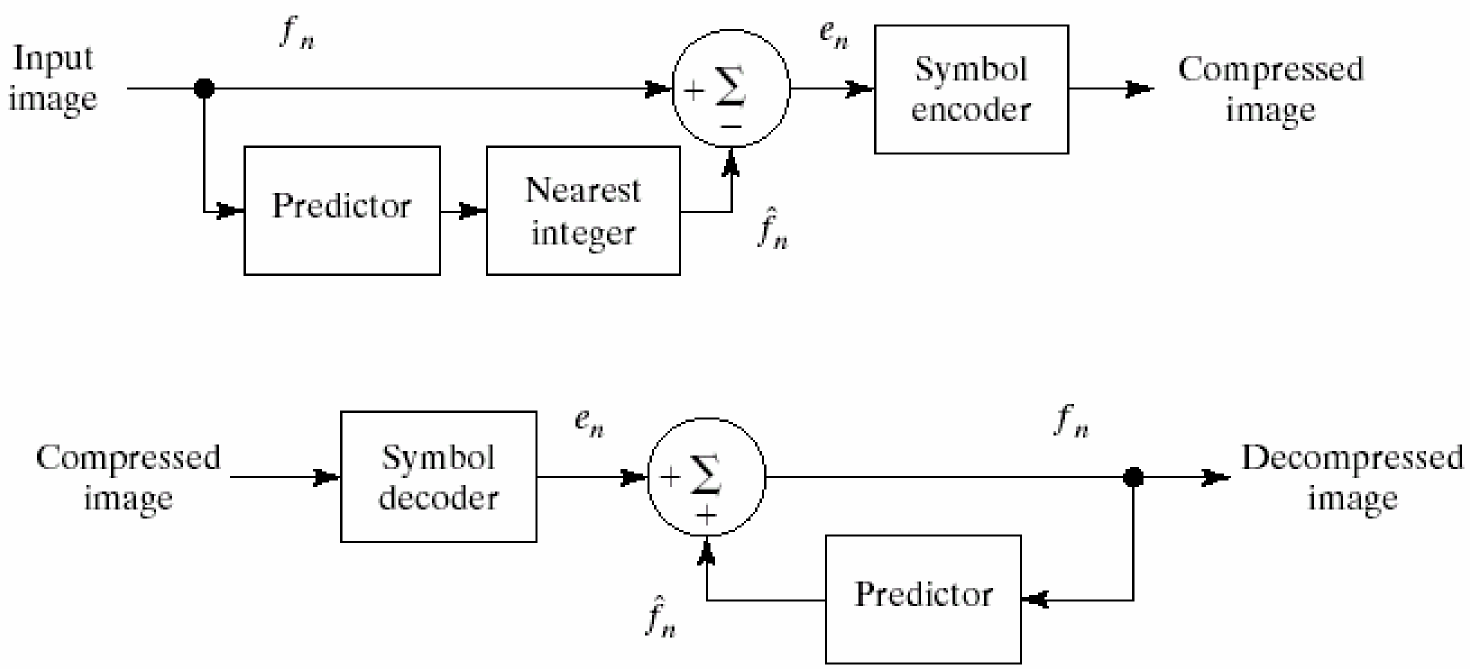
	WBS (1 × 8)	WBS (4 × 4)	RLC	PDQ	DDC	RAC
Code rate (bits/pixel)	0.48	0.39	0.32	0.23	0.22	0.23
Compression ratio	2.1:1	2.6:1	3.1:1	4.4:1	4.7:1	4.4:1

TABLE 8.9 Error-free binary image compression results for Fig. 8.14(b): $H \approx 0.55$ bits/pixel.

8.4.4 Lossless 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 $\hat{f}_n$
- The ***predictor error*** is $e_n = f_n - \hat{f}_n$
- Various local, global, and adaptive methods can be used to generate $\hat{f}_n$, in most of the case, the prediction is formed by a linear combination of m previous pixels as $\hat{f}_n = \textbf{\textit{round}}[\sum_{i=1}^{m} \alpha_i f_{n-i}]$

8.4.4 Lossless predictive coding



a
b

FIGURE 8.19 A lossless predictive coding model: (a) encoder; (b) decoder.

8.4.4 Lossless predictive coding

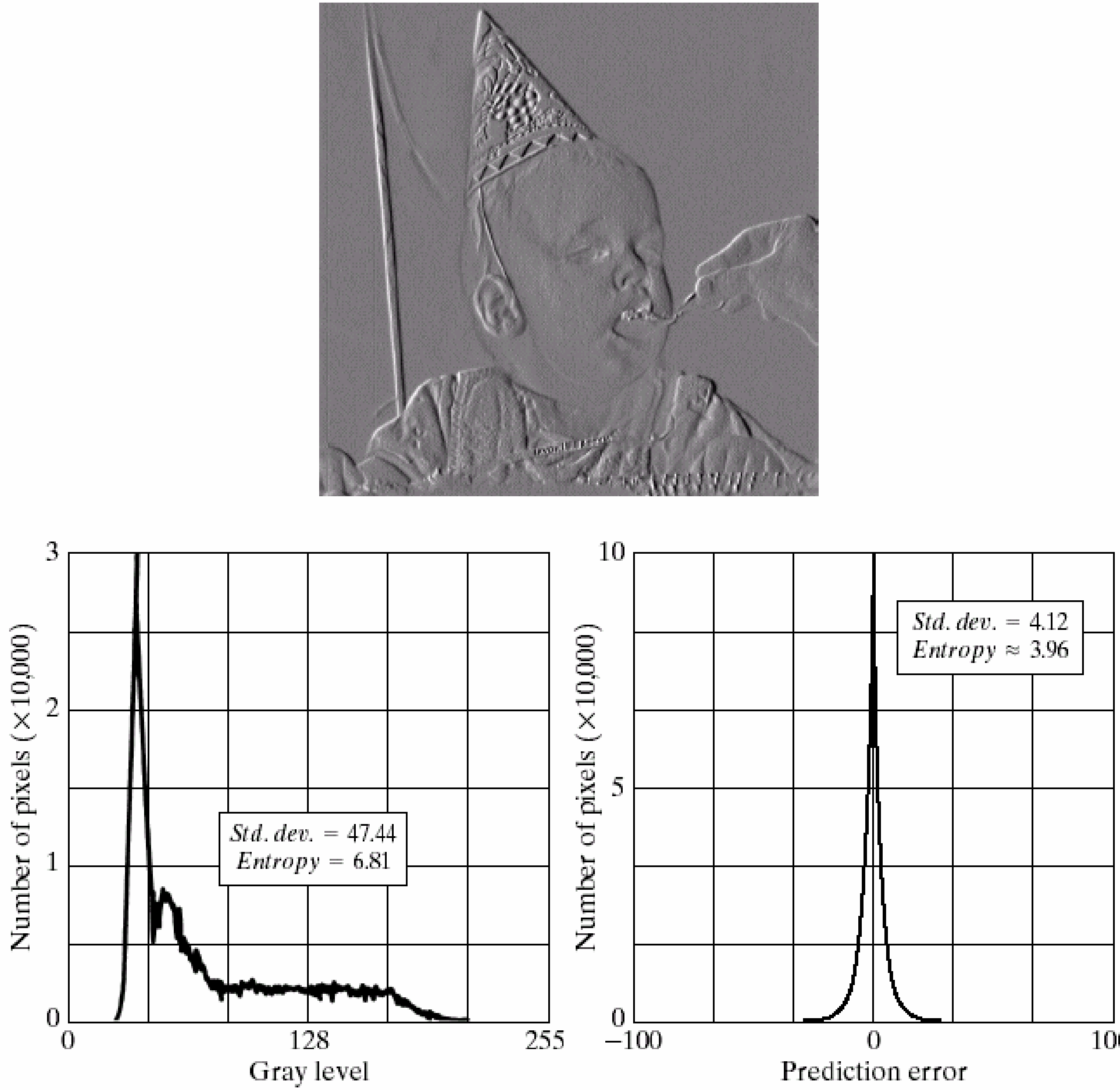


FIGURE 8.20 (a) The prediction error image resulting from Eq. (8.4-9). (b) Gray-level histogram of the original image. (c) Histogram of the prediction error.

8.5 Lossy compression

- 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.

8.5.1 Lossy predictive coding

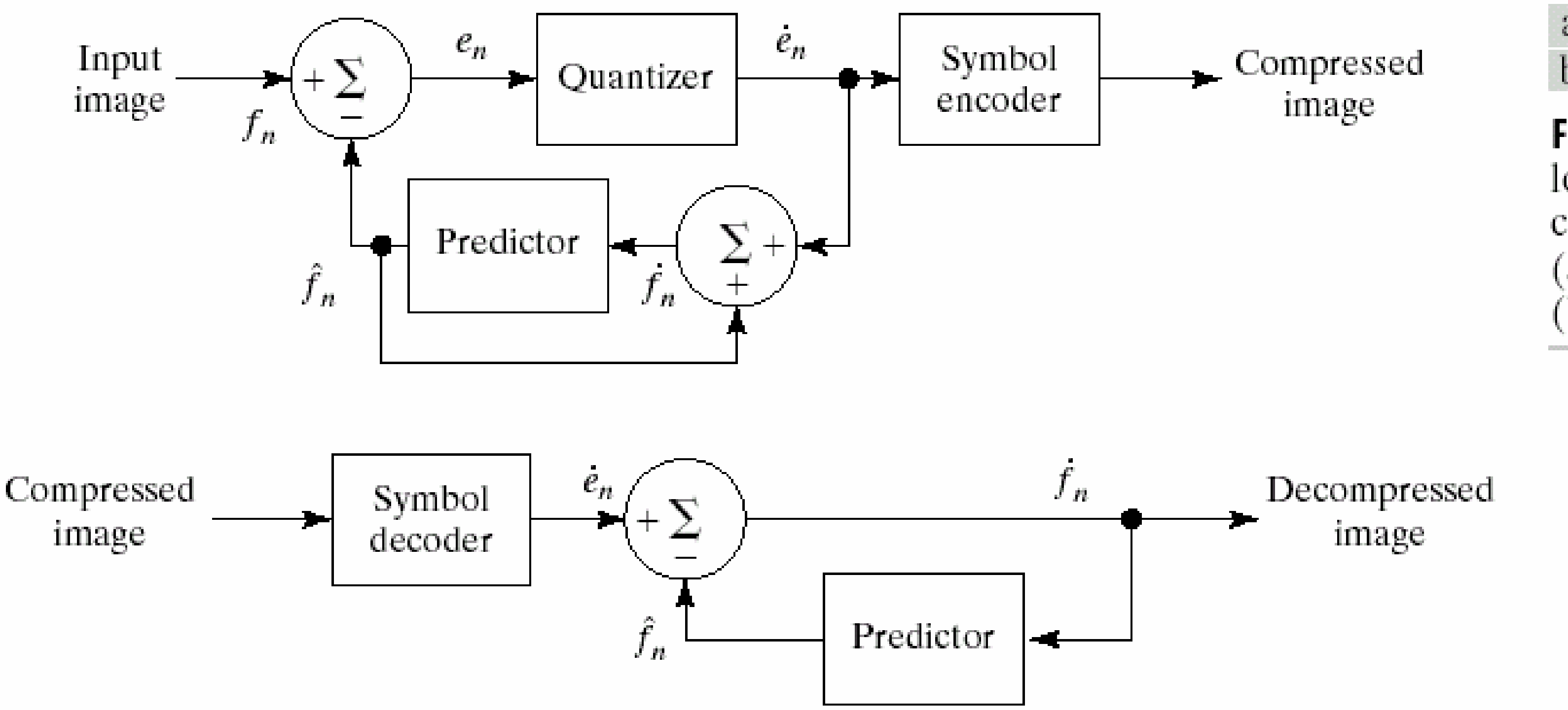


FIGURE 8.21 A lossy predictive coding model: (a) encoder and (b) decoder.

8.5.1 Lossy predictive coding

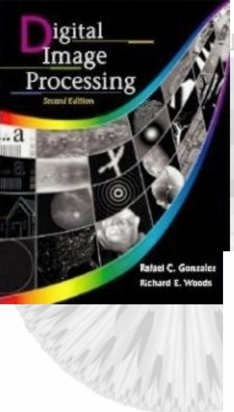
- The quantizer is inserted.
- The predictions generated by the encoder and the decoder must be equivalent.
- Replace the lossy encoder's predictor within a feedback loop, where its input denoted as, $\dot{f}_n$, is generated as a function of past predictions and the corresponding quantized errors, *i.e.*, $\dot{f}_n = \dot{e}_n + \hat{f}_n$
- ***Delta modulation***, the predictor is $\hat{f}_n = \alpha \dot{f}_{n-1}$

  and the quantizer is $\dot{e}_n = \begin{cases} +\zeta & for\ e_n > 0 \\ -\zeta & otherwise \end{cases}$

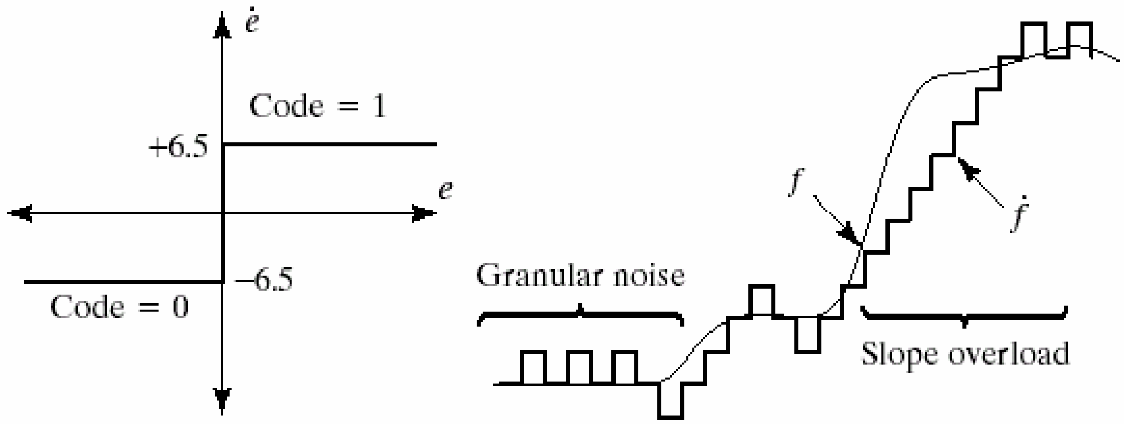
The output quantizer can be represented by single bit.

8.5.1 Lossy predictive coding-Delta Modulation

- 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……}
- α=1 and ζ=6.5
- Initial condition $\dot{f}_0 = f_0 = 14$
- when ζ is too small, the ***slope overload*** occurs, ζ is too large, the ***granular noise*** appears



8.5.1 Lossy predicitve coding



a b
c

FIGURE 8.22 An example of delta modulation.

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

8.5.1 Lossy predicitve coding

- Optimal predictor minimizes the encoder's mean prediction error $E\{e_n^2\} = E\{[f_n - \hat{f}_n]^2\}$
- Subject to the constraints

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

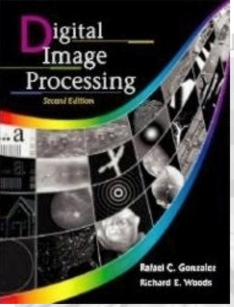
- $\partial E\{e_n^2\}/\partial \alpha_i = 0$

  where $E\{e_n^2\} = E\left\{\left[f_n - \sum_{i=1}^{m} \alpha_i f_{n-i}\right]^2\right\}$
- $\boldsymbol{\alpha} = \mathbf{R}^{-1}\mathbf{r}$ where $\mathbf{R}^{-1}$ is the inverse of the $m \times m$ autocorrelation matrix

8.5.1 Lossy predicitve coding

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

$$\mathbf{r}=\begin{bmatrix} E\{f_{n}f_{n-1}\} \\ E\{f_{n}f_{n-2}\} \\ : \\ E\{f_{n}f_{n-m}\} \end{bmatrix} \qquad \boldsymbol{\alpha}=\begin{bmatrix} \alpha_{1} \\ \alpha_{21} \\ : \\ \alpha_{m1} \end{bmatrix}$$



8.5.1 Lossy predicitve coding

- The variance of the prediction error is

$$\sigma^2_e = \sigma^2 - \alpha^T \mathbf{r} = \sigma^2 - \Sigma_i E\{f_n f_{n-i}\}\ \alpha_i$$

- The generalized four order prediction

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

- $\alpha_1 = \rho_h\ \ \alpha_2 = -\rho_v \rho,\ \alpha_3 = \rho_h,\ \alpha_4 = 0$

  Where ρ_h and ρ_v are the horizontal and vertical correlation coefficients.

8.5.1 Lossy predicitve coding

- ***Example*** : Consider four DPCM predictors

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

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

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

$$\hat{f}(x,y) = \begin{cases} 0.97f(x,y-1) & if\ \Delta h \le \Delta v \\ 0.97f(x-1,y) & otherwise \end{cases}$$

Where Δh = $|f(x-1,y)-f(x-1,y-1)|$ *and* Δv = $|f(x,y-1)-f(x-1,y-1)|$ denote the horizontal and vertical gradients at point *(x, y)*

8.5.1 Lossy predictive coding

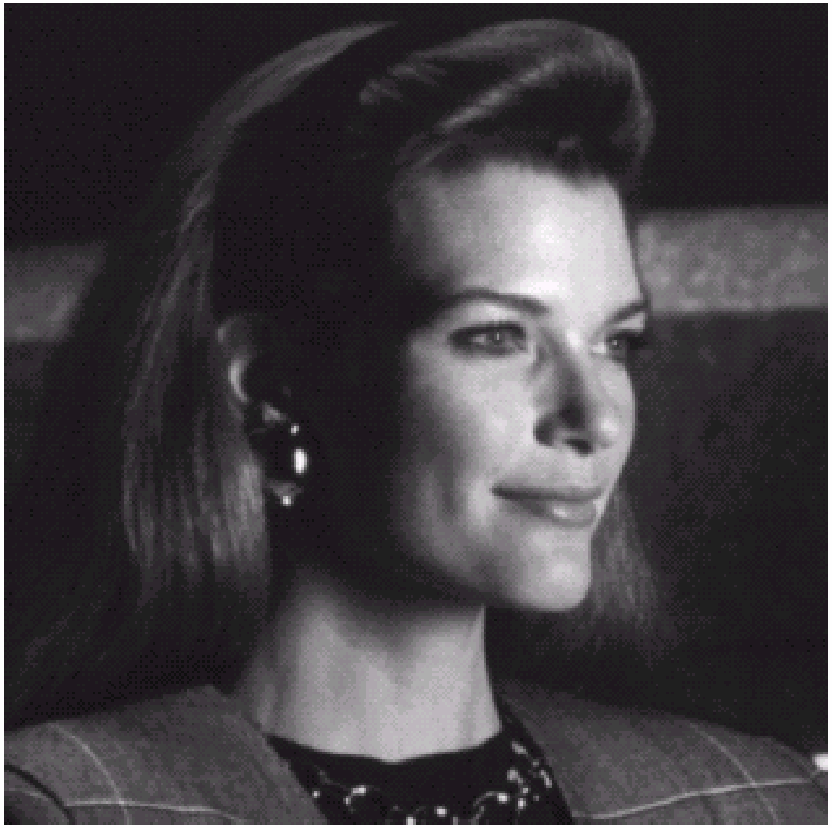
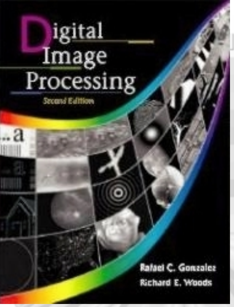


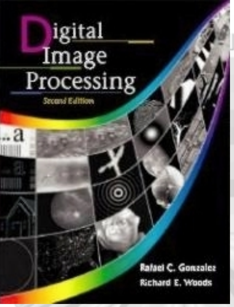
FIGURE 8.23 A 512×512 8-bit monochrome image.



8.5.1 Lossy compression

a b
c d

FIGURE 8.24 A comparison of four linear prediction techniques.



8.5.1 Lossy predictive coding

t=q(s) is an odd function

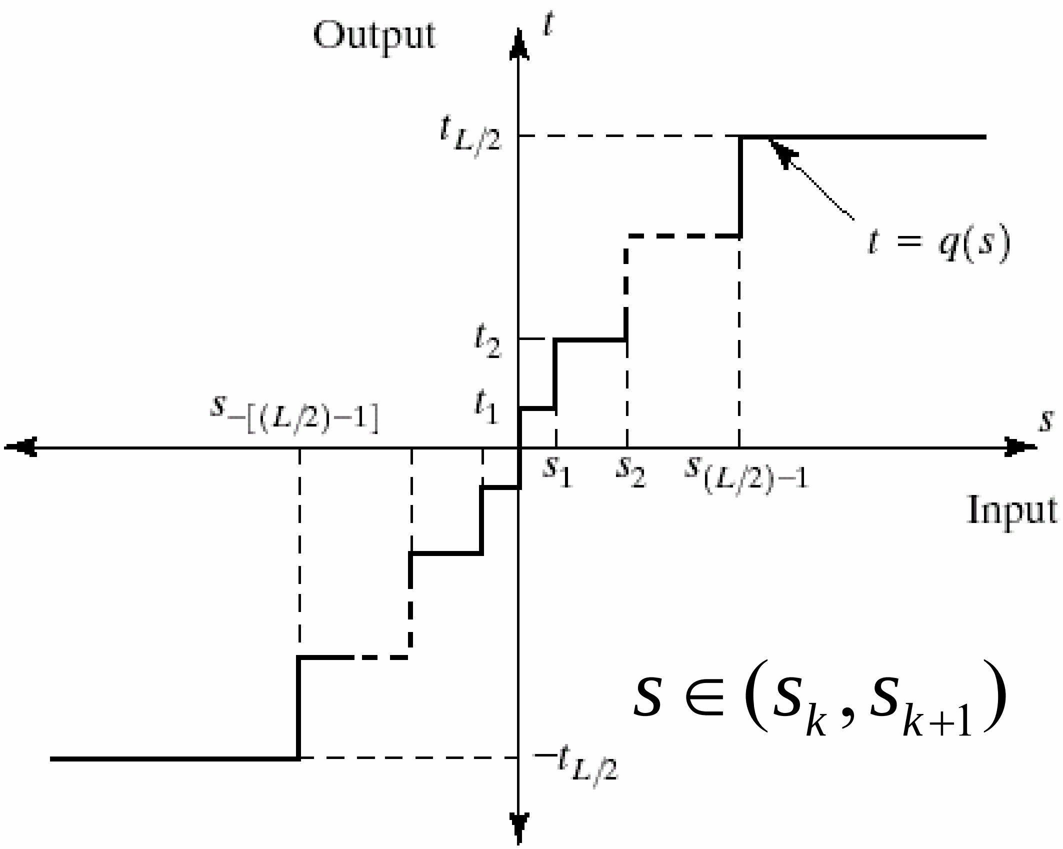


FIGURE 8.25 A typical quantization function.

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

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

8.5.1 Lossy predictive coding : optimal quantization

Example

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

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

$$t_k = s_k + \frac{5}{256}, k = 1, \ldots.256$$

Quantization interval $\theta \triangleq t_k - t_{k-1} = s_k - s_{k-1}$

Zero memory quantizer : one input sampled at one time output value depends only on that input.

8.5.1 Lossy predictive coding : optimal quantization

- **Optimal mean square Quantizer (or 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 ***m.s.e***. is minimized

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

to minimize $\varepsilon = \sum_{i=1}^{L} \int_{s_i}^{s_{i+1}} (s-t_i)^2 P(s)ds$

or $\dfrac{\partial \varepsilon}{\partial t_k} = \dfrac{\partial \varepsilon}{\partial s_k} = 0$

8.5.1 Lossy predictive coding : optimal quantization

- Under the conditions that

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

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

8.5.1 Lossy predictive coding

TABLE 8.10
Lloyd-Max quantizers for a Laplacian probability density function of unit variance.

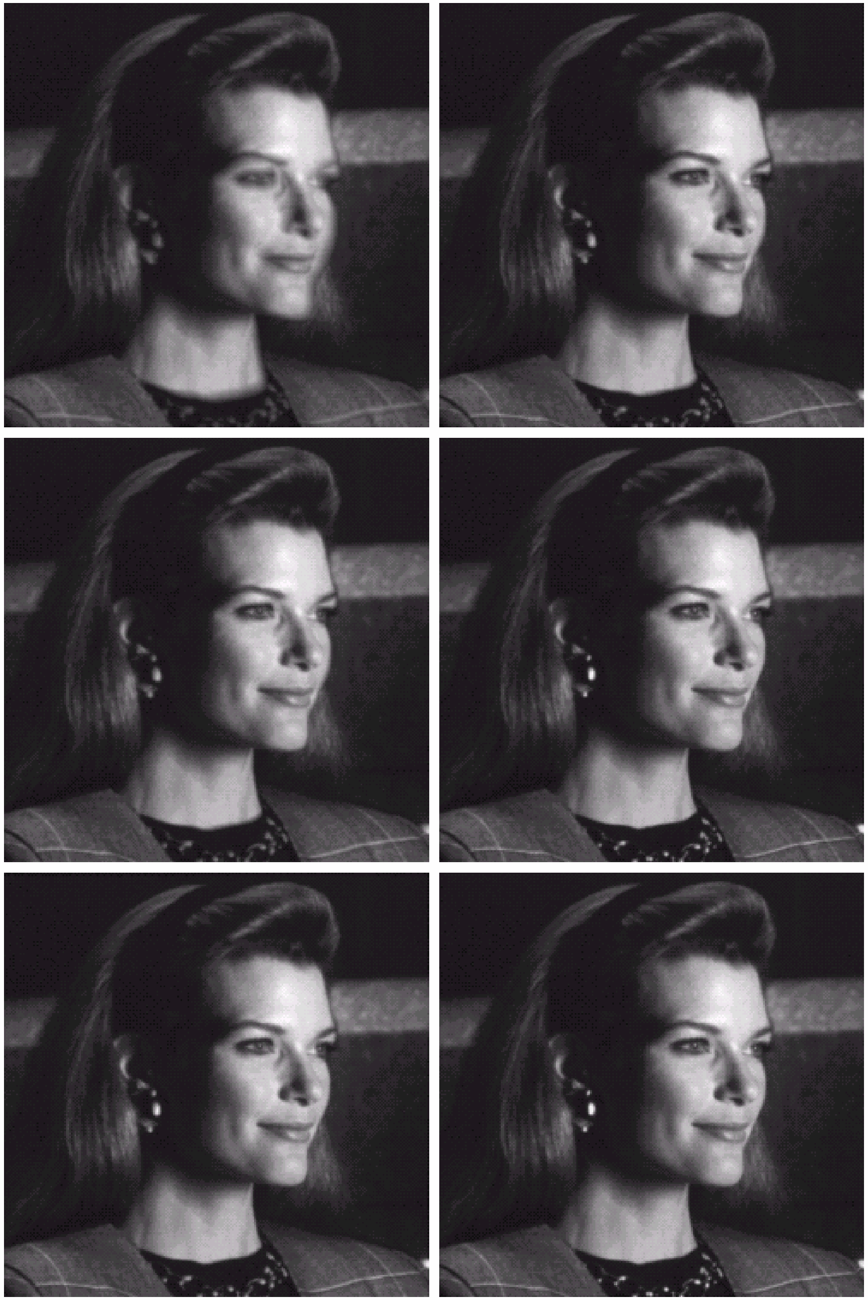
Levels	2		4		8	
i	s_i	t_i	s_i	t_i	s_i	t_i
1	∞	0.707	1.102	0.395	0.504	0.222
2			∞	1.810	1.181	0.785
3					2.285	1.576
4					∞	2.994
θ	1.414		1.087		0.731	

8.5.1 Lossy predictive coding

Predictor	Lloyd-Max Quantizer 2-level	Lloyd-Max Quantizer 4-level	Lloyd-Max Quantizer 8-level	Adaptive Quantizer 2-level	Adaptive Quantizer 4-level	Adaptive Quantizer 8-level
Eq. (8.5-16)	30.88	6.86	4.08	7.49	3.22	1.55
Eq. (8.5-17)	14.59	6.94	4.09	7.53	2.49	1.12
Eq. (8.5-18)	9.90	4.30	2.31	4.61	1.70	0.76
Eq. (8.5-19)	38.18	9.25	3.36	11.46	2.56	1.14
Compression	8.00:1	4.00:1	2.70:1	7.11:1	3.77:1	2.56:1

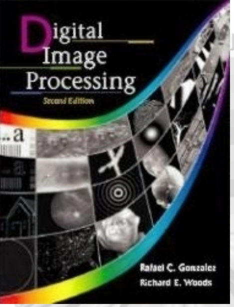
TABLE 8.11
Lossy DPCM root-mean-square error summary.

8.5.1 Lossy predictive coding

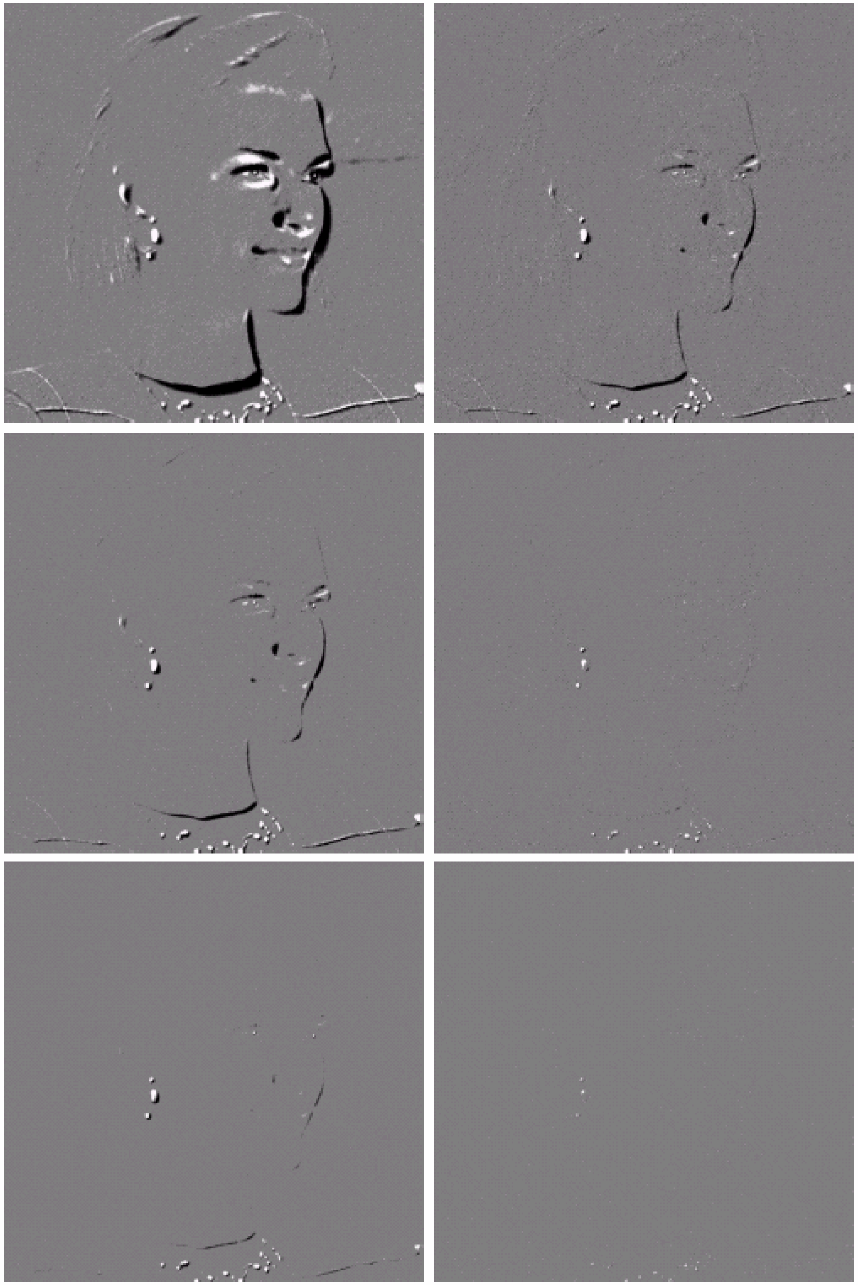


a b
c d
e f

FIGURE 8.26 DPCM result images: (a) 1.0; (b) 1.125; (c) 2.0; (d) 2.125; (e) 3.0; (f) 3.125 bits/pixel.

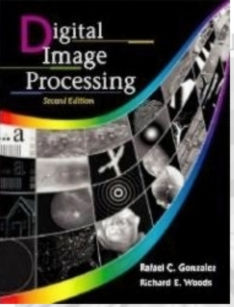


8.5.1 Lossy predictive coding



a	b
c	d
e	f

FIGURE 8.27 The scaled (×8) DPCM error images that correspond to Figs. 8.26(a) through (f).



8.5.2 Transform Coding

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

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

- The ***reverse transform*** *is*

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

- The transformation kernel is separable *as*

$$g(x, y, u, v) = g_1(x, u) g_2(y, v)$$

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

$$g(x, y, u, v) = e^{-j2\pi(ux+vy)/N}/N^2 = g_1(x, u) g_2(y, v)$$

$$h(x, y, u, v) = e^{j2\pi(ux+vy)/N} = h_1(x, u) h_2(y, v)$$

8.5.2 Transform Coding

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

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

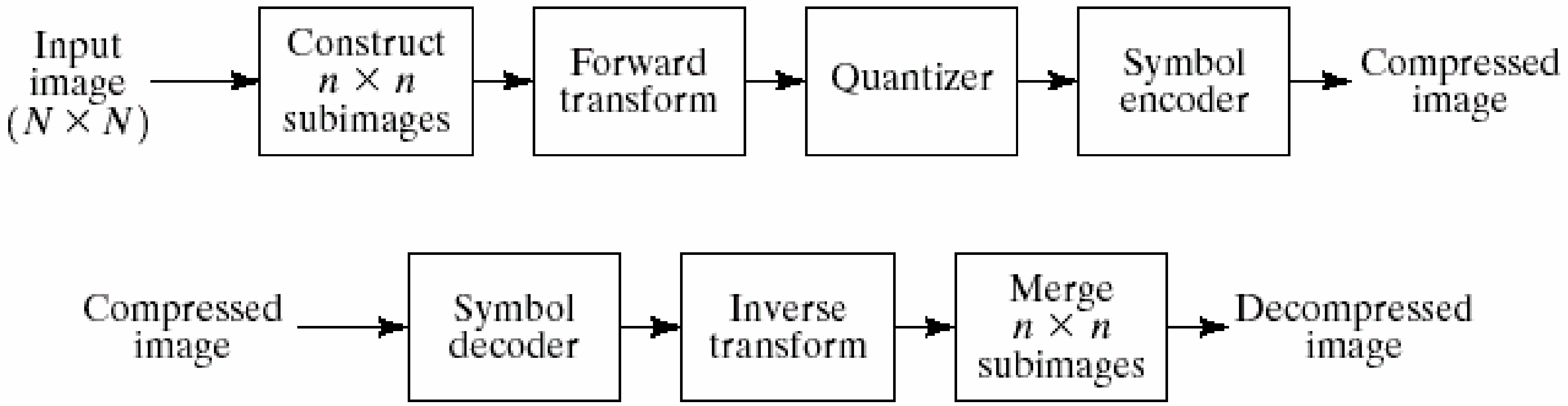
  where $N=2^m$, the summation is performed in *modulo 2* arithmetic and $b_k(z)$ is the kth bit in the binary representation of z.
- If $m=3$, $z=6(110)$, then $b_0(z)=0$, $b_1(z)=1$ and $b_2(z)=1$
- The $p_i(u)$ are defined as follows:

  $p_0(u)=b_{m-1}(u)$, $p_1(u)=b_{m-1}(u)+b_{m-2}(u)$,

  $p_2(u)=b_{m-2}(u)+b_{m-3}(u)$, ...,$p_{m-1}(u)=b_1(u)+b_0(u)$

  where the sums are performed in *modulo 2* arithmetic.

8.5.2 Transform Coding



a
b

FIGURE 8.28 A transform coding system: (a) encoder; (b) decoder.

8.5.2 Transform Coding

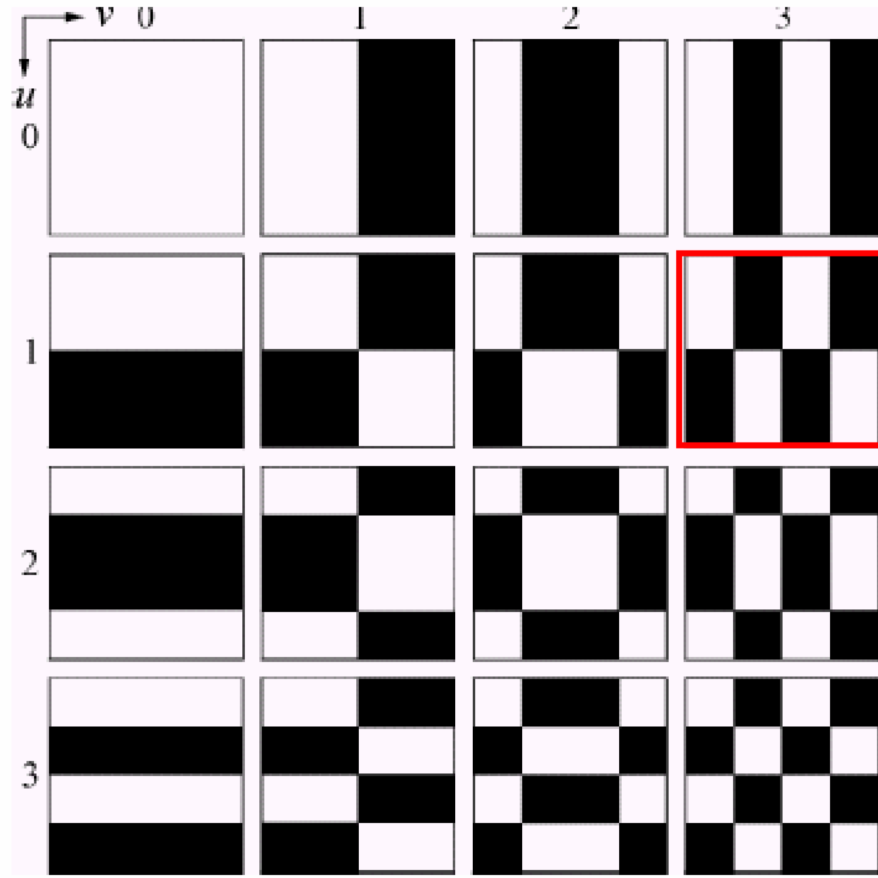
- Let $\boldsymbol{H} = [g_1(x, u,)] = [g_2(y, v)]$ is real, symmetry and orthogonal with the property $H = H^* = H^T = H^{-1}$

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

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

$$H_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \frac{1}{\sqrt{8}}\left[\begin{array}{cc:cc:cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ \hdashline 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hdashline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right]$$

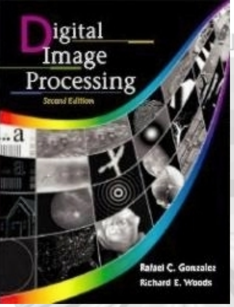
8.5.2 Transform Coding



$\mathbf{u}_1=[1\ 1\ -1\ -1]$

$\mathbf{v}_3=[1\ -1\ 1\ -1]$

FIGURE 8.29 Walsh-Hadamard basis functions for $N = 4$. The origin of each block is at its top left.



8.5.2 Transform Coding

- ***Discrete cosine transform***(DCT) is derived from the identical kernels as

$$g(x, y, u, v)=h(x, y, u, v)$$

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

where

$$\alpha(u)=\begin{cases}\sqrt{\dfrac{1}{N}} & for\ u=0\\[2ex] \sqrt{\dfrac{1}{N}} & for\ u=1,2,\dots N-1\end{cases}$$

8.5.2 Transform Coding

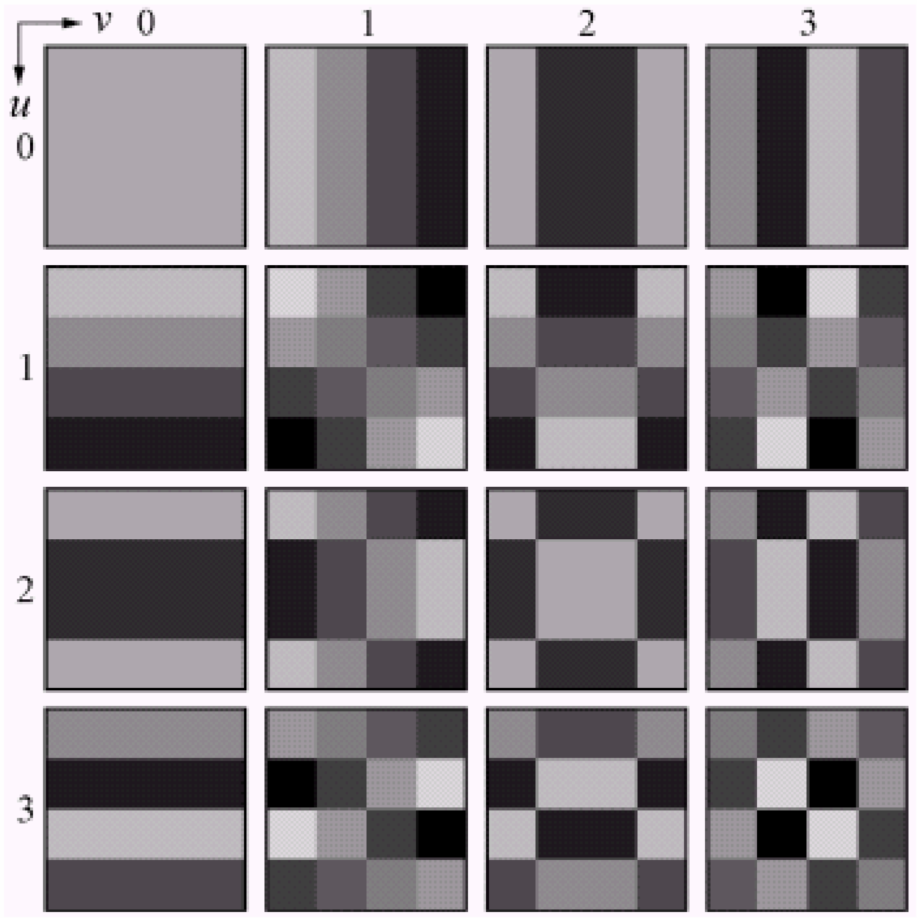
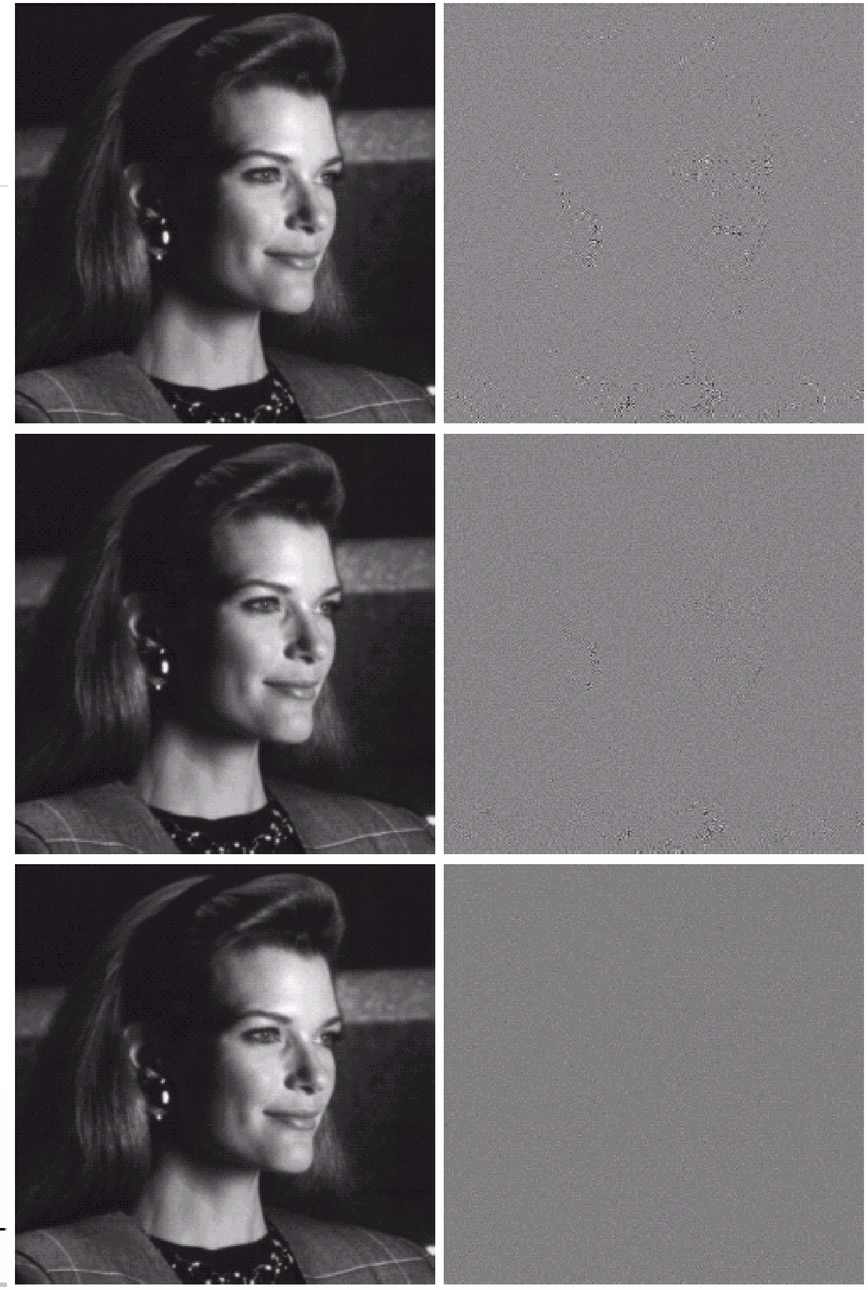


FIGURE 8.30 Discrete-cosine basis functions for $N = 4$. The origin of each block is at its top left.

8.5.2 Transform Coding

- Image of size 512×512 is divided into 8×8 subimages.
- Half of the transform coefficients are discarded.
- The actual ***rms*** errors are 1.28, 0.86, 0.68



a	b
c	d
e	f

FIGURE 8.31 Approximations of Fig. 8.23 using the (a) Fourier, (c) Hadamard, and (e) cosine transforms, together with the corresponding scaled error images.

8.5.2 Transform Coding

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

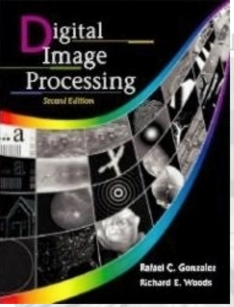
  or $$\mathbf{F}=\sum_{u=0}^{n-1}\sum_{v=0}^{n-1}T(u,v)\mathbf{H}_{uv}$$

- The image $\mathbf{F}=[f(x,y)]$ (a $n\times n$ matrix) is composed of a set of basis images, and

$$\mathbf{H}_{\mathrm{uv}}=\begin{bmatrix} h(0,0,0,0) & h(0,1,u,v) & \dots & h(0,n-1,u,v) \\ h(1,0,u,v) & . & \dots & . \\ . & . & \dots & . \\ h(n-1,0,u,v) & h(n-1,1,u,v) & \dots & h(n-1,n-1,u,v) \end{bmatrix}$$

- Transform coefficient masking function

$$\gamma(u,v)=\begin{cases}0 & \textit{if } \mathrm{T(u,\ v)\ satisfies\ a\ specified\ \ truncatio\ n\ criterion} \\ 1 & \textit{otherwise}\end{cases}$$



8.5.2 Transform Coding

- An approximation of **F** can be obtained from the truncation as

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

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

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

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

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

8.5.2 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.

8.5.2 Transform Coding

The advantage of DCT : It avoids the boundary discontinuity which may cause the Gibbs Phenomenon.

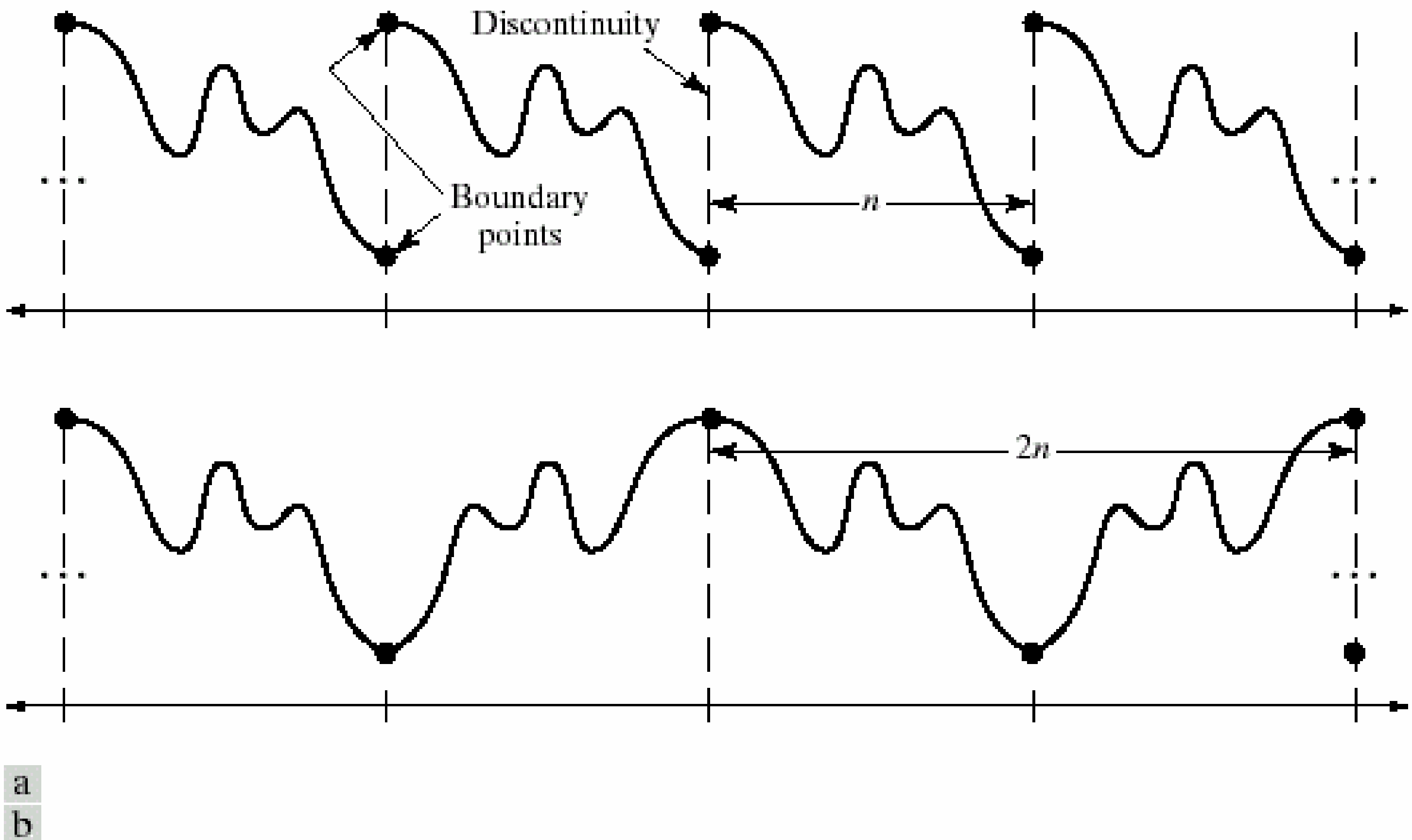
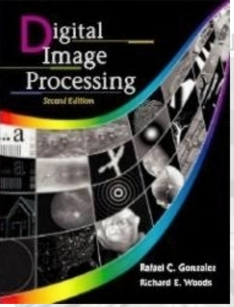


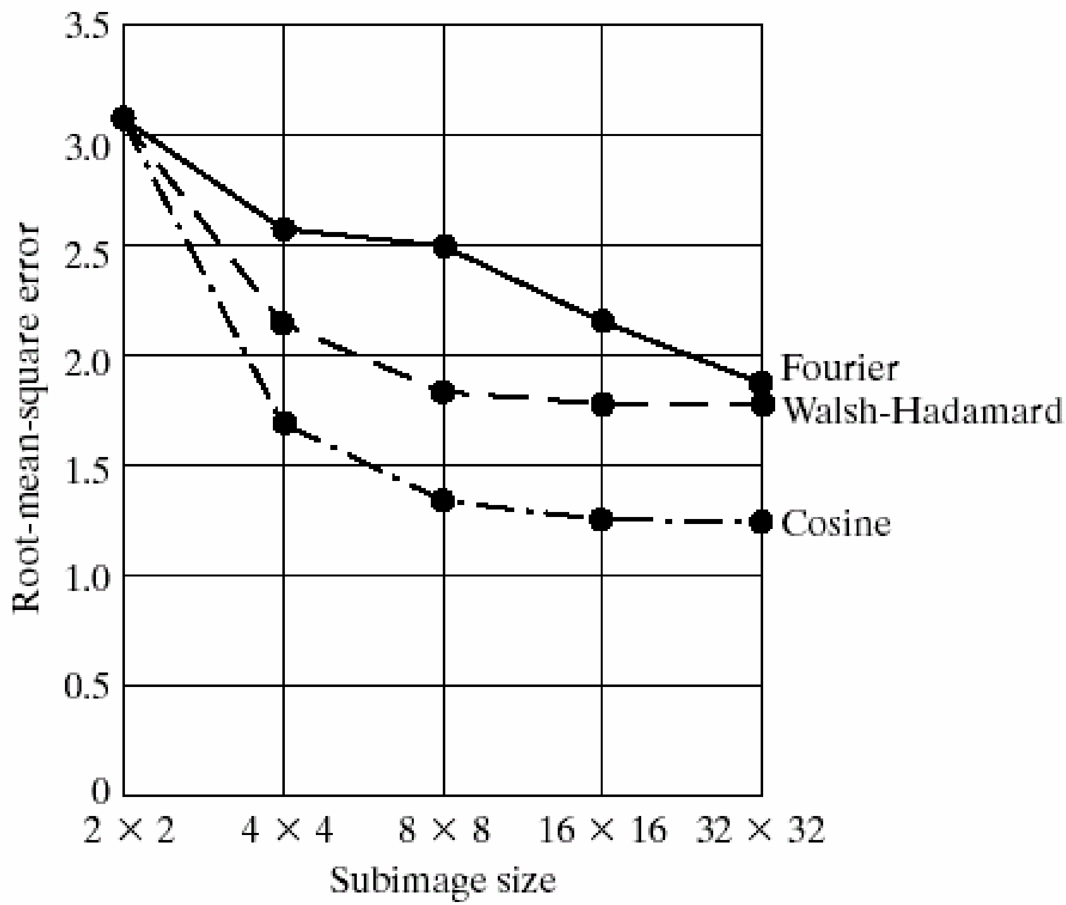
FIGURE 8.32 The periodicity implicit in the 1-D (a) DFT and (b) DCT.



8.5.2 Transform Coding

FIGURE 8.33 Reconstruction error versus subimage size.

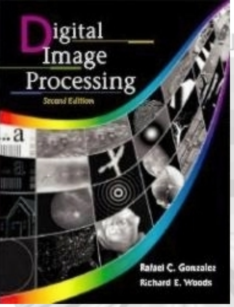
Images are subdivided so that the correlation (redundancy) between adjacent subimages is reduced to some acceptable level



8.5.2 Transform Coding

a b
c d
e f

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



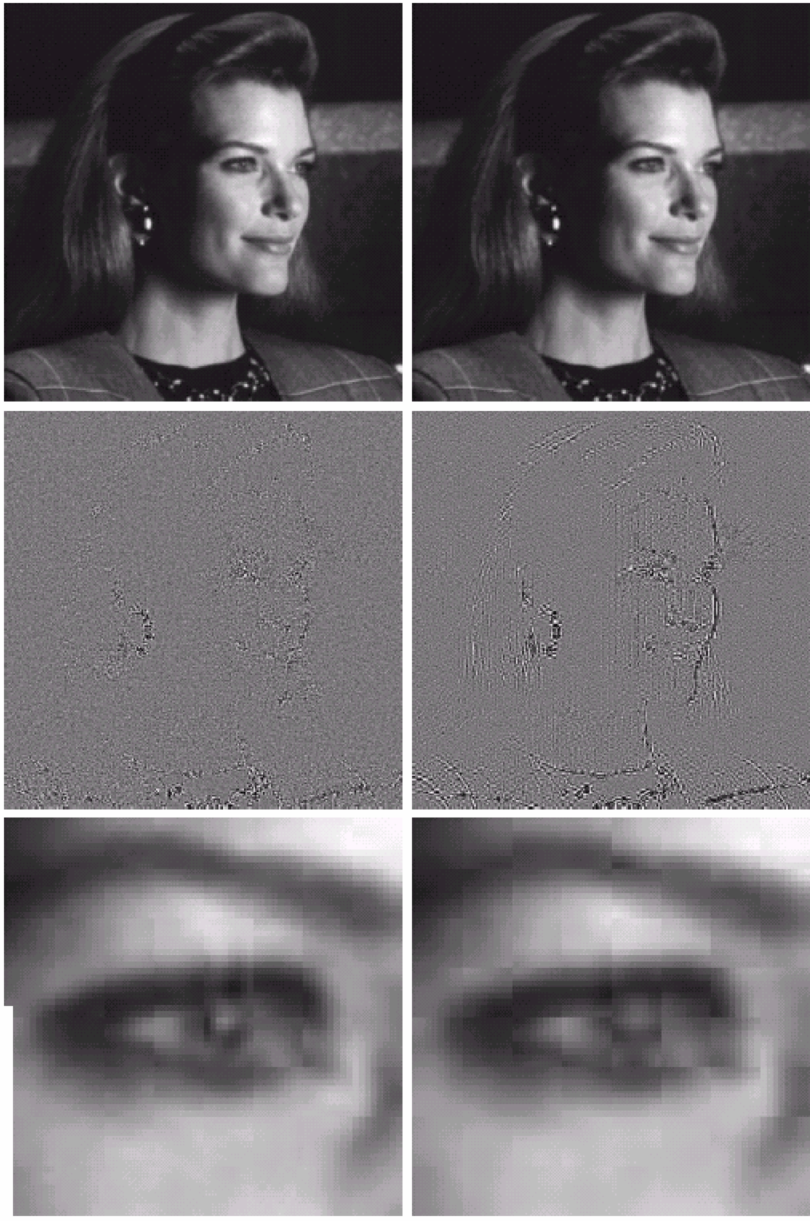
8.5.2 Transform Coding

Bit allocation: *Truncation*, *Quantization*, and *Coding* of the transform coefficients.

The retained coefficients are selected based on

(a) maximal variance – *zonal coding*

(b) maximum magnitude – *thresholding coding*



a b
c d
e f

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

8.5.2 Transform Coding- 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 can not 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 $\log_2 (\sigma^2/D)$.

8.5.2 Transform Coding

(a)

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

(b)

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

(c)

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

(d)

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

a b
c d

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

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

8.5.2 Transform Coding- Threshold coding

- The location of the transform coefficients retained for each subimage vary from one subimage to another.
- 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***.

8.5.2 Transform Coding- Zonal 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.

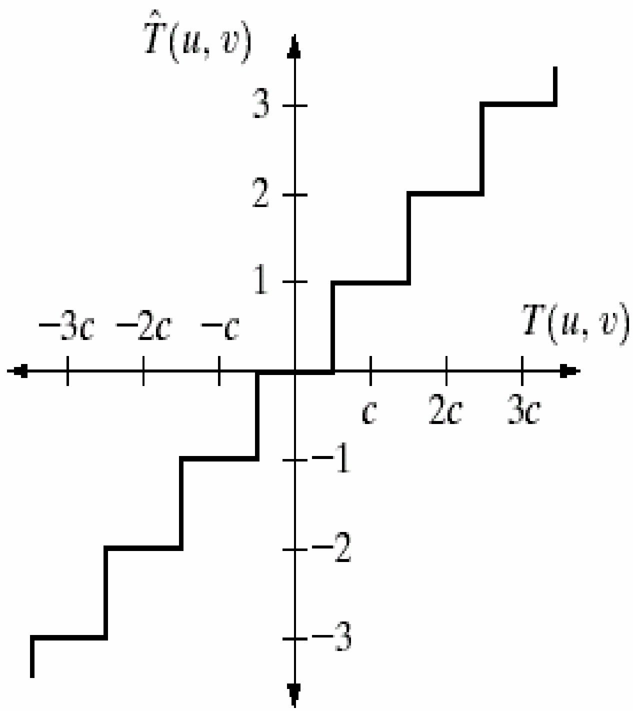
N-largest coding: the same number of coefficients is discarded for each subimage. ***The coding rate is constant***.

(3) The threshold can be varied as a function of location of each coefficient. It results in a variable code rate. The thresholding and quantization can be combined by

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

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

8.5.2 Transform Coding



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 8.37 (a) A threshold coding quantization curve [see Eq. (8.5-40)]. (b) A typical normalization matrix.

8.5.2 Transform Coding

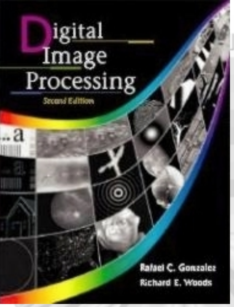
- The de-normalization (decompression) results are

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

- Assume $Z(u, v)=c$ and $\hat{T}(u,v) = k$ then

$$kc - c/2 \leq T(u, v) \leq kc + c/2$$

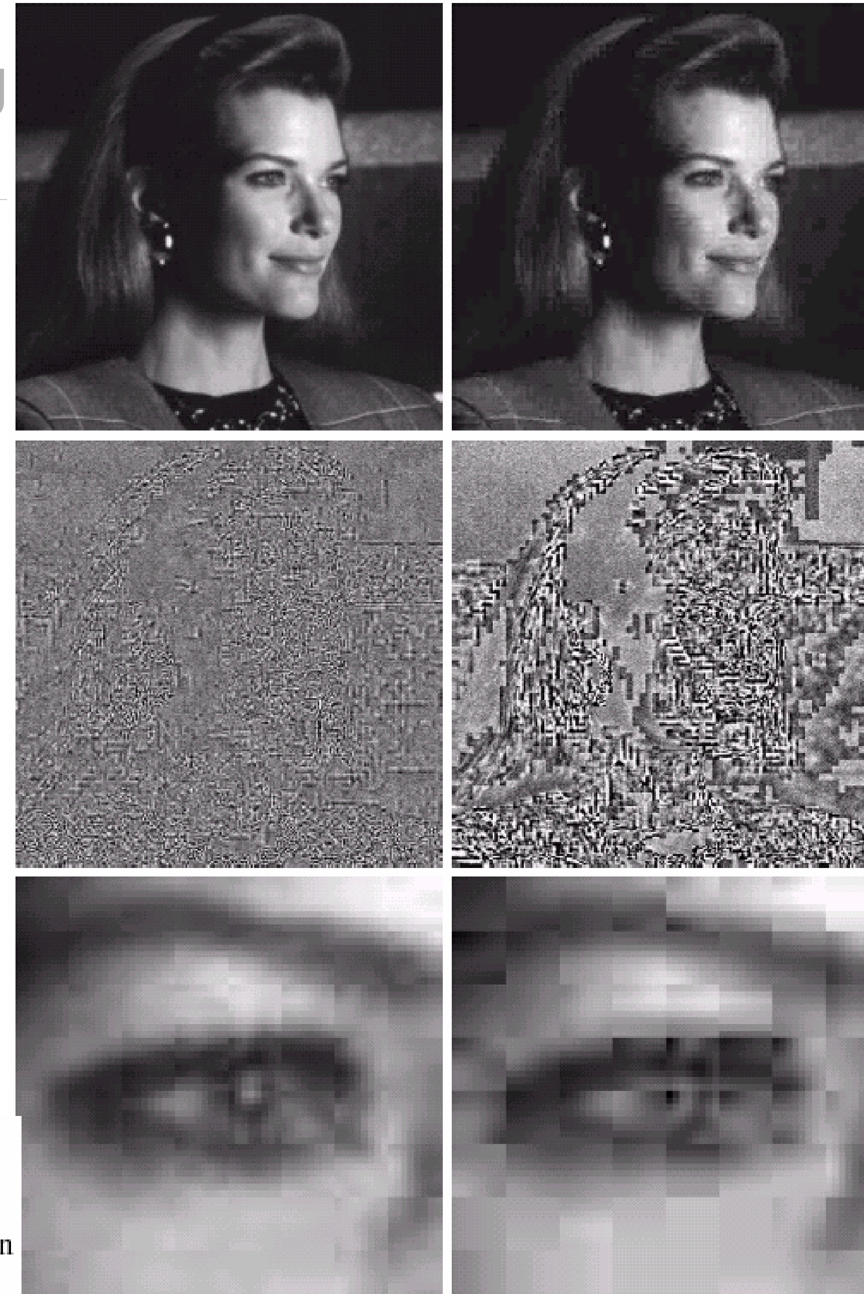
- If $Z(u, v)>2T(u, v)$ then $T(u, v)$ is completely truncated and discarded.
- $\hat{T}(u,v)$ is coded by variable length coding (*i.e.*, Huffman code), of which the code length increases with the magnitude of k.



8.5.2 Transform Coding

The threshold-coding uses 8×8 DCT and normalization array in Fig. 8.37(b).

Compression ratio 34:1 and 67:1 (4 times the normalization array), the corresponding ***rms*** errors are 3.42 and 6.33



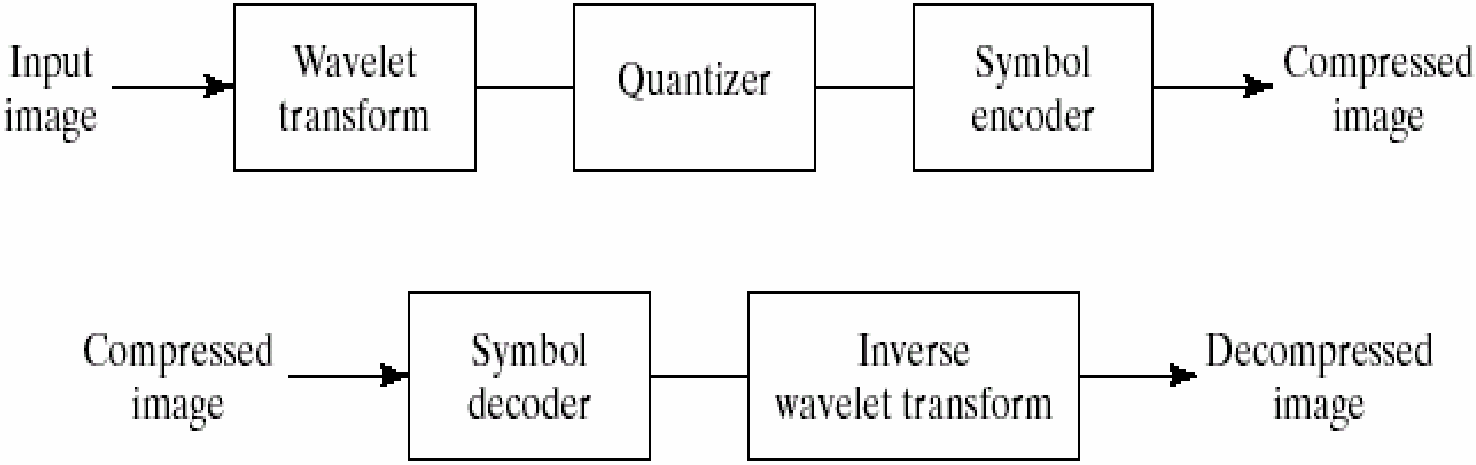
a b
c d
e f

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

8.5.3 Wavelet Coding

- Wavelet transform coefficients are ***decorrelated***.
- Packing most of the important visual information into a small number of coefficients- ***energy compaction***.
- *Difference* between *transform coding* and *wavelet coding* is the ***omission of subimages***.
- The wavelet transform is *inherently local* (*i.e*., wavelet transform inherit time as well as frequency resolutions, their ***basis functions are limited in duration***).
- The horizontal, vertical, and diagonal wavelet coefficients are zero mean and Laplacian-like distributions. Many of them carry little visual information, they can be quantized and coded using run-length, Huffman, or arithmetic coding.

8.5.3 Wavelet Coding



a
b

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

8.5.3 Wavelet Coding

Compression ratios are 34:1 and 67:1

The ***rms*** errors are 2.29 and 2.96

a b
c d
e f

FIGURE 8.40 (a), (c), and (e) Wavelet coding results comparable to the transform-based results in Figs. 8.38(a), (c), and (e); (b), (d), and (f) similar results for Figs. 8.38(b), (d), and (f).

8.5.3 Wavelet Coding

Compression ratios are 108:1 and 167:1

The ***rms*** errors are 3.72 and 4.73

a	b
c	d
e	f

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

8.5.3 Wavelet Coding- Wavelet selection

- The wavelet transformation can be implemented as a sequence of digital filtering operation.
- The ability of the wavelet to pack information into a small number of transform coefficients determines it compression and reconstruction performance.

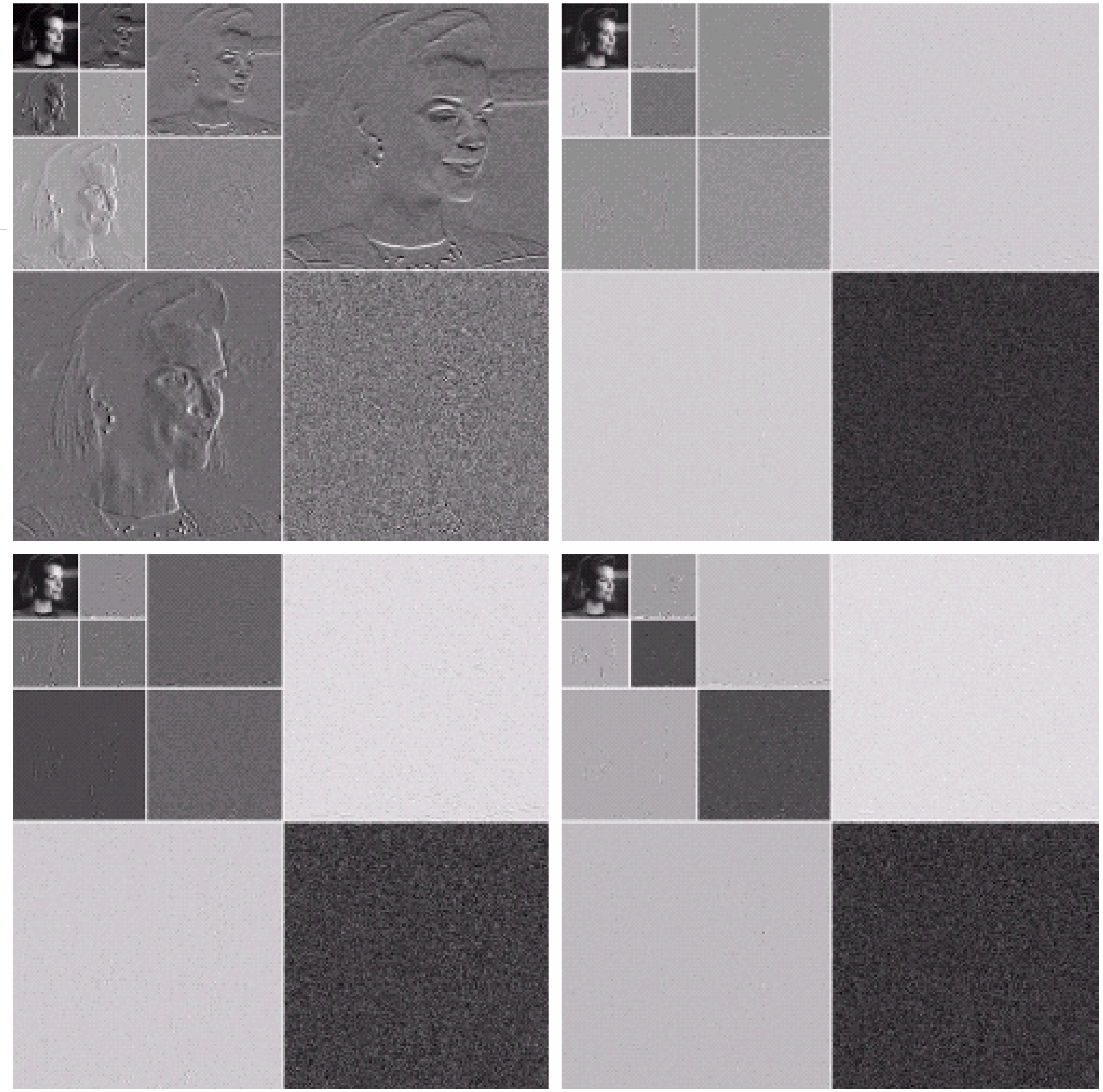
8.5.3 Wavelet Coding

(a) Harr wavelets

(b) Daubechies wavelets

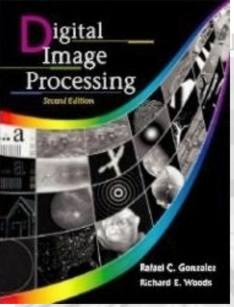
(c) Symlets: An extension of Daubechies wavelet.

(d) Biorthogonal wavlets



a b
c d

FIGURE 8.42 Wavelet transforms of Fig. 8.23 with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.



8.5.3 Wavelet Coding

Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar (see Ex. 7.10)	2 + 2	46%
Daubechies (see Fig. 7.6)	8 + 8	51%
Symlet (see Fig. 7.24)	8 + 8	51%
Biorthogonal (see Fig. 7.37)	17 + 11	55%

TABLE 8.12 Wavelet transform filter taps and zeroed coefficients when truncating the transforms in Fig. 8.42 below 1.5.

Scales and Filter Bank Iterations	Approximation Coefficient Image	Truncated Coefficients (%)	Reconstruction Error (rms)
1	256 × 256	75%	1.93
2	128 × 128	93%	2.69
3	64 × 64	97%	3.12
4	32 × 32	98%	3.25
5	16 × 16	98%	3.27

TABLE 8.13 Decomposition level impact on wavelet coding the 512 × 512 image of Fig. 8.23.

8.5.3 Wavelet Coding-decomposition level selection

- The number of operations in computation increase with the number of decomposition levels.
- Quantizing the increasingly low-scale coefficients that result with more reconstruction impact on increasing larger areas of the reconstructed image.
- From Table 8.13, the initial decompositions are responsible for the majority of the data compression.
- There is little change in the number of truncated coefficients above three decomposition levels.

8.5.3 Wavelet Coding-quantization

- The effectiveness of quantization can be improved by:

  (1) Introducing an enlarged guantization interval around zero, called ***dead zone.***

  (2) Adapting the size of quantization interval from scale to scale.

8.5.3 Wavelet Coding

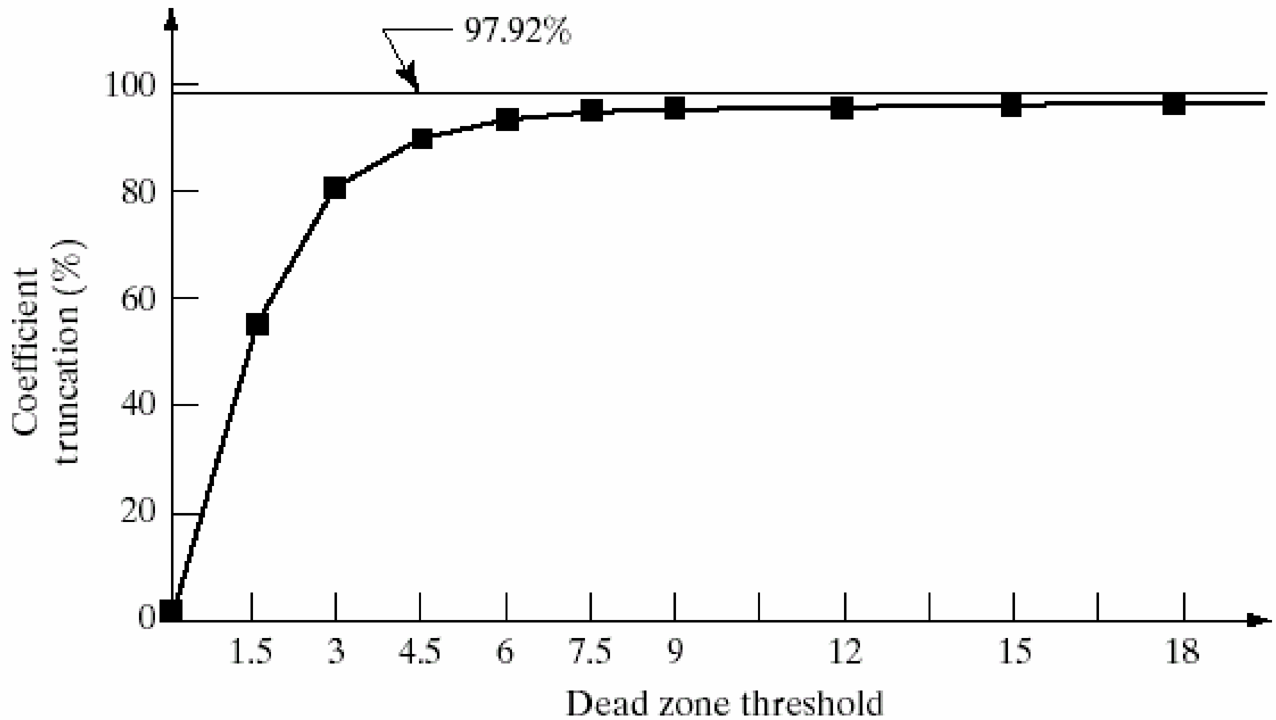


FIGURE 8.43 The impact of dead zone interval selection on wavelet coding.

8.6 Image Compression Standard –Binary image compression

- CCITT Group 3 and 4 standards for binary image compression, originally design for FAX.
- Group 3 applies a non-adaptive, 1-D run-length coding technique.
- Group 4 a streamlined version of group 3, in which 2-D coding is allowed.
- Group 3 and 4 are non-adaptive and sometimes results in data expansion (i.e., with half-tone images).
- CCITT joint with ISO propose JBIG, an adaptive arithmetic compression.

8.6.1 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 is less than 63, a terminating code (Table 8.14) is used.
- If the run length is larger than 63, the largest possible make-up code (Table 8.15) is used in conjunction with the terminating code that represent the difference between the makeup code and actual run-length.

8.6.1 Binary Image Compression Standard

TABLE 8.14
CCITT terminating codes.

Run Length	White Code Word	Black Code Word	Run Length	White Code Word	Black Code Word
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101100
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111

8.6.1 Binary Image Compression Standard

Table 8.14 (Cont’)

16	101010	0000010111	48	00001011	000001100100
17	101011	0000011000	49	01010010	000001100101
18	0100111	0000001000	50	01010011	000001010010
19	0001100	00001100111	51	01010100	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	00000110111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000
26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	0011000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111

8.6.1 Binary Image Compression Standard

Run Length	White Code Word	Black Code Word	Run Length	White Code Word	Black Code Word
64	11011	0000001111	960	011010100	0000001110011
128	10010	000011001000	1024	011010101	0000001110100
192	010111	000011001001	1088	011010110	0000001110101
256	0110111	000001011011	1152	011010111	0000001110110
320	00110110	000000110011	1216	011011000	0000001110111
384	00110111	000000110100	1280	011011001	0000001010010
448	01100100	000000110101	1344	011011010	0000001010011
512	01100101	0000001101100	1408	011011011	0000001010100
576	01101000	0000001101101	1472	010011000	0000001010101
640	01100111	0000001001010	1536	010011001	0000001011010
704	011001100	0000001001011	1600	010011010	0000001011011
768	011001101	0000001001100	1664	011000	0000001100100
832	011010010	0000001001101	1728	010011011	0000001100101
896	011010011	0000001110010			
	Code Word			Code Word	
1792	00000001000		2240	000000010110	
1856	00000001100		2304	000000010111	
1920	00000001101		2368	000000011100	
1984	000000010010		2432	000000011101	
2048	000000010011		2496	000000011110	
2112	000000010100		2560	000000011111	
2176	000000010101				

TABLE 8.15 CCITT makeup codes.

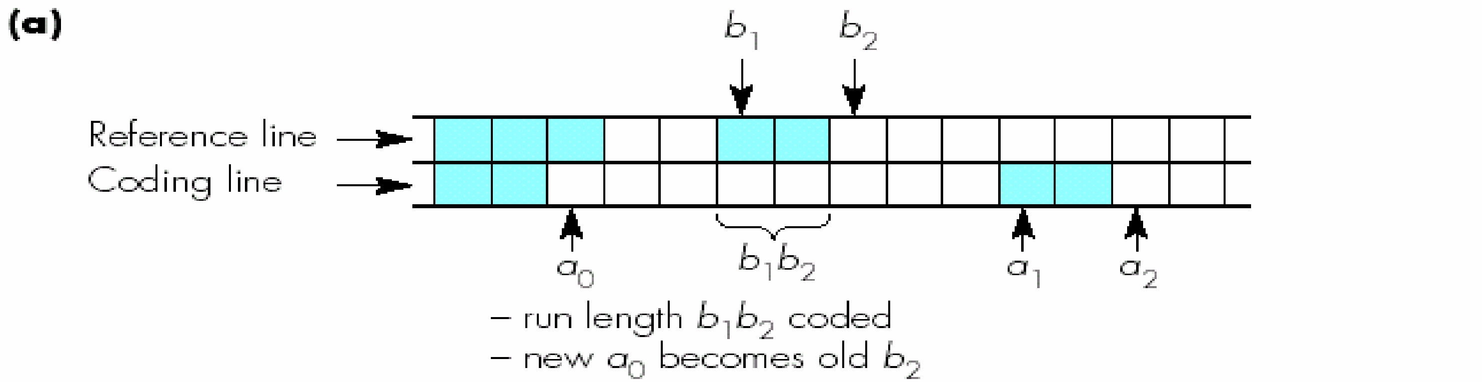
8.6.1 2-D run-length coding-MMR coding (ITU-T6)

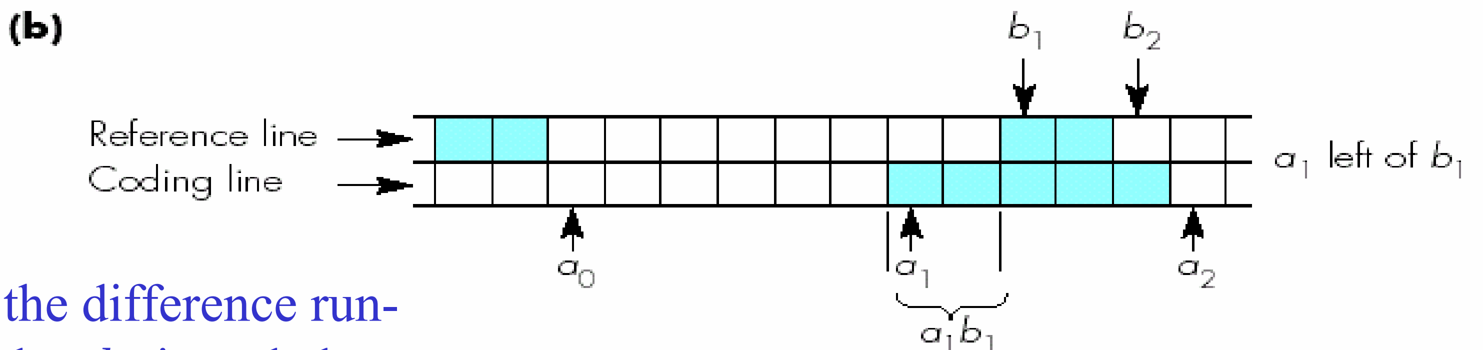
- T6 coding scheme: ***Modified-Modified READ*** (MMR) coding – ***2-D coding***
  - It identifies the black and white run-lengths by comparing adjacent scan lines.
  - READ stands for ***relative element address designate.***
  - A optional in Group 3 but a compulsory in Group 4.

8.6.1 2-D run-length coding- MMR coding (ITU-T6)

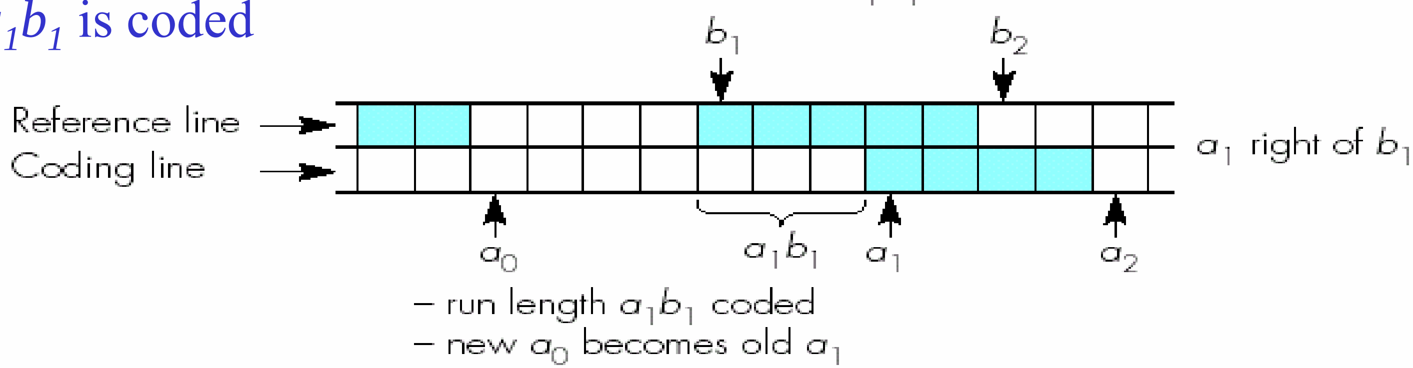
- MMR coding
  - The run-lengths associated with a line is identified by comparing the line contents, known as ***coding line*** (***CL***), relative to the immediately preceding line, known as ***reference line*** (***RL***)
  - The first reference line is all white line
  - Three different referring modes
  - ***Pass mode***
    - The run-length of the reference line is to the left of the next run-length of coding line.
  - ***Vertical Mode***
    - The run-length of the reference line overlaps the next run-length in the coding line by a maximum of plus or minus 3 pixels.
  - ***Horizontal mode***
    - The run-length of the reference line overlaps the next run-length in the coding line by more than plus or minus 3 pixels.

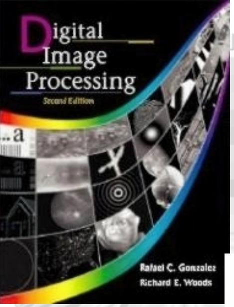
8.6.1 2-D run-length coding-MMR coding (ITU-T6) (a) pass mode; (b) horizonatal mode



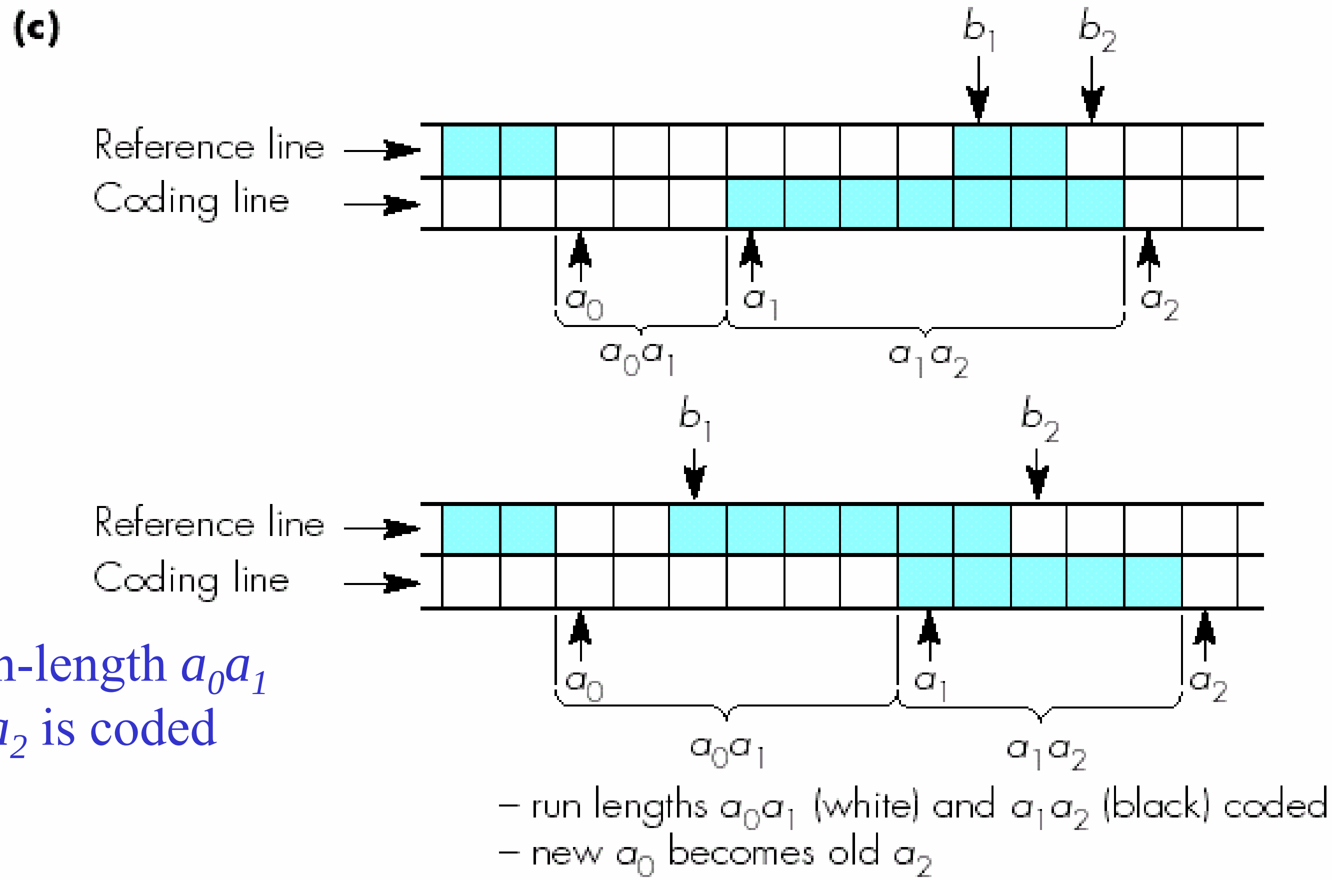


Just the difference run-length a_1b_1 is coded





8.6.1 2-D run-length coding-MMR coding (ITU-T6); (c) vertical mode



The run-length a_0a_1 and a_1a_2 is coded

Note: a_0 is the first pel of a new codeword and can be black or white

a_1 is the first pel to the right of a_0 with a different color

b_1 is the first pel on the reference line to the right of a_0 with a different color

b_2 is the first pel on the reference line to the right of b_1 with a different color

8.6.1 2-D run-length coding-MMR coding (ITU-T6)

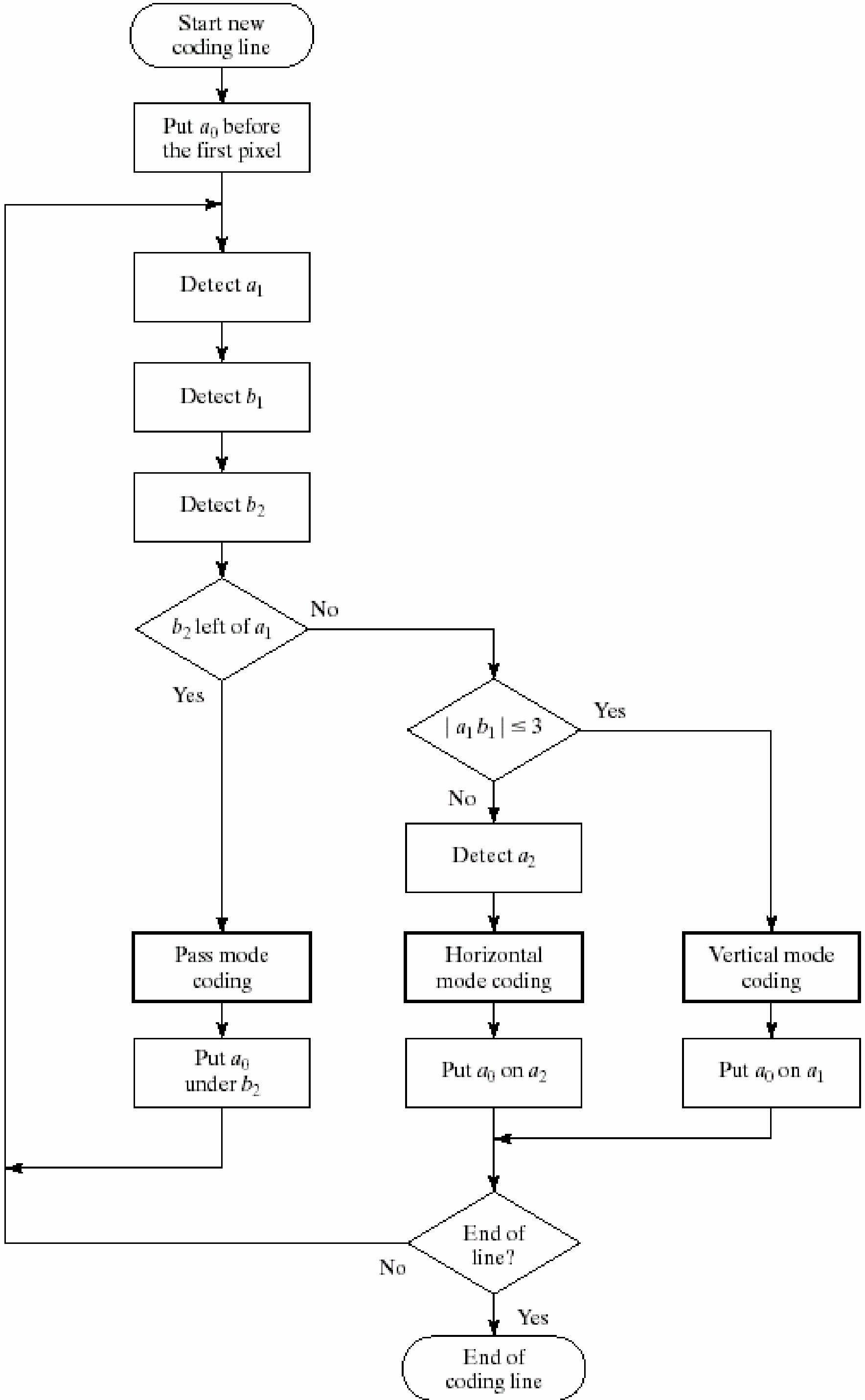


FIGURE 8.44 CCITT 2-D coding procedure. The notation $|a_1b_1|$ denotes the absolute value of the distance between changing elements a_1 and b_1.

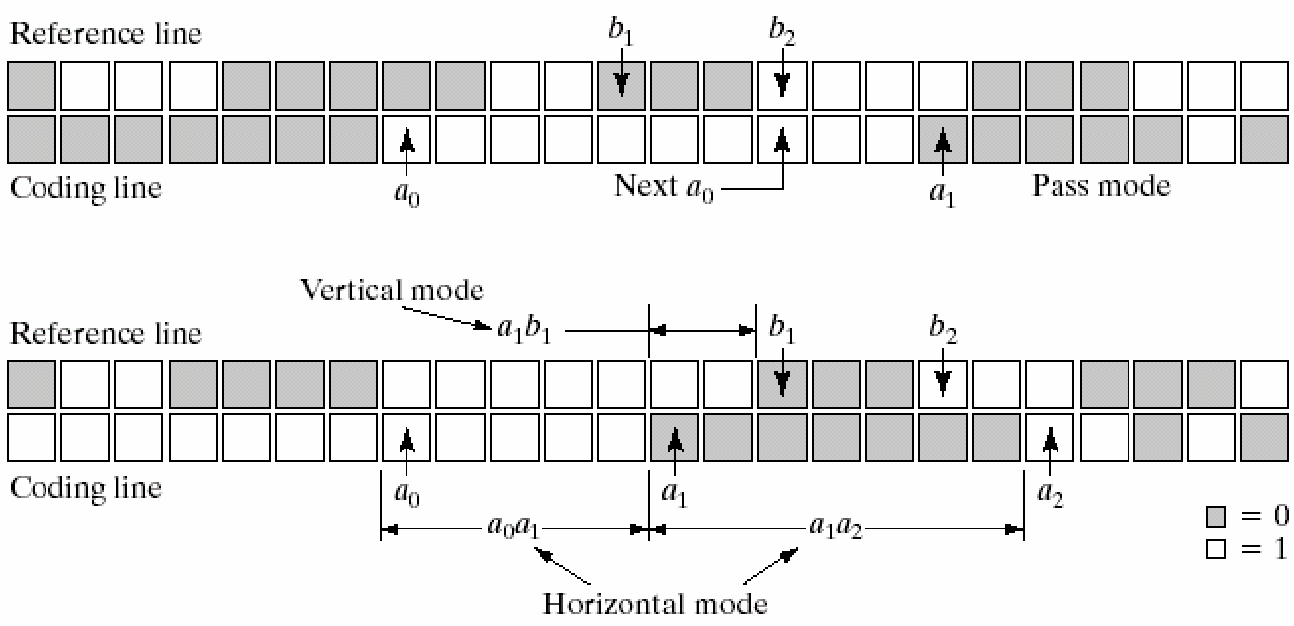
8.6.1 2-D run-length coding-MMR coding (ITU-T6)

- *Two-dimensional Code Table*
  - Additional codewords are used to indicates to which mode the following codewords relate, i.e., *011* indicates the horizontal mode.
  - ***Extension mode***
    - A unique codeword that aborts the encoding operation prematurely before the end of the page.
    - Allow a portion of a page to be sent in its uncompressed form or possibly with a different coding scheme.
    - For example *000001111*code is used to initiate an uncompressed mode of transmission

8.6.1 2-D run-length coding-MMR coding (ITU-T6)

Modes	Run-length to be encoded	abbreviation	codeword
Pass	b_1b_2	P	$0001+b_1b_2$
Horizontal	$a_0\,a_1,\,a_1\,a_2$	H	$001+a_0\,a_1+a_1\,a_2$
Vertical	$a_1b_1=0$	V(0)	1
	$a_1b_1=-1$	Vr(1)	011
	$a_1b_1=-2$	Vr(2)	000011
	$a_1b_1=-3$	Vr(3)	0000011
	$a_1b_1=1$	Vl(1)	010
	$a_1b_1=2$	Vl(2)	000010
	$a_1b_1=3$	Vl(3)	0000010
Extension			*0000001XXXX*

8.6.1 2-D run-length coding-MMR coding (ITU-T6)



a
b

FIGURE 8.45 CCITT (a) pass mode and (b) horizontal and vertical mode coding parameters.

8.6.1 2-D run-length coding-MMR coding (ITU-T6)

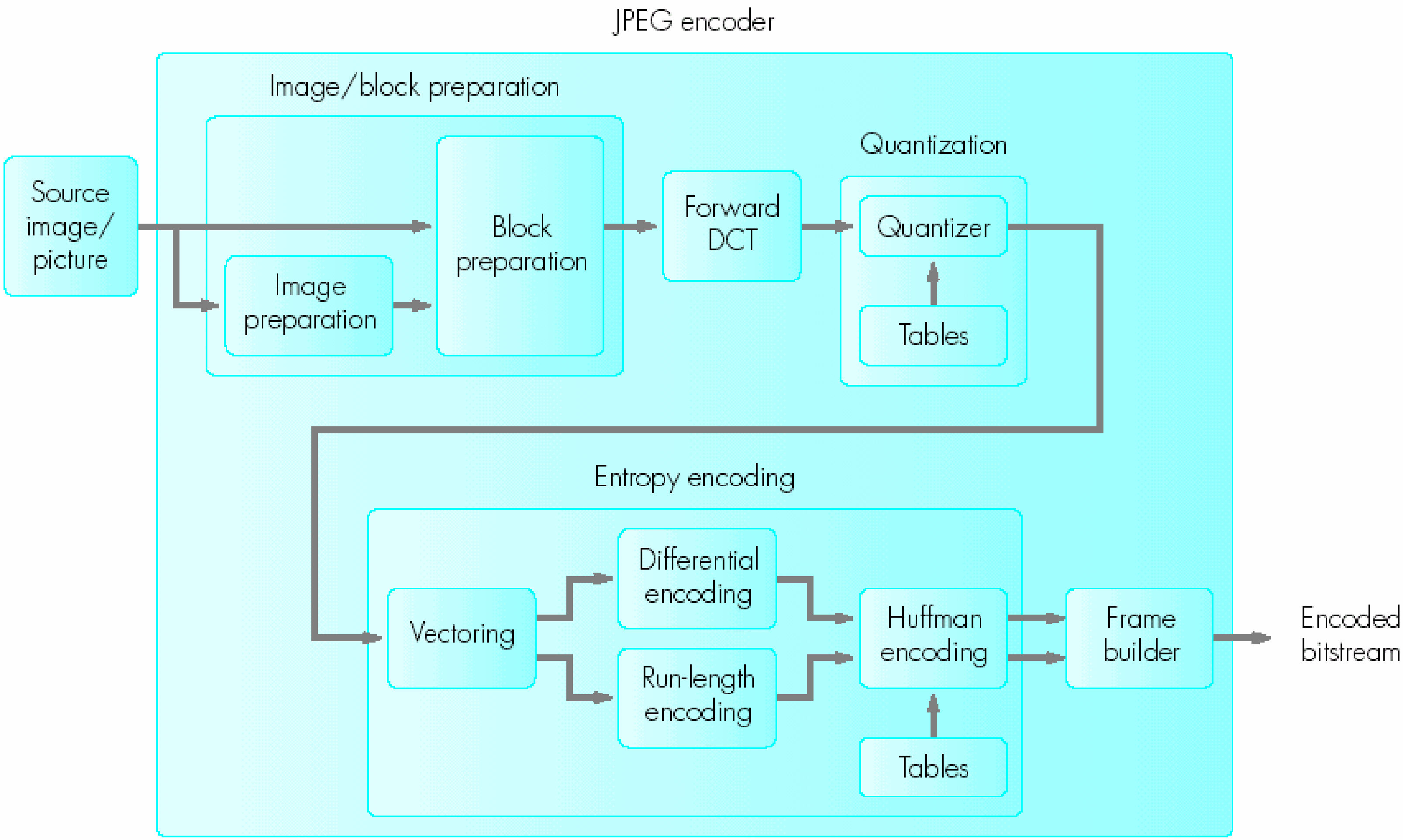
TABLE 8.16
CCITT two-dimensional code table.

Mode	Code Word
Pass	0001
Horizontal	$001 + M(a_0a_1) + M(a_1a_2)$
Vertical	
a_1 below b_1	1
a_1 one to the right of b_1	011
a_1 two to the right of b_1	000011
a_1 three to the right of b_1	0000011
a_1 one to the left of b_1	010
a_1 two to the left of b_1	000010
a_1 three to the left of b_1	0000010
Extension	0000001xxx

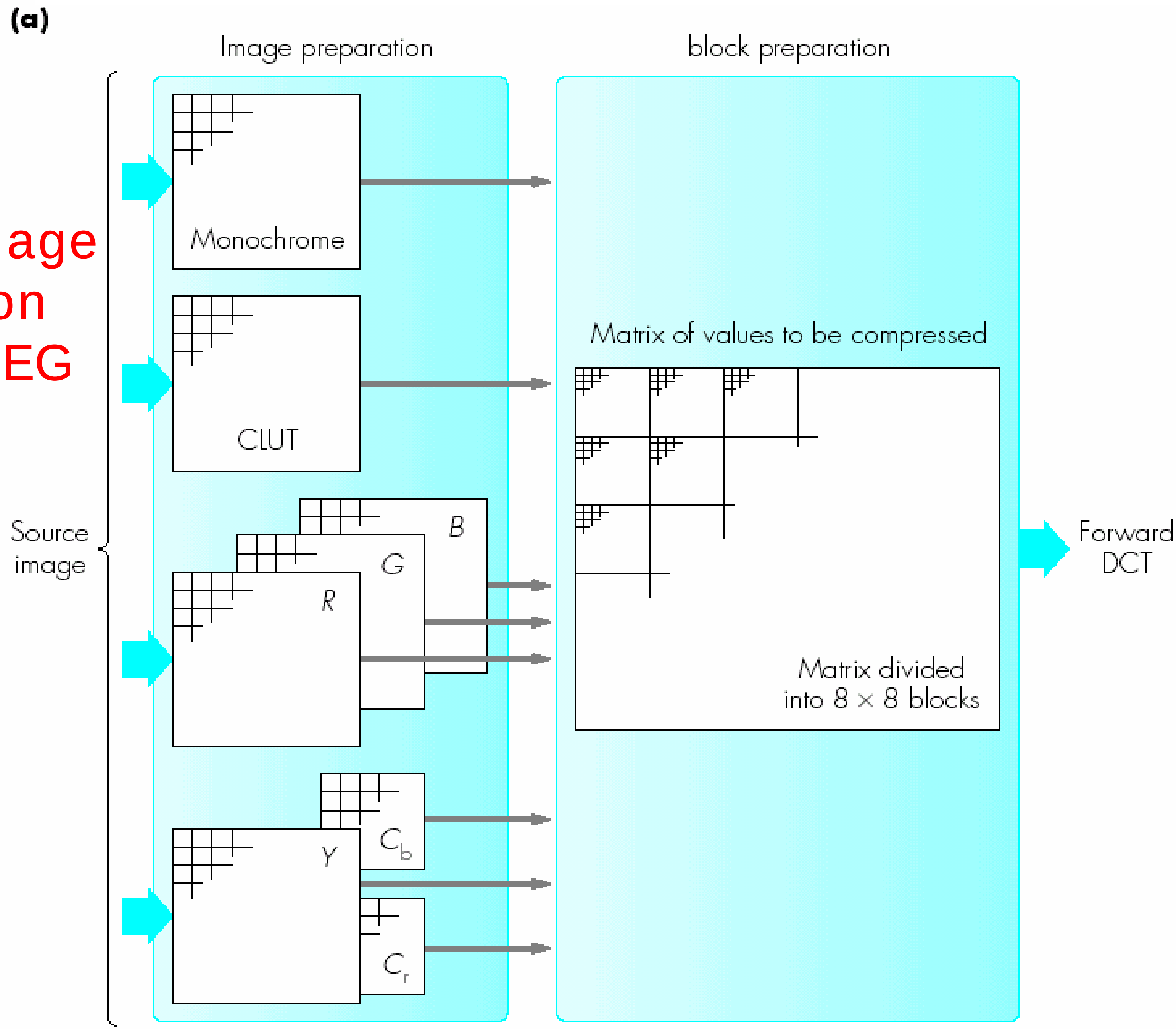
8.6.2 Continuous-Tone Still Image Compression Standard-JPEG

- Joint Photograph Expert Group (JPEG) worked on behalf of ISO, the ITU, and the IEC to define the international standard *JPEG* also known as *IS 10918.*
- *Baseline Mode* or *Lossy Sequential Mode*
  - *Image/block preparation*
  - *Forward DCT*
  - *Quantization*
  - *Entropy encoding*
  - *Frame building*

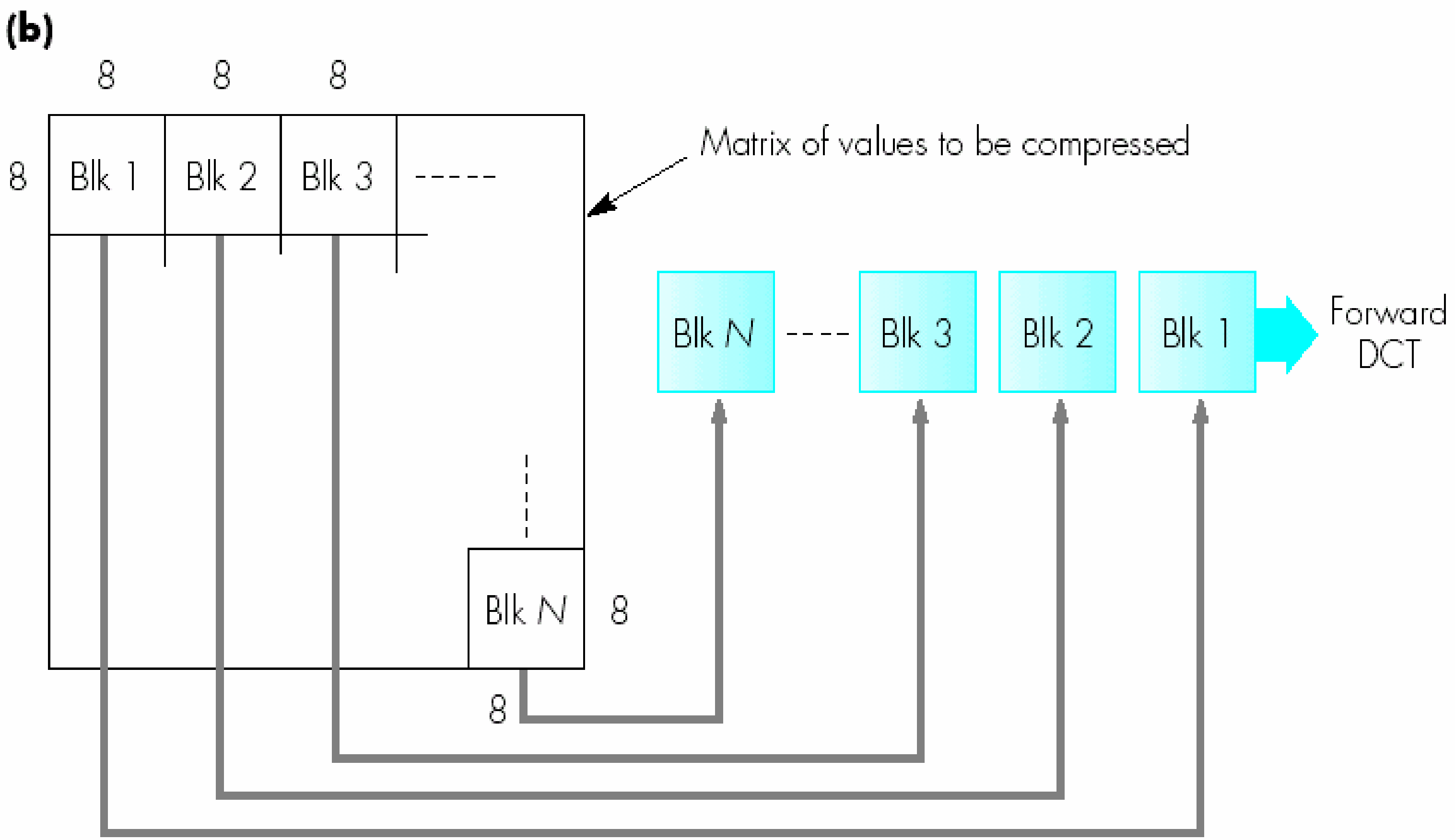
8.6.2 Still Image Compression Standard-JPEG



8.6.2 Still Image Compression Standard-JPEG



8.6.2 Still Image Compression Standard-JPEG



8.6.2 Still Image Compression Standard-JPEG

- All luminance/chrominance values are first subtracted by 128.
- The input 2D matrix represented by p*[x,y]*
- The transformed matrix F*[i, j]* are

$$F[i,j]=\frac{1}{4}C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7}P[x,y]\cos\frac{(2x+1)i\pi}{16}\cos\frac{(2y+1)j\pi}{16}$$

- Where $C(i)$ and $C(j)=1/2^{1/2}$ for $i, j=0$,

  $=0$ for other i *and* j

8.6.2 Still Image Compression Standard-JPEG

$$
\begin{bmatrix}
52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\
63 & 59 & 66 & 90 & & & & \\
62 & 59 & 68 & & & & & \\
63 & 58 & & & & & & \\
67 & 61 & & & & & & \\
79 & & & & & & & \\
85 & & & & & & & \\
87 & & & & & & &
\end{bmatrix}
\xrightarrow{-128}
\begin{bmatrix}
-76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\
-65 & -69 & -62 & -38 & & & & \\
-66 & -69 & -60 & & & & & \\
-65 & -70 & & & & & & \\
-61 & -67 & & & & & & \\
-49 & & & & & & & \\
-43 & & & & & & & \\
-41 & & & & & & &
\end{bmatrix}
$$

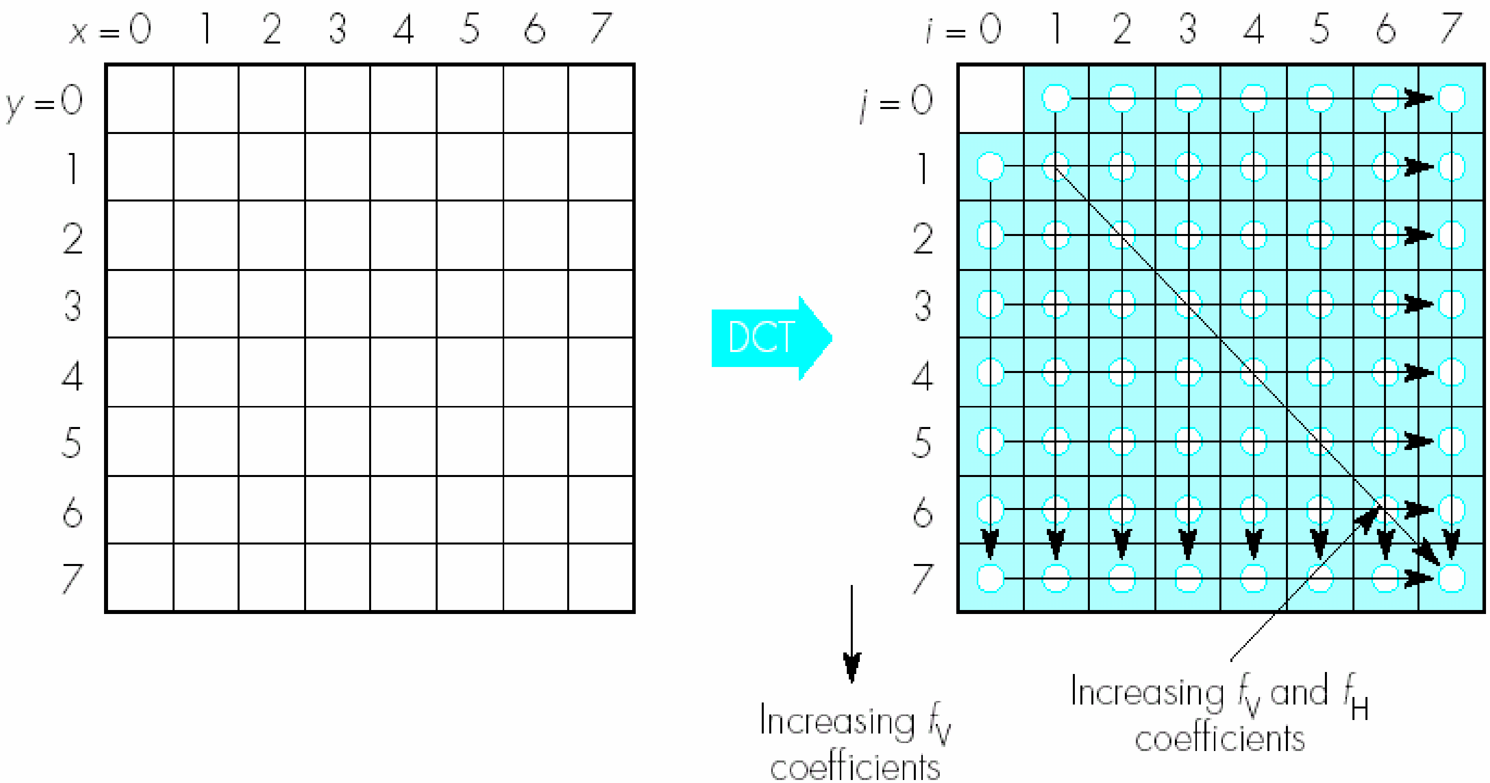
$\Downarrow$ DCT

$$
\begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\
1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\xLeftarrow{\hat{T}(u,v) = \mathrm{round}\left[\frac{T(u,v)}{z(u,v)}\right]}
\begin{bmatrix}
-415 & -29 & -62 & 25 & 55 & -20 & -1 & 3 \\
7 & -21 & -62 & 9 & 11 & -7 & -6 & 6 \\
-46 & 8 & 77 & -25 & & & & \\
-50 & 13 & 35 & & & & & \\
11 & -8 & -13 & & & & & \\
-10 & 1 & 3 & & & & & \\
-4 & -1 & & & & & & \\
-1 & -1 & & & & & &
\end{bmatrix}
$$

8.6.2 Still Image Compression Standard-JPEG

- 64 values of $p[x, y]$ contribute to each entry in $F[i, j]$
- For $i=j=0$, $F[0, 0]$ is a function of summation of all $p[x, y]$, the mean of 64 values, *DC coefficient*
- Other coefficients $F[i, j]$, where $i=1$ *to* 7, $j=1$ *to* 7, are AC coefficients.

8.6.2 Still Image Compression Standard-JPEG



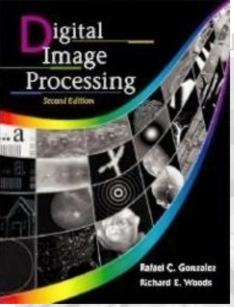
$P[x, y]$ = 8 × 8 matrix of pixel values

$F[i, j]$ = 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



8.6.2 Still Image Compression Standard-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-Quantization

- Real DCT Coefficients (real values) need to be quantized as an integer value before encoding.
- The quantization levels for DC coefficients and each AC coefficient are different due to the sensitivity of eye varies with spatial frequency.

JPEG-example

- Assuming a quantization threshold value of 16, derive the resulting quantization error for each of the following DCT coefficients:

  127, 72, 64, 56, –56, –64, –72, –128

- *Answer:*

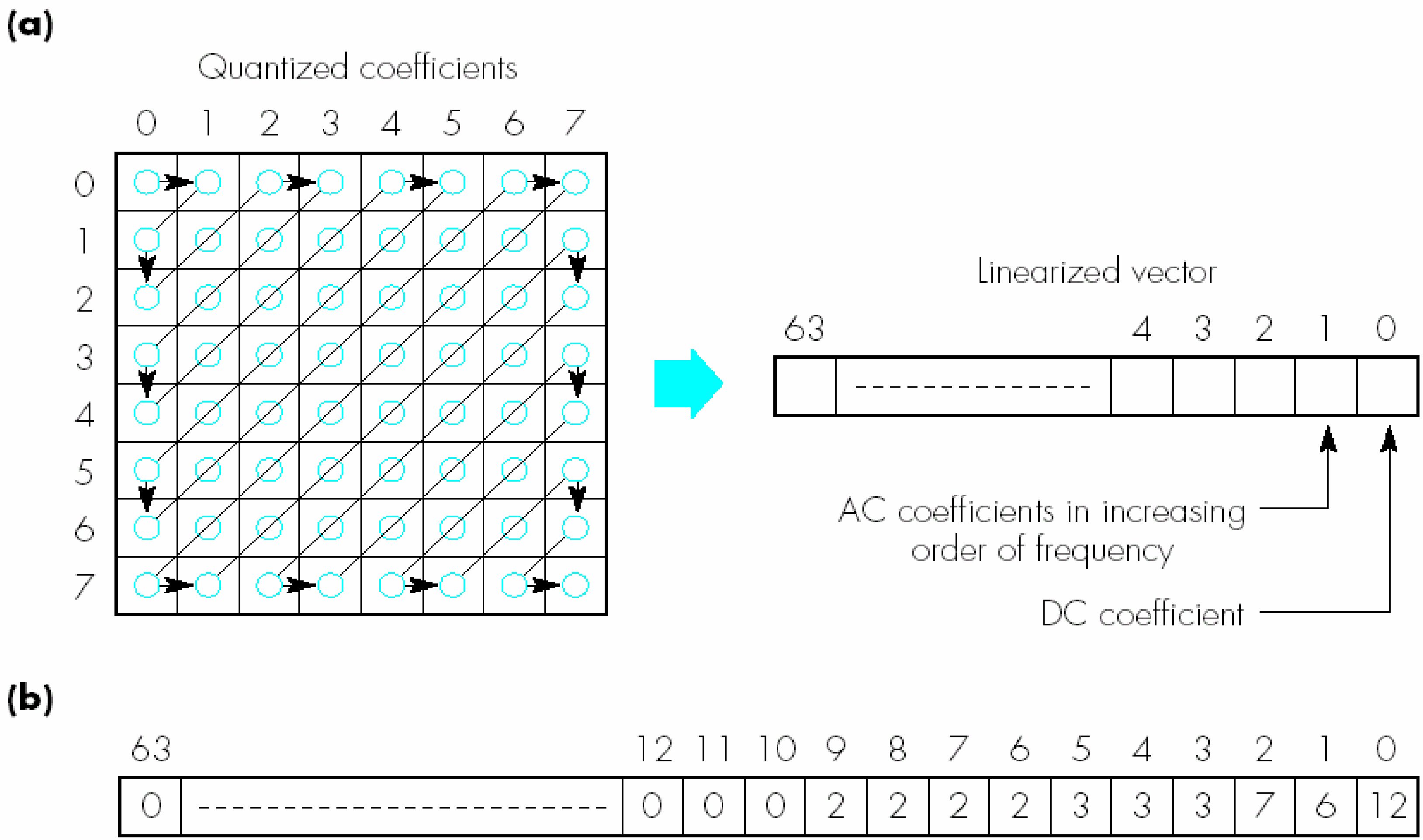
Coefficient	*Quantized value*	*Rounded value*	*Dequantized value.*	*Error*
127	127/16 = 7.9375	8	8×16 = 128	-1
72	4.5	5	80	+8
64	4	4	64	0
56	3.5	4	64	+8
-56	-3.5	-4	- 64	-8
-64	-4	- 4	- 64	0
-72	-4.5	-5	- 80	-8
-128	127/16 = 7.9375 8	8 。人	–128	0

- As we can deduce from these figures, the maximum quantization error is plus or minus 50% of the threshold value used

JPEG-Entropy Encoding

- 1-D Vectoring
- Differential encoding
  - For DC coefficients
- Run-Length Encoding
  - For AC coefficients
- Huffman Coding
- Vectoring
  - Represent the values in 2D coefficient matrix by a 1D vector
  - Zig-zag scanning, DC coefficient and lower-frequency AC coefficients are scanned first.

JPEG-Entropy Encoding



JPEG-differential coding

- The DC Coefficients vary only slowly from one block to the next.
- Only the difference in magnitude of the DC coefficient in a quantized block relative to the value in the preceding block is encoded.
- The difference values are then encoded in the form of *(SSS, value)*. *SSS* indicates the number of bits needed to encode the value and the *value* is the encoded bits.

JPEG-DC coefficient coding

- The sequence of ***DC coefficients*** in consecutive quantized blocks, one per block, was:

  12, 13, 11, 11, 10

- The corresponding difference values would be:

  12, 1, -2, 0, -1

- Determine the encoded version of the difference values which relate to the encoded DC coefficients from consecutive DCT blocks:

Answer: (From the table (a) in the next page)

Value	*SSS*	*Encoded Value*
12	4	1100
1	1	1
-2	2	01
0	0	
-1	1	0

JPEG-AC coefficients Encoding

- The AC coefficients are encoded in the form of string of pairs of values as *(skip, value)* where *skip* is the number of the zeros in the run and *value* is the next non-zero coefficient.
- The *value* field is encoded as *SSS/value*
- The 63 values in the vector will be encoded as

(0,6)(0,7)(0,3)(0,3)(0,3)(0,2)(0,2)(0,2)(0,2)(0,0)

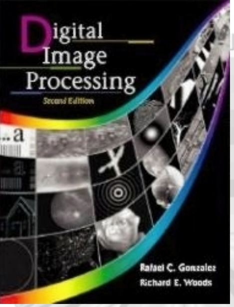
JPEG-AC coefficients Encoding

- Derive the binary form of the following run-length encoded AC coefficients:

  (0,6)(0,7)(3,3)(0,-1)(0,0)

- *Answer:*

AC coefficients	*Skip*	*SSS/Value*	
0,6	0	3	110
0,7	0	3	111
3,3	3	2	11
0,-1	0	1	0
0,0	0	0	



JPEG-DC coefficients Encoding

(a)

Difference value	Number of bits needed (SSS)	Encoded value
0	0	
−1, 1	1	1 = 1 , −1 = 0
−3, −2, 2, 3	2	2 = 10 , −2 = 01 3 = 11 , −3 = 00
−7..−4, 4..7	3	4 = 100 , −4 = 011 5 = 101 , −5 = 010 6 = 110 , −6 = 001 7 = 111 , −7 = 000
−15...−8, 8...15	4	8 = 1000 , −8 = 0111 ⋮ 12=1100 , -12=0011

(b)

Number of bits needed (SSS)	Huffman codeword
0	010
1	011
2	100
3	00
4	101
5	110
6	1110
7	11110
⋮	
11	111111110

DC coefficient code table

JPEG-DC coefficients Encoding

- Determine the Huffman-encoded version of the following difference values which relate to the encoded DCT coefficients from consecutive DCT blocks: 12, 1, -2, 0, -1
- Use the default Huffman codewords defined in DC code-table(b).

Answer:

Value	*SSS*	*Huffman-encoded SSS (Table (b))*	*Encoded value (Table (a))*	*Encoded bitstream*
12	4	101	1100	1011100
1	1	011	1	0111
-2	2	100	01	10001
0	0	010		010
-1	1	011	0	0110

Encoding bitstream : *1011100* for DC coefficient in block1
0111 for DC coefficient in block 2
10001 for DC coefficient in block 2

JPEG-DC coefficients decoding

- The decoder uses the same set of codewords to determine the *SSS* field from the received bitstream (e.g. 1011100……..) by searching the bitstream bit-by-bit – starting from the leftmost bit – until it reaches a valid codeword (101).
- The number of bits in the corresponding *SSS* value is then read from the DC coefficient coding table and this is used to determine the number of following bits (e.g. 4 bits) in the bitstream that represent the related *value*.
- Decoding the following bits (1100) using the DC code table to find the real *value* (=12).

8.6.2 Still Image Compression Standard-JPEG

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
$-7, \ldots, -4, 4, \ldots, 7$	3	3
$-15, \ldots, -8, 8, \ldots, 15$	4	4
$-31, \ldots, -16, 16, \ldots, 31$	5	5
$-63, \ldots, -32, 32, \ldots, 63$	6	6
$-127, \ldots, -64, 64, \ldots, 127$	7	7
$-255, \ldots, -128, 128, \ldots, 255$	8	8
$-511, \ldots, -256, 256, \ldots, 511$	9	9
$-1023, \ldots, -512, 512, \ldots, 1023$	A	A
$-2047, \ldots, -1024, 1024, \ldots, 2047$	B	B
$-4095, \ldots, -2048, 2048, \ldots, 4095$	C	C
$-8191, \ldots, -4096, 4096, \ldots, 8191$	D	D
$-16383, \ldots, -8192, 8192, \ldots, 16383$	E	E
$-32767, \ldots, -16384, 16384, \ldots, 32767$	F	N/A

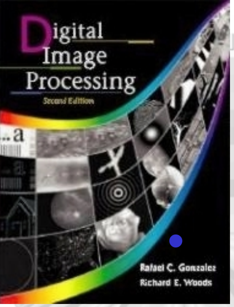
8.6.2 Still Image Compression Standard-JPEG

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

JPEG-AC coefficients Encoding

- For each run-length encoded *AC coefficients* in the block, the bits that make up the skip and *SSS* fields are treated as a single (Composite) symbol and encoded using a default table of Huffman codeword or a new table sent with the encoded bit stream.
- To enable the decoder to discriminate between the *skip* and *SSS* field, each combination of the skip and SSS field is encoded separately and the composite symbol is replaced by equivalent Huffman codeword.



JPEG-AC coefficients Encoding

- Derive the composite binary symbols for the following set of run-length encoded AC coefficients:

  (0,6)(0,7)(3,3)(0, –1)(0,0)
- Assuming the *skip* and *SSS* fields are both encoded as a composite symbol, use the Huffman codewords shown in Table 8.19 to derive the Huffman-encoded bitstream for this set of symbols.
- *Answer:*
- The *skip* and *SSS* fields for this set of AC coefficients were derived earlier
- ***AC coeff.*** ***Composite symbol*** ***Huffman codeword*** ***Run-length value***

AC coeff.	***Composite symbol skip***	***Composite symbol SSS***	***Huffman codeword***	***Run-length value***
0, 6	0	3	100	6 = 110
0, 7	0	3	100	7 = 111
3, 3	3	2	111110111	3 = 11
0, –1	0	1	00	-1 = 0
0, 0	0	0	1010	

The Huffman-encoded bit-stream is then derived by adding the runlength encoded value to each of the Huffman codewords:

100110 100111 *11111011110* 000 *1010*

8.6.2 Still Image Compression Standard- JPEG

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)*.

8.6.2 Still Image Compression Standard -JPEG

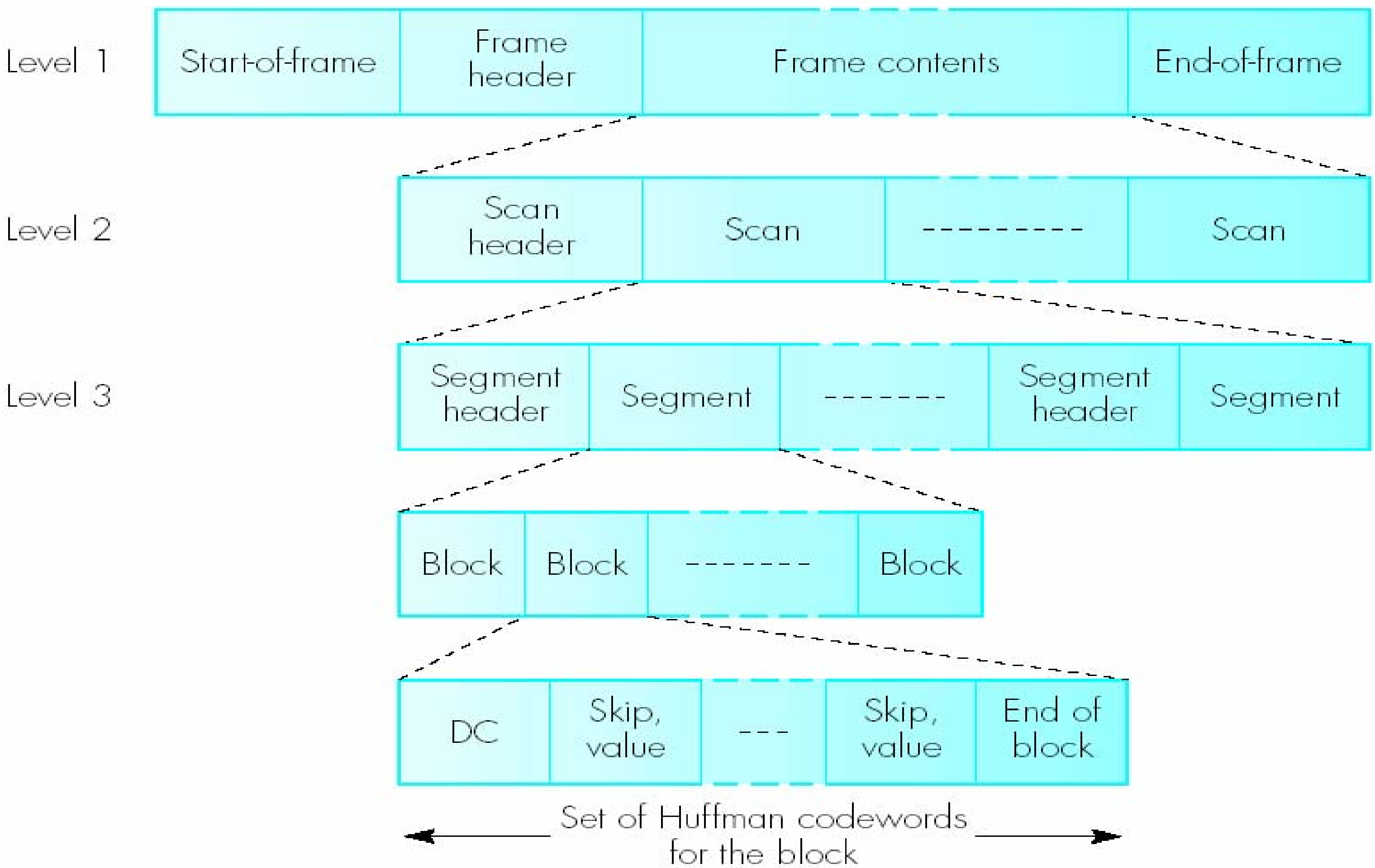
Table 8.19 (Con't)

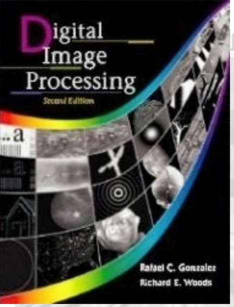
2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	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

8.6.2 Still Image Compression Standard - JPEG output bitstream format.

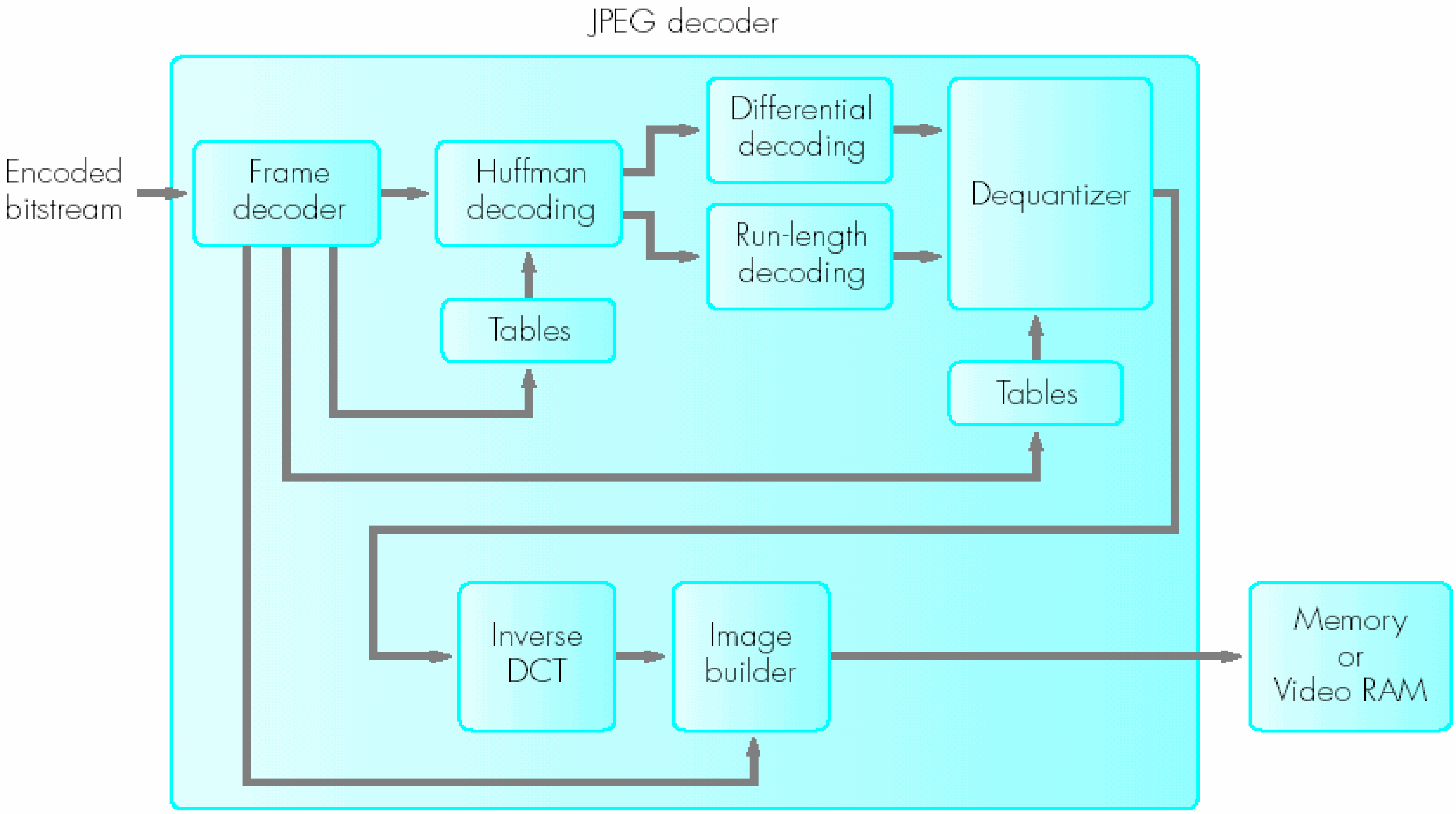
- Encapsulate all the information relating to an encoded image/picture in a frame.
- The structure of a frame is hierarchical
  - *Frame* consists of a number of *scans*
  - *Scan* consists of a number of *segments*
  - *Segment* consists of a number of *blocks*
- Frame header
  - Overall width and height of an image
  - The number and type of component (CLUT, R/G/B, $Y/C_r/C_b$)
  - Digitization format (4:2:2 or 4:2:0) :
- Scan header
  - Identity of the component (R/G/B etc)
  - The number of bits used to digitize each component
  - The quantization table of values
- Each segment can be decoded independently of the other to overcome the possibility of bit error propagation.

8.6.2 Still Image Compression Standard - JPEG output bitstream format.





8.6.2 Still Image Compression Standard –JPEG decoder



8.6.2 Still Image Compression Standard-JPEG2000

- 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.

8.6.2 Still Image Compression Standard-JPEG2000

- 1st step: ***DC level shift***: that shifts the samples of the *Ssiz*-bits unsigned image to be coded by substracting 2^{Ssiz-1}.
- For component images, using the component transform to transform correlated components (R, G, B) to three *uncorrelated components* Y_0, Y_1 and Y_2.
- The histogram of Y_1 and Y_2 are highly peaked around zero.
- ***Tiling process*** creates *tile component* that can be extracted and reconstructed independently.
- ***Tiles*** are rectangular arrays of pixels that contain the same relative portion of all components.
- 1-D FWT of the rows and columns of each tile component is computed.

8.6.2 Still Image Compression Standard-JPEG2000

- For error-free compression, the transform is based on *5-3 coefficient scaling-wavelet vector*.
- For lossy compression, the transform is based on *9-7 coefficient scaling-wavelet vector.*
- ***X*** is the tile component being transform, ***Y*** is the resulting transform.
- The even-indexed values of ***Y*** are equivalent to the FWT lowpass filtered output.
- The odd-indexed values of Y are equivalent to the FWT highpass filtered output.
- Repeat the transformation N_L times.

8.6.2 Still Image Compression Standard- JPEG2000

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

TABLE 8.20 Impulse responses of the low and highpass analysis filters for an irreversible 9-7 wavelet transform.

8.6.2 Still Image Compression Standard-JPEG2000

- The complementary lifting-based approach involves six sequential "lifting" and "scaling" operations:

$$Y(2n+1)=X(2n+1)+\alpha[X(2n)+X(2n+2)] \quad i_0-3\leq 2n+1<i_1+3$$

$$Y(2n)=X(2n)+\beta[Y(2n-1)+Y(2n+1)] \quad i_0-2\leq 2n<i_1+2$$

$$Y(2n+1)=Y(2n+1)+\gamma[Y(2n)+Y(2n+2)] \quad i_0-1\leq 2n+1<i_1+1$$

$$Y(2n)=Y(2n)+\delta[Y(2n-1)+Y(2n+1)] \quad i_0\leq 2n<i_1$$

$$Y(2n+1)=-K\,Y(2n+1) \quad i_0\leq 2n+1<i_1$$

$$Y(2n)=Y(2n)/K \quad i_0\leq 2n<i_1$$

- X is the tile component, Y is the resulting transform.
- i_0 and i_1 define the position of the tile component within a component.
- i_0 and i_1 are the indices of the first sample of the tile component row or column being transformed and the one immediately following the last sample.

8.6.2 Still Image Compression Standard-JPEG2000

- The variable n assumes values based on i_0, i_1 and which of the six operations is being operated.
- IF $n \geq i_1$ or $n < i_0$ X(n) is obtained by symmetrically extending $X(i_0-1)= X(i_0+1)$, $X(i_0-2)= X(i_0+2)$, $X(i_1)= X(i_1-2)$, $X(i_1+1)= X(i_1-3)$
- The even indexed value of Y are equivalent to the FWT lowpass filtered output
- The odd indexed value of Y are equivalent to the FWT highpass filtered output
- The ***lifting parameters***: α, β, γ, δ.
- The ***scaling factor***: K

8.6.2 Still Image Compression Standard-JPEG2000

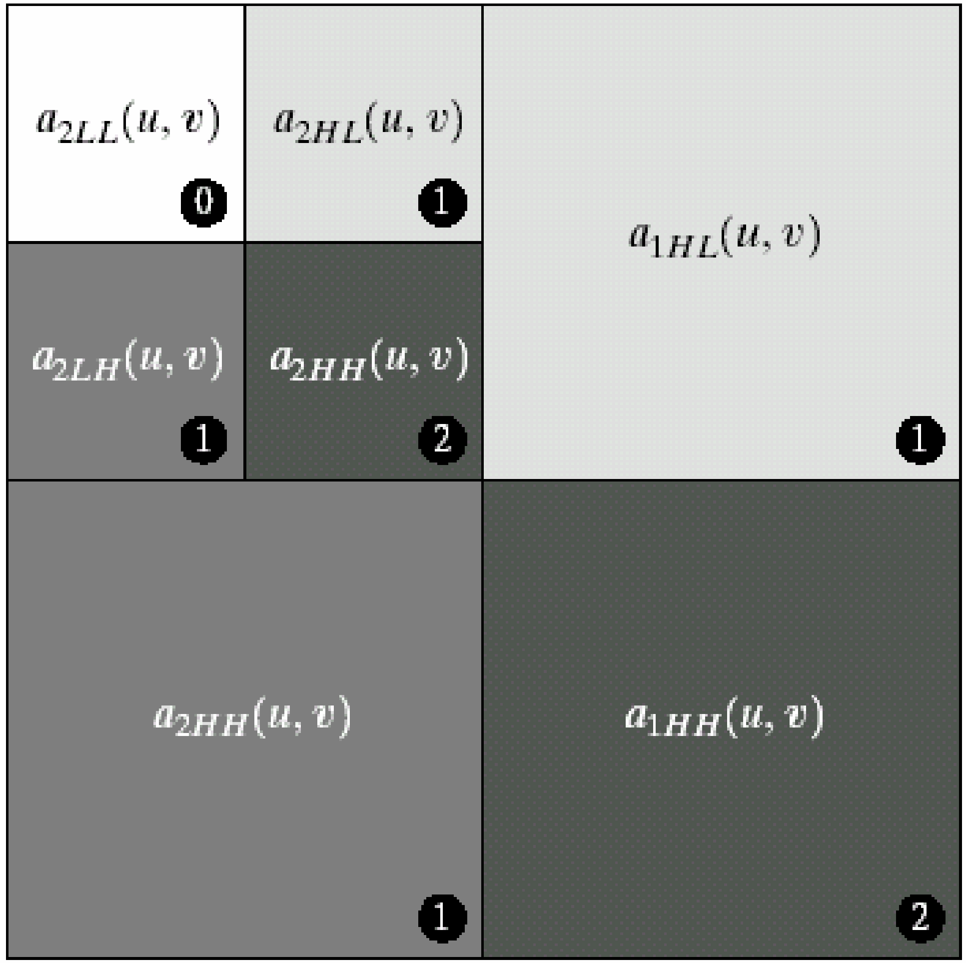


FIGURE 8.46 JPEG 2000 two-scale wavelet transform tile-component coefficient notation and analysis gain.

8.6.2 Still Image Compression Standard-JPEG2000

- To reduce the number of bits needed to represent the transform, coefficient $a_b(u, v)$ of subband b is quantized to $q_b(u, v)$ as $q_b(u, v) = sign[a_b(u, v)] floor[|a_b(u, v)|/\Delta_b]$, where the quantization step $\Delta_b = 2^{R_b - \varepsilon_b}(1 + \mu_b/2^{11})$, and R_b is the nominal dynamic range of subband b.
- R_b is the sum of the number of bits used to represent the original image and *analysis gain* bits for subband b.
- ε_b *and* μ_b are the number of bits allocated to the *exponent* and *mantissa* of the subband's coefficients.
- For error free compression: $R_b = \varepsilon_b$, $\mu_b = 0$ *and* $\Delta_b = 1$

8.6.2 Still Image Compression Standard-JPEG2000

- The final steps of the encoding process are:

1) *Coefficient bit modeling*:
   a) The coefficients of each transformed tile-component's subband are arranged into rectangular blocks called *code block*, which are individually coded a bit plane at a time.
   b) Staring from the most significant bit, each bit plane is processed by three passes: *significant propagation*, *magnitude refinement*, *cleanup*.
2) *Arithmetic coding*
3) *Bit-stream layering*
4) *packetizing*

8.6.2 Still Image Compression Standard-JPEG2000

- Although the encoder have encoded M_b bit planes for a particular subband, the decoder may choose only to decode N_b bit plane.
- This amount to quantizing the code block coefficients using only a step size of $2^{M_b\text{-}N_b}\,\Delta_b$.
- The result coefficients denoted as $\bar{q}_b(u,v)$ are dequantized using

$$R_{q_b}(u,v)=\begin{cases}(\bar{q}_b(u,v)+2^{M_b-N_b(u,v)})\Delta_b & \bar{q}_b(u,v)>0\\(\bar{q}_b(u,v)-2^{M_b-N_b(u,v)})\Delta_b & \bar{q}_b(u,v)>0\\0 & \bar{q}_b(u,v)=0\end{cases}$$

- The dequantized coefficients are then inverse transformed using inverse FWT.

8.7 Video Compression Standards

Video – moving pictures

- motion JPEG
  - JPEG applied to each frame independently to remove spatial redundancy –Considerable
- Temporal redundancy in video
  - 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 sent.

8.7 Video Compression Standards – Frame Types

- Intracoded frames or I frames
  - Coded without reference to other frame
  - Presented in the output stream at regular intervals, the number of frames between two successive I-frames is known as a group of picture (GOP)
- Intercoded frame or predicted frame
  - Predictive frames, or P-frames
    - Coded with reference to one previous frame.
  - Bidirectional frames, or B-frames
    - Coded with reference to two other frames

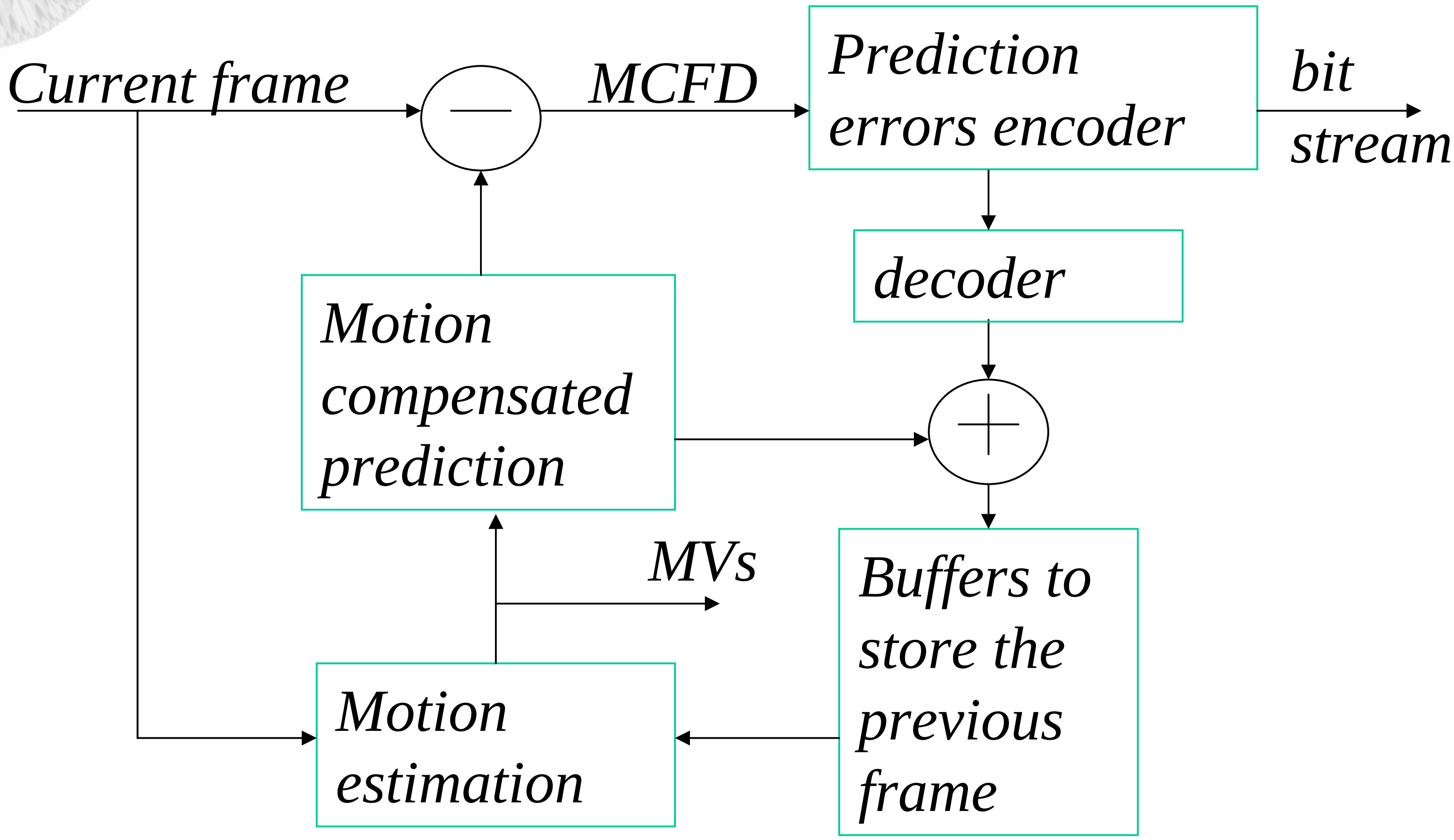
8.7 Video Compression Standards – Frame Types

- The encoded frame sequence
  - IBBPBBPBBIBBP….
- The reorder sequence for encoding
  - IPBBPBBIBBPBB….
- PB frame
  - The two neighboring P-frame and B-frame are encoded as if they are a single frame.
  - Increasing the frame rate without increasing the bit rate.
- D-frame for movie/video-on-demand
  - Inserted at a regular intervals throughout the stream
  - A highly compressed frames, which are ignored during the decoding of the I-frame and P-frame.

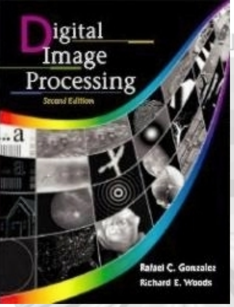
8.7 Video Compression Standards – Frame Types

- P-frame
  - Encoding relative to the previous I-frame or P-frame.
  - *Error propagation* - any error in the P-frame will be propagated to the next P-frame.
  - *The prediction span*, The number of frame between the P frame and it Preceding I-frame or P-frame.
  - For motion pictures, B-frame is needed, for occasional fast moving object.
- B-frames
  - Three frames are involved, the preceding I-frame or P-frame, the current frame, and the succeeding I-frame or P-frame- encoding delay.
  - Reducing the difference in encoding the uncover background.

8.7.1 Motion Estimation and Compensation



MCFD= motion compensated frame difference



8.7.2 Block matching

- ***Assumptions*** :

  1. *The object displacement is constant within a small 2-D block of pels.*

  2. *Different displacement for different block*

  3. *The same displacement for all pels in the corresponding block.*

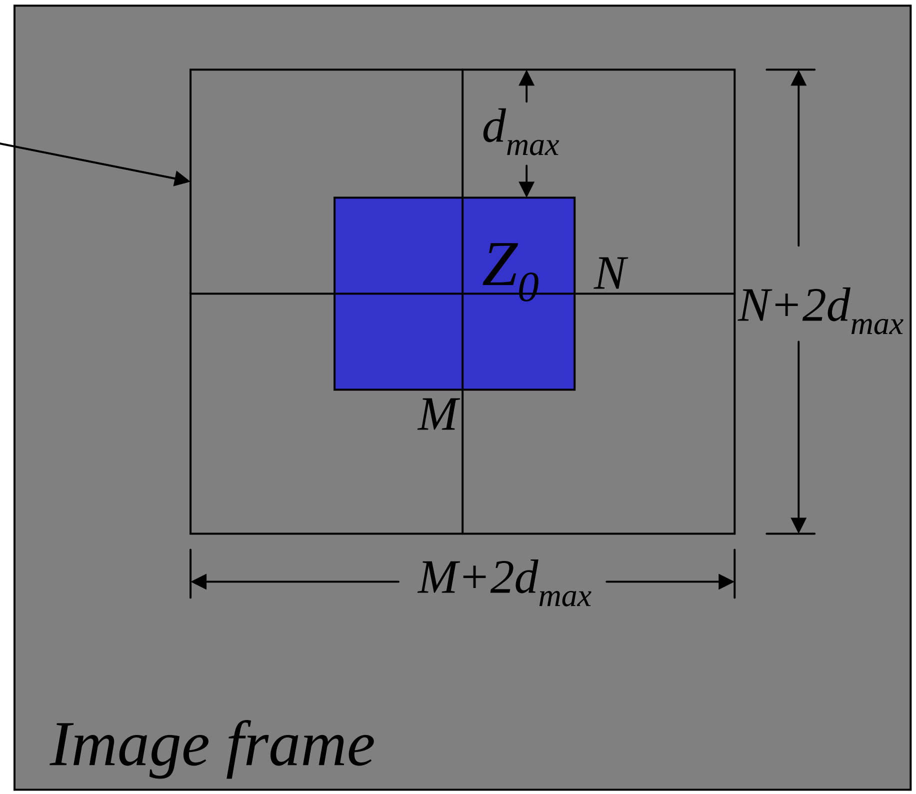
- Displacement D is estimated by choosing an optimal D that minimize the prediction error

$$PE(D) = \sum N\big(b(Z,t) - b(Z - D, t - \tau)\big)$$

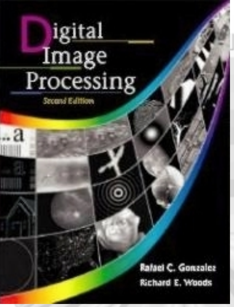
where $N(\cdot)$ is the distance metric.

8.7.2 block matching

- Search area:



Search area= $z_0 \pm \begin{bmatrix} \frac{1}{2}(M-1)+d_{\max} \\ \frac{1}{2}(N-1)+d_{\max} \end{bmatrix}$



8.7.2 Motion Estimation

- Evaluate the prediction error

$$PE(Z_0, i, j) = \frac{1}{MN} \sum_{|m| \le \frac{M}{2}} \sum_{|n| \le \frac{N}{2}} \left[b(Z_{m,n}, t) - b(Z_{m+i,n+i}, t-\tau) \right]^2$$

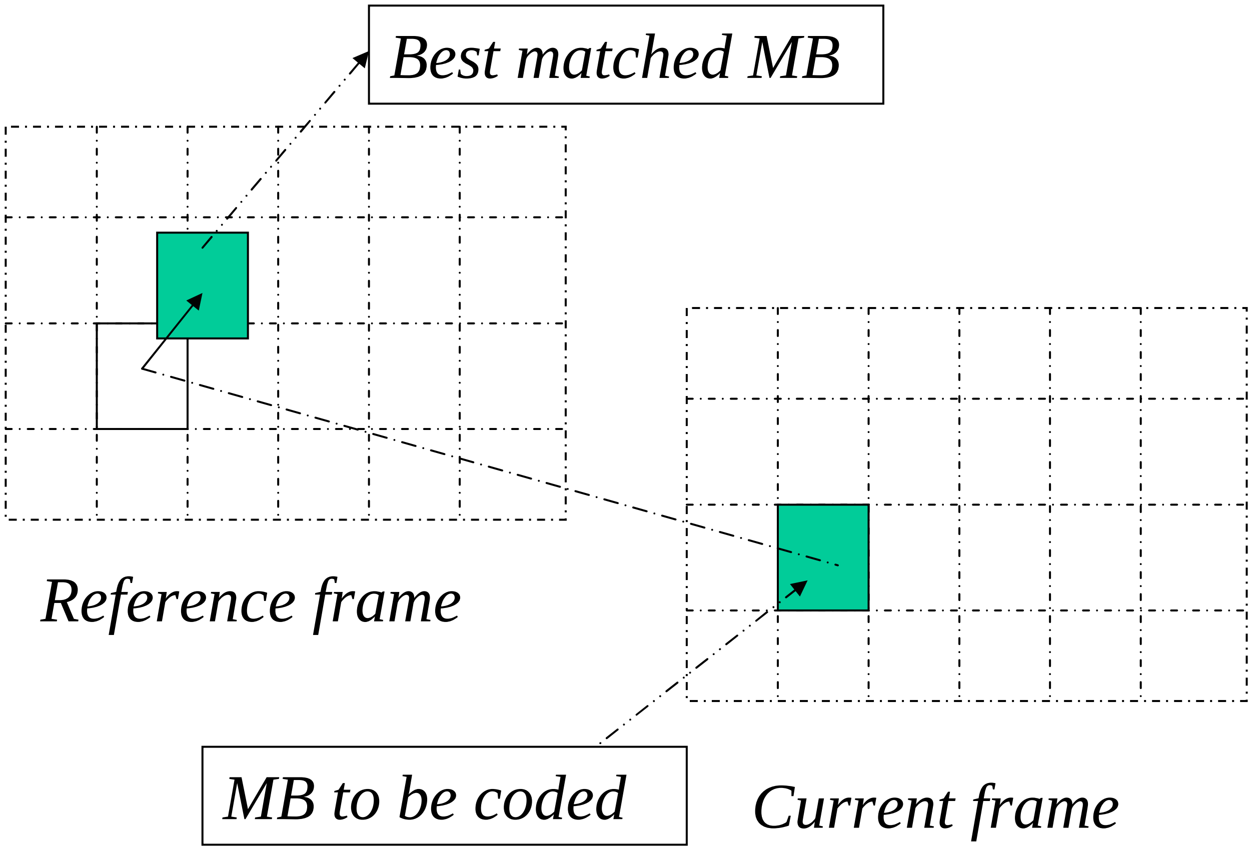
or

$$PE(Z_0, i, j) = \frac{1}{MN} \sum_{|m| \le \frac{M}{2}} \sum_{|n| \le \frac{N}{2}} \left| b(Z_{m,n}, t) - b(Z_{m+i,n+i}, t-\tau) \right|$$

where $\quad -d_{\max} \le i, j \le d_{\max} \quad Z_{m,n} = Z_0 + [m, n]$

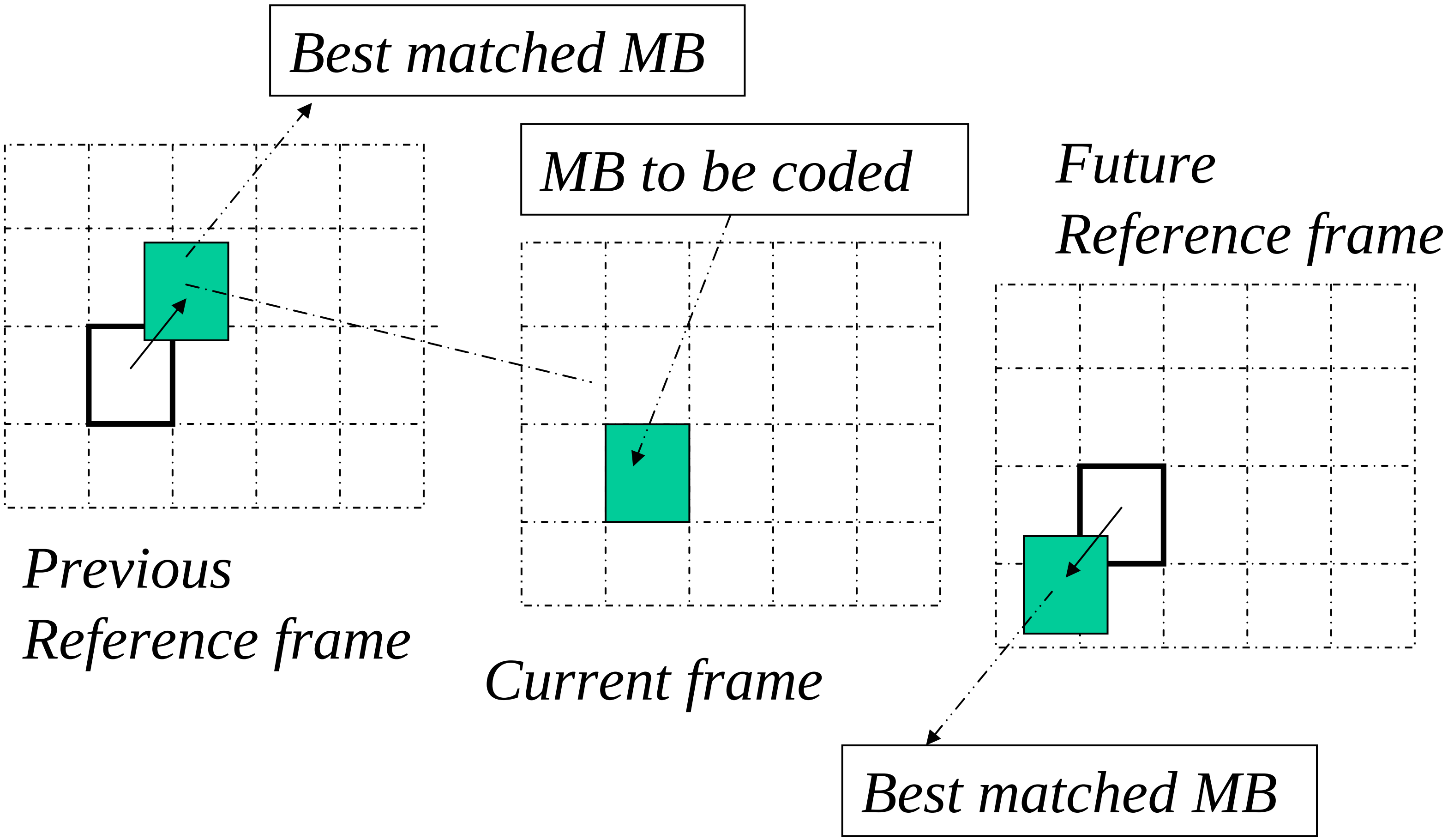
8.7.2 Motion Estimation

Unidirectional prediction



8.7.2 Motion Estimation

Bidirectional prediction



8.7.2 Block matching

- Matching Criterions:
-

$$\begin{cases} 1.\ \text{2-D logarithmic search} \\ 2.\ \text{three-step search} \\ 3.\ \text{modified conjugate direction} \end{cases}$$

- The goal is to require as few shifts as possible and to evaluate PE as few times as possible.
- Basic assumption: $PE(Z_0, i, j)$ increases monotonically as the shift (i, j) moves away from the direction of minimum distortion.
- ***Full search method***

8.7.2 Block matching

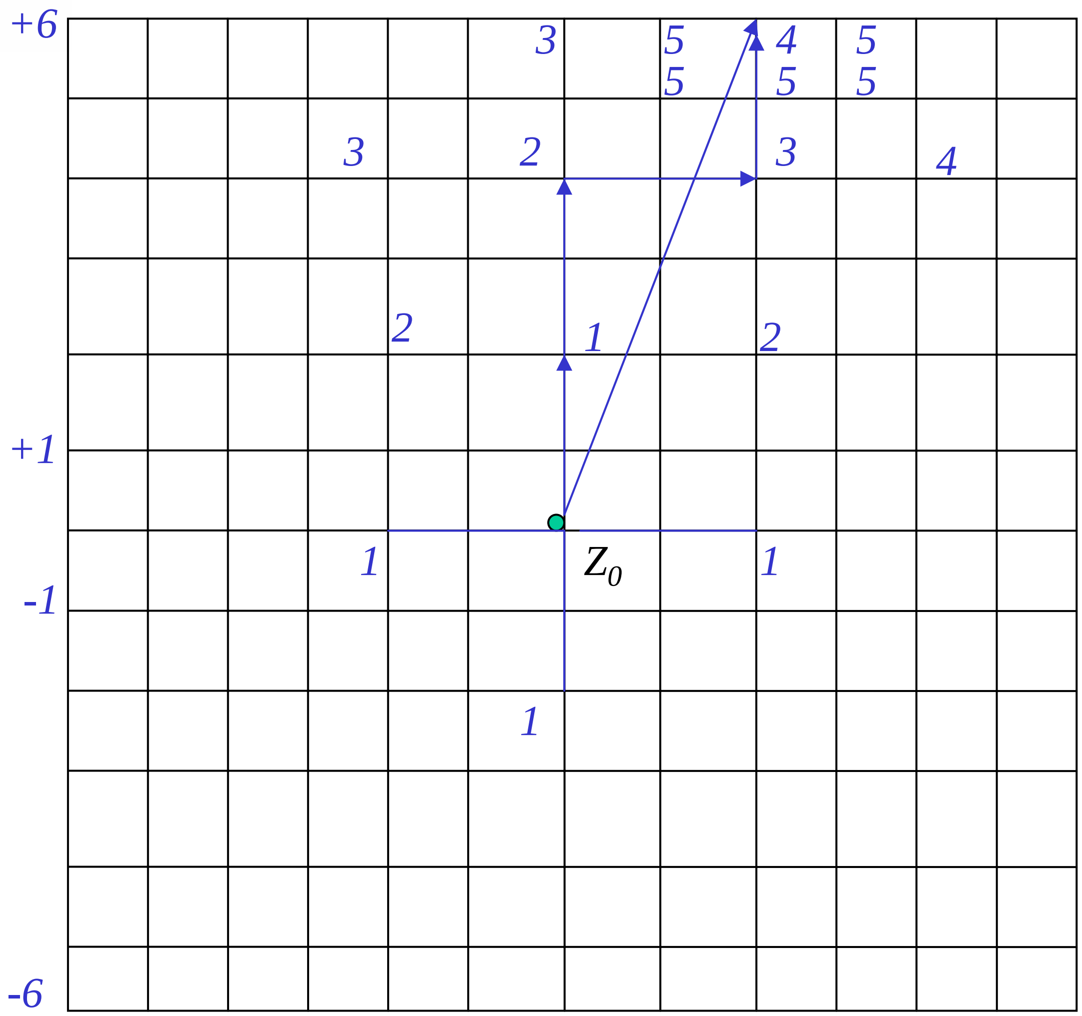
- (a) ***2-D logarithmic search***

  The distance between the search points is decreased if the minimum is at the center of search locations, or at the boundary of the search area. Each step, at most five search points are tried.
- (b) ***Three step search***: In each step, at most eight search points are tried.
- (c) ***Conjugate direction Search*** (Two-step search)

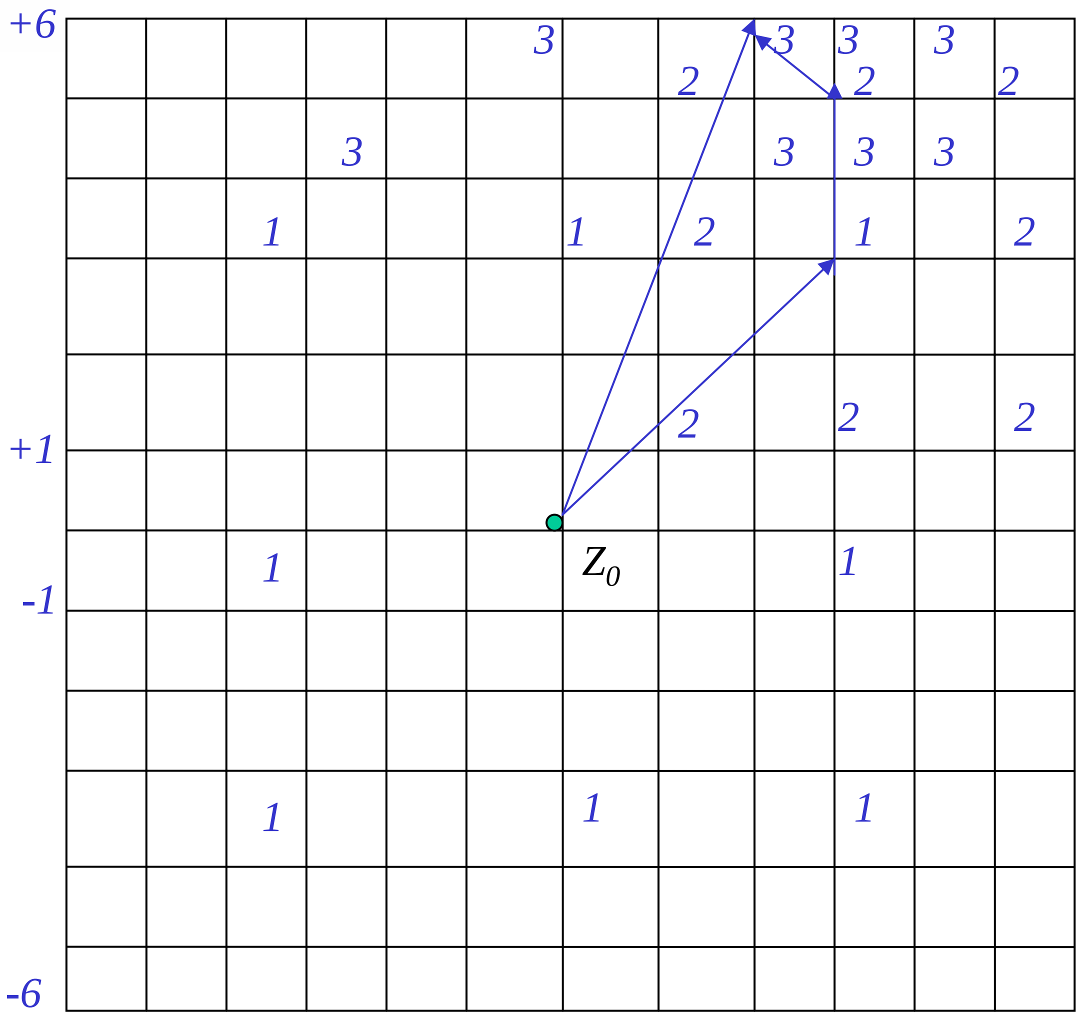
  1st step — search for minimum distortion in the horizontal direction displacement.

  2nd step — search for minimum distortion in the vertical direction displacement.

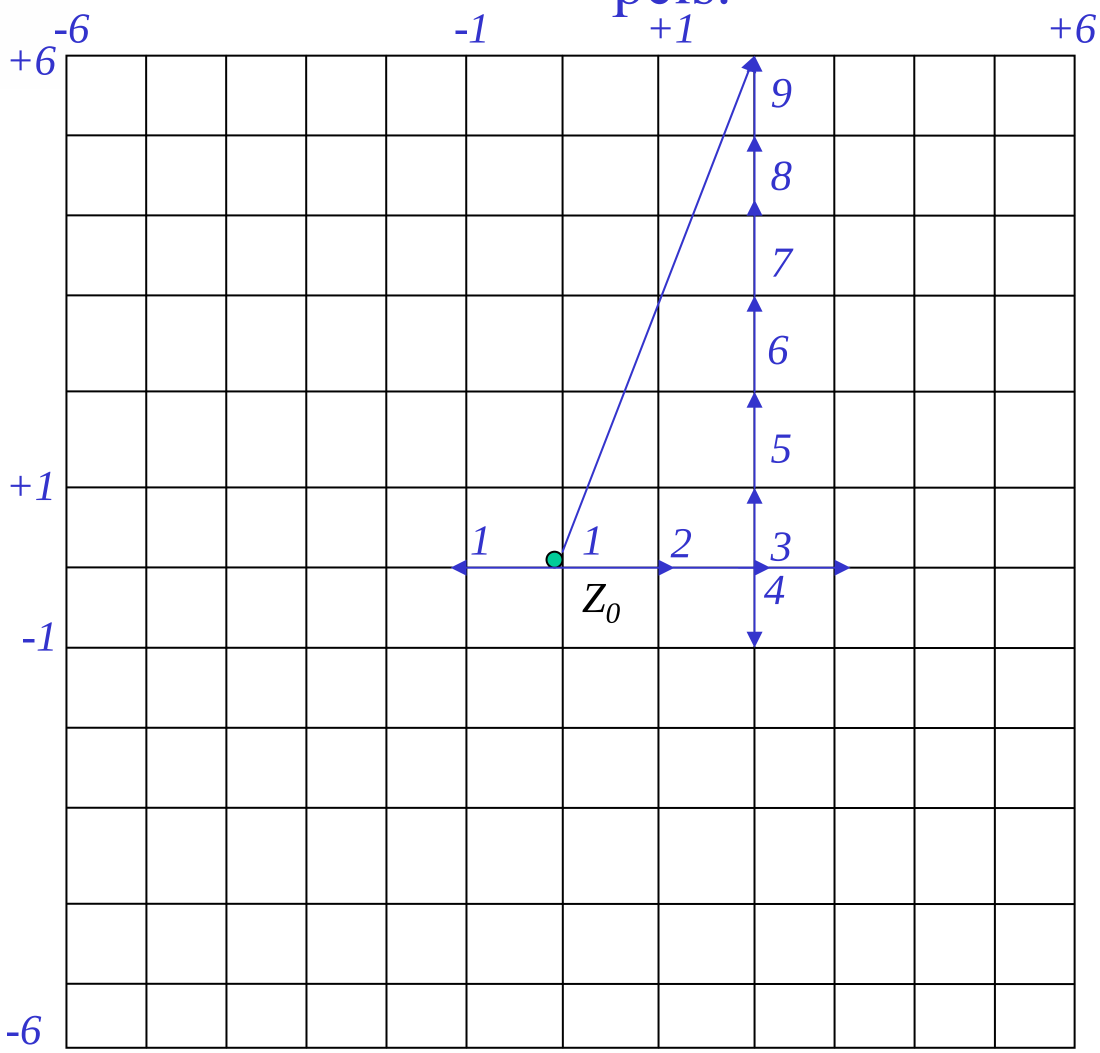
2D-logarithmic search procedure. The shifts in the search area of the previous frame are shown with respect to a pel (Z_0) in the present frame. Here the approximated displacement vectors (0,2)', (0,4)', (2,4)', (2,6)', (2,5)' are found in steps 1, 2, 3, 4 and 5. d_{max}=6 pels.



The three step search procedure. Here, (3,3)', (3,5)' and (2,6)' are the approximate displacement vectors found in steps 1, 2 and 3. d_{max} = 6 pels.



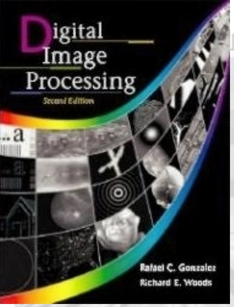
A simplified conjugate direction search method. Here, (2,6)' is the displacement vector found in step 9, i.e., $d_{max}=6$ pels.



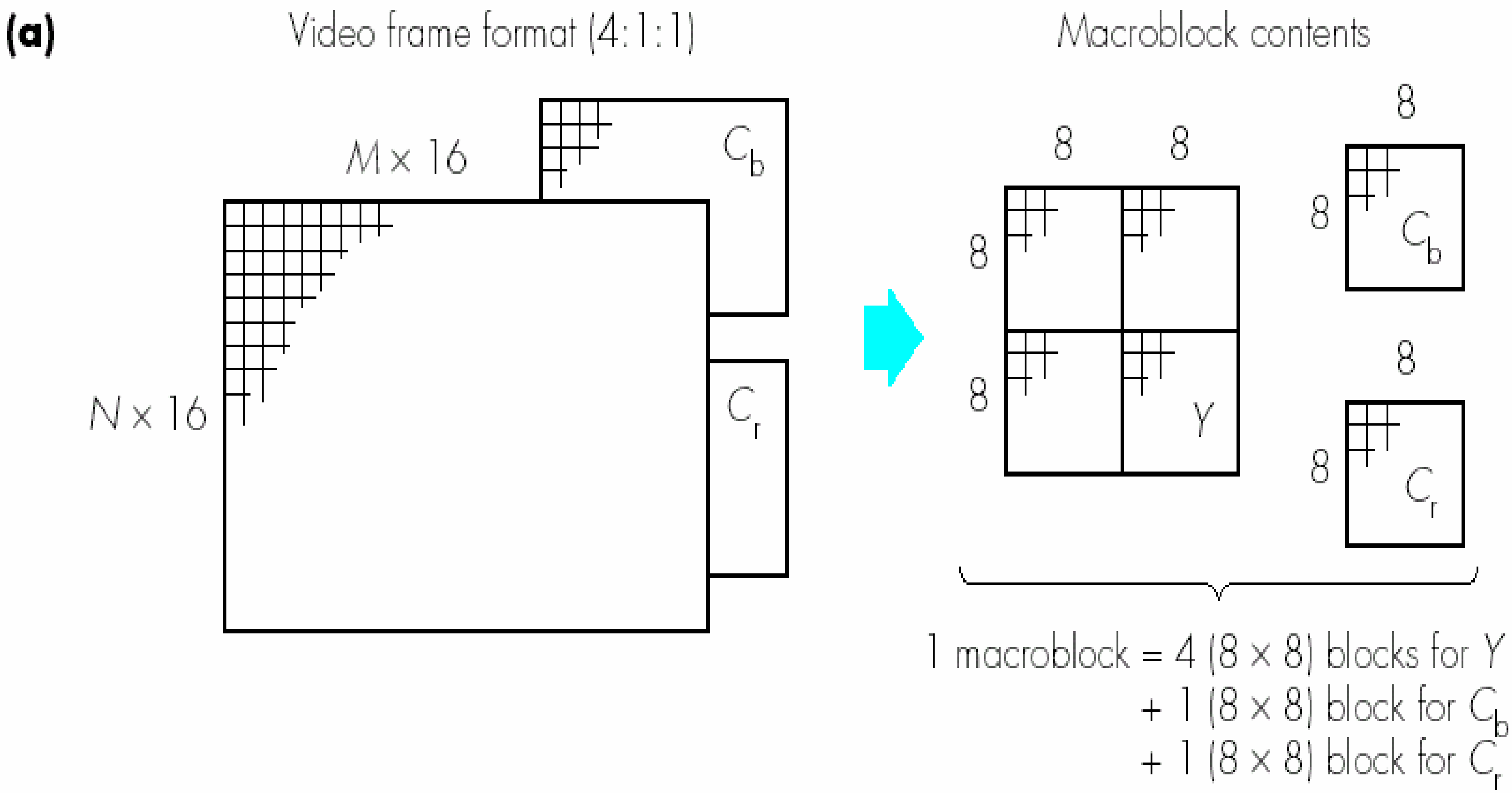
Required number of search points and sequential steps for various search procedures and a search area corresponding to a maximum displacement of 6 pels per frame. Total number of search points is Q=169.

Search procedures	Required number of search points		Required number of steps	
	a	b	a	b
2-D logarithmic	18	21	5	7
3-step search	25	25	3	3
Conjugate direction	12	15	9	12

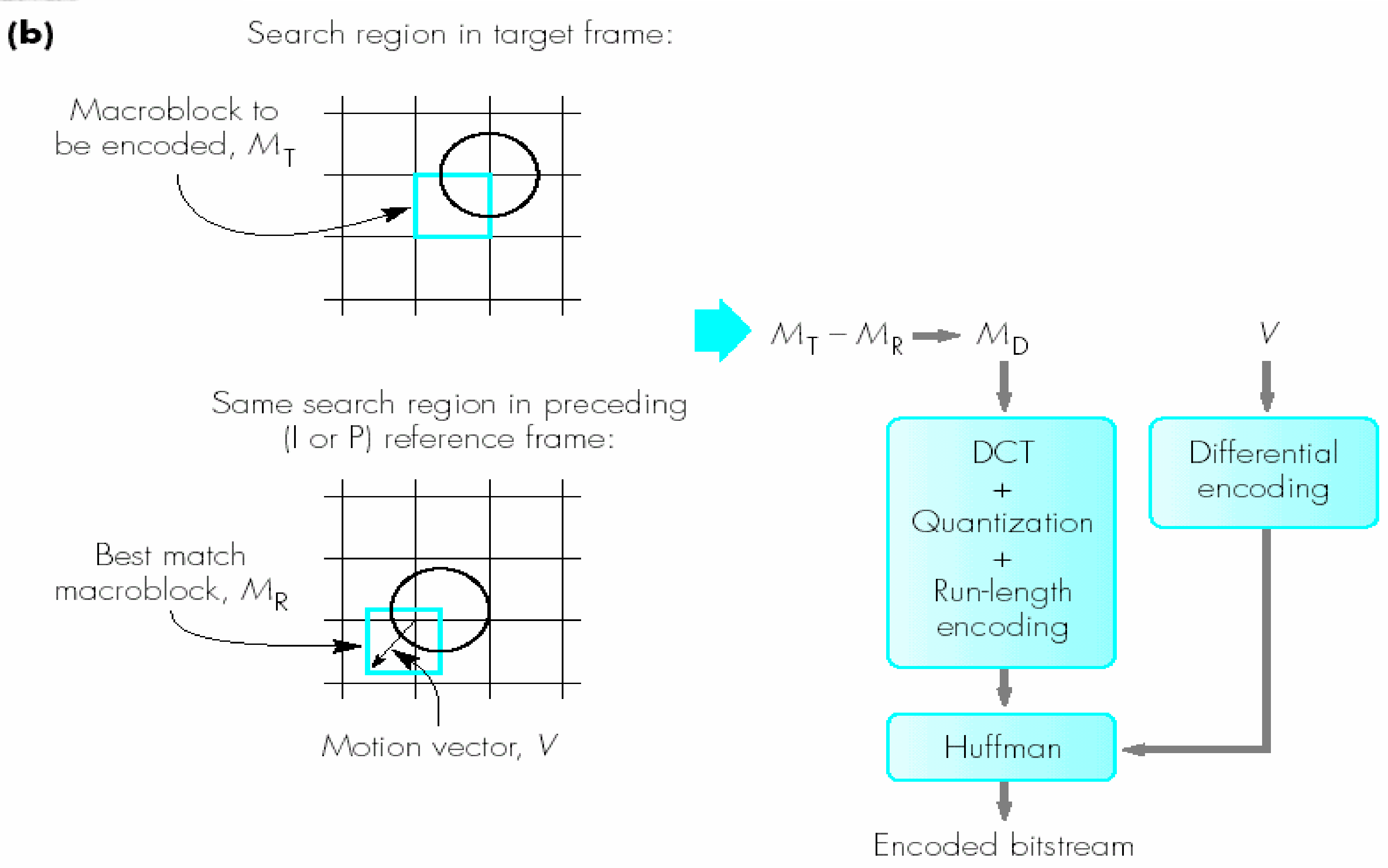
- Notes: (a) for a spatial displacement vector (2, -6)
- (b) for a worst case situation



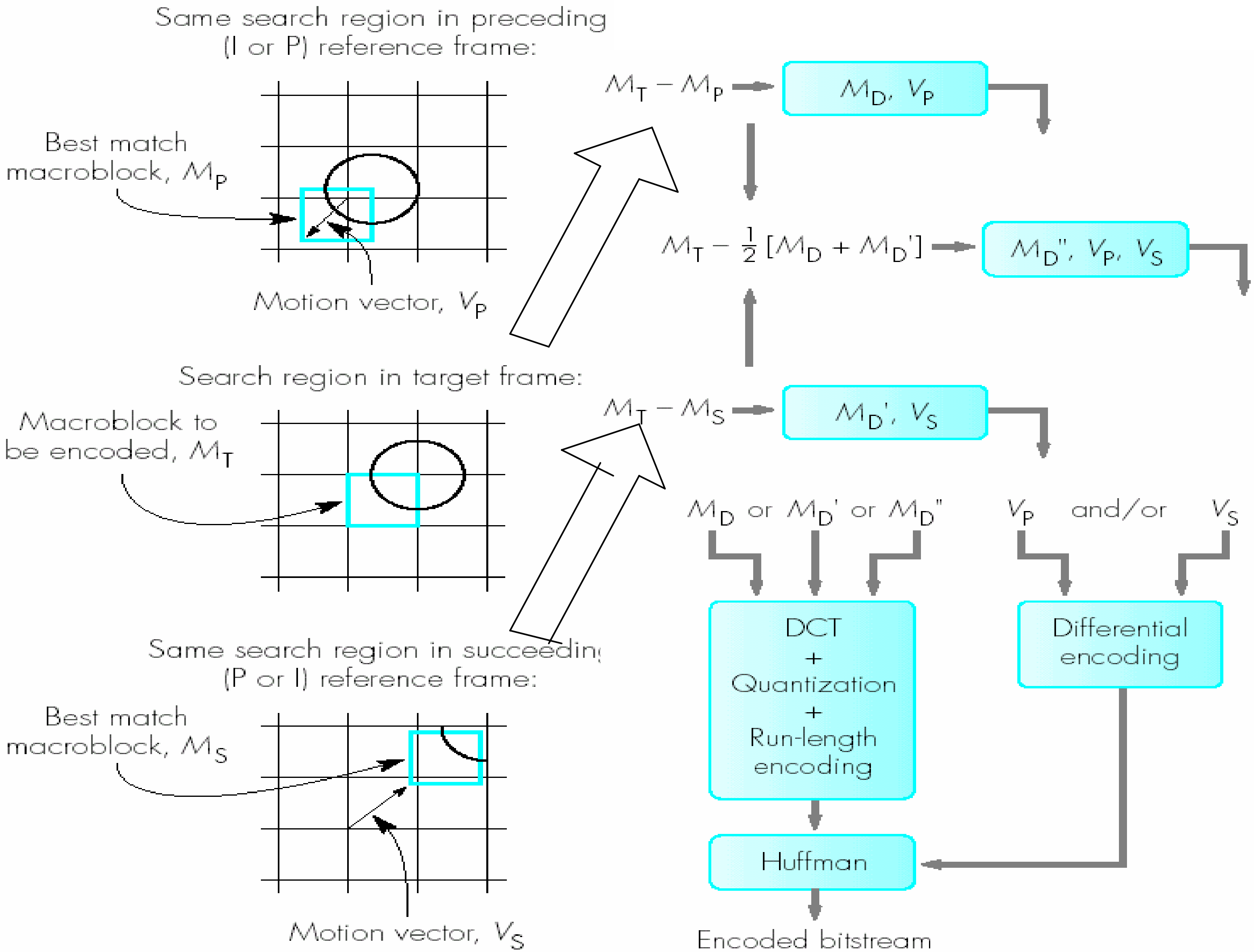
8.7.3 P-frame encoding: (a) macroblock structure;



(b) encoding procedure



8.7.4 B-frame encoding procedure.



8.7.5 Video Compression Standard-H.261

- Defined by ITU-T for the provision of video telephone and video conferencing services over ISDN
- Digitization format
  - CIF:Y=352×288, Cb=Cr=176 ×144, frame rate: 30fps
  - QCIF: :Y=176×144, Cb=Cr=88×72, frame rate: 15fps, or 7.5fps
- Use only I-frame and P-frame

8.7.5 Video Compression Standard-H.261

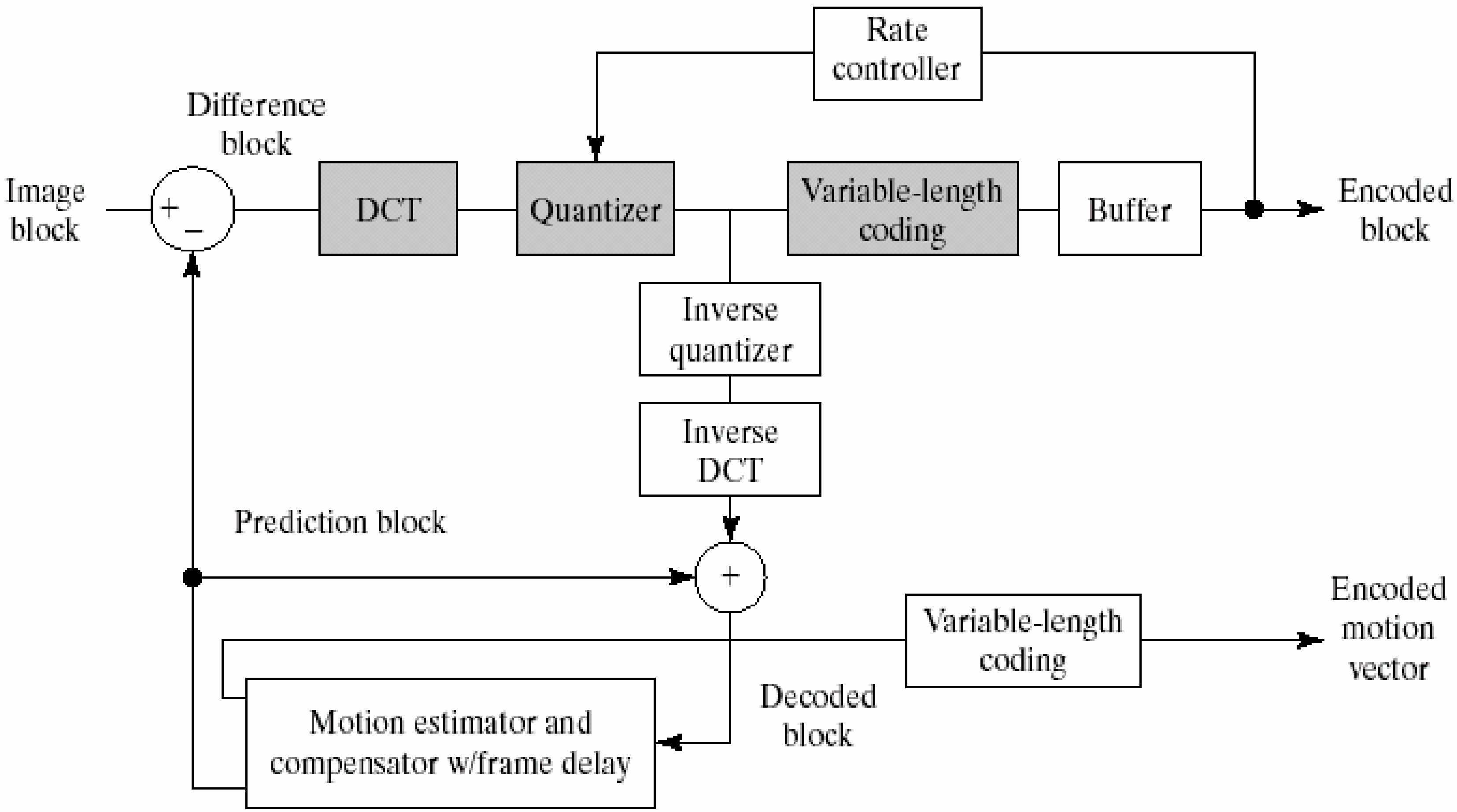
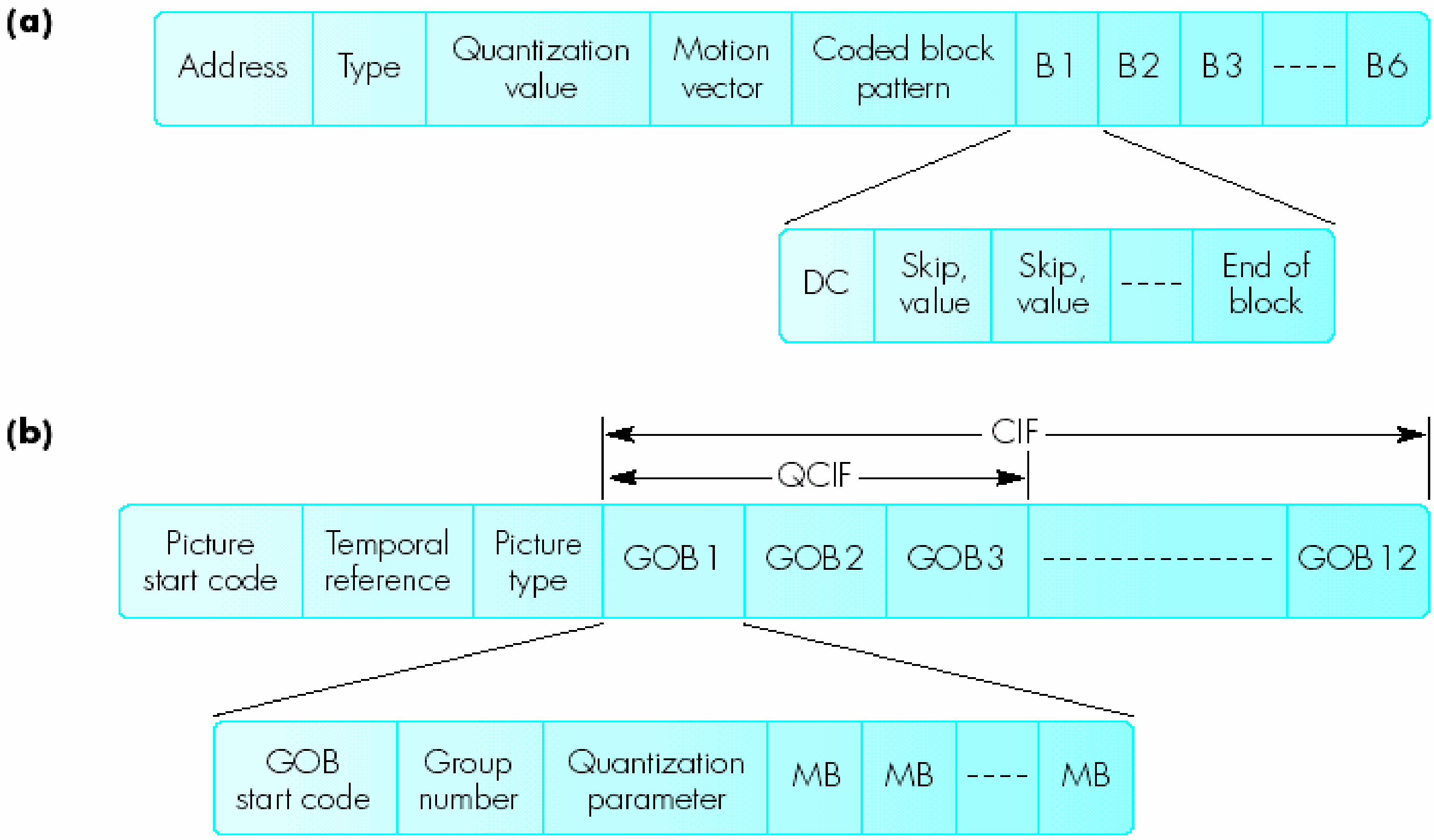


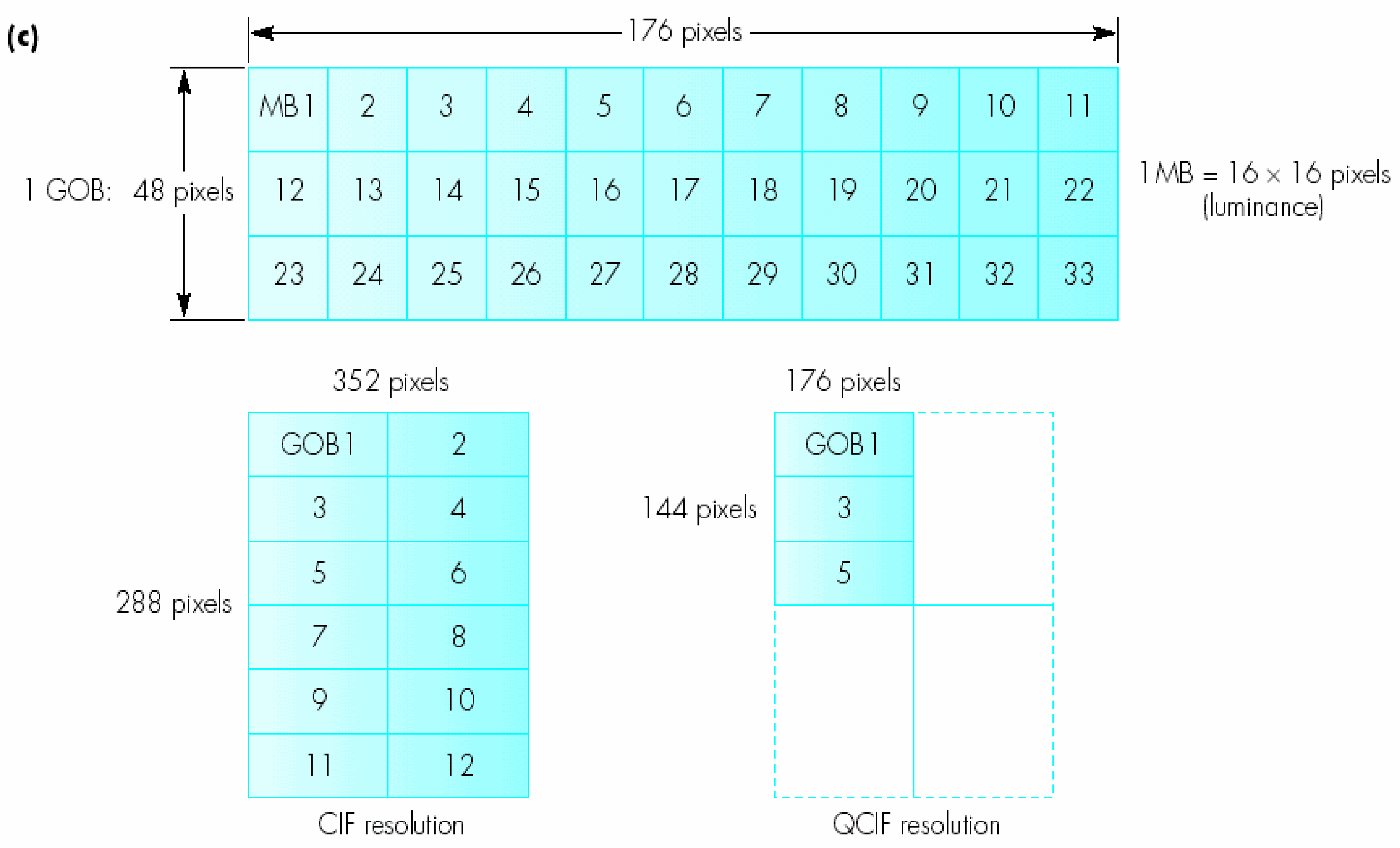
FIGURE 8.47 A basic DPCM/DCT encoder for motion compensated video compression.

8.7.5 Video Compression Standard-H.261



(a) macroblock format; (b) frame/picture format;

8.7.5 Video Compression Standard-H.261



(c) GOB structure.

8.7.6 Video Compression Standard-H.263

- Defined by ITU-I for use in a range of video applications over wireless and PSTN.
- Low-bit-rate, when bit rate lower than 64kbps, H.261 may generate the *blocking artifact*.
- Digitization format:
  - QCIF: :Y=176×144, Cb=Cr=88×72, frame rate: 15fps, or 7.5fps.
  - SQCIF: :Y=128×96, Cb=Cr=64×68, frame rate: 15fps, or 7.5fps
- Unrestricted motion vectors
  - The potential close matched macroblock that fall outside of the frame boundary.

8.7.7 Video Compression Standard-MPEG

- MPEG-1
  - VHS-quality audio and video on CD-ROM at bit rate 1.5Mbps.
  - Video resolution is based on SIF format 352×288 pels.
- MPEG-2
  - For the recording and transmission of studio-quality audio and video. It covers four levels of resolution:
    - Low: based on SIF format, target bit rate 4Mbps. It is compatible with MPEG-1.
    - Main: based on 4:2:0 format with a resolution 720 ×576. The target bit rate up to 15Mbps or 20Mbps with 4:2:2 format. It produces studio-quality video and multiple CD-quality audio channels.
    - Hight1440: based on 4:2:0 format with resolution 1440×1152. It is intended for HDTV at bit rate up to 60Mbps or 80Mbps with 4:2:2 format.
    - High 1920: based on 4:2:0 format, with a resolution of 1920×1152. It is intended for wide-screen HDTV (16:9) at bit rate up to 80Mbps or 100Mbps with 4:2:2 format.

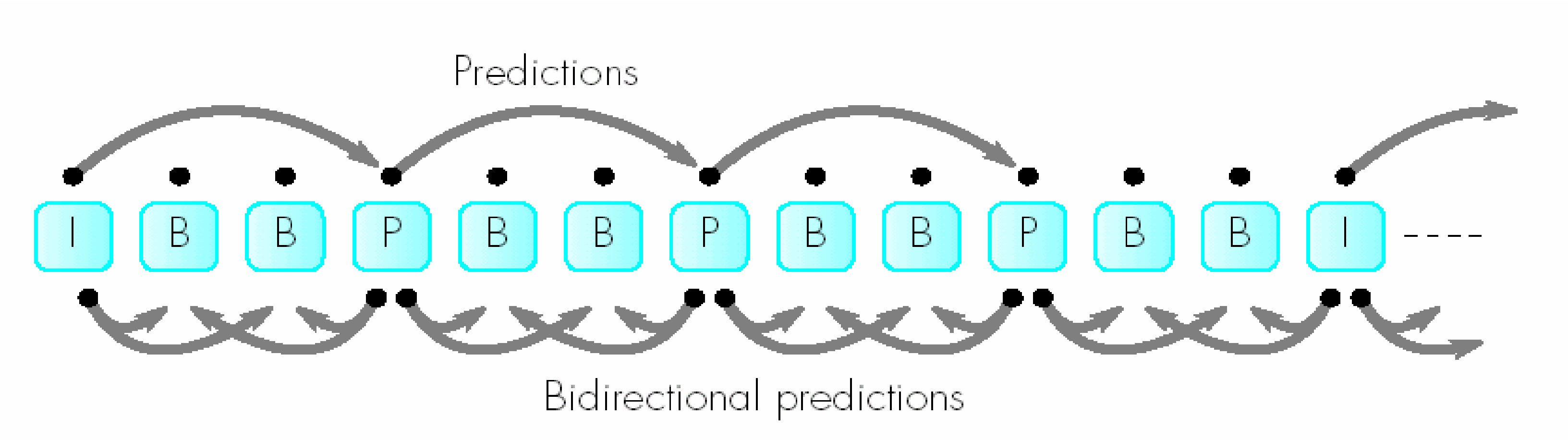
8.7.7 Video Compression Standard-MPEG

- MPEG-4
  - Similar to H.263 for very-low-bit rate applications with 4.8 to 64kbps.
  - Extended to wide range of interactive multimedia applications over the internet and other entertainment networks.
- MPRG-7
  - Describing the structure and features of the content of the compressed multimedia information produced by different standards
  - Search engine to locate the particular item and material that have defined features.

8.7.7 Video Compression Standard-MPEG-1

- Similar compression to H.261
- The microblock size is 16 ×16, the horizontal resolution reduced from 360 to 352.
- Video resolution
  - NTSC: Y= 352×240, Cr=Cb=172 ×120
  - PAL: Y= 352×288, Cr=Cb=172 ×144
- Frame types:
  - I-frame, P-frame, and B-frame
  - I-frame for various random access, with maximum access time 0.5 second, that influences the maximum separation of I-frame.
  - PAL, slow frame refresh time (1/25 sec)
    - IBBPBBPBBI….
  - NTSC, fast frame refresh rate (1/30 sec)
    - IBBPBBPBBPBBI…..

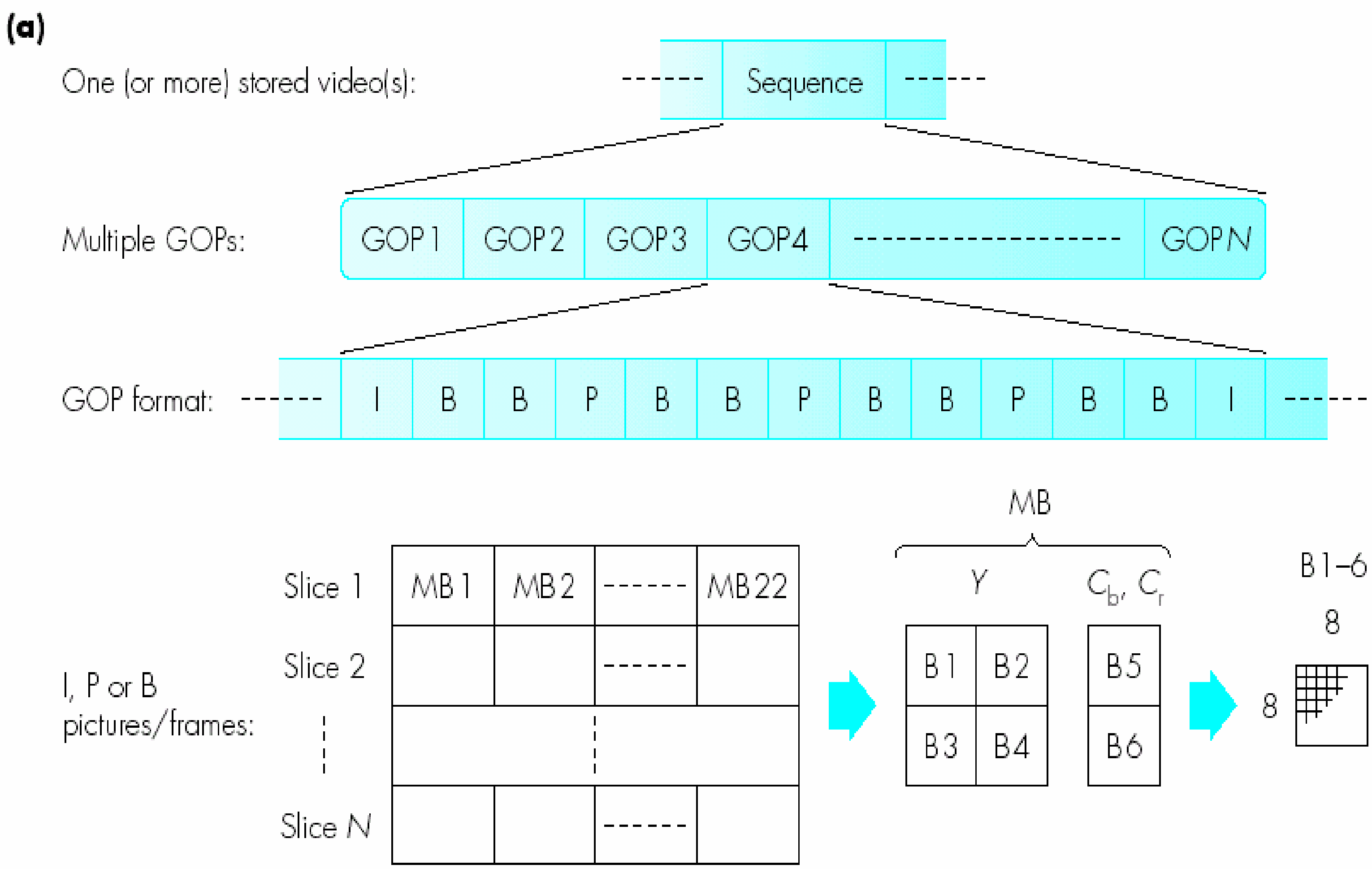
8.7.7 Video Compression Standard-MPEG-1



8.7.7 Video Compression Standard-MPEG-1

- Data structure
  - Macroblock:four 8 by 8 blocks
  - Slice: a number of macroblock between two time stamps. Normally there are 22 macroblocks for one slice.
  - Picture/frame: N slices
  - GOP (group of pictures), I, P, B frames
  - Sequence: a string of GOPs
    - Video parameters: screen size, aspect ratio,
    - bit-stream parameters: bit-rate, buffer size
    - Quantization parameters: the content of quantization table.
- Typical Compression ratios:
  - I-frame: 10:1, P-frame: 20:1, B-frame 50:1

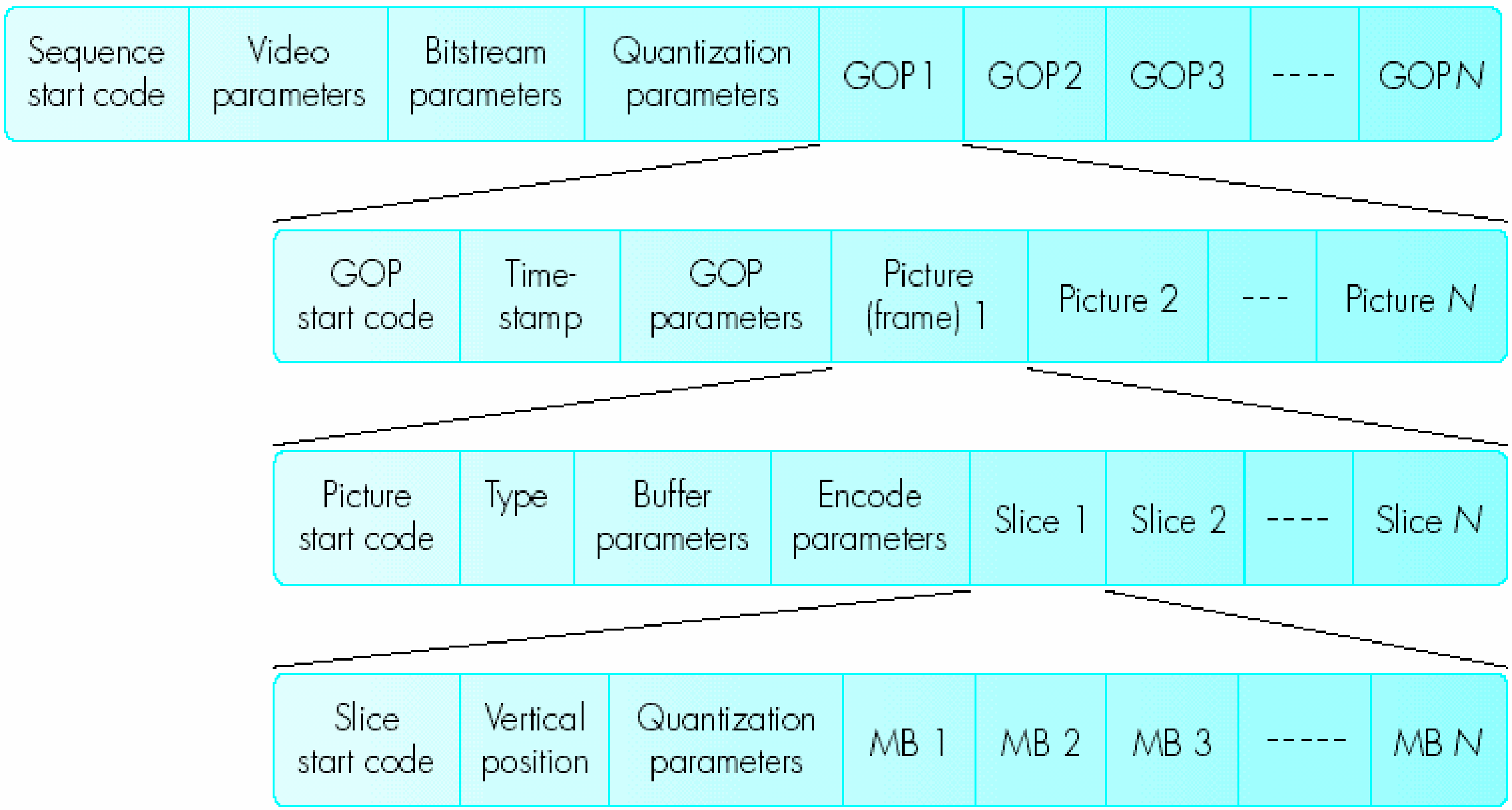
8.7.7 Video Compression Standard-MPEG-1



MPEG-1 video bitstream structure:(a) composition

8.7.7 Video Compression Standard-MPEG-1

(b)

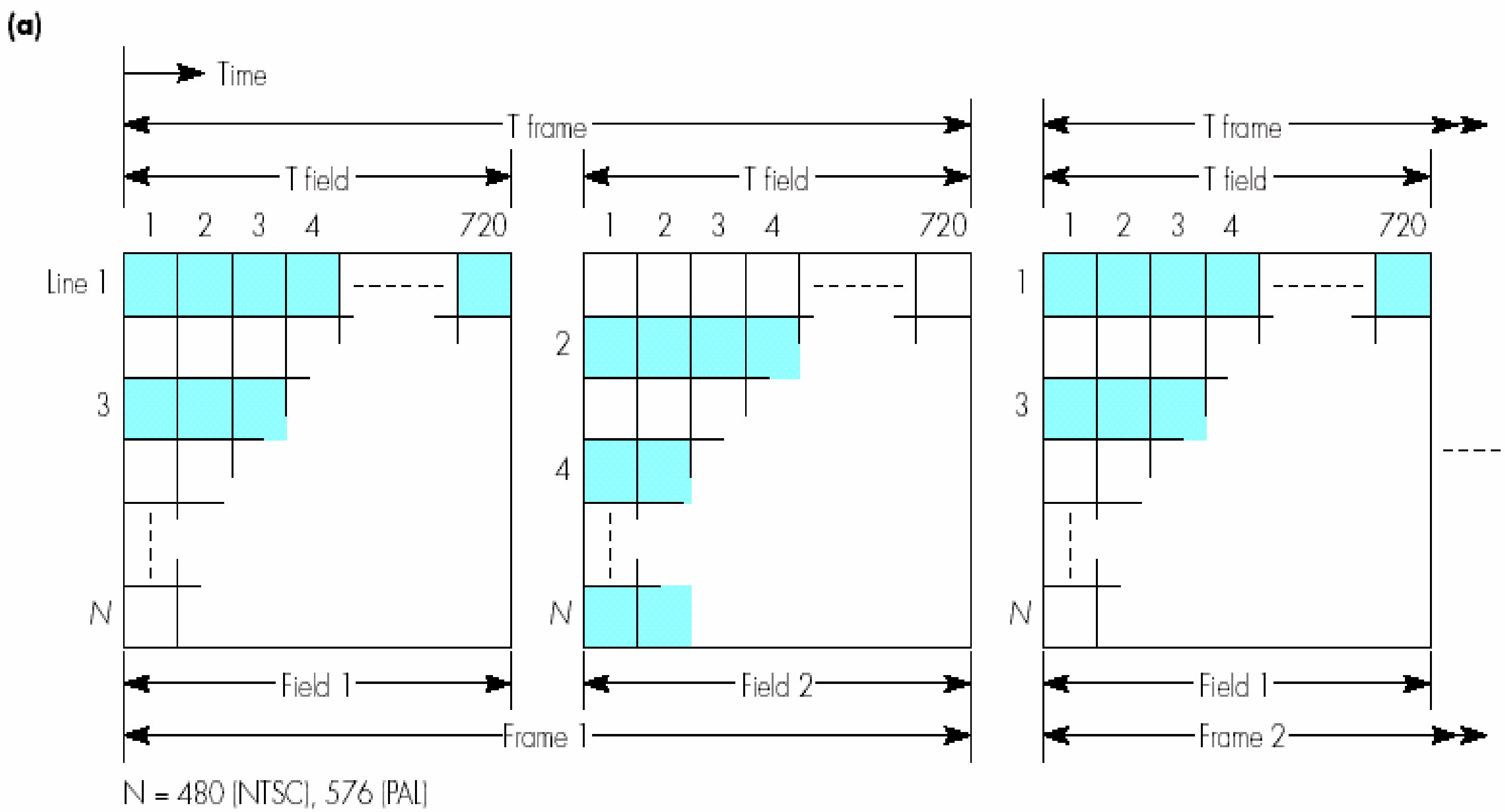


(b) format.

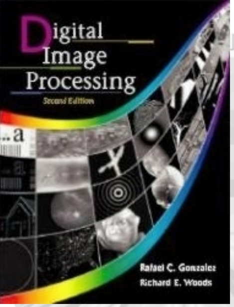
8.7.7 Video Compression Standard-MPEG-2

- Four levels: *low, main, high 1440, high*
- Five profiles:*simple, main, spatial resolution, quantization accuracy, high*
- Main profile at the main level (MP@ML)
  - Digital TV broadcasting
  - 4:2:0 format for NTSC (a resolution 720×480) and PAL(a resolution 720×576)
  - Interlaced scanning
  - DCT block (Field mode or Frame mode)
  - Motion estimation (Field mode, Frame mode or Mixed mode)
    - In the mixed mode, the motion vectors for the field and frame modes are computed and the best one is selected.

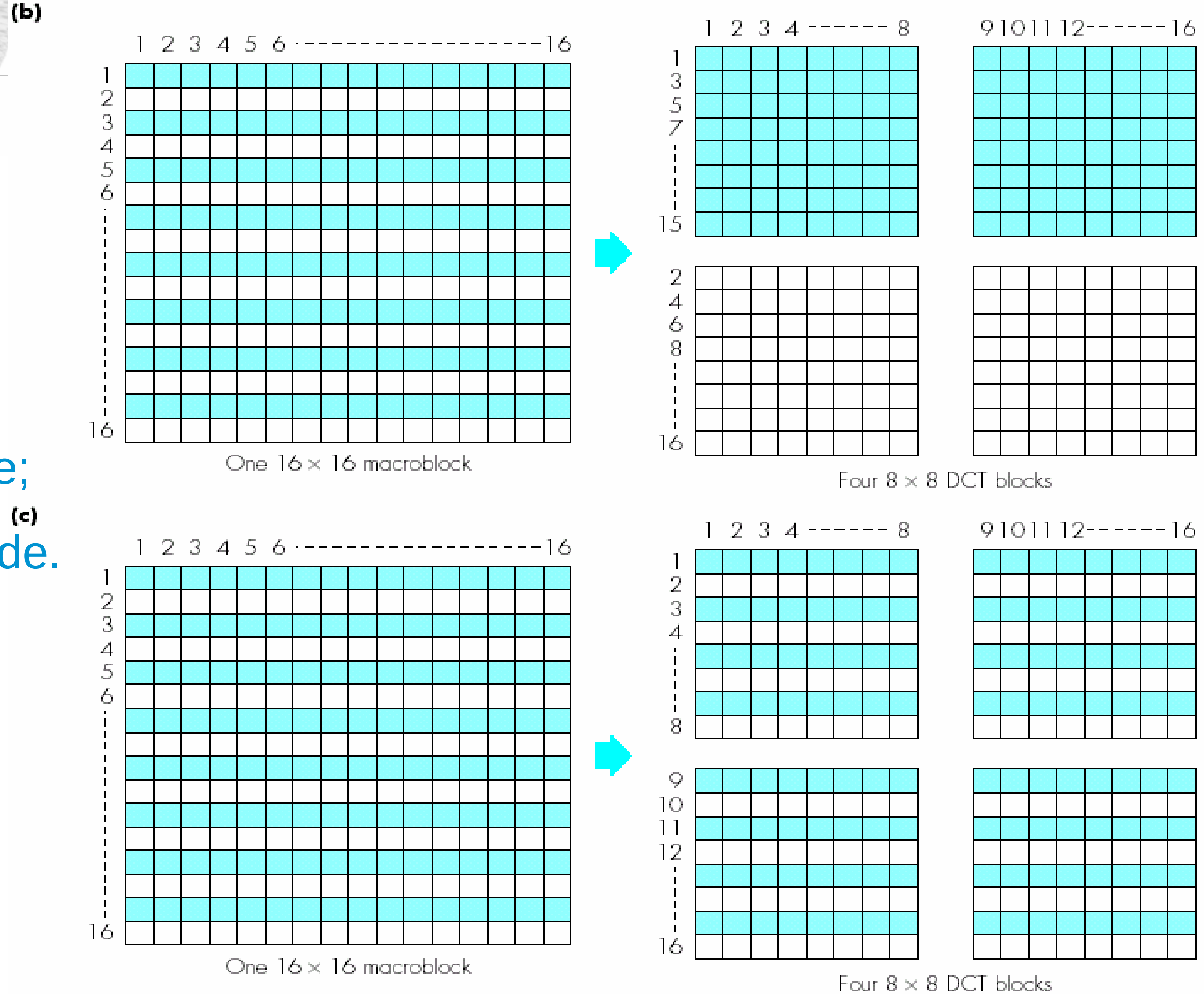
8.7.7 Video Compression Standard-MPEG-2



MPEG-2 DCT block derivation with I-frames: (a) effect of interlaced scanning;



8.7.7 Video Compression Standard-



(b) field mode;

(c) frame mode.

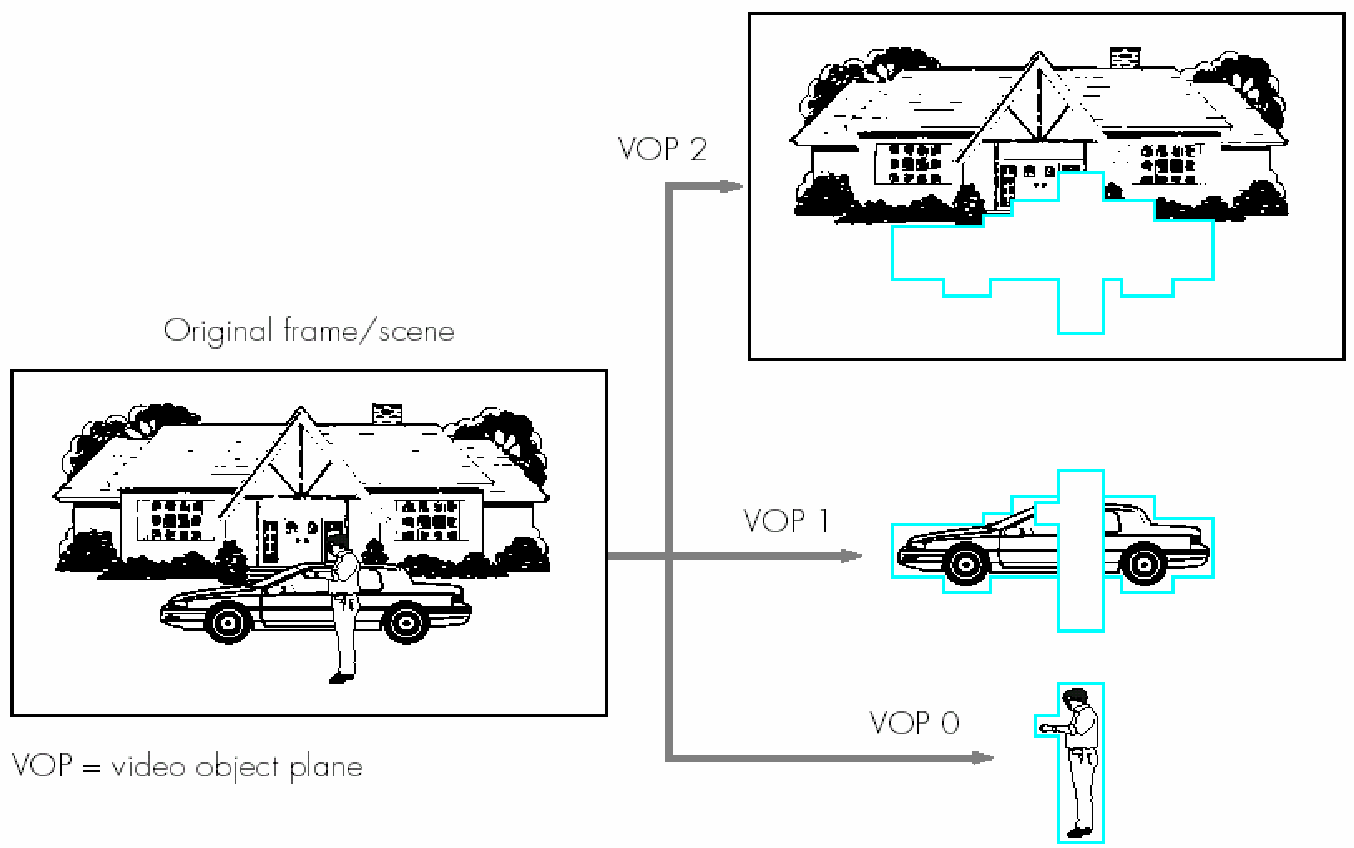
8.7.7 Video Compression Standard-MPEG-2

- HDTV-
  - ATV (Advance TV) in US
  - DVB (Digital Video Broadcast) in Europe
  - MUSE (Multiple sub-Nyquest sampling encoding) in Japan
  - ATV is formulated as a ***Grand Alliance Standard***
    - Main Profile at High Level (MP@HL) of MPEG-2
    - 16/9 aspect ratio, 1280 × 720
  - DVB
    - Spatial scalable profile at high 1440(SSP@H1440) of MPEG-2
    - 4/3 ratio 1440 × 1152
  - MUSE
    - HP@HL
    - 16/9 ratio, 1920 × 1035

8.7.7 Video Compression Standard-MPEG-4

- **Scene composition:** MPEG-4 has a number of *content-based functionalities*
  - ***Audio-visual objects (AVOs):*** each scene is defined in the form of background and one or more foreground AVOs. Each AVO is in turn defined in form of one or more video/audio objects.
  - ***Object descriptor*** : Each audio or video object has a associated object descriptor
  - ***Binary format for scene*** (BIFS): The language used to describe an modify objects.
  - ***Scene descriptor***: Define the way the various AVOs are related to each other in the context of complete scene.
  - ***Video object plane (VOPs)***: Each video frame is segmented into a number of VOPs, each corresponds to an AVO of interest.

8.7.7 Video Compression Standard-MPEG-4

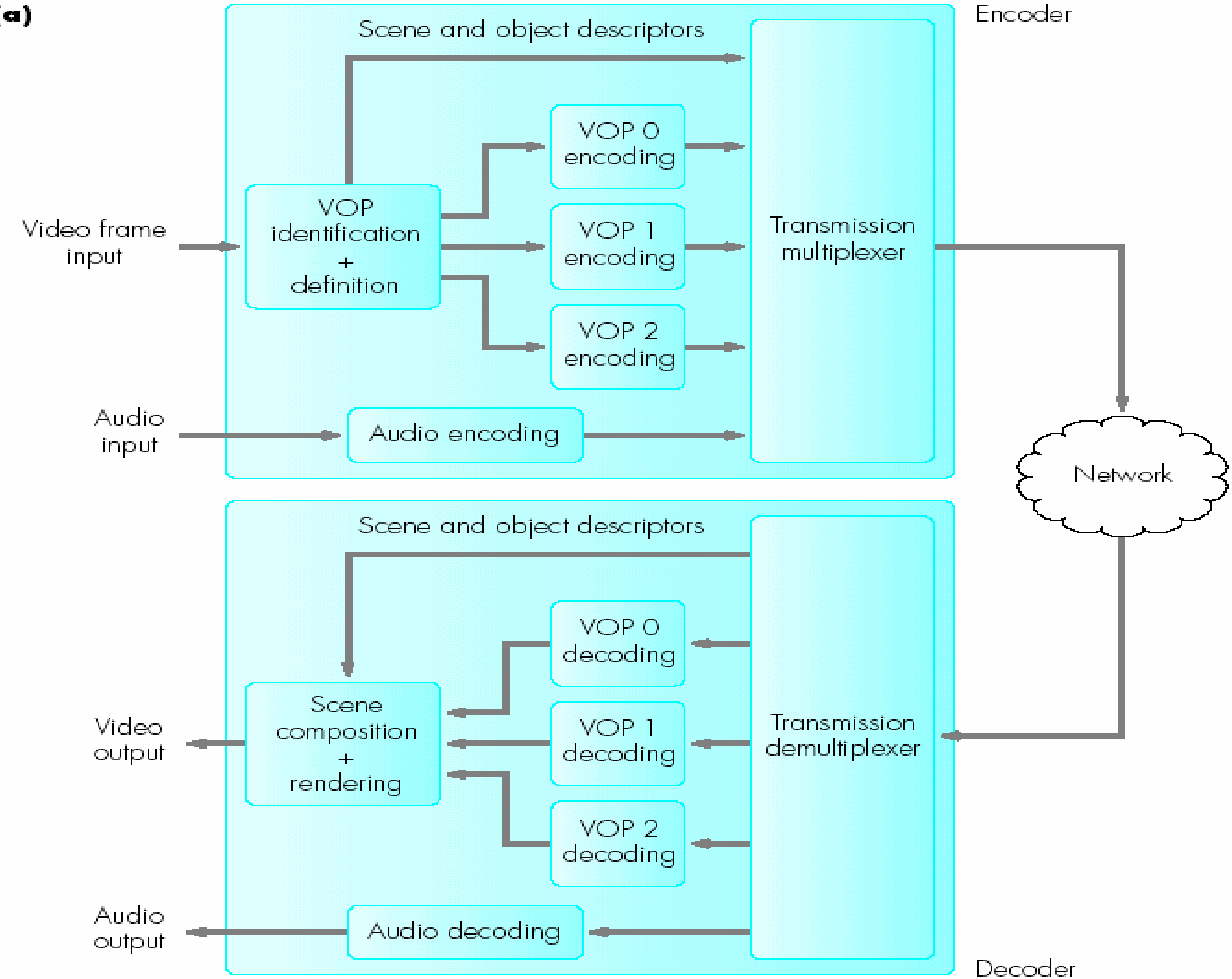


8.7.7 Video Compression Standard-MPEG-4

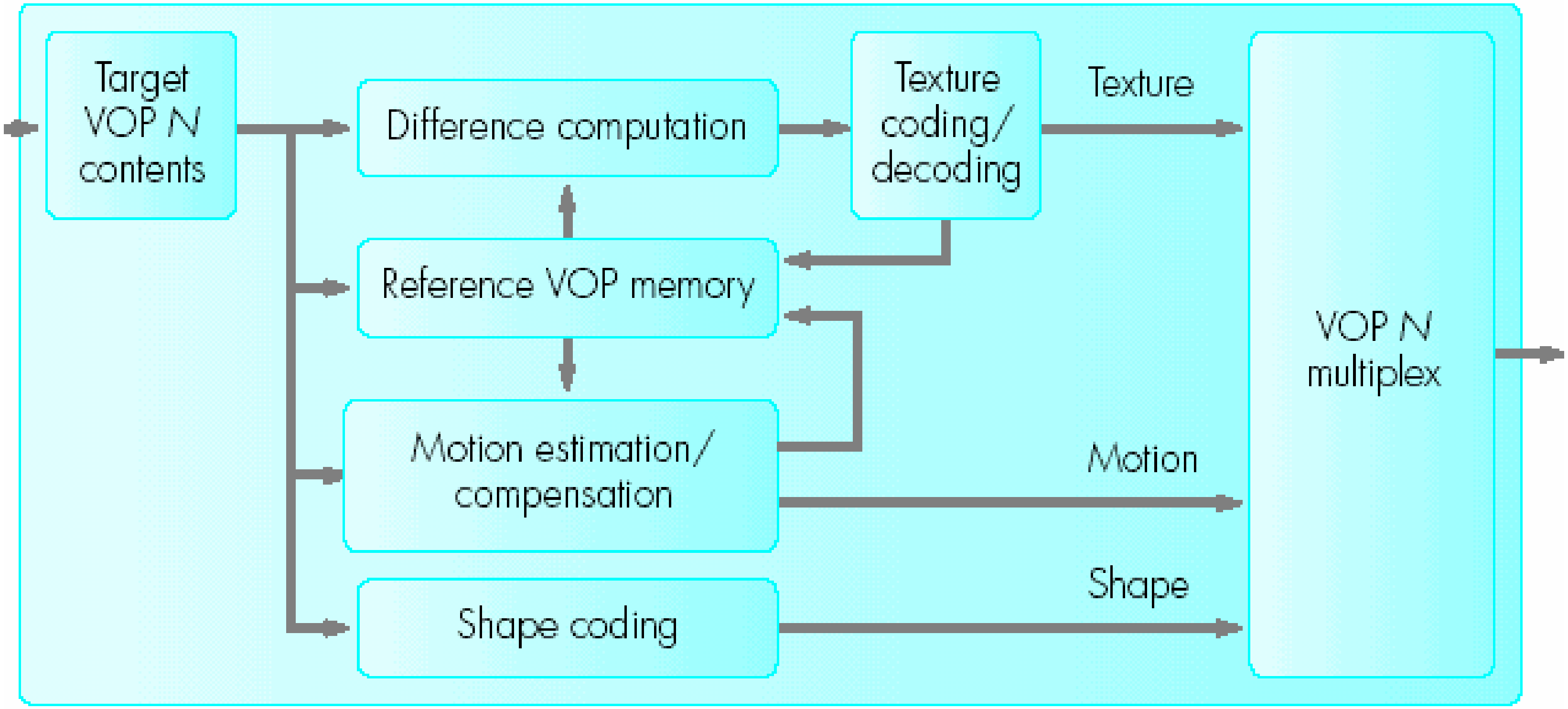
- **MPEG-4 Video Compression**
  - Each VOP is identified and encoded separately.
  - Identifying regions within a frame that have similar property such as color, texture, or brightness.
  - Each resulting object shapes is then bounded by a rectangle (which contains minimum number of macroblocks) to form the related VOP.
  - VOP is encoded based on its shape, texture and motion

8.7.7 Video Compression Standard-MPEG-4

(a)



8.7.7 Video Compression Standard-MPEG-4



VOP encoder schematic.