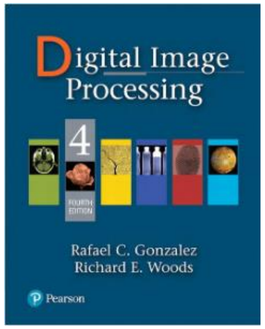


Fall, 2022

Digital Image Processing

Image Compression and Watermarking (I)

Dr. Tun-Wen Pai

1) **Understand information amount in an image**
2) **Understand data redundancy**
3) **Lossy and error-free compression**
4) **Huffman Codes/Runlength Encoding**

Dec. 20, 2022

Image Compression

- Images usually require an enormous amount of data to represent:
  Example: A standard 8.5” by 11” sheet of paper scanned at 100 samples per inch (dpi) and quantized to two gray levels (binary image) would require more than 100k bytes to represent.
  (8.5x100)(11x100)/8 = 116875 bytes.

- Image compression involves reducing the amount of data (bits) required to represent a digital image. This is done by removing **redundant** information.

- Image compression is crucial for efficient storage, retrieval, transmission, and manipulation of images.

- More generally, data compression involves the efficient representation of data in digital format.

- **Information theory** --- a field pioneered by Claude E. Shannon in the 1940s --- is the theoretical basis for most data compression techniques.

- Compression can be either **lossless (information preserving) or lossy**. In lossy compression, one can achieve higher levels of data reduction with less than perfect reproduction of the original image.
- Data compression refers to the process of reducing the amount of **data** required to represent a given quantity of **information**.

- Various amounts of data may be used to represent /describe the same information. Some representations maybe less efficient in the sense that they have **data redundancy**.

- If n_1 and n_2 are the number of **information carrying units** (ex. bits) in two datasets that represent the **same** information, the relative redundancy R_D of the first dataset is defined as

$$R_D = 1 - \frac{1}{C_R}, \quad \text{where } C_R = \frac{n_1}{n_2}$$

C_R is called the **compression ratio**.

- For images, data redundancy can be of three types:
  – **Coding redundancy**: This refers to the binary codewords used to represent grayvalues.
  – **Spatial and Temporal redundancy (Interpixel redundancy)**: This refers to the correlation between adjacent pixels in an image.
  – **Irrelevant information (Psychovisual redundancy)**: This refers to the unequal sensitivity of the human eye to different visual information.

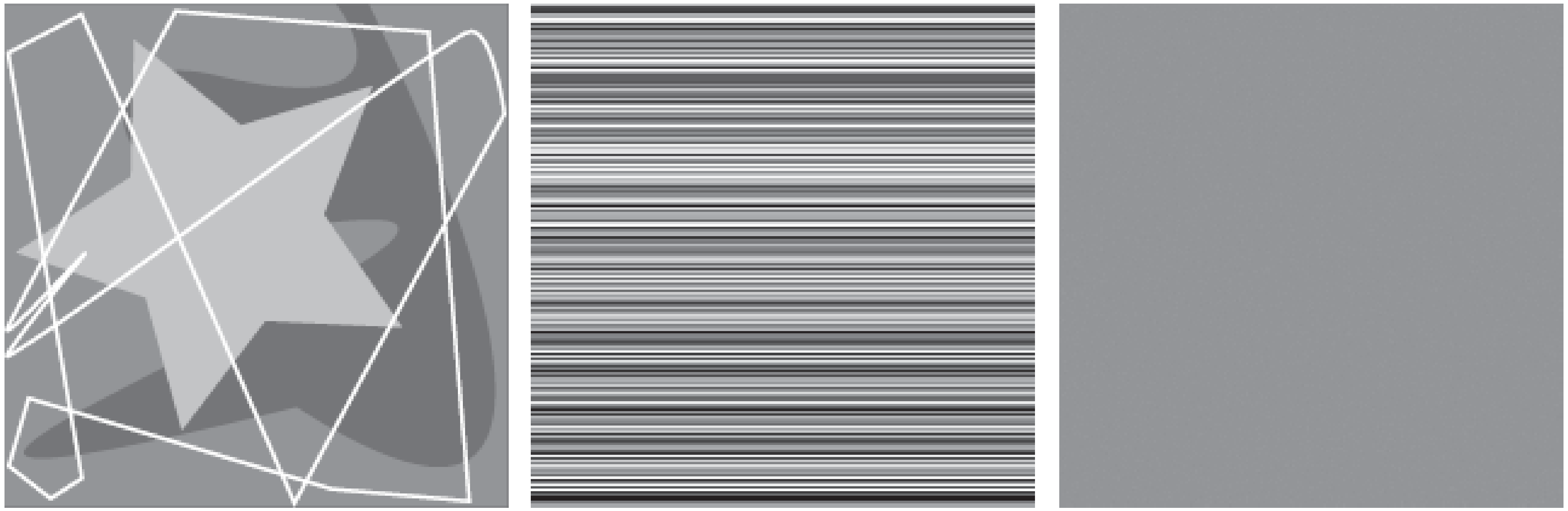


FIGURE 8.1 Computer generated $256 \times 256 \times 8$ bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy, but may exhibit others as well.)

TABLE 8.1 Example of variable-length coding.

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

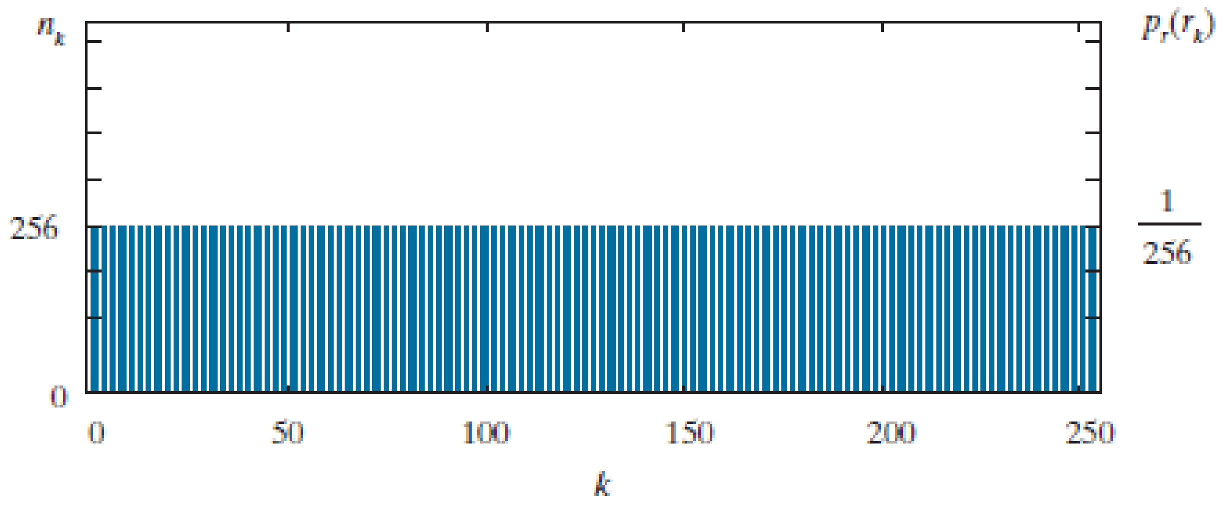
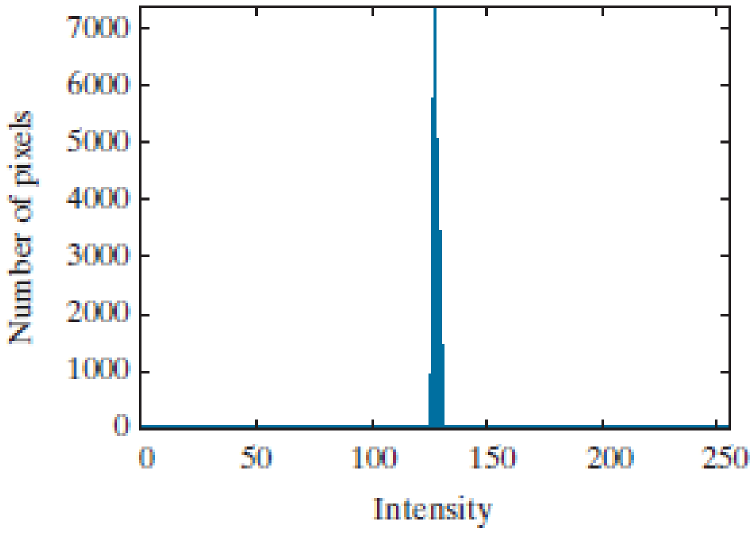
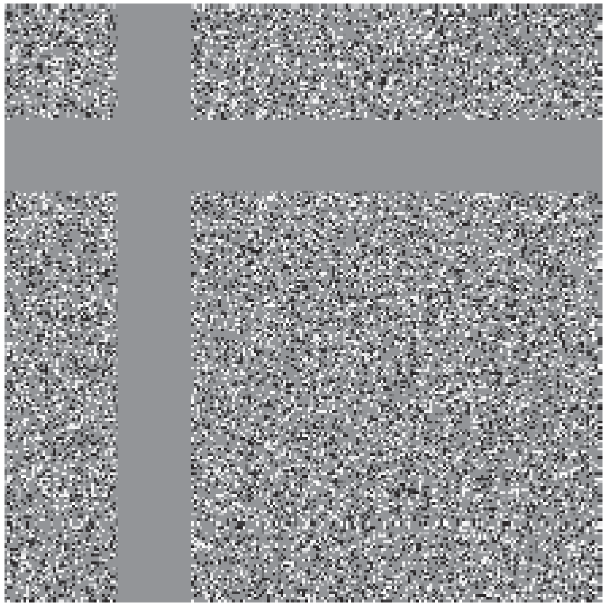


FIGURE 8.2 The intensity histogram of the image in Fig. 8.1(b).

a b

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





- A fidelity criterion or error criterion is required to quantify the loss of information (if any) due to a compression scheme.

- Objective fidelity criteria are based on some quantitative function of the original input image and the compressed and subsequently decompressed image. **Example**: Root-mean-square (RMS) error, which is defined as the square-root of the MSE.

$$e(m,n) = \hat{f}(m,n) - f(m,n)$$

$$f(m,n) : \text{Original image}$$

$$\hat{f}(m,n) : \text{Reconstructed image}$$

$$e(m,n) : \text{Error image}$$

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

- A related measure is the mean-square signal-to-noise ratio ($\mathrm{SNR}_{\mathrm{ms}}$):

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

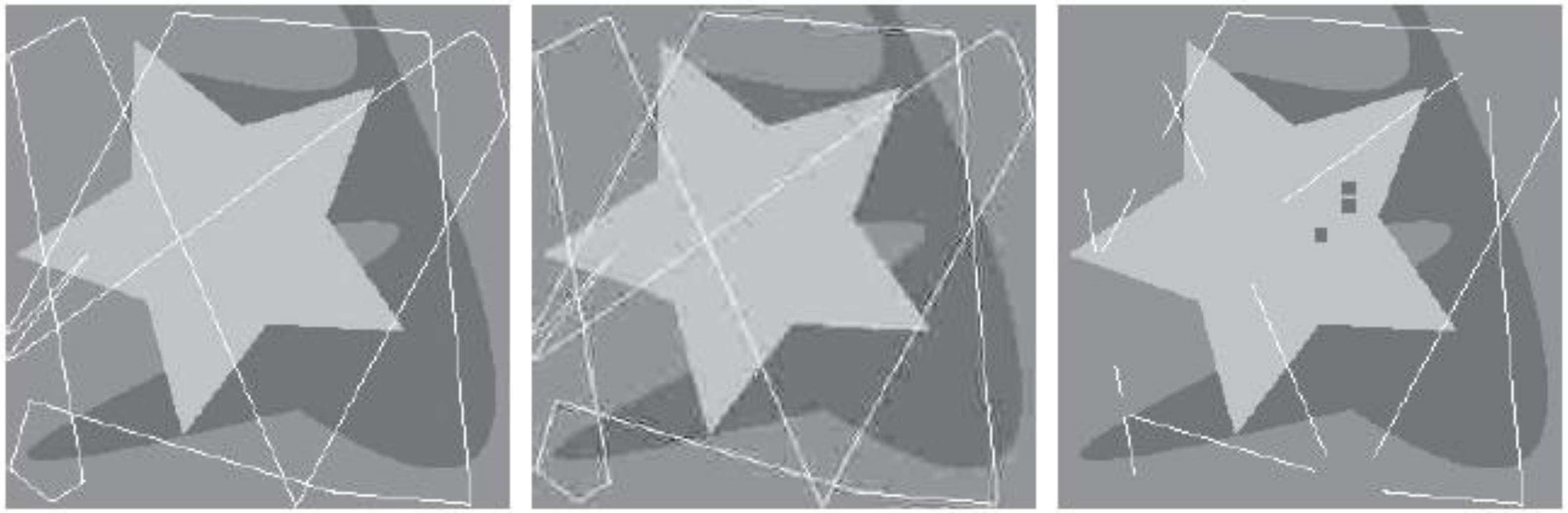
- The rms value of SNR, denoted $\mathrm{SNR}_{\mathrm{ms}}$, is the square-root of $\mathrm{SNR}_{\mathrm{ms}}$
- When the ultimate image is to be viewed by a human, **subjective fidelity criterion** may be more appropriate.
- Here image quality is measured by subjective evaluations by a **human observer**.
- Ratings by a number of human observers, based on “typical” decompressed images, are averaged to obtain this subjective fidelity criterion.

- Example of an absolute comparison scale:

Value	Rating	Description
1	Excellent	An image of extremely high quality --- as good as desired.
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.

- Example of relative comparison scale, based on a “side-by-side” comparison of the original image and the decompressed image:

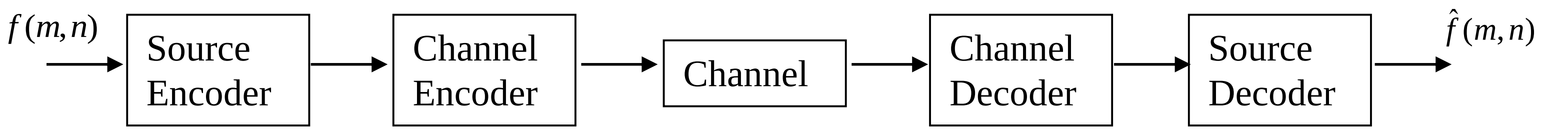
Value	-3	-2	-1	0	1	2	3
Rating	Much Worse	Worse	Slightly Worse	Same	Slightly better	Better	Much Better



a b c

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

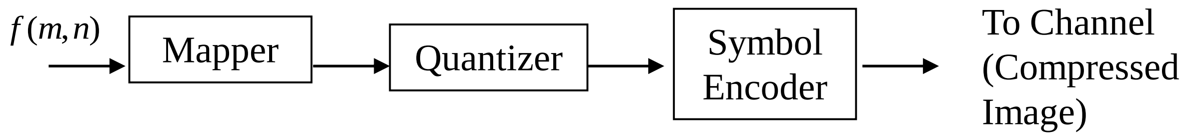
Image Compression model



- Source Encoder is used to remove redundancy in the input image.
- Channel Encoder is used to introduce redundancy in a controlled fashion to help combat noise. Example: Parity bit.
- This provides a certain level of immunity from noise that is inherent in any storage/transmission system. If the channel is not prone to noise, this block maybe eliminated.
- The Channel could be a communication link or a storage/retrieval system.
- Channel Decoder and Source Decoder invert the operations of the corresponding encoder blocks.
- We will mainly concentrate on the source encoder/decoder blocks and not on the channel encoder/decoder steps.

Source Encoder and Decoder

- Source encoder is responsible for reducing or eliminating any coding, interpixel, or psychovisual redundancy.

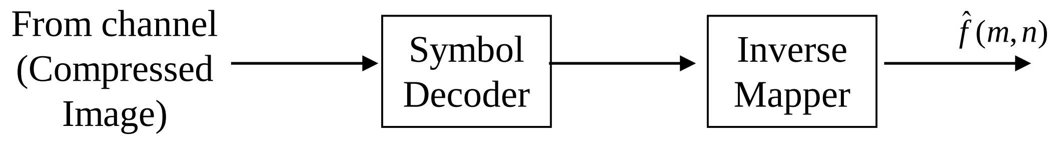


Source Encoder

- The first block “Mapper” transforms the input data into a (usually nonvisual) format, designed to reduce interpixel redundancy. This block is reversible and may or may not reduce the amount of data.
  Example: run-length encoding, image transform.

- The quantizer reduces accuracy of the mapper output in accordance with some fidelity criterion. This block reduces psychovisual redundancy and is usually not invertible.

- The Symbol Encoder creates a fixed or variable length codeword to represent the quantizer output and maps the output in accordance with this code. This block is reversible and reduces coding redundancy.



Source Decoder

- The decoder blocks are inverse operations of the corresponding encoder blocks (except the quantizer block, which is not invertible).

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

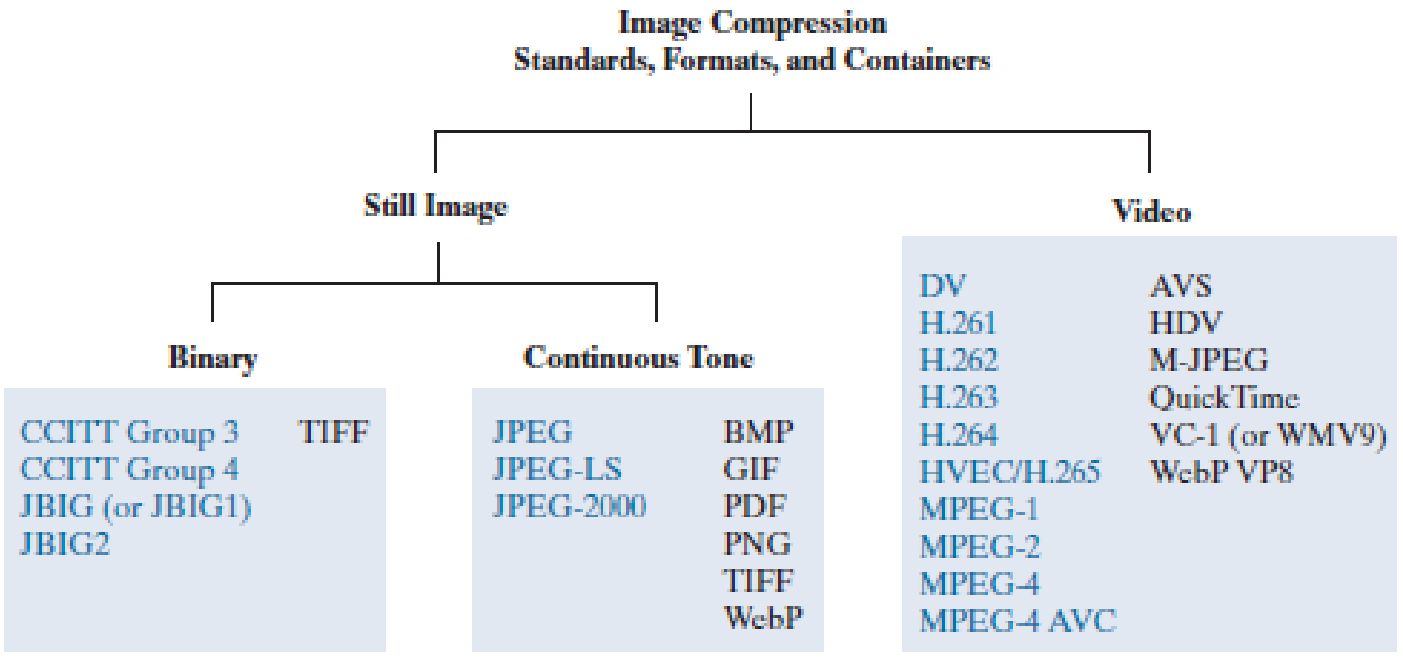


TABLE 8.3
Internationally sanctioned image compression standards. The numbers in brackets refer to sections in this chapter.

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

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

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

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

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

Elements of Information Theory

- What is information --- how to quantify it?
- What is the minimum amount of data that is sufficient to represent an image without loss of information?
- What is theoretically the best compressoin possible?
- What is the theoretically best possible transmission rate for **reliable** communication over a noisy channel?
- Information theory provides answers to these and other related fundamental questions.
- The fundamental premise of information theory is that the generation of information can be modeled as a probabilistic process.

- A discrete source of information generates one of *N* possible symbols from a source alphabet set $A = \{a_0, a_1, \ldots, a_{N-1}\}$, in unit time.
  Example: $A = \{a, b, c, \ldots, z\}$, $\{0, 1\}$, $\{0, 1, 2, .., 255\}$

- The source output can be modeled as a discrete random variable *E*, which can take values in set $A = \{a_0, a_1, \ldots, a_{N-1}\}$, with corresponding probabilities $\{p_0, p_1, .., p_{N-1}\}$.

- We will denote the symbol probabilities by the vector

$$z = \left[p(a_0), p(a_1), \ldots, p(a_{N-1})\right]^T = \left[p_0, p_1, \ldots, p_{N-1}\right]^T$$

- Naturally, $p_t \geq 0,$ and $\sum_{t=0}^{N-1} p_t = 1$

- The information source is characterized by the pair (A, z).

- Observing an occurrence (or realization) of the random variable *E* results in some gain of information denoted by *I(E)*. This gain of information was defined to be (Shannon):

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

- The base for the logarithm depends on the units for measuring information. Usually, we use base 2, which gives the information in units of “binary digits” or “bits”. Using a base 10 logarithm would give the entropy in the units of **decimal digits**.

- The amount of information attributed to an event *E* is inversely related to the probability of that event.

- Examples:
  - **Certain event**: *P*(*E*) = 1.0. In this case *I*(*E*) = log(1/1) = 0. This agrees with intuition, since if the event *E* is certain to occur (has probability 1), knowing that it has occurred has not led to any gain of information.
  - **Coin toss**: *P*(*E* = Heads) = 0.5. In this case I(*E*) = log(1/0.5) = log(2) = 1 bit. This again agrees with intuition.
  - **Rare event**: *P*(*E*) = 0.001. In this case *I*(*E*) = log(1/0.001) = log(1000) = 9.97 bits. This again agrees with intuition, since knowing that a rare event has occurred leads to a significant gain of information.

- The entropy *H*(z) of a source is defined as the **average amount of information gained by observing a single source symbol**:

$$H(z) = -\sum_{t=0}^{N-1} p_t \log p_t$$

- By convention, in the above formula, we set 0log0 = 0.
- The entropy of a source quantifies the “randomness” of a source.
- Higher the source entropy, more the uncertainty associated with a source output, and higher the information associated with a source.
- For fixed number of source symbols the entropy is maximized if all symbols are equally likely (recall uniform histogram).

Example:

Symbol a_t	Probability p_t	Information (in bits) $I(a_t) = -\log p_t$
0	½	1
1	¼	2
2	1/8	3
3	1/16	4
4	1/32	5
5	1/64	6
6	1/64	6

Source entropy:

$$H(z) = -\sum_{t=0}^{6} p_t \log\ p_t = -\left[1/2 \log 1/2 + 1/4 \log 1/4 + \cdots + 1/64 \log 1/64\right]$$
$$= \left[1/2 + 1/2 + \cdots + 3/32\right] = 63/32 = 1.96875 \text{ bits}$$

- Given that a source produces the above symbols with indicated probabilities, how do we represent them using binary strings?

Symbol a_t	Probability	Binary String (Codeword)	Length of codeword l_t
0	½	000	3
1	¼	001	3
2	1/8	010	3
3	1/16	011	3
4	1/32	100	3
5	1/64	101	3
6	1/64	110	3

Average length of a fixed codeword:

$$L_{avg} = \sum_{t=0}^{6} p_t l_t = \sum_{\mathrm{t}=0}^{6} \mathrm{p}_{\mathrm{t}}(3) = 3\sum_{t=0}^{6} p_t = 3\,\mathrm{bits}$$

- Is this the best we can do (in terms of L_{avg})? For a fixed length codeword scheme, yes. How about if we employ a variable length scheme?

- **Idea**: Since the symbols are not all equally likely, assign shorter codewords to symbols with higher probability and longer codewords to symbols with lower probability, such that the average length is smaller.

- Consider the following scheme

Symbol a_t	Probability p_t	Binary String (Codeword)	Length of codeword l_t
0	½	0	1
1	¼	10	2
2	1/8	110	3
3	1/16	1110	4
4	1/32	11110	5
5	1/64	111110	6
6	1/64	111111	6

$$L_{avg} = \sum_{t=0}^{6} p_t l_t = [1/2 + 1/2 + \cdots + 3/32] = 63/32 = 1.96875 \text{ bit}$$

- Notice that this is the same as the source entropy!

Shannon's noiseless coding theorem

Let (A, z) be a discrete source with probability vector z and entropy H(z). The average codeword length of any **distortionless** (uniquely decodable) coding is bounded by

$$\boxed{L_{avg} \geq H(z)}$$

In other words, no codes exist that can **losslessly** represent the source if $L_{\text{avg}} < H$(z)

Error-free compression

- Useful in application where no loss of information is tolerable. This maybe due to accuracy requirements, legal requirements, or less than perfect quality of original image.

- Compression can be achieved by removing coding and/or interpixel redundancy.

- Typical compression ratios achievable by lossless techniques is from 2 to 10.

Variable Length Coding

- This is used to reduce coding redundancy.

- Coding redundancy is present in any image with a non-uniform histogram (i.e. when all the graylevels are not equally likely).

- Given an image with, say 256 graylevels, $\{a_0, a_1, \ldots, a_{255}\} = \{0, 1, \ldots, 255\}$. This is our set of source symbols.

- For each graylevel a_k, we need its probability $p(a_k)$ in the image. This may be obrained from the image histogram: $p(a_k) = n_k / n$, n_k = # pixels with value a_k, n = total # pixels.

- To each graylevel a_k, we need to assign a codeword (a binary string). Suppose l_k is the length of codeword (= #bits required to represent a_k) for symbol a_k .

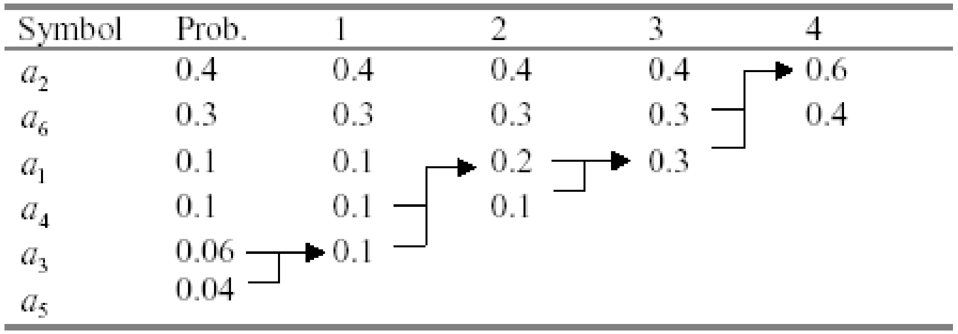
- Total number of bits required to represent the image is

$$\sum_{k=0}^{N-1} l_k n_k = n \sum_{k=0}^{N-1} l_k \left(\frac{n_k}{n} \right) = n \sum_{k=0}^{N-1} l_k \, p(a_k) = n L_{avg}$$

Huffman Code

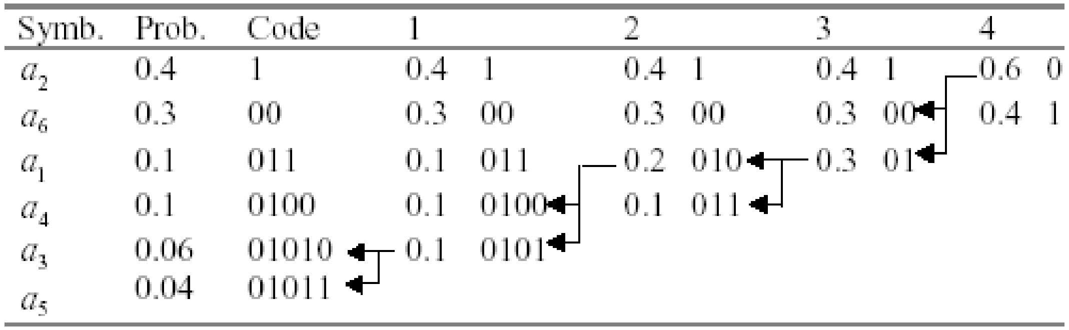
- The algorithm is best illustrated by means of an example.
- Given a source which generates one of six possible symbols $A = \{a_1, a_2, \ldots, a_6\}$ with corresponding probabilities {0.1, 0.4, 0.06, 0.1, 0.04, 0.3}.
- Arrange the symbols in descending order of their probability of occurrence.
- Successively reduce the number of source symbols by replacing the two symbols having least probability, with a “compound symbol.” This way, the number of source symbols is reduced by one at each stage.
- The compound symbol is placed at an appropriate location in the next stage, so that the probabilities are again in descending order. Break ties using any arbitrary but consistent rule.
- Code each reduced source starting with the smallest source and working backwards.

Source Reduction:



Symbol	Prob.	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				

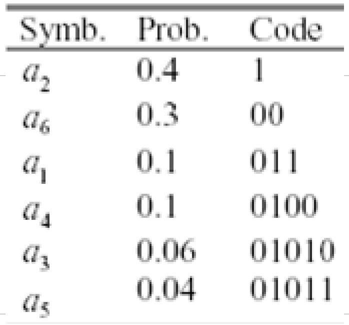
Code Assignment:



Symb.	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								

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

$$H(z) = -\sum_t p_t \log(p_t) = -\begin{bmatrix} 0.4(\log(0.4)) + 0.3(\log(0.3)) + 0.1(\log(0.1)) + \\ 0.1(\log(0.1)) + 0.06(\log(0.06)) + 0.04(\log(0.04)) \end{bmatrix}$$
$$= 2.14 \text{ bits/symbol}$$



Symb.	Prob.	Code
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010
a_5	0.04	01011

- The resulting code is called a Huffman code. It has some interesting properties:
  - The source symbols can be encoded (and decoded) one at time.
  - It is called a **block code** because each source symbol is mapped into a fixed sequence of code symbols.
  - It is **instantaneous** because each codeword in a string of code symbols can be decoded without referencing succeeding symbols.
  - It is **uniquely decodable** because any string of code symbols can be decoded in only one way.
- **Example**: Given the encoded string, 010100111100, it can be decoded as follows:

  010100111100 → **01010**0111100 → a_3**011**1100 → a_3a_1**1**100
  → $a_3a_1a_2$**1**00 → $a_3a_1a_2a_2$**00** → $a_3a_1a_2a_2a_6$

- The Huffman code is optimal (in terms of average codeword length) for a given set of symbols and probabilities, subject to the constraint that the symbols be coded one at a time.

Disadvantage:

- For a source with *J* symbols, we need *J* – 2 source reductions. This can be computationally intensive for large *J* (ex. *J* = 256 for an image with 256 graylevels).

Bit-plane Coding

- A grayscale image is decomposed into a series of binary images and each binary image is compressed by some binary compression method.

- This removes coding and interpixel redundancy.

- **Bit-plane decomposition**:
- Given a grayscale image with 2^m graylevels, each grayvalue can be represented by m-bits, say (a_{m-1}, a_{m-2},… a_1, a_0).

- The grayvalue *r* represented by (a_{m-1}, a_{m-2},… a_1, a_0) is given by the base 2 polynomial

$$r = a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \cdots + a_1 2^1 + a_0 2^0$$

- This bit representation can be used to decompose the grayscale image into *m* binary images (bit-planes).

- Alternatively, one can use the ***m*-bit Gray code** ($g_{m-1}, g_{m-2}, \ldots, g_1, g_0$) to represent a given gray-value.

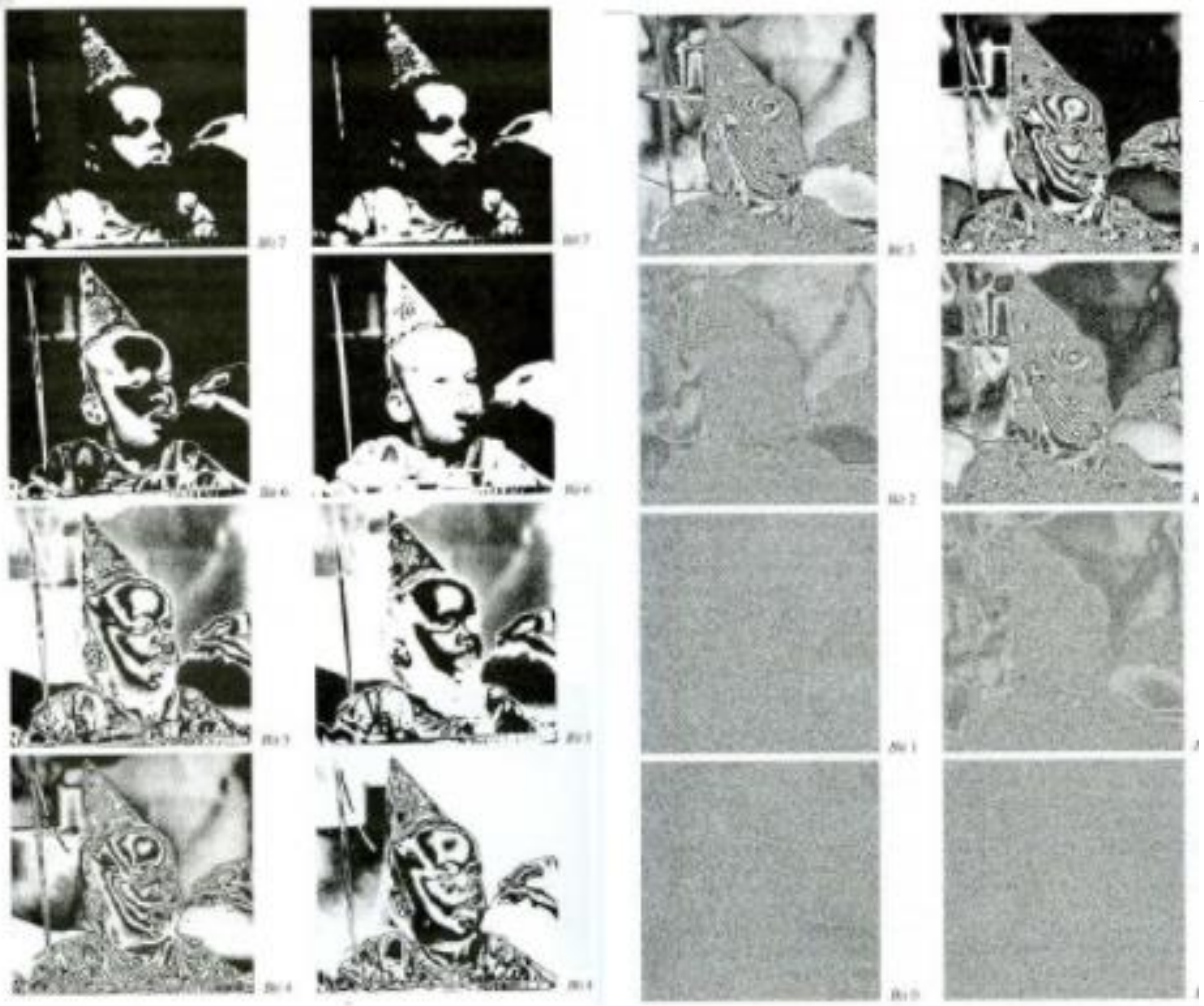
- The Gray code ($g_{m-1}, g_{m-2}, \ldots, g_1, g_0$) can be obtained from ($a_{m-1}, a_{m-2}, \ldots, a_1, a_0$) by the following relationship:

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

$$\text{where } \oplus \text{ denotes exclusive OR of bits.}$$

- The Gray code of successive gray-levels differ at only one position.

127 $\rightarrow$ 01111111 (binary representation) 01000000 (Gray code)
128 $\rightarrow$ 10000000 (binary representation) 11000000 (Gray code)



- The resulting binary images are then compressed (error-free).
- We will study a popular encoding scheme called run-length encoding (RLC).

Runlength encoding

- Each row of a bit plane (or binary image) is represented by a sequence of lengths (integers) that denote the successive runs of 0 and 1 pixels.

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

- Two approaches:
  - Start position and lengths of runs of 1s for each row is used:

    (1, 3)(7, 2)(12, 4)(17, 2)(20, 3)
    (5, 13)(19, 4)
    (1, 3)(17, 6)

  - Only lengths of runs, starting with the length of 1 run is used:

    3,3,2,3,4,1,2,1,3
    0,4,13,1,4
    3,13,6

- This technique is very effective in encoding binary images with large contiguous black and white regions, which would give rise to a small number of large runs of 1s and 0s.

- The run-lengths can in turn be encoded using a variable length code (ex. Huffman code), for further compression.

- Let a_k be the fraction of runs of 0s with length *k*. Naturally, $(a_1, a_2, \ldots, a_M)$, would represent a vector of probabilities (the probability of a run of 0s being of length *k*).

- Let $H_0 = -\sum_{t=1}^{M} a_t \log(a_t)$ be the entropy associated with $(a_1, a_2, \ldots, a_M)$ and $L_0 = \sum_{t=1}^{M} t a_t$ be the average length of runs of 0s.

- Let b_k be the fraction of runs of 1s with length *k*. Naturally, $(b_1, b_2, \ldots, b_M)$, would represent a vector of probabilities (the probability of a run of 1s being of length *k*).

- Let H_1 be the entropy associated with $(b_1, b_2, \ldots, b_M)$ and $L_1 = \sum_{t=1}^{M} t b_t$ be the average length of runs of 1s.

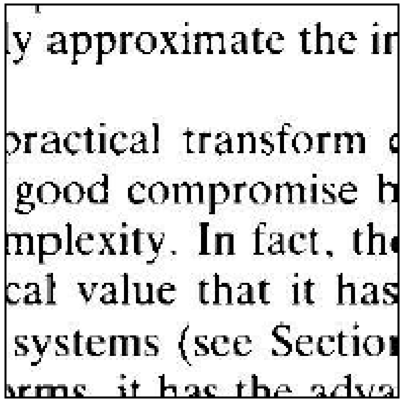
- The approximate runlength entropy of the image is

$$H_{RL} = \frac{(H_0 + H_1)\text{bits/run}}{(L_0 + L_1)\text{symbols/run}}$$

- H_{RL} provides an estimate of the average number of bits per pixel required to code the run lengths in a binary image, using a variable-length code.

- The concept of run-length can be extended to a variety of 2-D coding procedures.

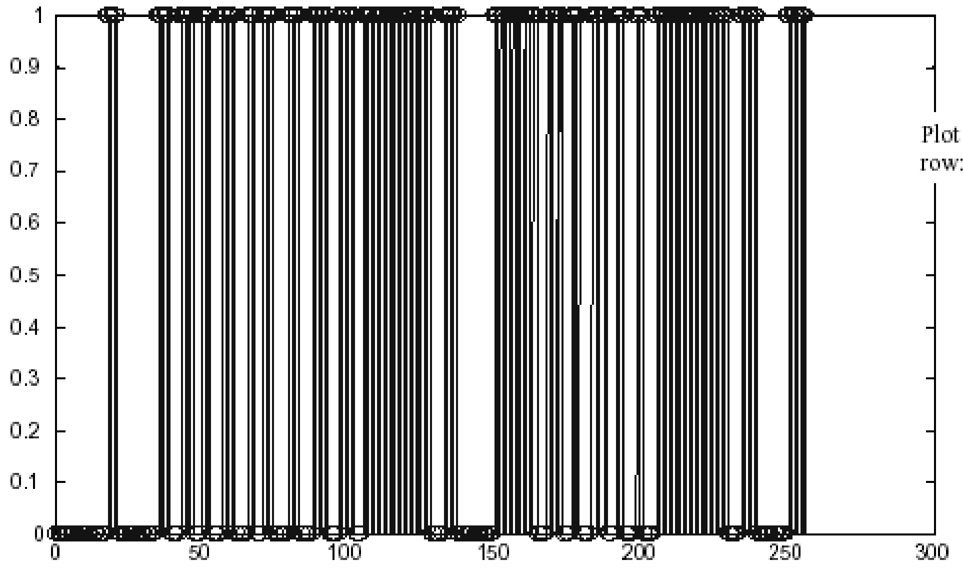
Runlength Example



ly approximate the ir

practical transform c
good compromise b
mplexity. In fact, th
cal value that it has
systems (see Sectio
rms, it has the adva

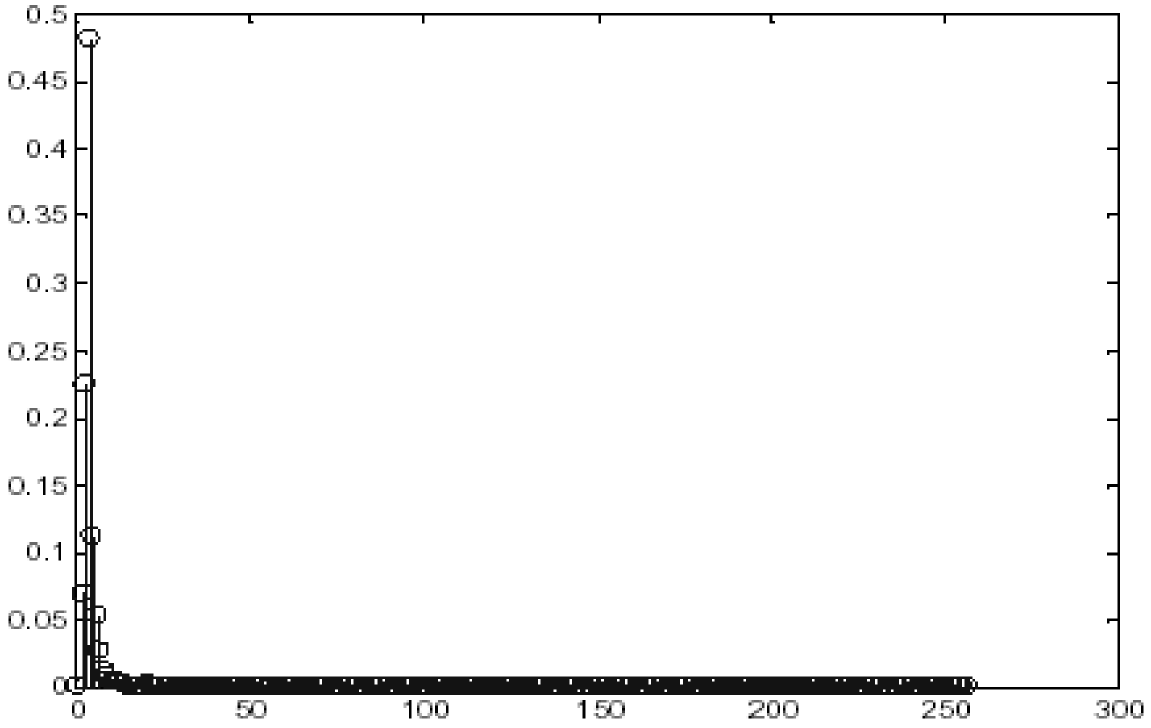
Binary text image



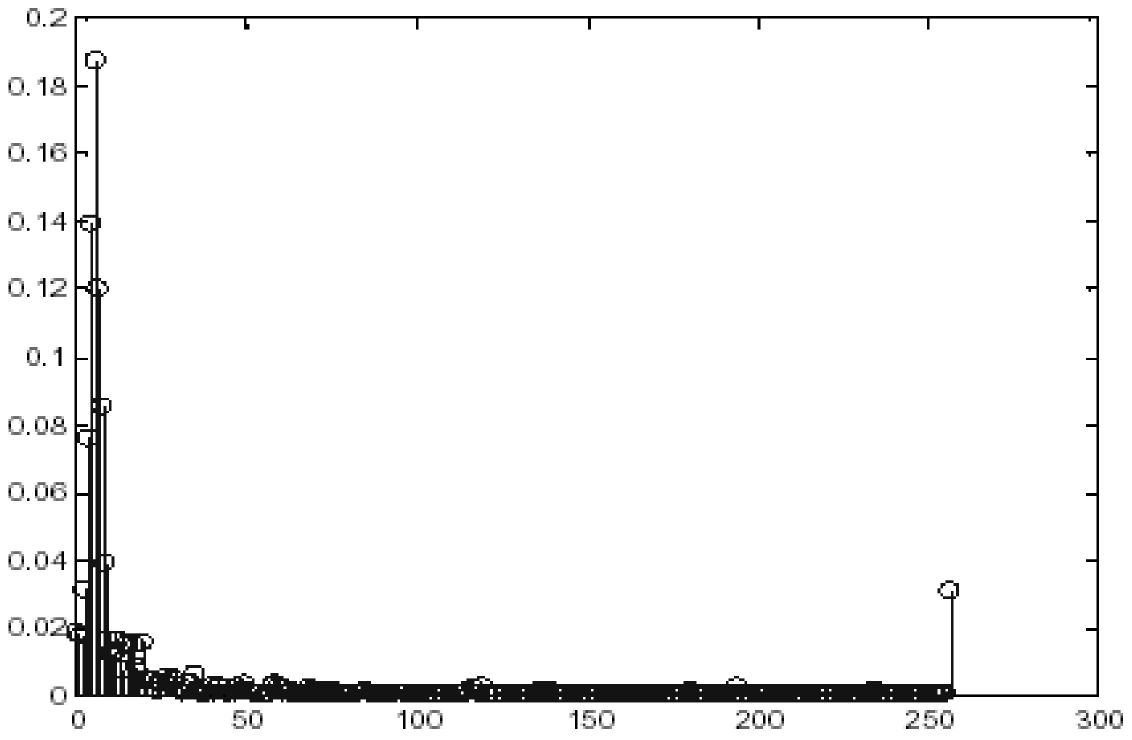
Plot of a typical row: (row # 152)

Runlength encoding of row # 152:

0, 17, 4, 13, 5, 3, 5, 2, 4, 3, 5, 4, 3, 2, 4, 5, 4, 4, 6, 3, 6, 2, 24, 4, 5, 12, 15,3, 6, 2, 4, 3, 6, 2, 4, 3, 3, 4, 25, 4, 6, 10, 7 (43 runs)



Distribution of lengths of runs of 0 in image
L_0 = 3.2 pixels/run
H_0 = 2.2 bits/run



Distribution of lengths of runs of 1 in image
L_1 = 20.3 pixels/run
H_1 = 4.4 bits/run

$$H_{RL} = \frac{H_0 + H_1}{L_0 + L_1} = 0.28 \text{ bits/pixel}$$