



Chapter08 Image Compression

Ping Zhang



Outline

- ◆ **Background**
- ◆ **Fundamentals**
- ◆ **Some Basic Compression Methods**
- ◆ **Digital Image Watermarking***



Background

数字图像通常要求很大的比特数，这给图像的传输和存储带来一定的困难。

如：一幅灰度图像，512x512大小，
其比特数为：

$$512 \times 512 \times 8 = 2,097,152 \text{ bit} = \text{256KB}$$



如：一幅卫星图像，采用12bit灰度
级的比特数为：

$$2340 \times 3240 \times 12 = 90,979,200 \text{ bit} = \text{10MB}$$





Background

如：一部2小时的标准清晰度(SD)电影，每秒放映30帧。把它数字化，每帧720x480像素，每像素的R、G、B三分量分别占8bit，总比特数为：

$$(2 \times 60 \times 60) \times 30 \times 720 \times 480 \times 3 \text{byte} \approx 224 \text{GB}$$

如一张DVD光盘可存8.5GB字节数据，这部电影就需要27张DVD光盘用来存储。

Compression Anywhere



Outline

- ◆ **Background**
- ◆ **Fundamentals**
- ◆ **Some Basic Compression Methods**
- ◆ **Digital Image Watermarking***



8.1 Fundamentals



Agenda

-  ***Coding Redundancy***
-  ***Spatial and Temporal Redundancy***
-  ***Irrelevant Information***
-  ***Measuring Image Information***
-  ***Fidelity Criteria***
-  ***Image Compression Models***
-  ***Image Formats, Containers, and Compression Standards***





8.1 Fundamentals

- The term *data compression* refers to the process of reducing the amount of data required to represent a given quantity of information
- Data \neq Information
- Various amount of data can be used to represent the same information
- Data might contain elements that provide no relevant information : *data redundancy*



8.1 Fundamentals

- Let b and b' denote the number of information carrying units in two data sets that represent the same information
- The **relative redundancy** R is define as :




$$R = 1 - 1 / C$$

where C , commonly called the **compression ratio**, is

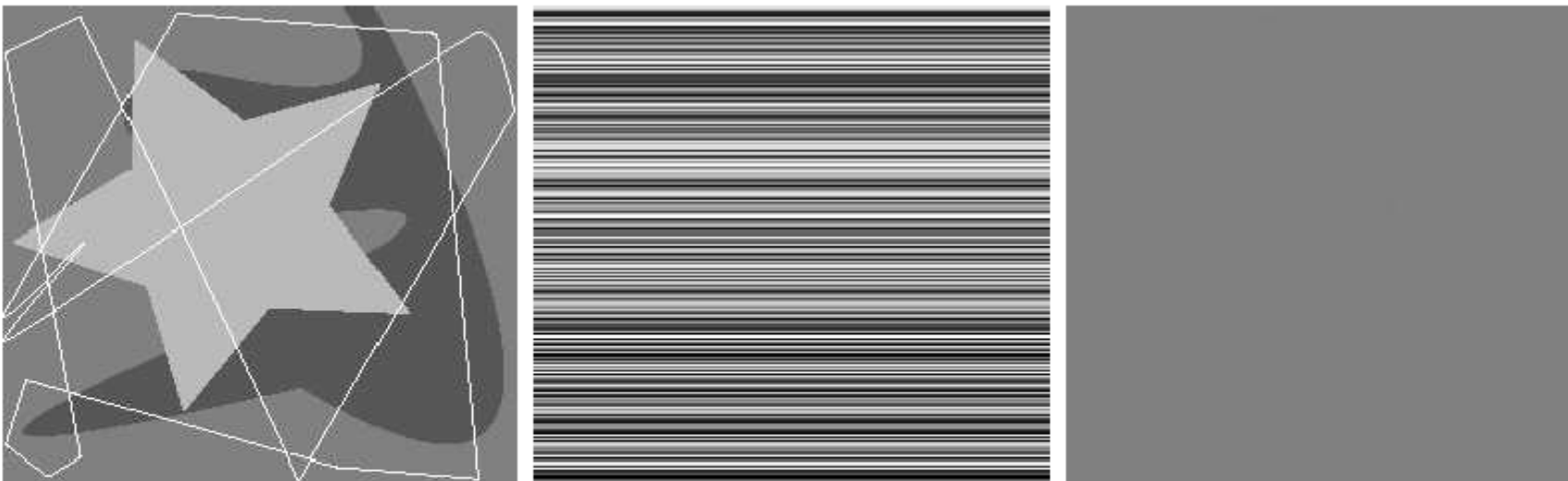
$$C = b / b'$$



8.1 Fundamentals

- If $b = b'$, $C = 1$ and $R = 0$  *no redundancy*
 - If $b \gg b'$, $C \rightarrow \infty$ and $R \rightarrow 1$  *high redundancy*
 - If $b \ll b'$, $C \rightarrow 0$ and $R \rightarrow -\infty$  *undesirable*
-
- In Image compression, 3 basic redundancy can be identified
 - *Coding Redundancy*
 - *Spatial and Temporal Redundancy*
 - *Irrelevant Information*

8.1 Fundamentals



a b c

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

- *Coding Redundancy*
- *Spatial and Temporal Redundancy*
- *Irrelevant Information*



1. Coding Redundancy

- Recall from the histogram calculations

$$p_r(r_k) = \frac{n_k}{MN} \quad k = 0, 1, 2, \dots, L-1$$

$p_r(r_k)$ is the probability of a pixel in $M \times N$ image to have a certain value r_k ;

n_k is the number of times that the k th intensity appears in the image;

L is the number of intensity value.

- If the number of bits used to represent r_k is $l(r_k)$, then

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

Example 8.1 A simple illustration of variable-length coding

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

$$\begin{aligned}
 L_{avg} &= \sum_{k=0}^4 l(r_k) P_r(r_k) \\
 &= 2(0.25) + 1(0.47) + 3(0.25) + 3(0.03) \\
 &= 1.81 \text{ bit}
 \end{aligned}$$

$$C = \frac{256 \times 256 \times 8}{256 \times 256 \times 1.81} \approx 4.42$$

$$R = 1 - \frac{1}{4.42} = 0.774$$

2. Spatial and Temporal Redundancy

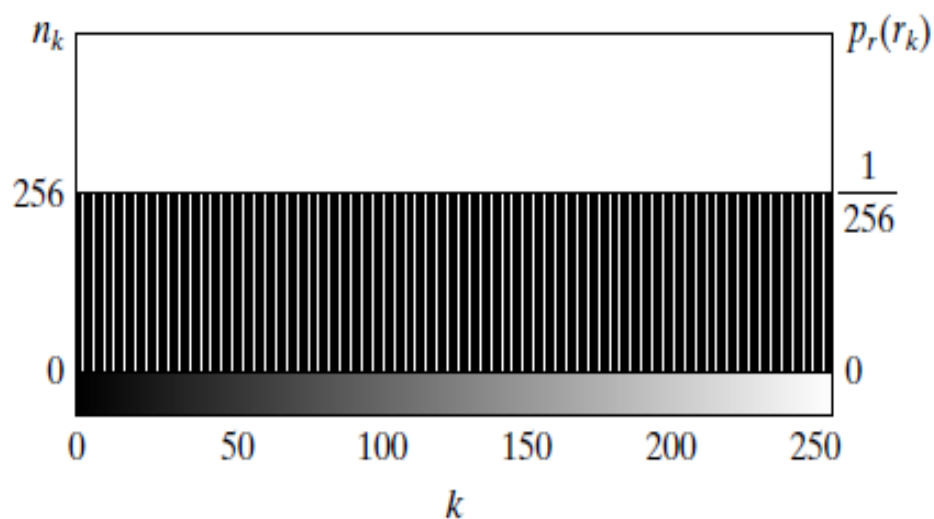


FIGURE 8.2 The intensity histogram of the image in Fig. 8.1(b).

Run Length Coding:

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

3. Irrelevant Information

- Information that is ignored by the human visual system or is extraneous to the intended use of an image are obvious candidates for omission.





4. Measuring Image Information

- a random event E with probability $P(E)$ is said to contain this units of information:

$$I(E) = \log \frac{1}{P(E)} = -\log P(E)$$

- a discrete set of possible $\{a_1, a_2, \dots, a_J\}$ with associated probabilities $\{P(a_1), P(a_2), \dots, P(a_J)\}$, the average information per source output, called the **entropy** of the source:

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



4. Measuring Image Information

Example:

$$A = \left\{ a_1, a_2, a_3, a_4 \right\} \\ \left\{ \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8} \right\}$$

$$H = -\sum_{j=1}^4 P(a_j) \log P(a_j) = \frac{7}{4}$$

4. Measuring Image Information

- Image Entropy:



$$\tilde{H} = - \sum_{k=0}^{L-1} p_r(r_k) \log_2 p_r(r_k)$$

r_k	$p_r(r_k)$
$r_{87} = 87$	0.25
$r_{128} = 128$	0.47
$r_{186} = 186$	0.25
$r_{255} = 255$	0.03
$r_k \text{ for } k \neq 87, 128, 186, 255$	0

$$\begin{aligned}
 \tilde{H} &= -[0.25 \log_2 0.25 + 0.47 \log_2 0.47 + 0.25 \log_2 0.25 + 0.03 \log_2 0.03] \\
 &\approx -[0.25(-2) + 0.47(-1.09) + 0.25(-2) + 0.03(-5.06)] \\
 &\approx 1.6614 \text{ bits / pixel}
 \end{aligned}$$



4. Measuring Image Information

Shannon's first theorem:

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

- $L_{avg,n}$ is the average number of code symbols required to represent all n symbol groups.
- $L_{avg,n}/n$ can be made arbitrarily close to **H** by encoding infinitely long extensions of the single-symbol source.



5. Fidelity Criteria

- The error between two functions is given by:

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

- So, the total error between the two images is

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

- The **root-mean-square error** averaged over the whole image is

$$e_{rms} = \frac{1}{MN} \sqrt{[\hat{f}(x, y) - f(x, y)]^2}$$



5. Fidelity Criteria

- A closely related objective fidelity criterion is the mean square signal to noise ratio of the compressed-decompressed image

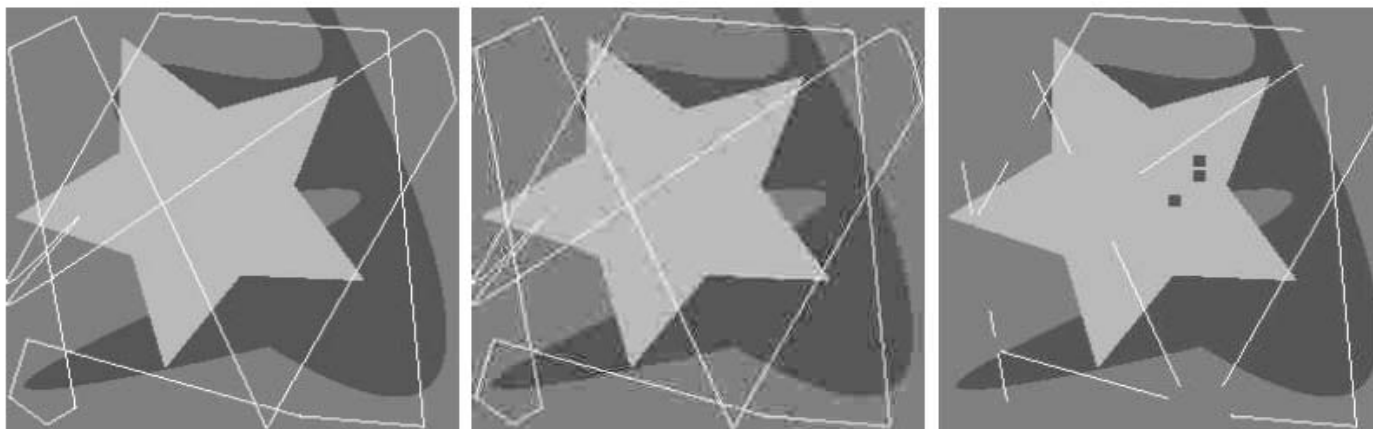
$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}$$

5. Fidelity Criteria

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

TABLE 8.2

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



a b c

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

6. Image Compression Models

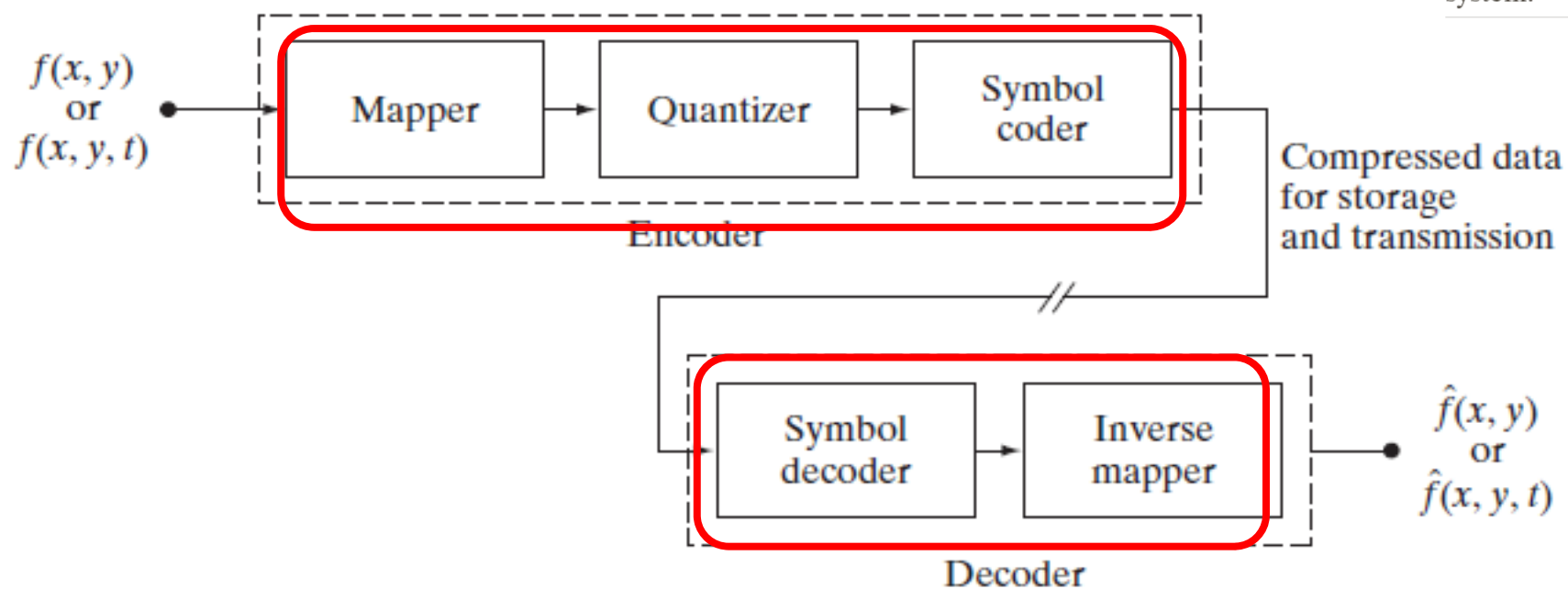


FIGURE 8.5
Functional block
diagram of a
general image
compression
system.

7. Image Formats, Containers, and Compression Standards

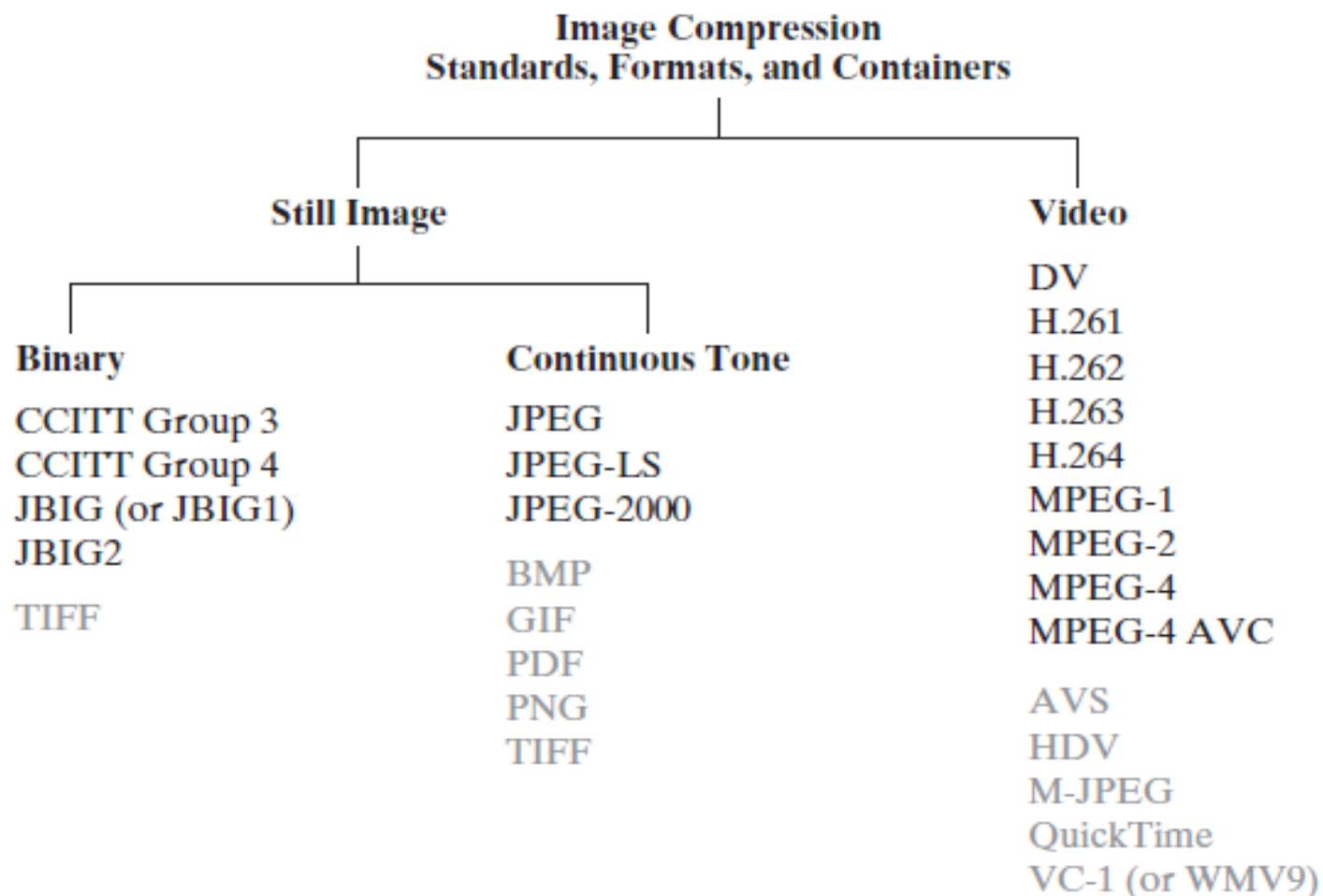


FIGURE 8.6 Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in black; all others are grayed.

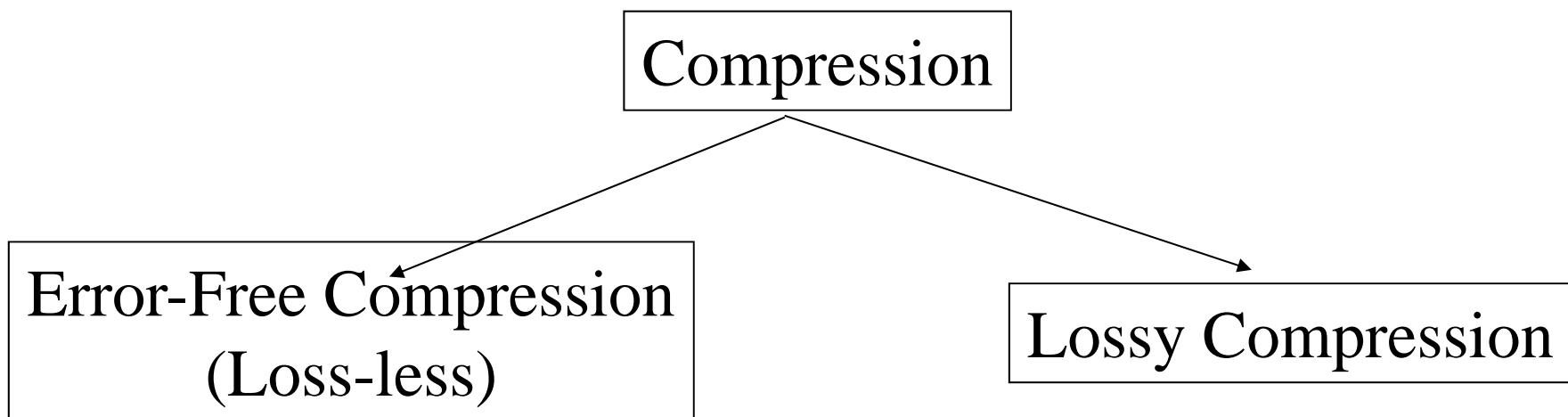
Outline

- ◆ Background
- ◆ Fundamentals
- ◆ Some Basic Compression Methods
- ◆ Digital Image Watermarking*

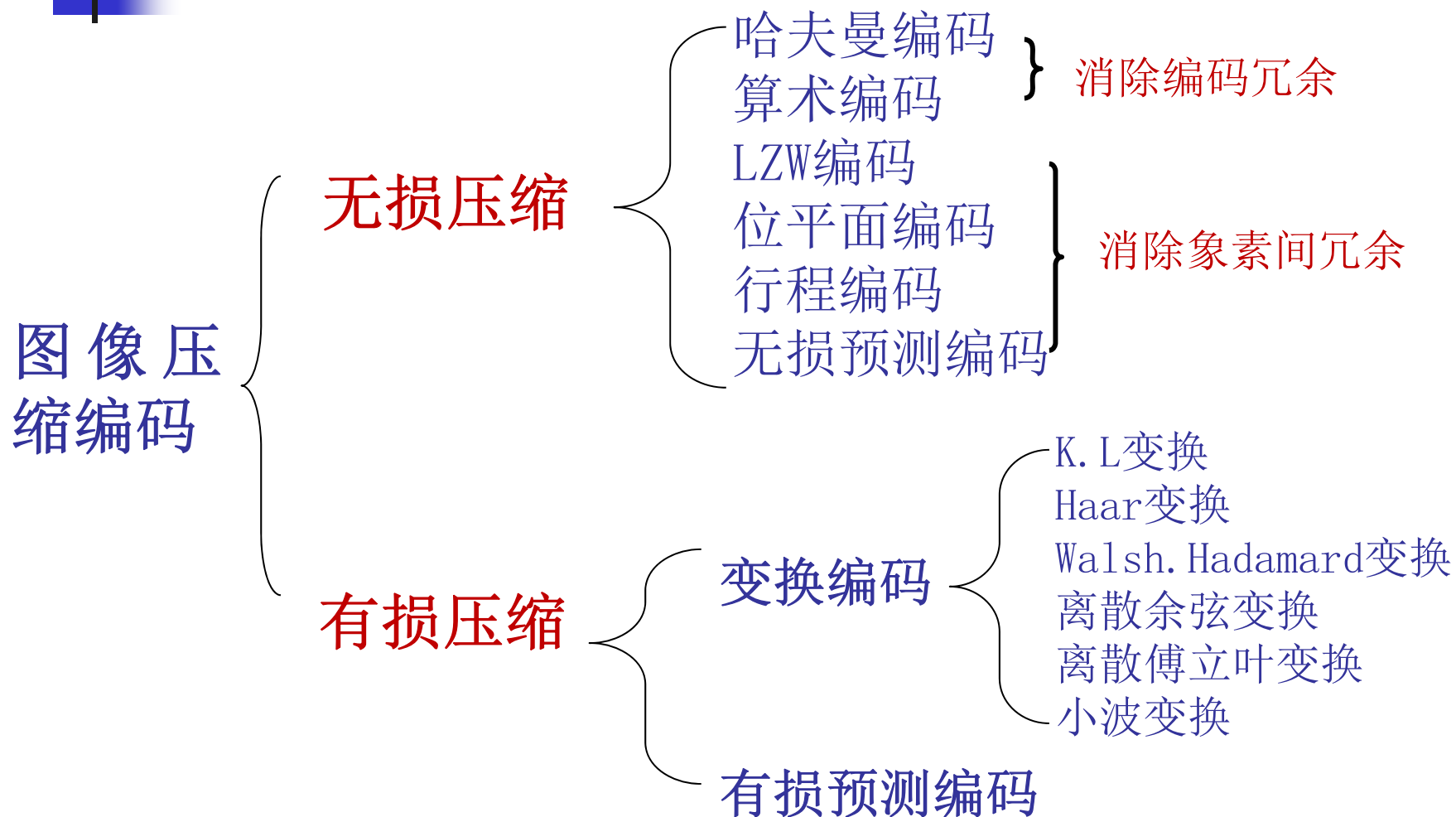




8.2 Some Basic Compression Methods



8.2 Some Basic Compression Methods



8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





1. Huffman Coding

- The most popular technique for removing *coding redundancy* is due to Huffman (1952)
- **Huffman Coding** yields the smallest number of code symbols per source symbol
- The resulting code is *optimal*
- Huffman Coding is used in CCITT, JBIG2, JPEG, MPEG-1, 2, 4, H.261, H.262, H.263, H.264

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				

1. Huffman Coding

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				

1. Huffman Coding

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								

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



课堂练习：

设有一幅灰度图像，其灰度级为5且概率如下：

1	2	3	4	5
0.1	0.2	0.42	0.16	0.12

- (1) 进行霍夫曼编码，写出编码过程。
- (2) 计算平均编码长度和压缩比。

课堂练习：

(1) 编码过程如下：（编码结果不唯一）

符号	概率	编码	概率	编码	概率	编码	概率	编码
3	0.42	1	0.42	1	0.42	1	0.58	0
2	0.2	000	0.22	01	0.36	00	0.42	1
4	0.16	001	0.2	000	0.22	01		
5	0.12	010	0.16	001				
1	0.1	011						

(2) 平均编码长度和压缩比

$$L_{avg} = \sum_{k=0}^{L-1} L(r_k) p_r(r_k) = 0.42 \times 1 + 0.58 \times 3 = 2.16 \quad c = 3/2.16 \approx 1.39$$

8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





2. Golomb Coding

- The coding of nonnegative integer inputs with exponentially decaying probability distributions. **Golomb** first proposed for the representation of nonnegative run lengths.(1966)
- Golomb codes can only be used to represent nonnegative integers and there are many Golomb codes to choose from.
- Golomb codes can be shown to be optimal and seldom used for the coding of intensities.
- Golomb Coding is used JPEG and AVS.



2. Golomb Coding

$G_m(n)$ is constructed as follows:

- Step1: Form the unary code of quotient $\lfloor n/m \rfloor$ (The unary code of an integer q is defined as q 1s followed by a 0).
- Step2: Let $k = \lceil \log_2 m \rceil$, $c = 2^k - m$, $r = n \bmod m$ and compute truncated remainder r'

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

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



2. Golomb Coding

Example: $G_4(9)$

- Step1: $\lfloor 9/4 \rfloor = \lfloor 2.25 \rfloor = 2$, the unary is **110**.
- Step2: $k = \lceil \log_2 4 \rceil = 2, c = 2^2 - 4 = 0, r = 9 \bmod 4 = 0001$
$$r' = \begin{cases} r \text{ truncated to } k-1 \text{ bits} & 0 \leq r < c \\ r+c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases} \quad r' = \text{01}$$
- Step 3: $G_4(9) = \text{11001}$

2. Golomb Coding

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

TABLE 8.5

Several Golomb codes for the integers 0 – 9.

8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





3. Arithmetic Coding

- In **arithmetic coding**(1963), an entire sequence of source symbols is assigned a single arithmetic code word.
- The code word defines an interval of real numbers between 0 and 1.
- Each symbol of the message reduces the size of the interval in accordance with its probability of occurrence.
- Arithmetic Coding is used in JBIG1, JBIG2, JPEG-2000, H.264, MPEG-4.

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	$[0.8, 1.0)$

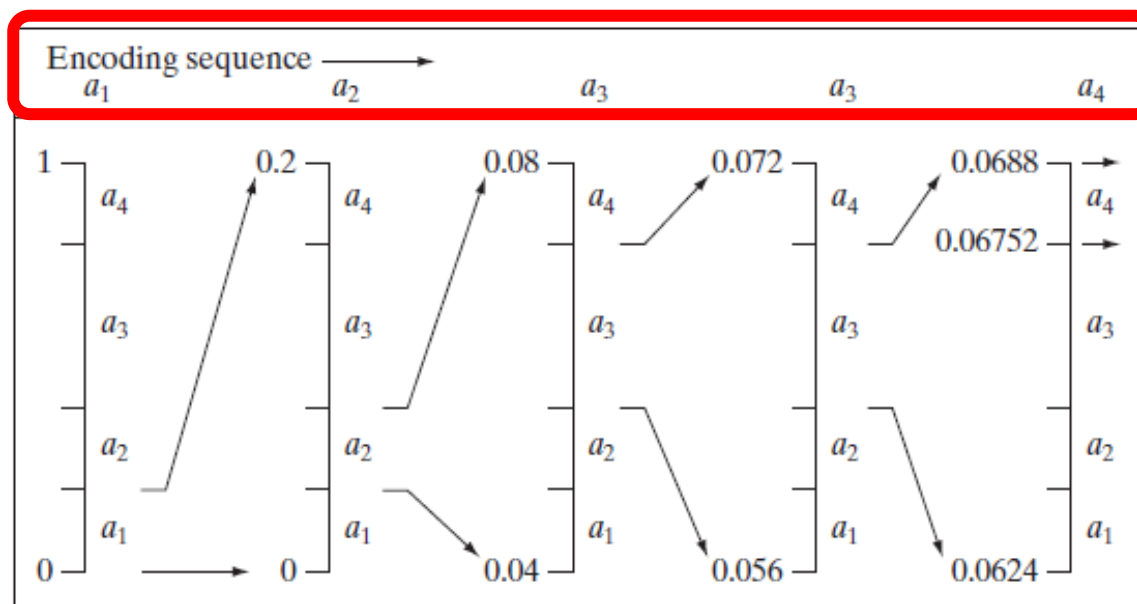


FIGURE 8.12
Arithmetic coding procedure.

$$\begin{cases} Low_N = Low_O + range_O \times rlow_N \\ High_N = Low_O + range_O \times rhigh_N \end{cases}$$



$$\begin{aligned} Low_3 &= 0.04 + (0.08 - 0.04) \times 0.4 = 0.056 \\ High_3 &= 0.04 + (0.08 - 0.04) \times 0.8 = 0.072 \end{aligned}$$

3. Arithmetic Coding

a_1	→	$[0, 0.2)$
a_2	→	$[0.04, 0.08)$
a_3	→	$[0.056, 0.072)$
a_3	→	$[0.0624, 0.0688)$
a_4	→	$[0.06752, 0.0688)$

- Any number within $[0.06752, 0.0688)$ can be used to represent the message. For example, 0.068

8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





4. LZW Coding

- **Lempel-Ziv-Welch(LZW) coding**, assigns fixed-length code words to variable length sequences of source symbols.
- A key feature of LZW coding is that it requires no a priori knowledge of the probability of occurrence of the symbols to be encoded.
- LZW coding has been integrated into a variety of mainstream imaging file formats, including GIF, TIFF PDF.



4. LZW Coding

- A codebook or a **dictionary** has to be constructed.
- For an 8-bit monochrome image, the first 256 entries are assigned to the gray levels 0,1,2,...,255.
- As the encoder examines image pixels, gray level sequences that are not in the dictionary are assigned to a new entry.
- The size of the dictionary is an important system parameter.

4. LZW Coding

■ Example

Consider the following 4x4 8 bit image

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

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

Initial Dictionary

4. LZW Coding

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

- Is 39 in the dictionary.....Yes
- What about 39-39.....No
- Then add 39-39 in entry 256
- And output the last recognized symbol...39

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

4. LZW Coding

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

TABLE 8.7
LZW coding example.

4. LZW Coding

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



39 39 126 126 256
258 260 259 257 126

$$4 \times 4 \times 8 = 128 \text{ bit}$$

$$10 \times 9 = 90 \text{ bit}$$

$$C = \frac{128}{90} = 1.42$$

8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





5. Run-Length Coding

- Images with repeating intensities along their rows (or columns) can often be compressed by **run-length coding**. (1950)
- Each **run-length pair** specifies the start of a new intensity and the number of consecutive pixels that have that intensity.
- Compression is achieved by eliminating a simple form of spatial redundancy- groups of identical intensities.
- Run-length coding is used in CCITT, JBIG2, JPEG, M-JPEG, MPEG-1,2,4, BMP

5. Run-Length Coding

Example

aaaaaabbbbbcccccccc → 7a6b8c



8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**

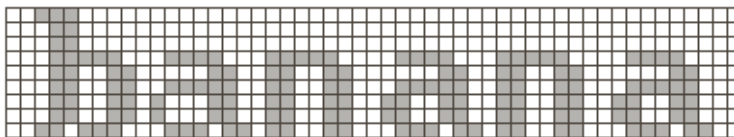




6. Symbol-Base Coding

- In **symbol-based coding**, an image is represented as a collection of frequently occurring sub-images, called **symbols**.
- Each such symbol is stored in a symbol dictionary and the image is coded as a set of triplet.
- Storing repeated symbols only once can compress images significantly- particularly in document storage and retrieval applications.

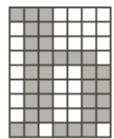
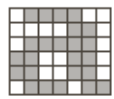

6. Symbol-Base Coding



a b c

FIGURE 8.17

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

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



7. Bit-Plane Coding

- **Bit-plane coding:** the image is decomposed into a series of binary images.

$$\textcircled{1} \quad a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \dots + a_1 2^1 + a_0 2^0$$

$$\textcircled{2} \quad g_{m-1} = a_{m-1}$$

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

- Each binary image is compressed using one of well known binary compression techniques.

7. Bit-Plane Coding

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

TABLE 8.10

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

a b
 c d
 e f
 g h

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



8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





8. Block Transform Coding

1、变换编码的基本思想

- 用一个可逆的、线性的变换(如FFT、DFT), 把图像映射到变换系数集合;
- 然后对该系数集合进行量化和编码;
- 对于大多数自然图像, 重要系数的数量是比较少的, 因而可以用量化(或完全抛弃), 且仅以较小的图像失真为代价。

8. Block Transform Coding

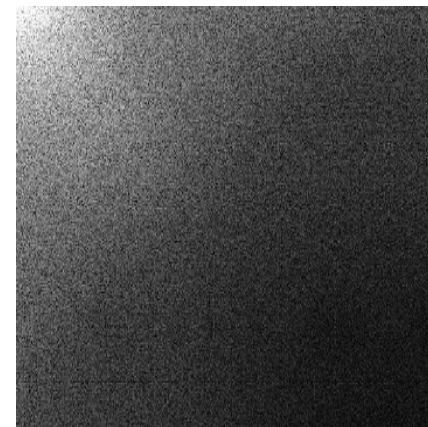
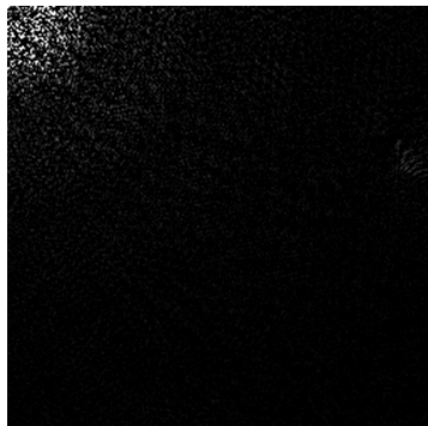
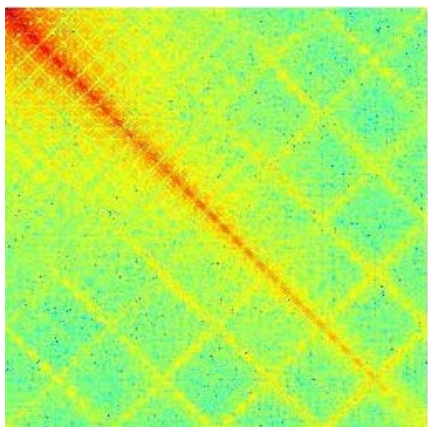
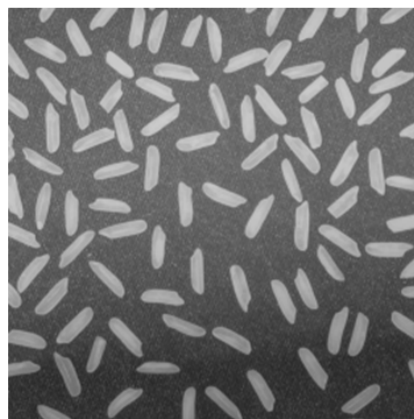
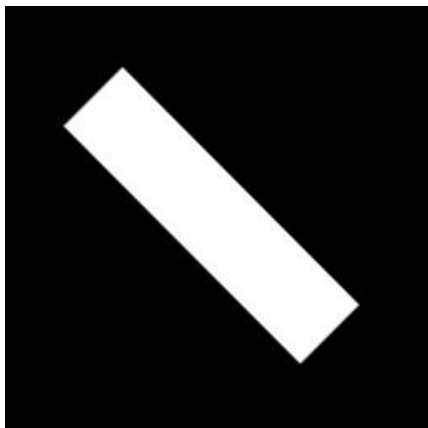
例如：

52	55	61	66	70	61	64	73	610	-29	-61	26	55	-19	0	3
63	59	66	90	109	85	69	72	7	-20	-61	9	12	-6	-6	7
62	59	68	113	144	104	66	73	-46	8	77	-25	-29	11	7	-4
63	58	71	122	154	106	70	69	-48	12	35	-14	-9	7	2	2
67	61	68	104	126	88	68	70	11	-7	-12	-2	0	2	-4	2
79	65	60	70	77	68	58	75	-9	2	4	-3	0	1	2	0
85	71	64	59	55	61	65	83	-2	-1	2	1	1	-3	2	-2
87	79	69	68	65	76	78	94	-1	0	0	-2	0	0	0	1

原图像

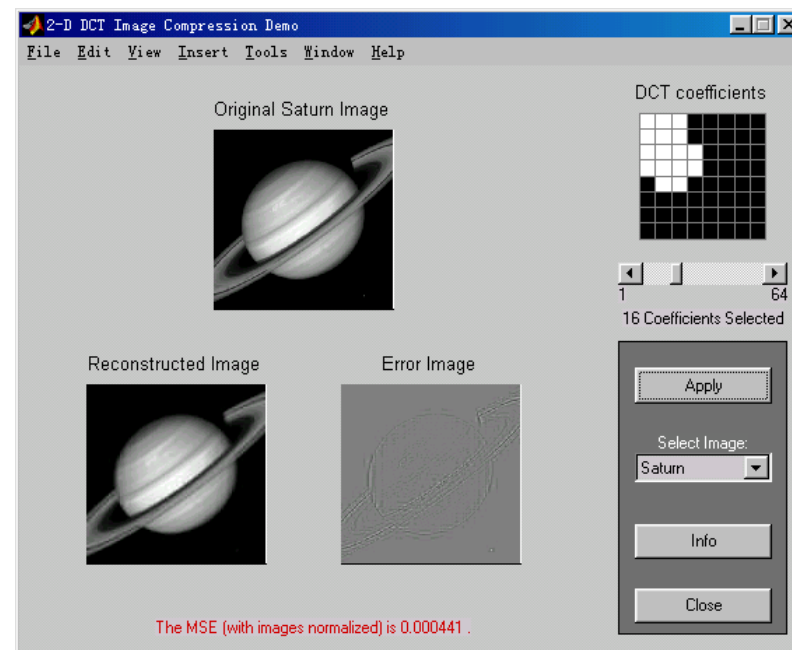
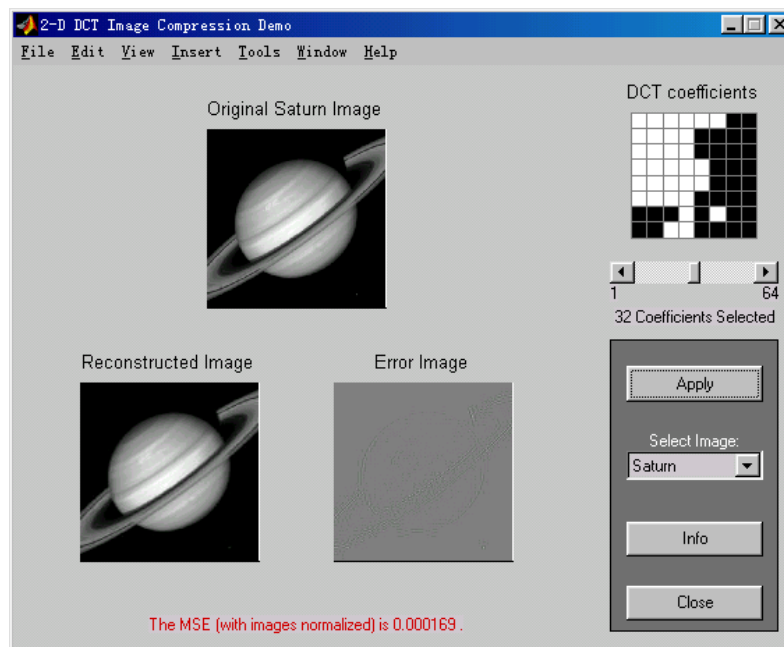
DCT变换系数

DCT



DCT编码

dctdemo.m

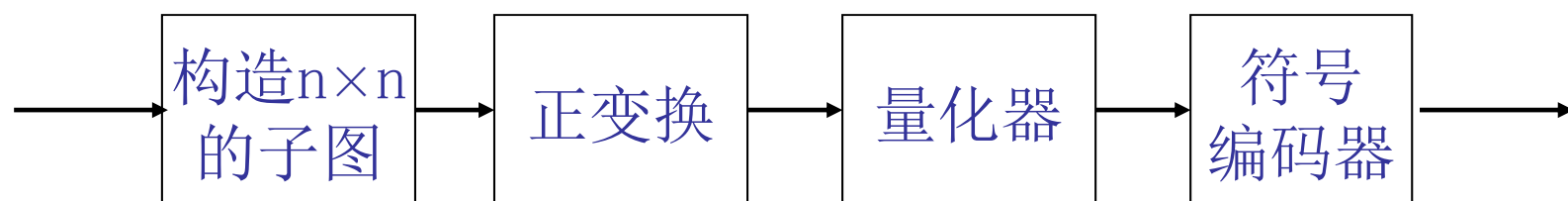


8. Block Transform Coding

■ 编码、解码流程图

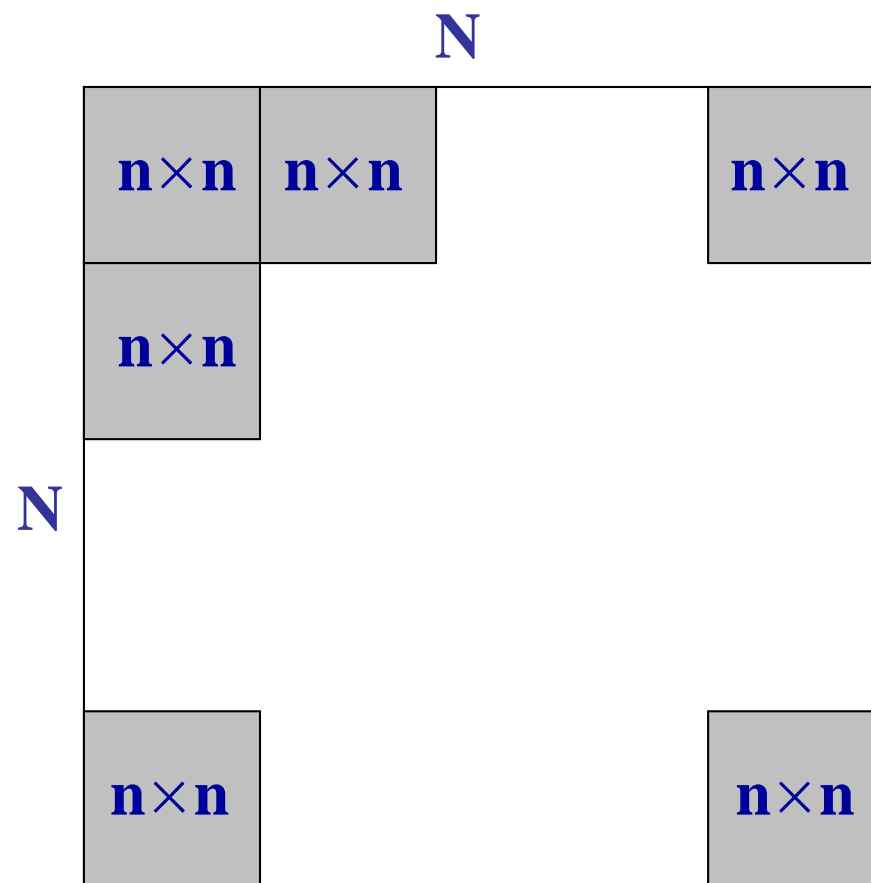
输入图像 $N \times N$

压缩图像



8. Block Transform Coding

■ 构造 $n \times n$ 的子图





8. Block Transform Coding

2、变换编码的基本原理

将FFT逆变换表达式进行改写：

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi(ux + vy) / N]$$

$F(u, v)$ 记为： $T(u, v)$

$\exp[j2\pi(ux + vy) / N]$ 记为： $H(x, y, u, v)$

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

变换编码,即要用等式的右部近似原图像。



8. Block Transform Coding

进一步改写：

$$F = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) H_{uv}$$

其中：

- 1) F 是一个包含了 $f(x, y)$ 的像素的 $n \times n$ 的矩阵；
- 2) H_{uv} 的值只依赖坐标变量 x, y, u, v ，与 $T(u, v)$ 和 $f(x, y)$ 的值无关。被称为基图像。可以在变换前一次生成，对每一个 $n \times n$ 的子图变换都可以使用。



8. Block Transform Coding

基图像 H_{uv} :

$$H_{uv} = \begin{bmatrix} h(0,0,u,v) & h(0,1,u,v) & \cdots & h(0,n-1,u,v) \\ h(1,0,u,v) & h(1,1,u,v) & \cdots & h(0,n-1,u,v) \\ \cdots & \cdots & \cdots & \cdots \\ h(n-1,0,u,v) & h(n-1,1,u,v) & \cdots & h(n-1,n-1,u,v) \end{bmatrix}$$

8. Block Transform Coding

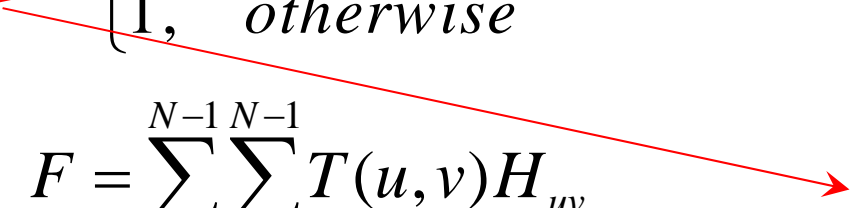
■ 变换系数截取模板函数

通过定义变换系数截取模板函数，消去冗余。

$$m(u, v) = \begin{cases} 0, & \text{如果 } T(u, v) \text{ 满足特定的截断条件} \\ 1, & \text{otherwise} \end{cases}$$

对于：
$$F = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) H_{uv}$$

近似：
$$\hat{F} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) m(u, v) H_{uv}$$



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



8. Block Transform Coding

■ 误差评估:

$$\begin{aligned} e_{ms} &= E \left\{ \left\| F - \hat{F} \right\|^2 \right\} \\ &= E \left\{ \left\| \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) H_{uv} - \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) m(u, v) H_{uv} \right\|^2 \right\} \\ &= E \left\{ \left\| \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) H_{uv} [1 - m(u, v)] \right\|^2 \right\} \\ &= \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \sigma_{T(u, v)}^2 [1 - m(u, v)] \end{aligned}$$



8. Block Transform Coding

■ 误差评估

其中， $\|F - \hat{F}\|$ 是 $(F - \hat{F})$ 的矩阵范数， $\sigma_{2T(u,v)}$ 是变换在 (u,v) 位置上的系数方差。

最后的简化是基于基图像的规范正交，并假设 F 的像素是通过一个具有0均值和已知协方差的随机处理产生的。



8. Block Transform Coding

■ 误差评估小结

- (1) 总的均方近似误差是丢弃的变换系数的方差之和（也即对于 $m(u,v) = 0$ 的系数方差之和）。
- (2) 能把大多数信息封装到最少的系数里去的变换，可得到最好的子图像的近似，同时重构误差也最小。
- (3) 在导致等式成立的假设下，一个 $N \times N$ 的图像的 $(N/n)^2$ 个子图像的均方误差是相同的。因此， $N \times N$ 图像的均方误差（是平均误差的测量）等于一个子图像的均方误差。



8. Block Transform Coding

3、变换编码——几个关键问题

- 变换的选择
- 对变换的评价
- 子图尺寸的选择
- 压缩的位分配（编码）



8. *Block Transform Coding*

■ 变换的选择

- (1) Karhunen-Loeve变换(KLT)
- (2) 离散傅立叶变换 (DFT)
- (3) 离散余弦变换 (DCT)
- (4) Walsh-Hadamard变换 (WHT)



8. Block Transform Coding

■ 对变换的评价

按信息封装能力排序：

KLT, DCT, DFT, WHT

但KLT的基图像是数据依赖的，每次都要重新计算Huv。因而很少使用。DFT的块效应严重。常用的是DCT，现已被国际标准采纳，作成芯片。其优点有：

- (1) 基本没有块效应。
- (2) 信息封装能力强，把最多的信息封装在最少的系数中。



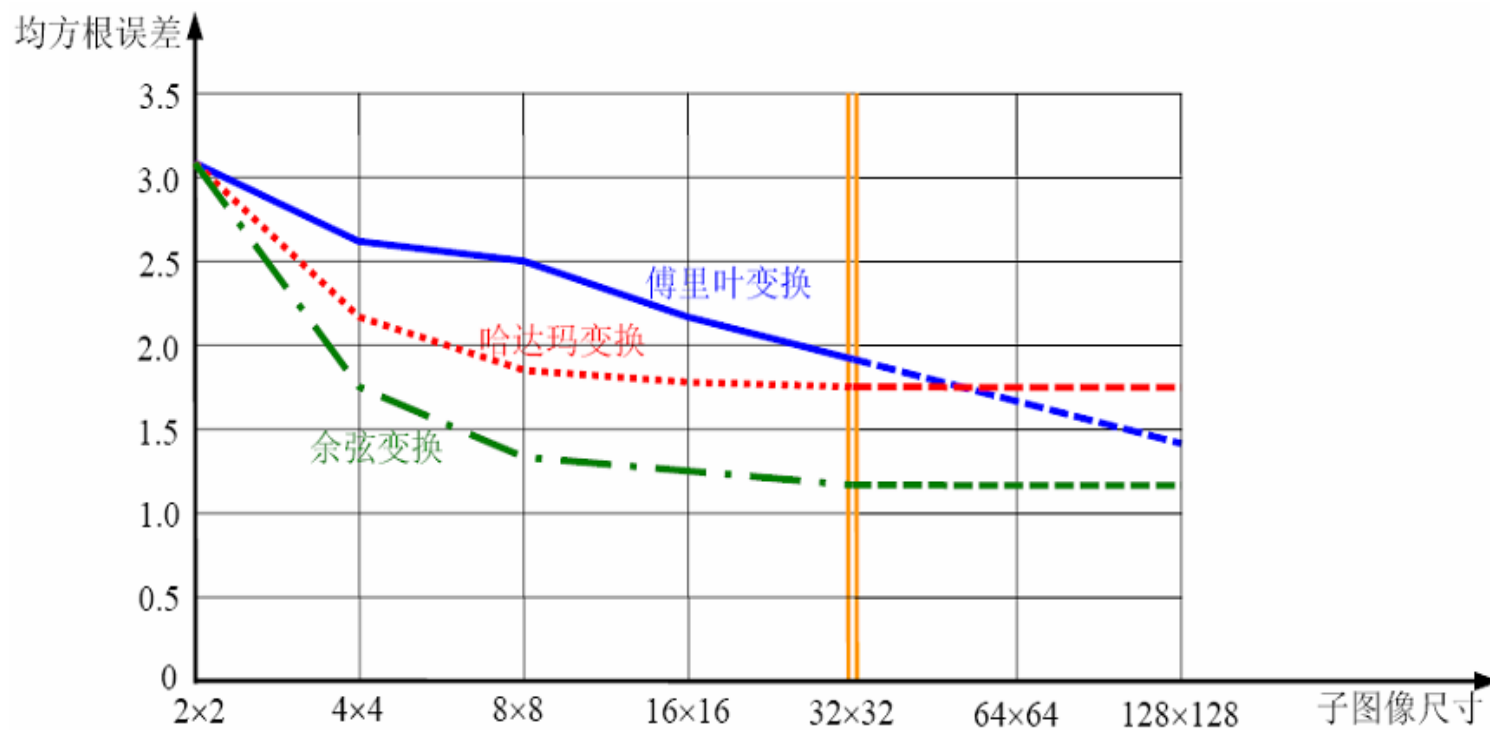
8. Block Transform Coding

■ 子图尺寸的选择

子图尺寸的选择有两个原则：

- (1) 如果 n 是子图的维数， n 应是2的整数次方。为便于降低计算复杂度。
- (2) n 一般选为 8×8 或 16×16 ，可由试验得到。
- (3) 随着 n 的增加，块效应相应减少。

8. Block Transform Coding



变换编码重建误差与子图像尺寸的关系



8. Block Transform Coding

■ 压缩位的分配（编码）

定义：截取、量化、系数编码统称为位分配。

主要解决 $m(u,v)$ 的设计、编码问题。截取和量化一般有两种方法：

- (1) 区域编码。
- (2) 阈值编码(适应性编码)。

8. Block Transform Coding

■ (1) 区域编码

基本思想:

所有子图像使用相同的编码模板。

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

可消去87.5%的系数的模板



8. Block Transform Coding

(a) 大部分的信息应该包含在最大方差的变换系数中。

- ◆ 每一个DCT变换系数被认为是一个随机变量；
- ◆ 该变量的分布可以在所有变换子图像的集合上进行计算。

(b) 在 $(N/n)^2$ 个子图找出取最大方差的 m 个系数的位置。

- ◆ 同时确定了系数的坐标 u 和 v ；
- ◆ 对所有子图像，这 m 个系数的 $T(u,v)$ 值是保留的，其他的 T 值被抛弃；
- ◆ m 是一个可选常数。



8. Block Transform Coding

■ 算法的实现:

- 〈1〉 计算模板：方差最大的地方置1，其它地方置0;
- 〈2〉 量化系数：例如最优Lloyd-Max量化器
- 〈3〉 结果编码：有两种分配二进制位的编码方法：
 - ① 系数被赋予相同数量的二进制位。
 - ② 系数之间固定地分配一定的二进制位。



8. Block Transform Coding

例如：

系数之间固定地分配一定的二进制位的用位模板。

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



8. Block Transform Coding

■ (2) 阈值编码(适应性编码)

基本思想：没有一个取舍系数的固定模板。不同的子图保留不同的系数。通过一个阈值T，来决定一个系数的去留。

If $a(\text{系数}) > T(\text{阈值})$ $m(u,v) = 1$
else $m(u,v) = 0$

由于其简单性，阈值编码是实际应用中经常使用的编码方法。



8. Block Transform Coding

■ 理论根据：

- (1) 取值最大的变换系数，在重构子图的质量中起的作用也最重要。
- (2) 最大系数的分布随子图的不同而不同。

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



8. Block Transform Coding

■ 算法的实现:

1. 阈值的选取，常有三种取法。

(1) 所有子图使用同一个全局阈值。压缩率的大小随图像的不同而不同。由超过全局阈值的系数的个数所决定。

(2) 对每个子图使用不同的阈值。每个子图保留的系数的个数事先确定，即总保留N个最大的。称为N-最大化编码。对于每个子图同样多的系数被丢弃。因此，每个子图的压缩率是相同的，并且是预先知道的。



8. Block Transform Coding

(3) 阈值作为子图系数位置的函数。所有子图使用同一个全局阈值模板，但阈值的选取，与系数的位置相关，阈值模板给出了不同位置上系数的相应阈值。例如：

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

8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**





9. Predictive Coding

无损预测编码(Lossless Predictive Coding)

(1) 基本思想

- 去除像素冗余。
- 认为相邻像素的信息有冗余。当前像素值可以用先前的像素值来获得。
- 用当前像素值 f_n ，通过预测器得到一个预测值 \hat{f}_n ，对当前值和预测值求差。对差编码，作为压缩数据流中的下一个元素。

无损预测编码(Lossless Predictive Coding)

■ 例：利用前一个像素预测

100	102	101	100	100
101	100	102	102	100
100	103	100	102	101
100	100	100	102	100
101	101	100	100	100

100	2	-1	-1	0
1	-1	2	0	-2
-1	3	-3	2	-1
0	0	0	2	-2
1	0	-1	0	0



无损预测编码(Lossless Predictive Coding)

- 一般情况下，利用前m个像素的线性组合来预测，即

$$f_n = \text{round} \left[\sum_{i=1}^m \alpha_i f_{n-i} \right]$$

因此，在一维线性(行预测)预测编码中，预测器为：

$$\hat{f}_n(x, y) = \text{round} \left[\sum_{i=1}^m \alpha_i f(x, y-i) \right]$$

其中，*round*为取最近整数， α_i 为预测系数(可为1/m)，y是行变量。

注意：前m个像素不能用此法编码，可用哈夫曼编码。

无损预测编码(Lossless Predictive Coding)

例析: $F = \{154, 159, 151, 149, 139, 121, 112, 109, 129\}$

取 $m = 2, \alpha = 1/2$

预测值:

$$\begin{aligned} f_2 &= 1/2 \times (154 + 159) \approx 156, & e_2 &= 151 - 156 = -5 \\ f_3 &= 1/2 \times (159 + 151) = 155, & e_3 &= 149 - 155 = -6 \\ f_4 &= 1/2 \times (151 + 149) = 150, & e_4 &= 139 - 150 = -11 \\ f_5 &= 1/2 \times (149 + 139) = 144, & e_5 &= 121 - 144 = -23 \\ f_6 &= 1/2 \times (139 + 121) = 130, & e_6 &= 112 - 130 = -18 \\ f_7 &= 1/2 \times (121 + 112) \approx 116, & e_7 &= 109 - 116 = -7 \\ f_8 &= 1/2 \times (112 + 109) \approx 110, & e_8 &= 129 - 110 = 19 \end{aligned}$$



无损预测编码(Lossless Predictive Coding)

(2) 编码步骤:

第①步: 压缩头处理。

第②步: 预测值。

第③步: 求预测误差值。

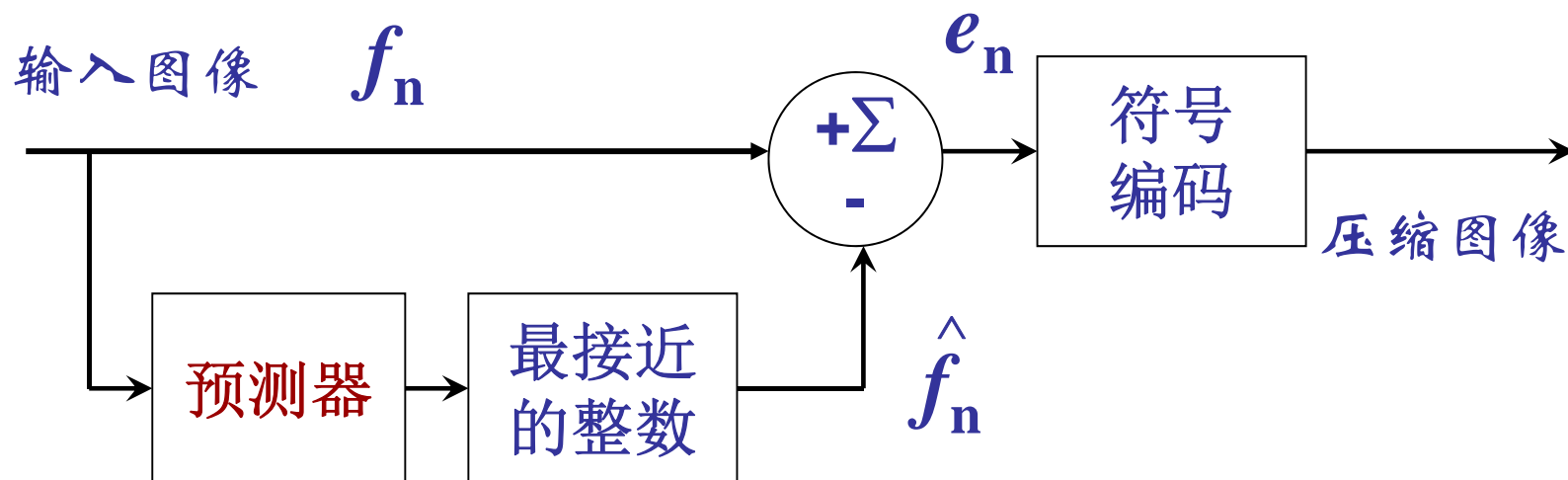
$$e(x, y) = f(x, y) - \hat{f}(x, y)$$

第④步: 对误差 $e(x, y)$ 编码, 作为压缩值。

重复②、③、④步

无损预测编码(Lossless Predictive Coding)

(3) 编码原理图





9. Predictive Coding

有损预测编码(Lossy Predictive Coding)

□ 基本概念

□ 量化器

□ 编码原理

有损预测编码(Lossy Predictive Coding)

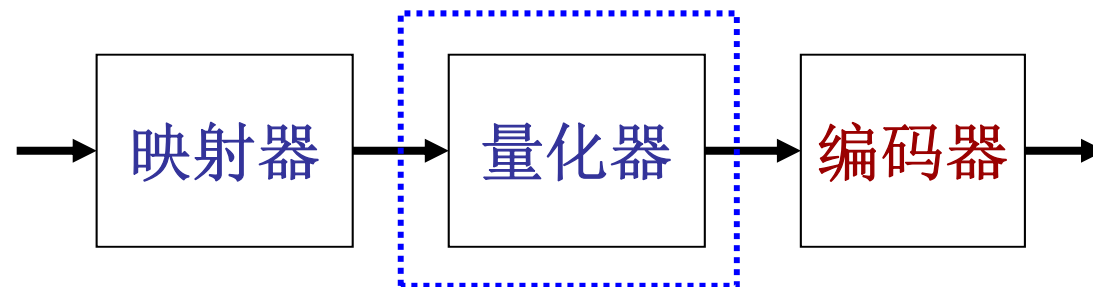
(1) 有损压缩的基本概念

- 有损压缩是：
 - 通过牺牲图像的准确率来达到增大压缩率的目的。
 - 如果容忍解压后的结果中有一定的误差，那么压缩率可以显著提高。
- 有损压缩方法的压缩比：
 - 在图像压缩比大于30:1时，仍然能够重构图像。
 - 在图像压缩比为10:1到20:1时，重构图像与原图几乎没有差别。
 - 无损压缩的压缩比很少有能超过3:1的。
- 有损与无损压缩的根本差别在于有没有量化器模块。

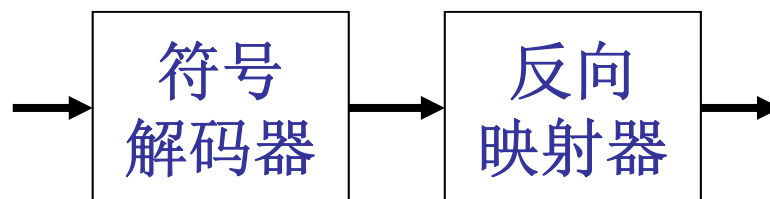
有损预测编码(Lossy Predictive Coding)

数据源编、解码的一般模型——

■ 编码模型



■ 解码模型





有损预测编码(Lossy Predictive Coding)

(2) 量化器:

- 减少数据量的最简单的办法是将图像量化成较少的灰度级，通过减少图像的灰度级来实现图像的压缩；
- 这种量化是不可逆的，因而解码时图像有损失。

例如： 如果输入是256个灰度级，对灰度级量化后输出，只剩下4个层次，数据量被大大减少。

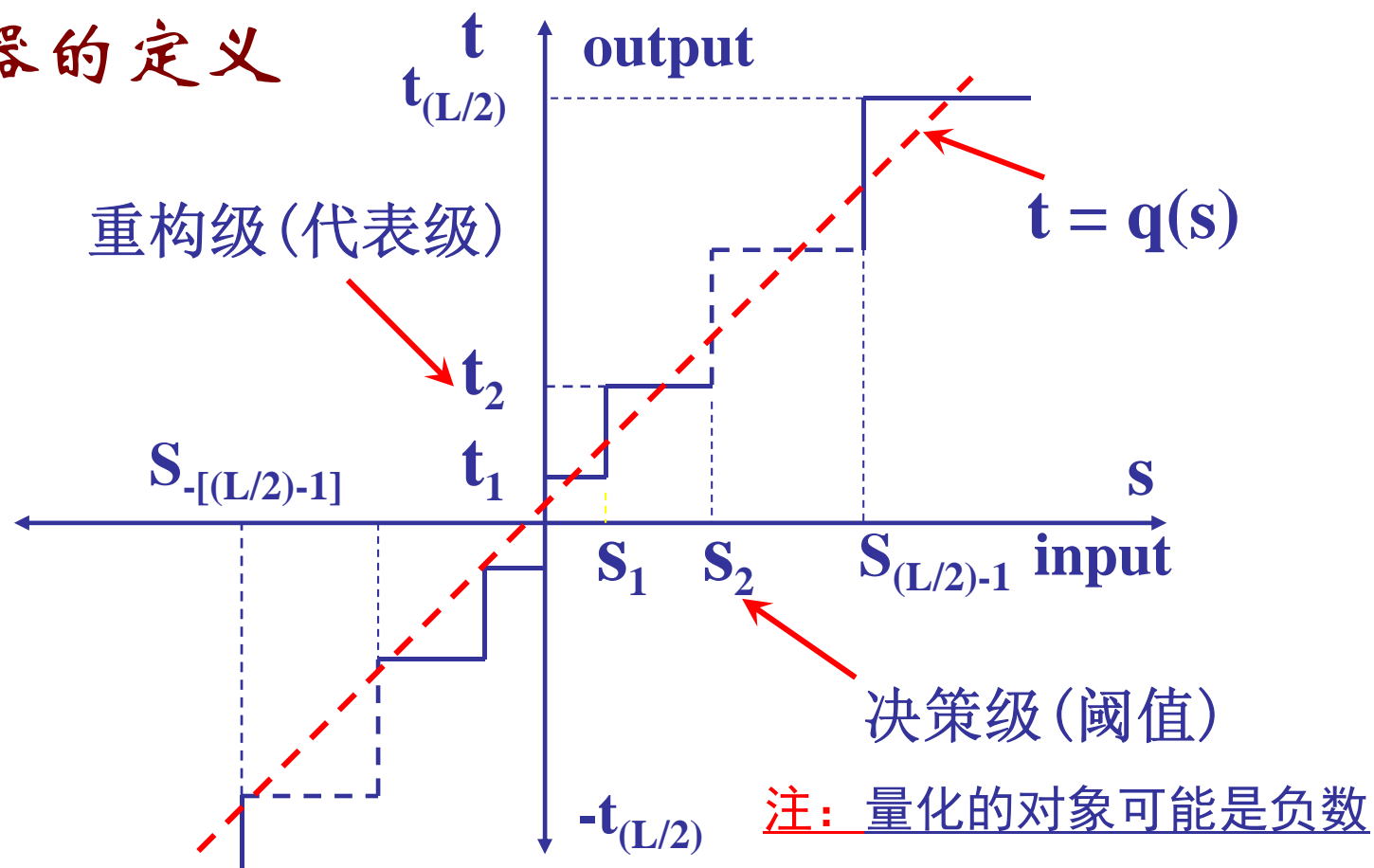
有损预测编码(Lossy Predictive Coding)

■ 量化器的定义

- 阶梯形量化函数 $t=q(s)$ ，是一个 s 的奇函数（即 $q(-s)=-q(s)$ ），它可以通过 $L/2$ 、 s_i 和 t_i 来完全描述，从而定义了一个量化器。
- s_i 被称为量化器的决策级（阈值）；
 t_i 被称为量化器的重构级（代表级）。
- L ：是量化器的级数。
- 由于习惯的原因， s_i 被认为是映射到 t_i ，如果它在半开区间 $(s_i, s_{i+1}]$ 。

有损预测编码(Lossy Predictive Coding)

■ 量化器的定义





有损预测编码(Lossy Predictive Coding)

(3)无损到有损——算法演变

基本思想：

对无损预测压缩的误差进行量化，通过消除视觉心理冗余，达到对图像进一步压缩的目的。

——引入量化(Quantification)

有损预测编码(Lossy Predictive Coding)

■ 编、解码原理及过程

➤ 将 e_n 量化: $\hat{e}_n = Q(e_n)$

➤ 用: $f'_n = \hat{e}_n + \hat{f}_n$

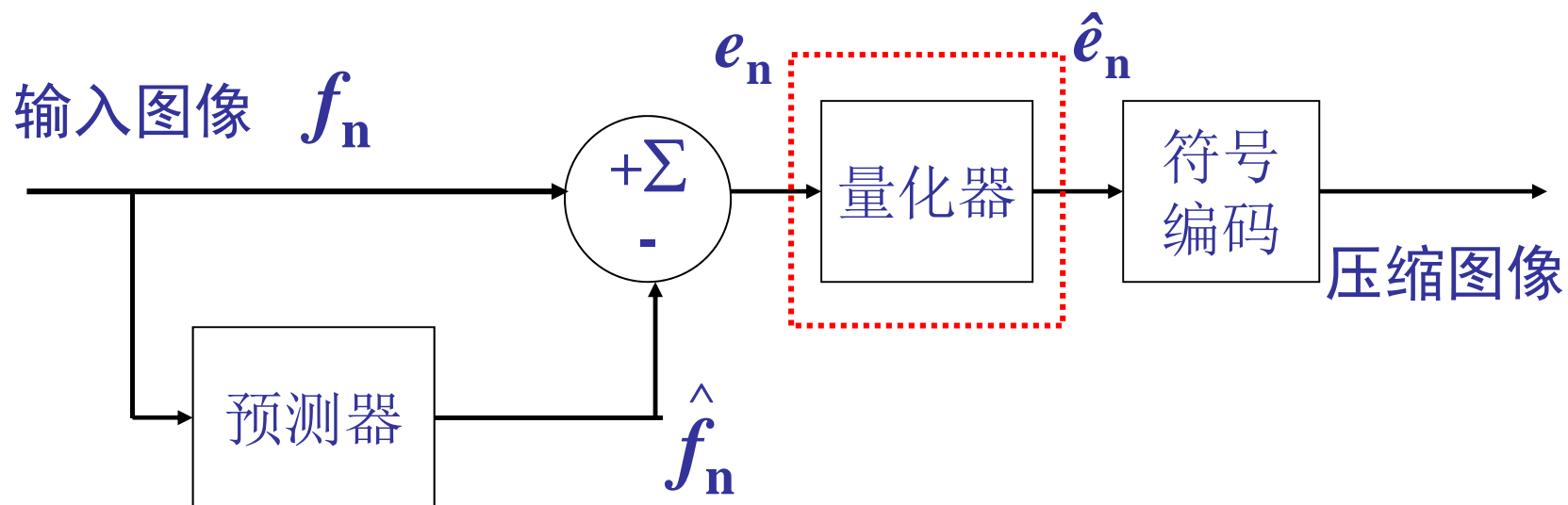
➤ 近似 $f_n \approx f'_n$

➤ 编码: $\hat{e}_n = Q(f_n - \hat{f}_n)$

➤ 解码: $f'_n = \hat{e}_n + \hat{f}_n$

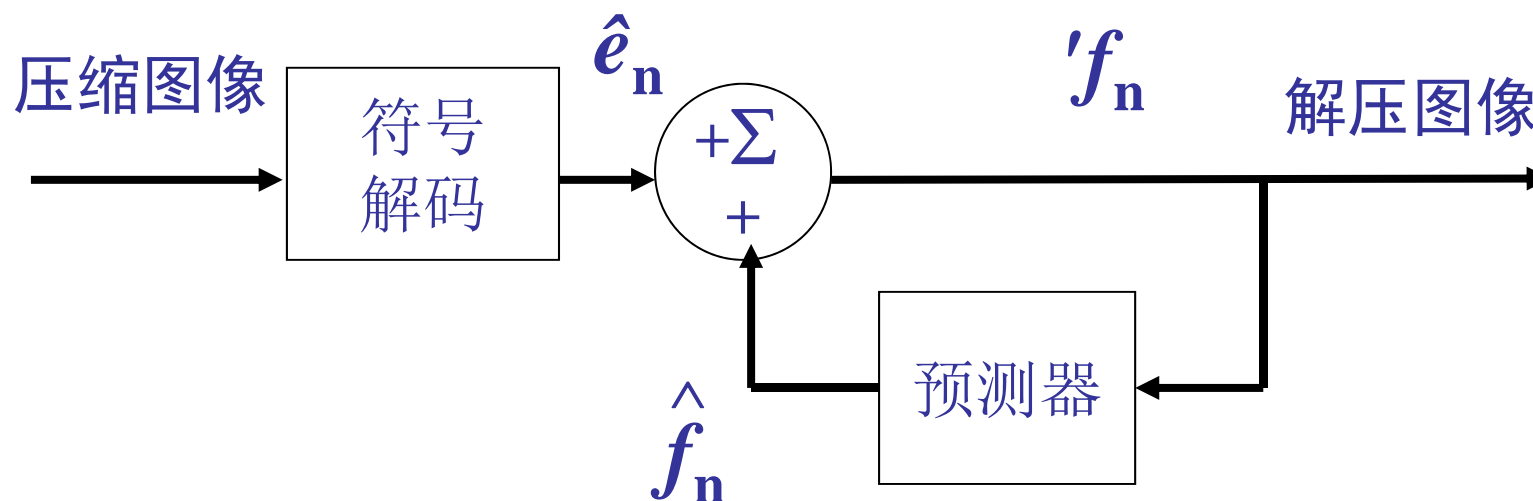
有损预测编码(Lossy Predictive Coding)

■ 编码流程图: $\hat{e}_n = Q(f_n - \hat{f}_n)$



有损预测编码(Lossy Predictive Coding)

- 解码流程图: $'f_n = \hat{e}_n + \hat{f}_n$



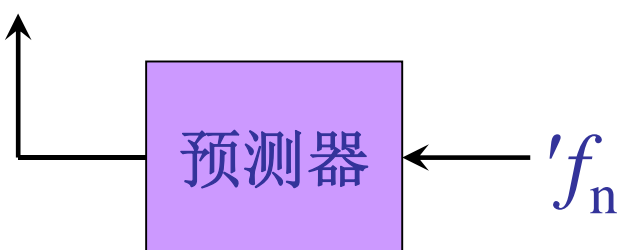
有损预测编码(Lossy Predictive Coding)

注意：上述方案的压缩编码中，预测器的输入是 f_n ，而解压中的预测器的输入是 f'_n ，要使用相同的预测器，编码方案要进行修改。

编码

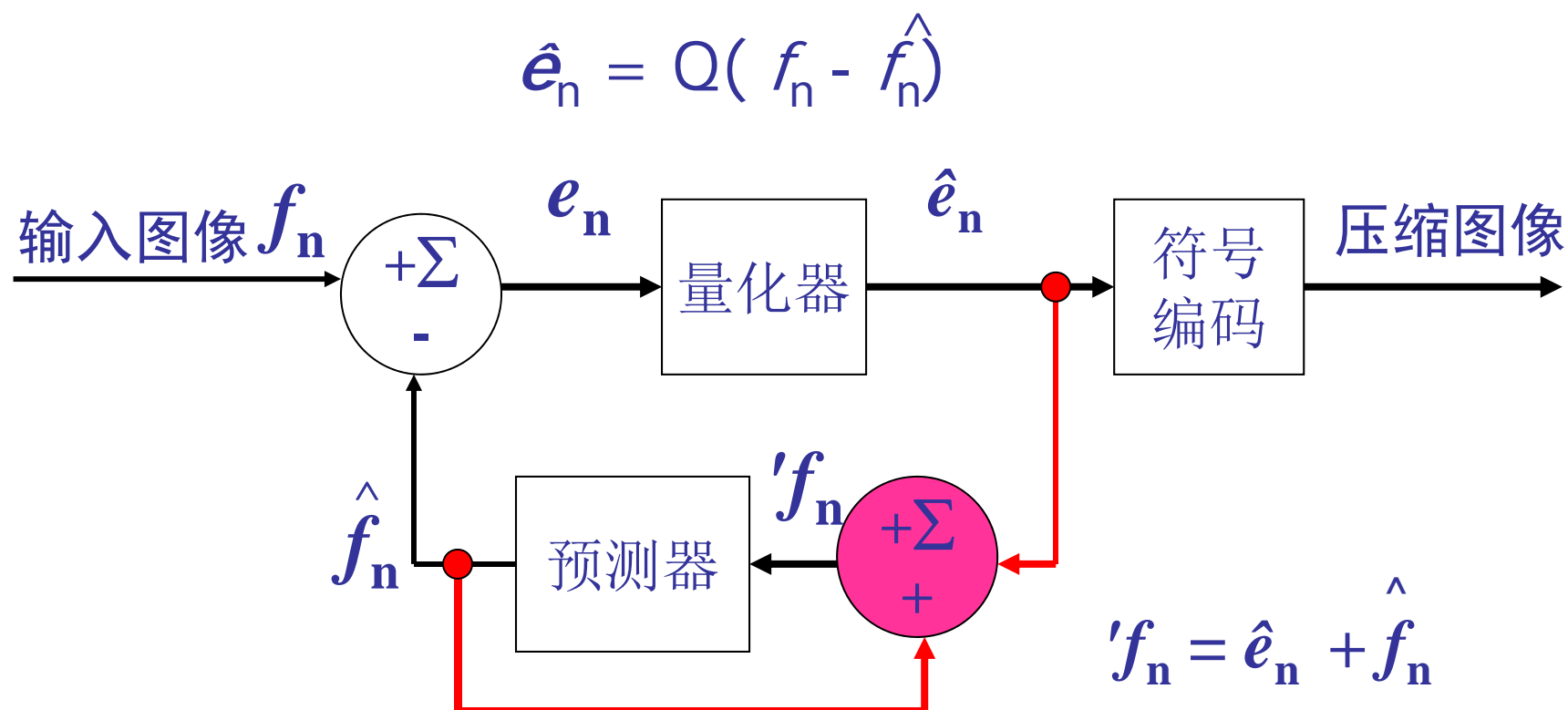


解码



有损预测编码(Lossy Predictive Coding)

■ 修改后的有损预测编码





有损预测编码(Lossy Predictive Coding)

■ DPCM 简介

差分脉冲编码调制(Differential Pulse Code Modulation, DPCM), 采用反馈方法预测估值。

1950年, 卡特勒申请专利;

1952年, 获得批准;

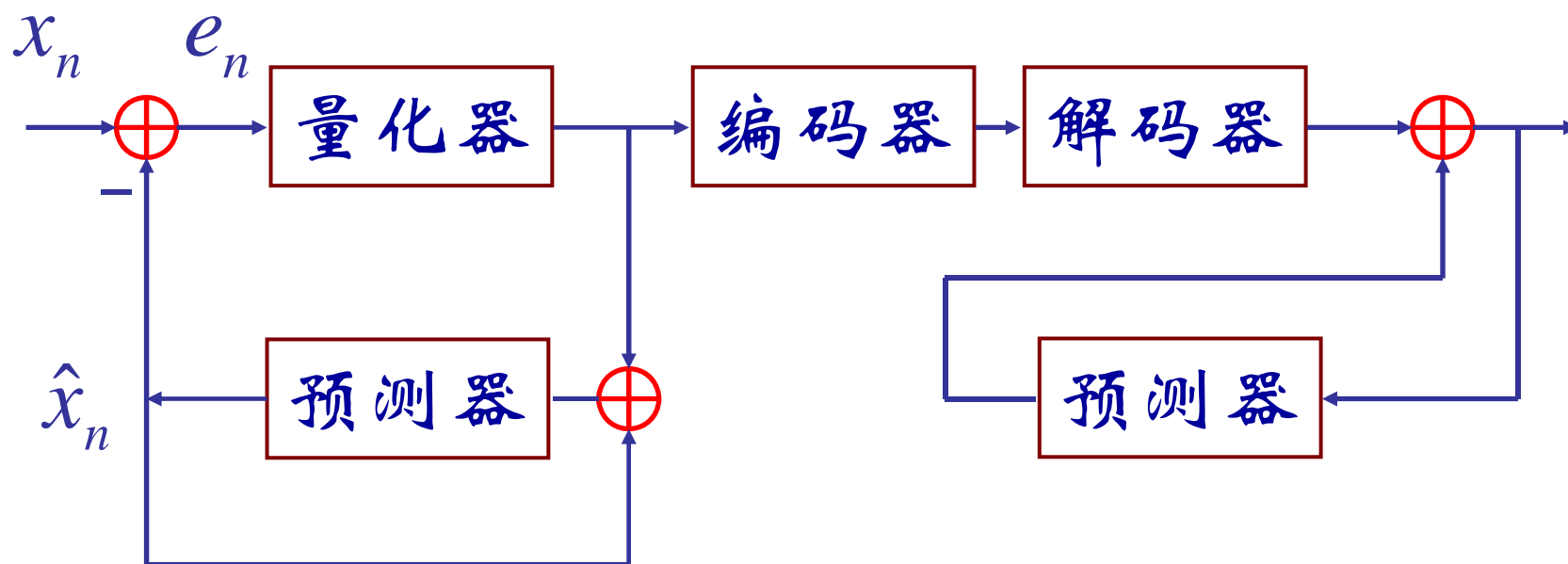
1958年, 格雷厄姆 (Graham) 开始计算机模拟研究编码方法;

1966年, 奥尼尔 (O'Neal) 对电视信号预测编码进行理论分析以及计算机模拟;

1971年投入使用。

有损预测编码(Lossy Predictive Coding)

■ 编码原理图



有损预测编码(Lossy Predictive Coding)

■ 量化器:

$$e'_n = \begin{cases} +\zeta, & e_n > 0, \\ -\zeta, & \text{else,} \end{cases} \quad \begin{array}{l} \zeta \text{ 是一个正常数} \\ e'_n \text{ 用 1 位编码} \end{array}$$

■ 预测器:

$$\hat{f}_n = \alpha f'_{n-1}$$

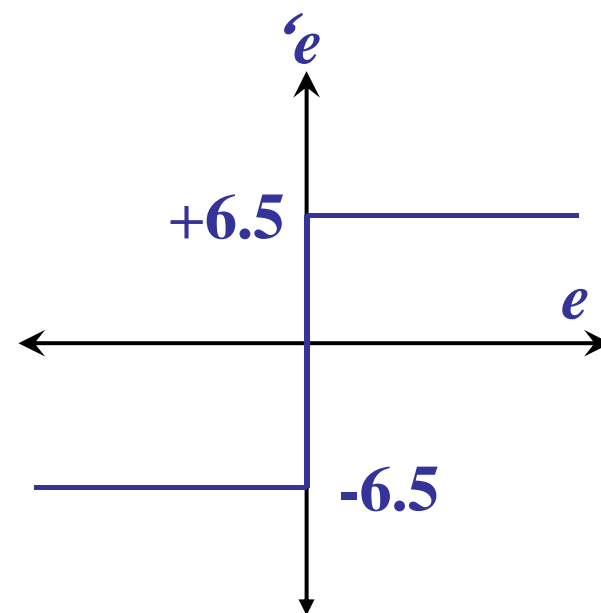
α 一般是一个小于1的预测系数。

有损预测编码(Lossy Predictive Coding)

DPCM:

例如：量化器

设： $\zeta = 6.5$



有损预测编码(Lossy Predictive Coding)

DPCM: ■ 举例: $\alpha = 1$, $\zeta = 6.5$

计算: 两个像素 $f_0 = 14$ 、 $f_1 = 15$, 则:

$$n = 0, \hat{f}_0 = f_0 = 14$$

$$n = 1, \hat{f}_1 = \alpha f_0 = 1 \times 14$$

—————→ (预测结果)

编码: $e_1 = 15 - 14 = 1$

—————→ (预测误差)

$$e'_1 = +6.5 (\because e_1 > 0)$$

—————→ (预测误差)

解码: $f'_1 = e'_1 + \hat{f}_1 = 6.5 + 14 = 20.5$

—————→ (重构结果)

$$f_1 - f'_1 = (15 - 20.5) = -5.5$$

—————→ (重构误差)

有损预测编码(Lossy Predictive Coding)

DPCM:

输入		编码				解码		误差
n	f	\hat{f}	e	'e	'f	\hat{f}	'f	f-'f
0	14	-	-	-	14.0	-	14.0	0.0
1	15	<u>14.0</u>	1.0	<u>6.5</u>	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

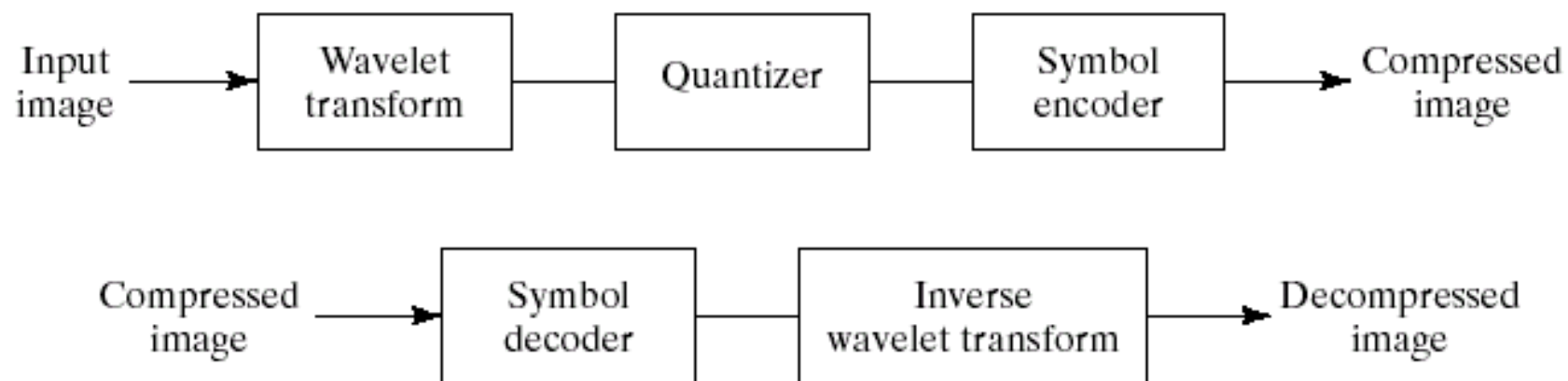
8.2 Some Basic Compression Methods

Agenda

-  **Huffman Coding** 
-  **Golomb Coding**
-  **Arithmetic Coding** 
-  **LZW Coding** 
-  **Run-Length Coding** 
-  **Symbol-Based Coding**
-  **Bit-Plane Coding**
-  **Block Transform Coding** 
-  **Predictive Coding** 
-  **Wavelet Coding**



10. Wavelet Coding



10. Wavelet Coding



a b
 c d

FIGURE 8.46

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

10. Wavelet Coding



JPEG-DCT



**JPEG2000-
Wavelet**



Outline

- ◆ **Background**
- ◆ **Fundamentals**
- ◆ **Some Basic Compression Methods**
- ◆ **Digital Image Watermarking***





制定图像和视频标准的国际组织

■ 制定图像标准的国际组织：

- ISO（国际标准化组织）
- CCITT（国际电报电话咨询委员会）
→ 联合组织下进行制定的

■ 制定视频标准的国际组织：

- ISO（国际标准化组织）
- IEC（国际电工委员会）
- ITU（国际电信联盟）
→ 分别或联合组织下进行制定的



图像压缩标准

◆ JPEG——静态图像压缩标准

Joint Photographic Experts Group(联合图像专家组)

◆ JPEG2000——新一代静态图像压缩标准

适用范围:

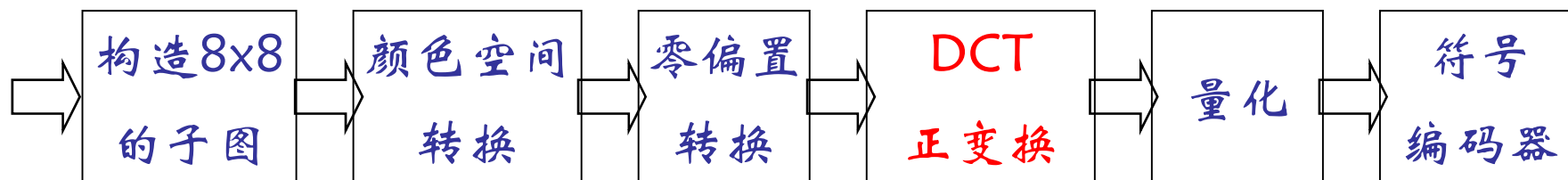
- 灰度图像，彩色图像
- 静止图像的压缩，视频序列帧内图像压缩

JPEG压缩标准(ISO 10918-1)

JPEG压缩流程

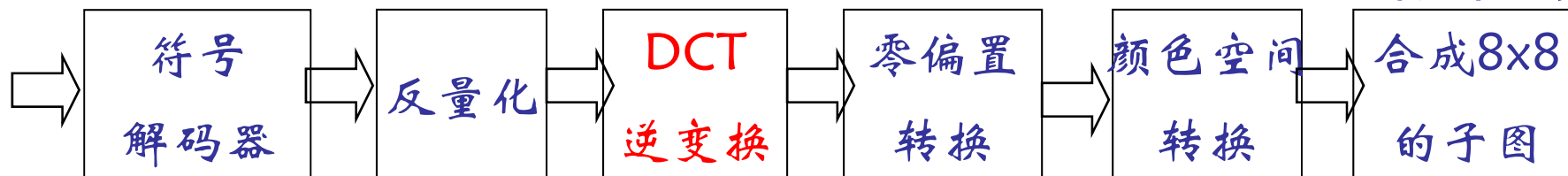
输入图像 $N \times N$

压缩图像



压缩的图像

解压图像





JPEG标准的划分

- 4种编码模式：
 - lossless encoding mode
 - DCT_based sequential encoding
 - DCT_based progressive encoding
 - DCT_based hierarchical encoding
- 3种技术层次(按算法的复杂性):
 - 基本系统 (Baseline System)
 - 扩展系统 (Extended System, 提供二进制算术编码)
 - 专用无损失系统 (Independent System)

JPEG

DCT_based progressive encoding





JPEG2000 压缩标准 (ISO 15444)

- 核心技术是离散小波变换(DWT)
- 高压缩率
- 同时支持有损和无损压缩
- 实现了渐进传输
- 支持“感兴趣区域”压缩

JPEG2000与JPEG对比

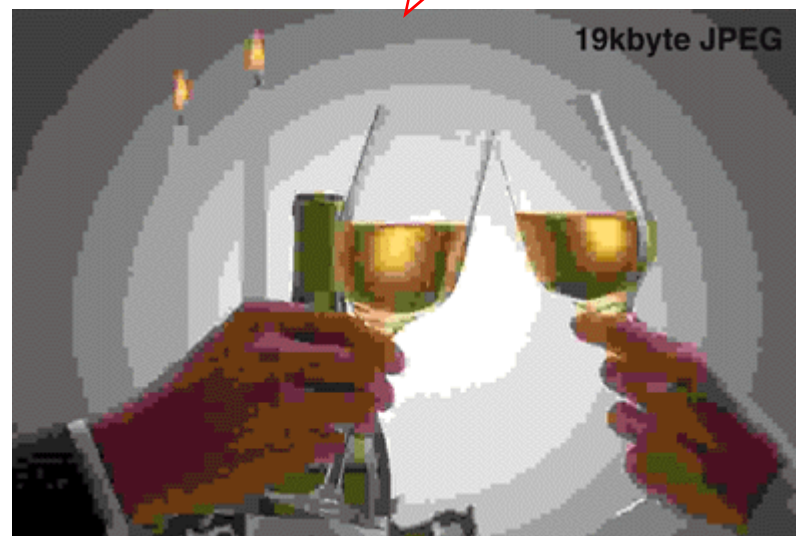


3MB的原始画面

19kB的JPEG2000



19kB的JPEG画面



JPEG2000与JPEG对比

(Bitrate=0.125bpp)



JPEG2000



JPEG

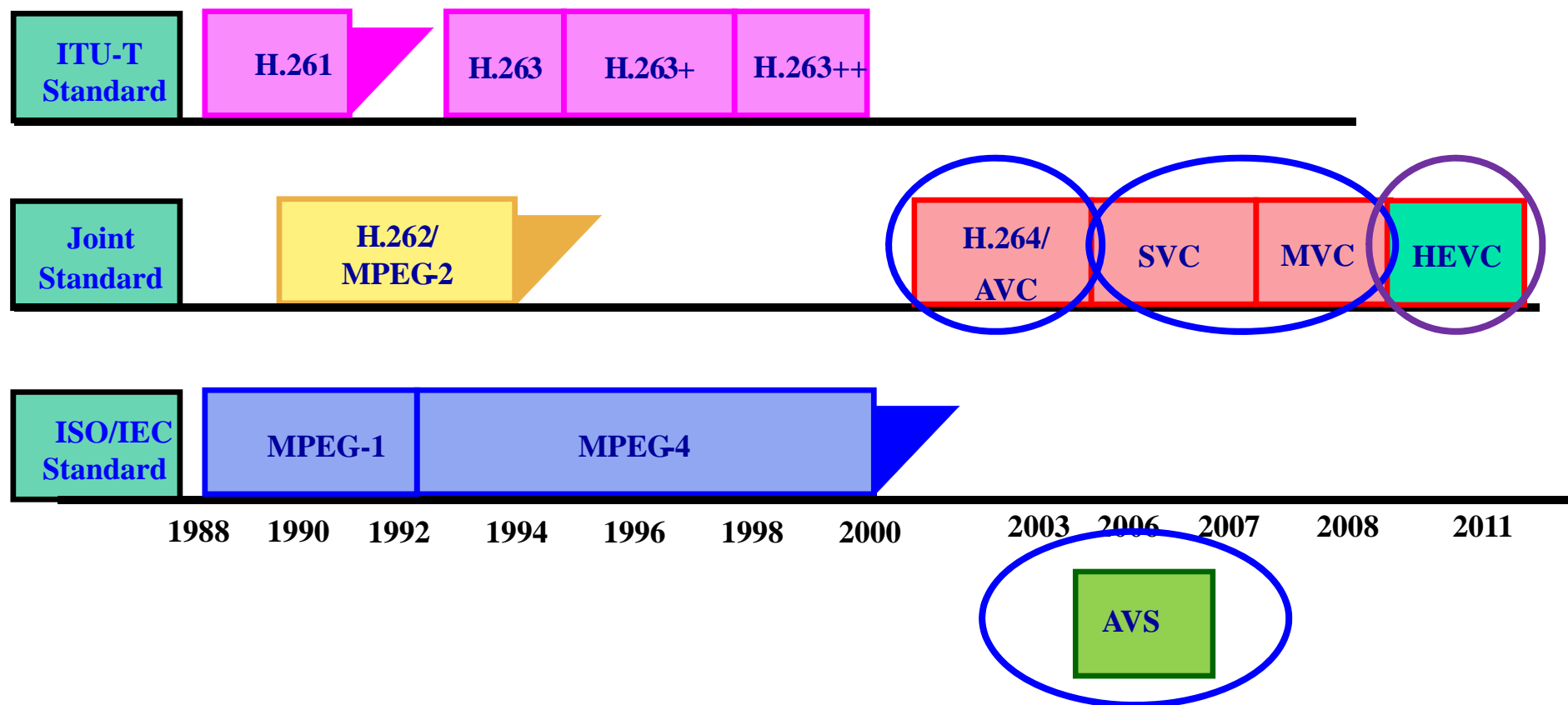
视频压缩

■ 视频的定义

- 由多幅尺寸相同的静止图像组成的序列。
- 与静止图像相比，视频多了一个时间轴，成为三维信号。



视频编解码标准



视频压缩标准对比

	颁布日期	技术特点	应用场合
H.261	1991	8×8DCT; 16×16的运动补偿块; 整像素补偿精度; 支持前向预测; 采用环路滤波器	ISDN(综合业务数字网)视频会议
H.263 H.263+ H.263++	1995 1998 2000	在H.261基础上, 支持1/2像素的补偿精度; 支持8×8运动补偿块; 支持双向预测;	因特网的视频会议、可视电话业务等
MPEG-1	1992	8×8 DCT; 16×16运动补偿块; 1/2像素补偿精度; 支持双向预测; 无环路滤波器	VCD、家用视频、视频监控等
MPEG-2	1994	在MPEG-1基础上, 支持16x8的运动补偿块; 支持隔行视频编码; 采用帧预测和场预测及场DCT等技术; 支持可分级性工具。	数字电视、DVD、数字视频存储、宽带视频会议等
MPEG-4	1999	基于对象的编码技术; 8×8DCT; 1/4像素的补偿精度; 支持双向预测; 支持全局运动补偿; 变长编码/算术编码/SPRITE编码;	因特网视频、标准交互式视频、专业视频、移动通信等

视频压缩标准对比(续)

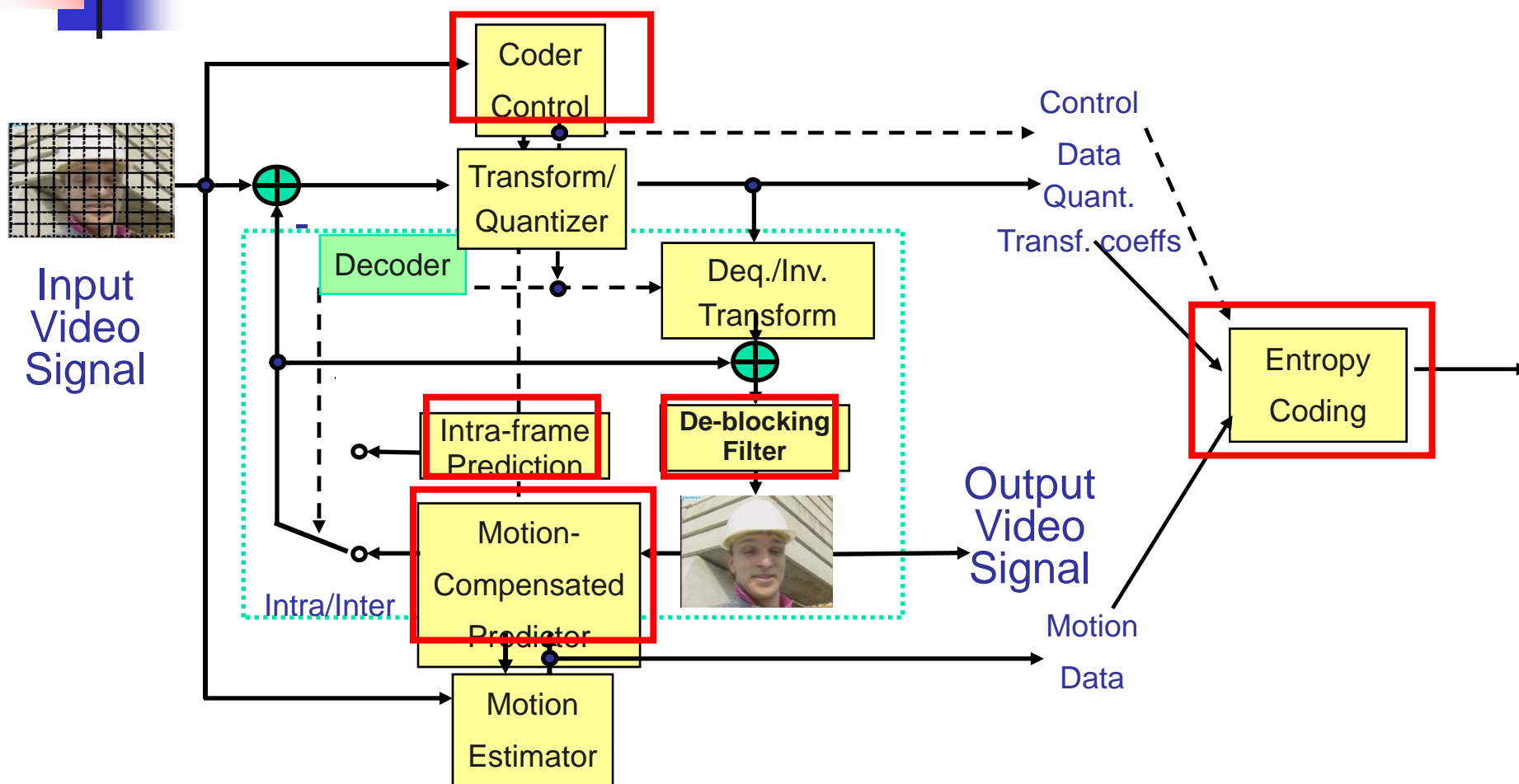
H. 264/AVC	2003	采用7种可变块大小；支持多参考帧；支持1/4像素的补偿精度；支持4×4的整数DCT；支持双向预测编码模式；支持CAVLC/CABAC；支持环内滤波和加权预测	视频会议、因特网视频、数字电视广播、高清电视、移动视频、视频点播、视频监控等等
SVC	2007	支持时间、空间和质量可分级技术	
MVC	2008	支持多视点编码	
AVS	2006	采用4种可变尺寸的块大小；支持2个参考帧；支持1/4像素的补偿精度；支持8×8的整数DCT；支持双向预测编码模式；支持CAVLC/CABAC；支持环内滤波和加权预测	数字电视广播、视频会议、视频存储、高清电视、移动视频、视频点播、视频监控等等



H.264/AVC标准

2001年12月，ITU的视频编编码专家组（Video Coding Experts Group，简称VCEG）与ISO/IEC的MPEG(Motion Picture Experts Group)组成联合视频组（Joint Video Team，简称JVT），于2003年4月最终制定了H.264/AVC编码标准，其在ITU系列中称为H.264，在MPEG系列中称为MPEG-4第10部分高级视频编码模式（Advanced Video Coding，简称AVC）。

H.264/AVC编解码器的基本架构

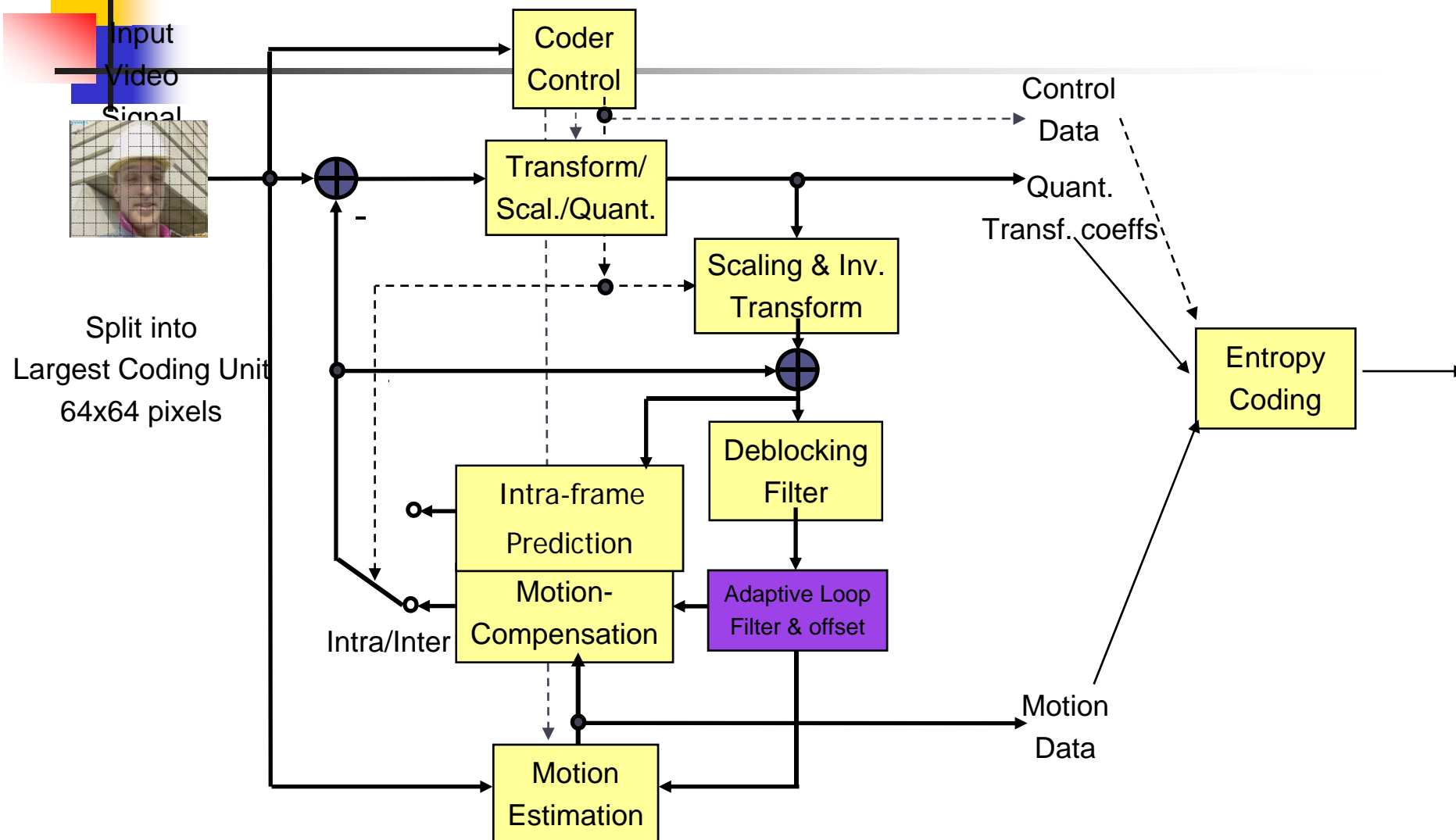




视频编解码技术的下一步发展趋势探讨

- **HEVC (High Efficiency Video Coding)**
 - 随着网络技术和终端处理能力的不断提高，人们对目前广泛使用的MPEG-2, MPEG-4, H. 264等，提出了新的要求。希望能够提供：1) 高清，2) 3D，3) 移动无线，以满足新的家庭影院、远程监控、数字广播、移动流媒体、便携摄像、医学成像等心领域的应用。
 - 新一代视频压缩标准的核心目标是在H. 264/AVC high profile 的基础上，**压缩效率提高一倍**。即在保证相同视频图像质量的前提下，视频流的码率减少50%。在提高压缩效率的同时，可以允许编码端适当提高复杂度。

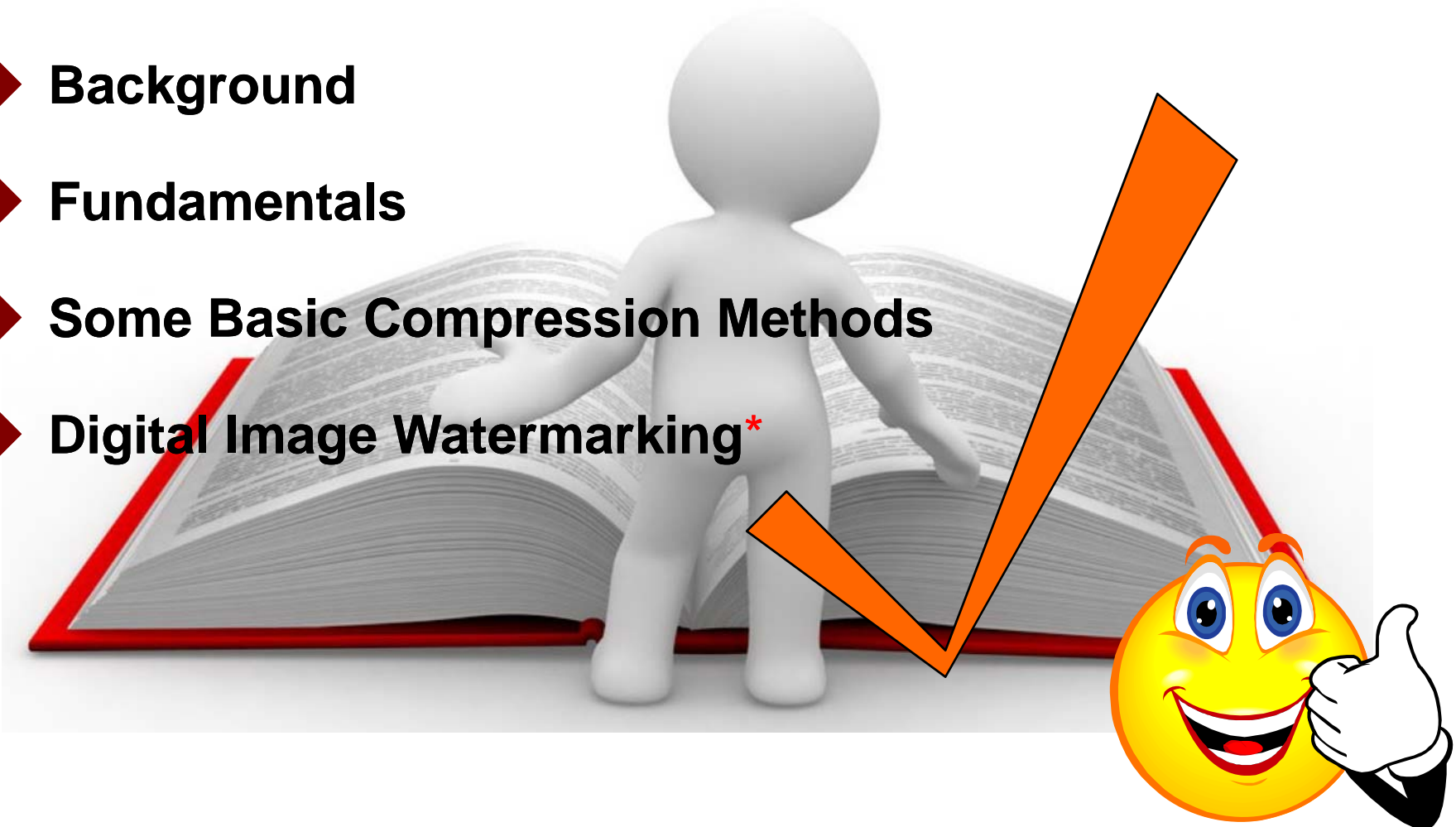
Structure of the current HEVC model (HM3)





Outline

- ◆ Background
- ◆ Fundamentals
- ◆ Some Basic Compression Methods
- ◆ Digital Image Watermarking*





本章作业

➤ 1. 书后 8.2, 8.19

◆ 2. 课后MATLAB编程练习

读取一幅灰度或者彩色图像，实现下列算法：

首先将图像分成许多 8×8 的子图像，对每个子图像进行DCT，对每个子图像的64个系数，按照每个系数的大小来排序后，舍去小的变换系数，只保留16个系数，实现图像4:1的压缩