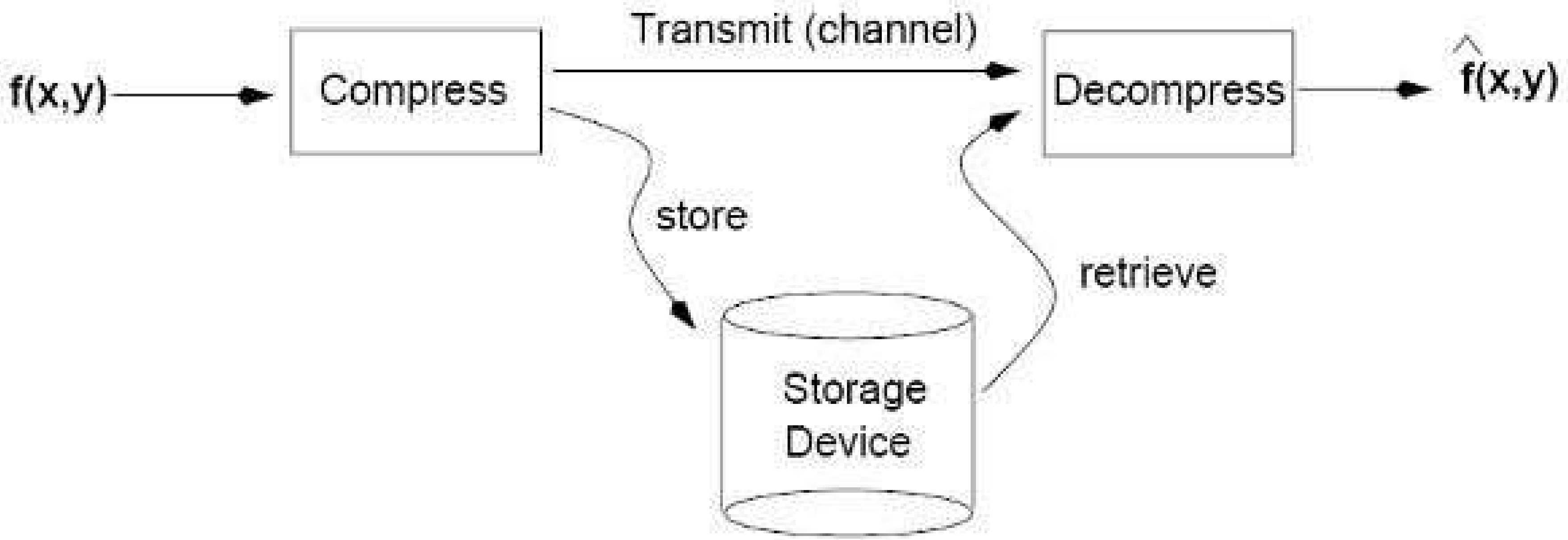
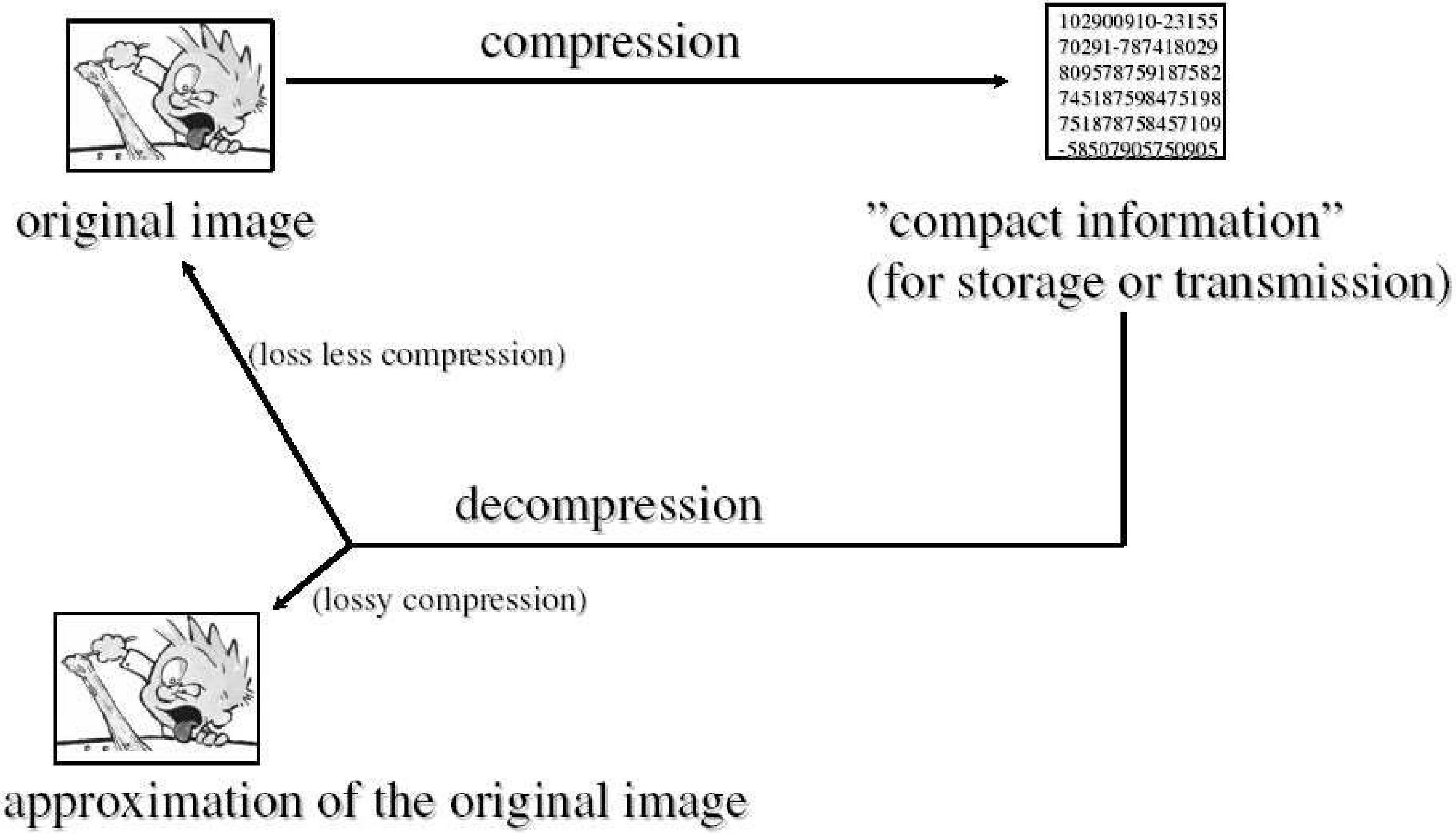
# Image Compression

- Image Compression is the art and science of reducing amount data required to represent an image.
- Digital images require huge amounts of space for storage and large bandwidths for transmission.
  - A 640 x 480 color image requires close to 1MB of space.
- The goal of image compression is to reduce the amount of data required to represent a digital image.
  - Reduce storage requirements and increase transmission rates.


f(x,y)
Compress
Transmit (channel)
Decompress
$\hat{f}(x,y)$
store
retrieve
Storage
Device



compression
102900910-23155
70291-787418029
809578759187582
745187598475198
751878758457109
-58507905750905
original image
"compact information"
(for storage or transmission)
(loss less compression)
decompression
(lossy compression)
approximation of the original image

Lossless

- Information preserving
- Low compression ratios

Lossy

- Not information preserving
- High compression ratios

# Data and Information:

- Data and information are not synonymous terms!
- Data is the means by which information is conveyed.
- Data compression aims to reduce the amount of data required to represent a given quantity of information while preserving as much information as possible.

- The same amount of information can be represented by various amount of data, e.g.:

Ex1: *Your wife, Helen, will meet you at Logan Airport in Boston at 5 minutes past 6:00 pm tomorrow night*

Ex2: *Your wife will meet you at Logan Airport at 5 minutes past 6:00 pm tomorrow night*

Ex3: *Helen will meet you at Logan at 6:00 pm tomorrow night*

# Data redundancy and compression ratio.

Relative data redundancy R

$R = 1 - 1/C$ where C commonly called the compression ratio is defined as

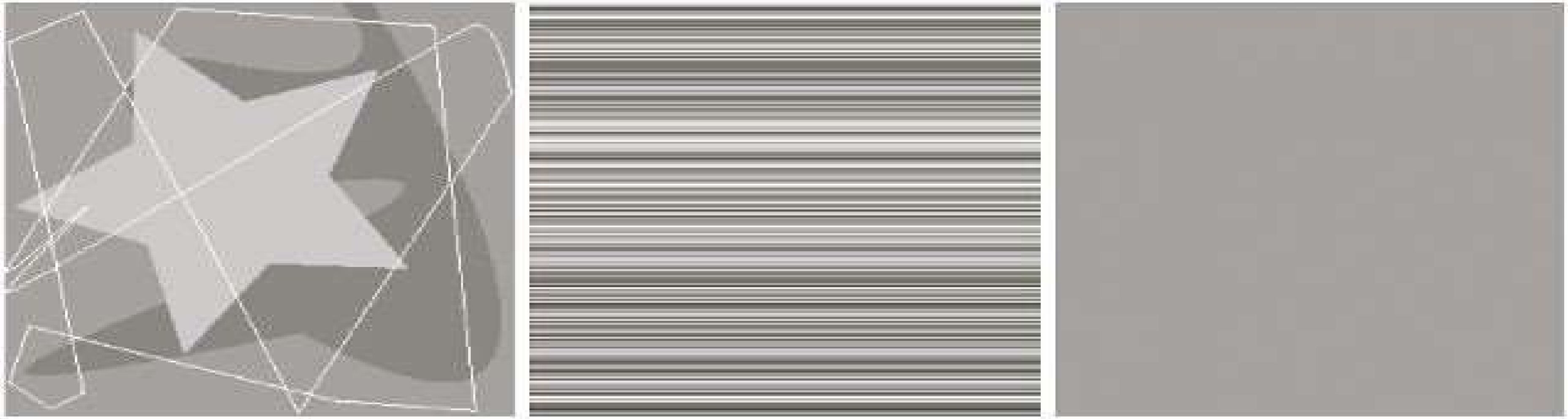
$C = b/b'$ where b and b' denote the number of bits in two representations of the same information

If C = 10,for instance, means larger representation has 10 bits of data for every 1 bit of data in the smaller representation

Corresponding relative data redundancy of the larger representation is 0.9 indicating 90% of its data is redundant

# Types of data redundancy

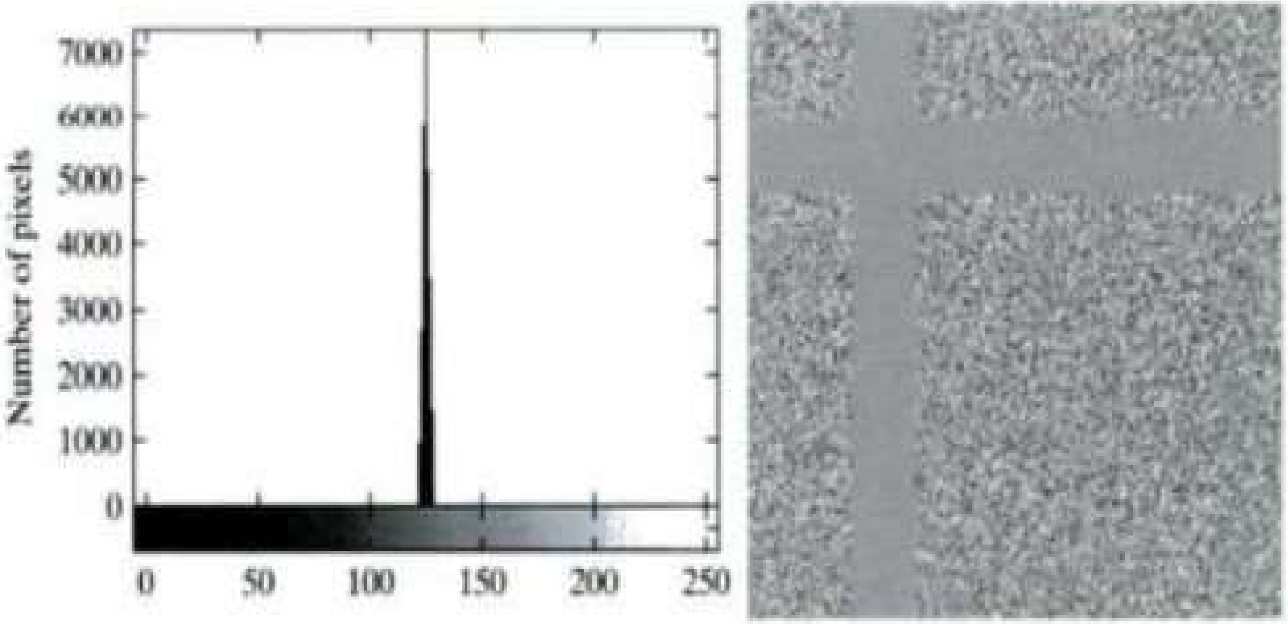
1. Coding redundancy
2. Spatial redundancy
3. Irrelevant information



a b c

**FIGURE 8.1** Computer generated 256 × 256 × 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.)

# Irrelevant Information




a b

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

1)Coding Redundancy: A code is a system of symbols (letters, numbers, bits) used to represent a body of information or set of events.

Each piece of information or event is assigned a sequence of code symbols, called a code word.

Number of symbols in each code word is its length

2)Spatial and temporal redundancy: Pixels of most 2-D intensity arrays are correlated spatially (i.e. each pixel is similar to or dependent on neighboring pixels.In a video sequence, temporally correlated pixels also duplicate information

3)Irrelevant information: Most 2-D intensity arrays contain information that is ignored by human visual system and/or extraneous to the intended use of the image. It is redundant in the sense that it is not used

# Coding Redundancy

Let $0 \le r_k \le 1$ : gray levels (discrete random variable)

$p_r(r_k)$ : probability of occurrence of $r_k$

$n_k$ : number of pixels that $r_k$ appears in the image

$n$ : total number of pixels in an image

$L$ : number of gray levels

$l(r_k)$ : number of bits used to represent $r_k$

$L_{avg}$ : average length of code words assigned to the grey levels

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k) \quad \text{where} \quad p_r(r_k) = \frac{n_k}{n}, k = 0,1,\ldots\ldots, L-1$$

Hence, total number of bits required to code an $M \times N$ image is $MNL_{avg}$

For a natural $m$-bit coding $L_{avg} = m$.

## Examples of variable length encoding

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

$$L_{avg} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

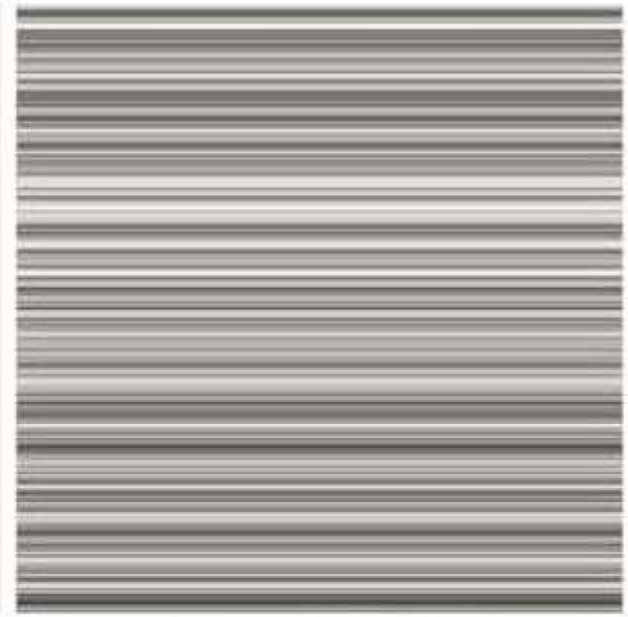
The total number of bits needed to represent the entire image is $MNL_{avg} = 256 \times 256 \times 1.81$ or 118,621. From Eqs. (8.1-2) and (8.1-1), the resulting compression and corresponding relative redundancy are

$$C = \frac{256 \times 256 \times 8}{118{,}621} = \frac{8}{1.81} \approx 4.42$$

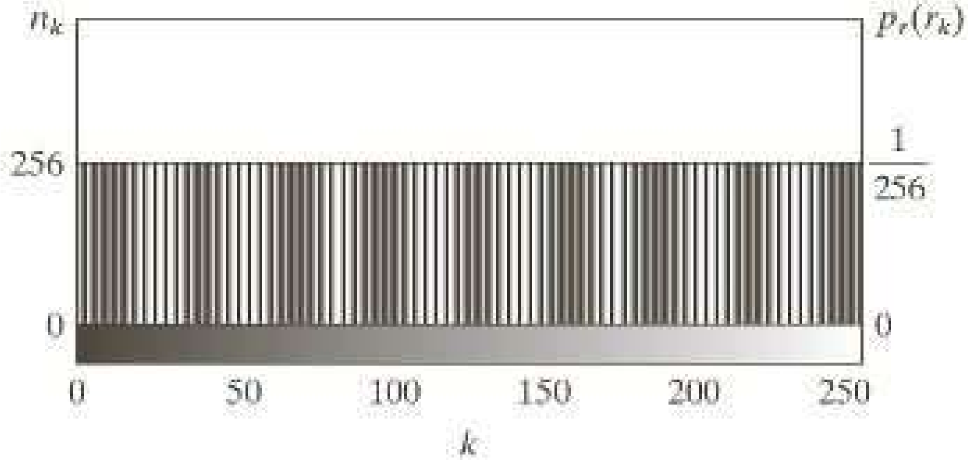
and

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

# Spatial redundancy



Spatial Redundancy

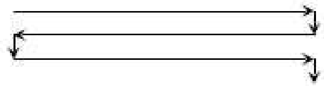


# Run-length coding

Every code word is made upp of a pair **(g,l)** where **g** is the graylevel and **l** is the number of pixels with that graylevel (length, or ”run”).

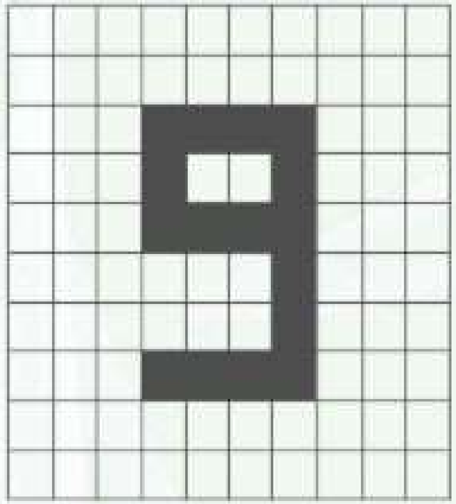
Ex 56 56 56 82 82 82 83 80
56 56 56 56 56 80 80 80

creates the runlength code (56,3) (82,3) (83,1) (80,4) (56,5)



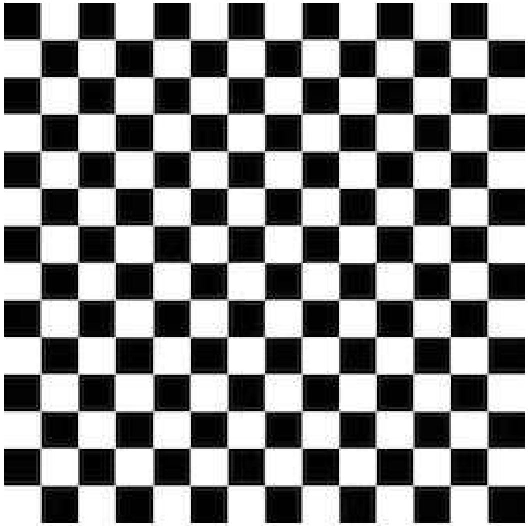
-The code is calculated row by row.
-Very efficient coding for binary data.
-Important to know position, and the image dimensions must be stored with the coded image.

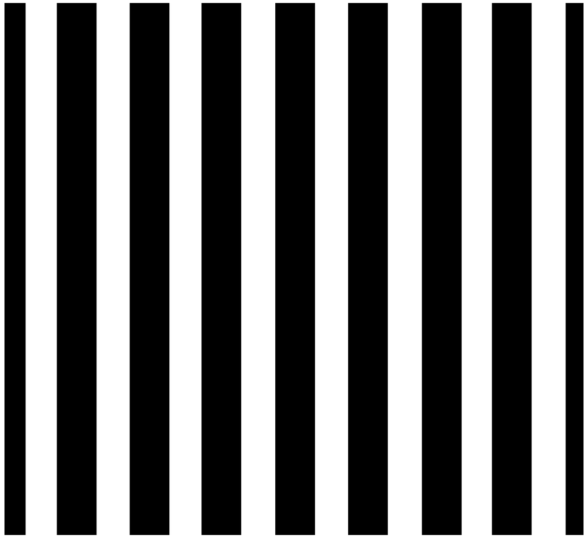
Can run length encoding be beneficial for every image?



(a)Image

Fig: Run-Length Code for a 100-pixel image





# Measuring Image Information

A random event E with probability P(E) contains:

$I(E) = \log(1/P(E)) = -\log(P(E))$ units of information

If the base 2 is selected, the unit of information is the bit.

Note: $I(E)=0$ when $P(E)=1$

# How much information does a pixel contain?

Suppose that gray level values are generated by a random variable, then rk contains:

$$I(r_k) = -\,log(P(r_k))$$

units of information!

Entropy of the image is defined as:

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

## Examples of variable length encoding

| $r_k$ | $p_r(r_k)$ | 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 |

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

# Example:

Consider the simple $4 \times 8$, 8-bit image:

$$\begin{array}{cccccccc}
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243
\end{array}$$

**(a)** Compute the entropy of the image.

**(a)** The entropy of the image is estimated using Eq. (8.1-7) to be

$$
\begin{aligned}
\tilde{H} &= -\sum_{k=0}^{255} p_r(r_k)\log_2 p_r(r_k) \\
&= -\left[\frac{12}{32}\log_2\frac{12}{32}+\frac{4}{32}\log_2\frac{4}{32}+\frac{4}{32}\log_2\frac{4}{32}+\frac{12}{32}\log_2\frac{12}{32}\right] \\
&= -[-0.5306-0.375-0.375-0.5306] \\
&= 1.811 \text{ bits/pixel.}
\end{aligned}
$$

# Solve by Yourself

**Can variable length coding procedure be used to compress a histogram equalized image?**

**Can such image contain spatial or temporal redundancies?**

# Fidelity Criteria

Quantify the nature and extent of information loss.

## Objective fidelity criteria:

Level of information loss can be expressed as a function of the original (input) and compressed-decompressed (output) image.

Given an $M \times N$ image $f(x,y)$ (original image), its compressed-then-decompressed image: $\hat{f}(x,y)$ , then the error between corresponding values are given as:

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

Total error is given by:

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

Normally the objective fidelity criterion parameters are as follows:

$e_{rms}$ (root-mean-square error):

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

$SNR_{ms}$ (mean-square signal-to-noise ratio):

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

# Subjective 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).

Consider an 8-pixel line of intensity data, $\{108, 139, 135, 244, 172, 173, 56, 99\}$. If it is uniformly quantized with 4-bit accuracy, compute the rms error and rms signal-to-noise ratios for the quantized data.

| $f(x,y)$ | | $\hat{f}(x,y)$ |
|---|---|---|
| Base 10 | Base 2 | Base 2 |
| 108 | 01101100 | 0110 |
| 139 | 10001011 | 1000 |

**Table P8.3**

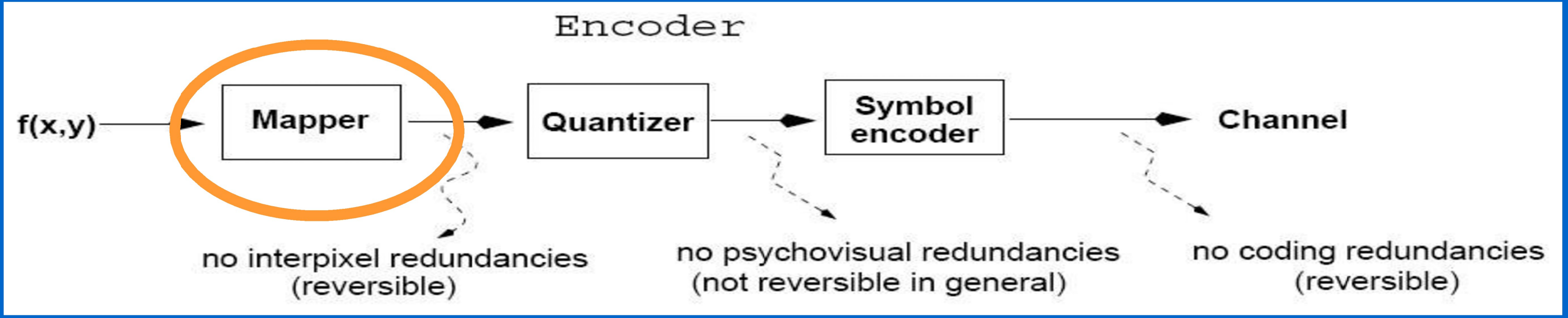
| $f(x,y)$ | | $\hat{f}(x,y)$ | | $16\hat{f}(x,y)-f(x,y)$ |
|---|---|---|---|---|
| Base 10 | Base 2 | Base 2 | Base 10 | Base 10 |
| 108 | 01101100 | 0110 | 6 | -12 |
| 139 | 10001011 | 1000 | 8 | -11 |
| 135 | 10000111 | 1000 | 8 | -7 |
| 244 | 11110100 | 1111 | 15 | -4 |
| 172 | 10101100 | 1010 | 10 | -12 |
| 173 | 10101101 | 1010 | 10 | -13 |
| 56 | 00111000 | 0011 | 3 | -8 |
| 99 | 01100011 | 0110 | 6 | -3 |

Using Eq. (8.1-10), the rms error is

$$
\begin{aligned}
e_{rms} &= \sqrt{\frac{1}{8}\sum_{x=0}^{0}\sum_{y=0}^{7}\left[16\hat{f}(x,y) - f(x,y)\right]^2} \\
&= \sqrt{\frac{1}{8}\left[(-12)^2+(-11)^2+(-7)^2+(-4)^2+(-12)^2+(-13)^2+(-8)^2+(-3)^2\right]} \\
&= \sqrt{\frac{1}{8}(716)} \\
&= 9.46
\end{aligned}
$$

$$
\begin{aligned}
SNR_{ms} &= \frac{\sum_{x=0}^{0}\sum_{y=0}^{7}\left[16\hat{f}(x,y)\right]^2}{\sum_{x=0}^{0}\sum_{y=0}^{7}\left[16\hat{f}(x,y)-f(x,y)\right]^2} \\
&= \frac{96^2+128^2+128^2+240^2+160^2+160^2+48^2+96^2}{716} \\
&= \frac{162304}{716} \\
&\simeq 227.
\end{aligned}
$$

# Image Compression Model (cont'd)




**Mapper**: transforms input data in a way that facilitates reduction of interpixel redundancies (spatial and temporal redundancy)

- Example - Run length coding

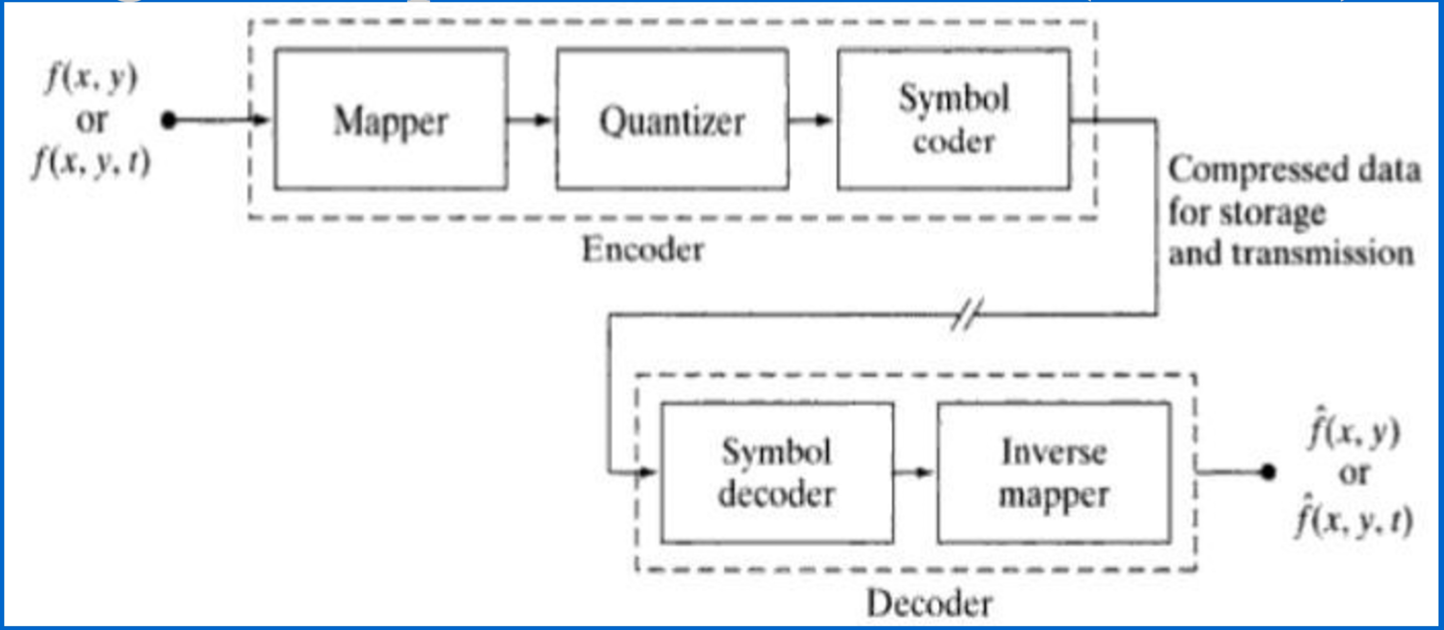
# Image Compression Model (cont'd)

- **<u>Quantizer</u>:** reduces the accuracy of the mapper's output in accordance with some pre-established fidelity criteria.
  - The goal is to keep irrelevant information out of the compressed representation.
  - It is irreversible but it must be omitted when error free compression is desired.

# Image Compression Model (cont'd)

- **Symbol encoder:** assigns the shortest code to the most frequently occurring output values – thus minimizing coding redundancy.

# Image Compression Models (cont'd)




- Inverse operations are performed.