# Image Compression

# Preview

- Methods of compressing data prior to storage and / or transmission are of significant practical and commercial
- Image compression addresses the problem of reducing the amount of data required to a digital image .
- The underlying basis of the reduction process is the removal of redundant data .

# Fundamentals

- The data compression refers to the process of reducing the amount of data required to represent a given quantity of information .
- The difference of data and information .
- Data are the means by which information is conveyed .
- Data redundancy is a central issue in digital image compression

# Fundamentals

- The relative data redundancy $R_D$ :

$$R_D = 1 - \frac{1}{C_R}$$

Where $C_R$ is compression ratio .

$$C_R = \frac{n_1}{n_2}$$

n1 , n2 donate the number of information – carrying units in two data set that represent the same information .

| | | | |
|---|---|---|---|
| n1=n2 | CR =1 | RD =0 | n1 contains no redundant data |
| n2 << n1 | CR → ∞ | RD→ 1 | significant compress & High redundant data |
| n2 >> n1 | CR →0 | RD→ -∞ | n2 contains much more data than n1 undesirable case |

CR=10 every 10 information in n1 represented by 1 bit in n2 , n1 has 90% redundancy

# 3 basic Data redundancies

- Coding Redundancy .

- Spatial and Temporal Redundancy.
- i.e. Video sequence (Correlated pixels are not repeated.)
- Irrelevant Information.
- Information that ignored by human visual system

# Coding Redundancy

- Lets assume , that a discrete random variable $r_K$ in the interval [0 , 1] represents the gray levels of an image and each $r_K$ occurs with probability

$$P_r(r_K)$$

$$P_r(r_K) = \frac{n_K}{n} \qquad k=0,1,2,\ldots\ldots L-1$$

Where L is the number gray levels ,

$n_K$ is the number of times that the $K^{th}$ gray level appears in image .

n is the total number of pixel in the image .

# Coding Redundancy

The average length of the code words assigned to the various gray level values

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

where $l(r_k)$ *no.\of bits used to represent each gray*

$p_r(r_k)$ *probability that gray level occurs*

# Example

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

$$L_{avg} = \sum l(r_K)\, p_r(r_K)$$

$$= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08)$$

$$+ 5(0.06) + 6(0.03) + 6(0.02)$$

$$= 2.7 bits.$$

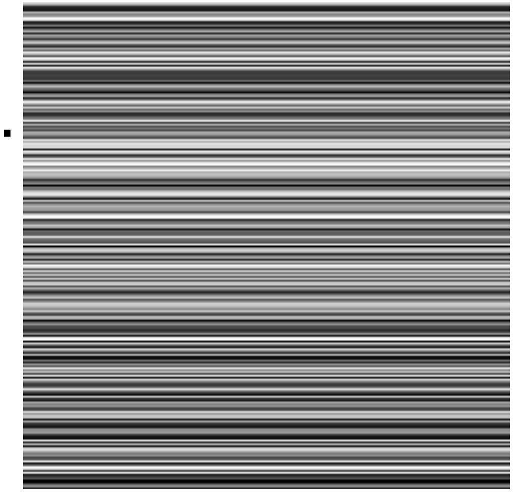
# Example

- The resulting compression ratio $C_R$ is 3/2.7 or 1.11 .
- Thus approximately 10% of the data resulting from the use of code 1 is redundant .
- The exact level of redundancy can be determine from

$$R_D = 1 - \frac{1}{1.11} = 0.099$$

# Spatial and Temporal redundancy

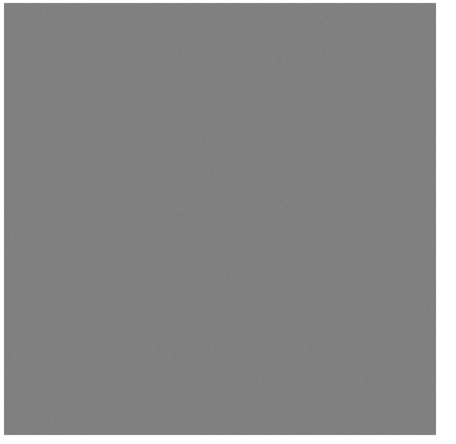
- Each line has the same intensity
- All 256intensity are of equal probability.
- Pixels intensity are independent of each other
- Pixels are correlated vertically
- Pixels intensity can be predicted from their
- Neighbor intensities, so the information carried
- By one pixel is small.

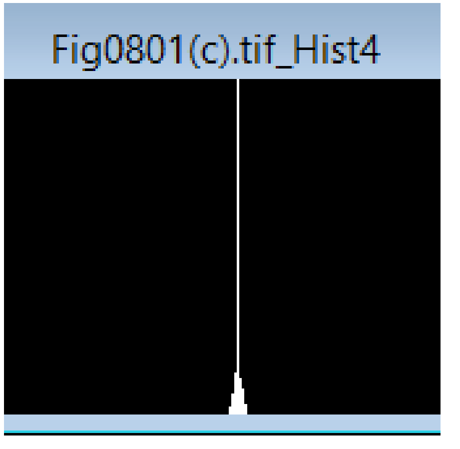


Histogram

# Irrelevant Information

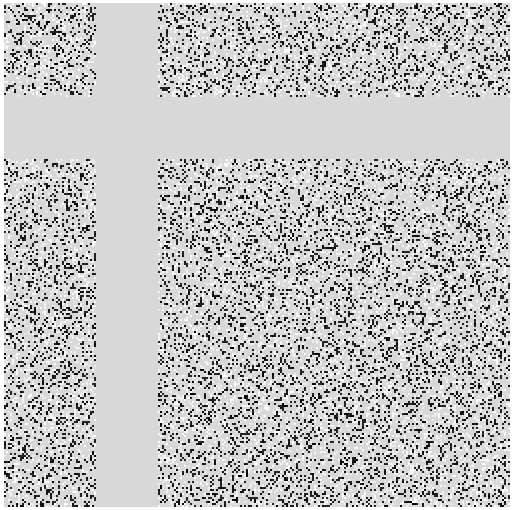
- Information that ignored by HVS are obvious candidates for omission.
- Original size is 256X256X8
- All are seems to be of the same color
- Compression =256X256X8/8
- = 65536:1





# Irrelevant Information

- this type of redundancy is different from the other 2 types
- Its elimination is possible because the information itself is not essential for HVS.
- Its removal referred to Quantization
- This means mapping of a broad range of intensity into limited range
- Quantization is irreversible operation.



Equalized Histogram

# How do we measure information?

- What is the information content of a message/image?
- What is the minimum amount of data that is sufficient to describe completely an image without loss of information?

# Modeling Information

Information generation is assumed to be a probabilistic process.

Idea: associate information with probability!

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

$$I(E) = log(\frac{1}{P(E)}) = -log(P(E)) \text{ units of information}$$

Note: I(E)=0 when P(E)=1 The event always occurs

# How much information does a pixel contain?

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

$$I(r_k) = -log(P(r_k))$$ units of information!

# How much information does an image contain?

Average information content of an image:

[redacted]

using $I(r_k) = -log(P(r_k))$

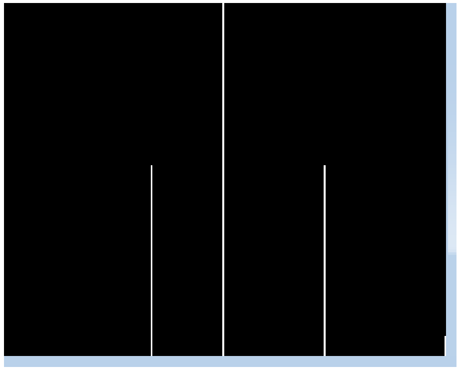
Entropy $H = -\sum_{k=0}^{L-1} P(r_k) log(P(r_k))$ units/pixel

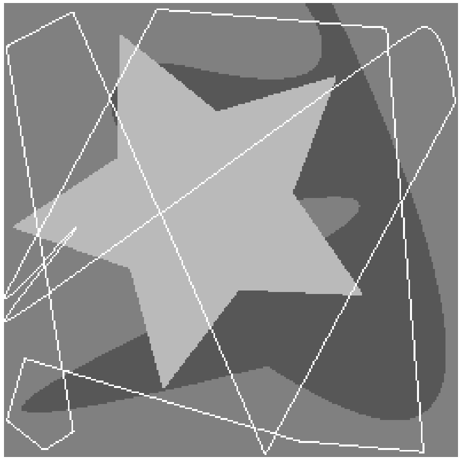
It is not possible to code an image with fewer than H bits/pixel

(**assumes** statistically independent random events)

# Example:

- $H = -[.25 \log_2 0.25 + .47 \log_2 0.47 +$
- $.25 \log_2 0.25 + .03 \log_2 0.03]$
- $= [-0.25(-2) + .47(-1.09) + .25(-2) + .03(-5.06)]$
- $= 1.6614$ bits/pixel





- What about H for the second type of redundancy?

# Fidelity Criteria

- Objective fidelity criterion

Loss of information – compress – decompress .

- Subjective fidelity criteria

Quality of image .

# Fidelity criteria

- Irrelevant information represents a loss, so we need a mean of quantifying the nature of loss
- When the level of information loss can be expressed as a function of the original or input image and the compressed and decompressed output image , it is based on an objective fidelity criterion .
- Example : the error at any x,y

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

Approximate Original

- The total error

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

- The square root

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

- The mean square signal – to – noise

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

# Fidelity Criteria(Subjective)

**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. |

# Huffman Coding (coding redundancy)

- A variable-length coding technique.
- Optimal code (i.e., minimizes the number of code symbols per source symbol).

- Assumption: symbols are encoded one at a time!

# Huffman Coding (cont'd)

- *Forward Pass*
  1. Sort probabilities per symbol
  2. Combine the lowest two probabilities
  3. Repeat *Step2* until only two probabilities remain.

| 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 | | | | |

# Huffman Coding (cont'd)

- ***Backward Pass***

  Assign code symbols going backwards

| 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 | | | | | | | | |

# Huffman Coding (cont'd)

- $L_{avg}$ using Huffman coding:

$$L_{avg} = E(l(a_k)) = \sum_{k=1}^{6} l(a_k)P(a_k) =$$

- $L_{avg}$ assuming binary codes:

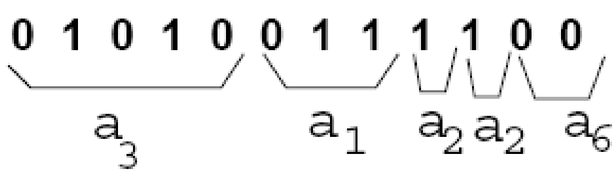
6 symbols, we need a 3-bit code

$(a_1: 000,\ a_2: 001,\ a_3: 010,\ a_4: 011,\ a_5: 100,\ a_6: 101)$

$$L_{avg} = \sum_{k=1}^{6} l(a_k)P(a_k) = \sum_{k=1}^{6} 3P(a_k) = 3\sum_{k=1}^{6} P(a_k) = 3 \text{ bits/symbol}$$

# Huffman Coding/Decoding

- After the code has been created, *coding/decoding* can be implemented using a look-up table.
- Note that decoding is done unambiguously.




| Original source | | |
|---|---|---|
| Sym. | 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 |

## Arithmetic (or Range) Coding
### (coding redundancy)

- Instead of encoding source symbols one at a time, sequences of source symbols are encoded together.
  - There is no one-to-one correspondence between source symbols and code words.
- Slower than Huffman coding but typically achieves better compression.

# Arithmetic Coding (cont.)

- A sequence of source symbols is assigned to a sub-interval in [0,1) which corresponds to an arithmetic code, e.g.,

arithmetic code

$\alpha_1 \alpha_2 \alpha_3 \alpha_3 \alpha_4$ → [0.06752, 0.0688) → 0.068

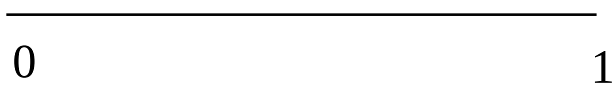
- We … as the number of symbols in the message increases, the interval used to represent the message becomes smaller.

# Arithmetic Coding (cont.)

Encode message: $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

| Source Symbol | Probability |
|---|---|
| $a_1$ | 0.2 |
| $a_2$ | 0.2 |
| $a_3$ | 0.4 |
| $a_4$ | 0.2 |

1) Start with interval [0, 1)



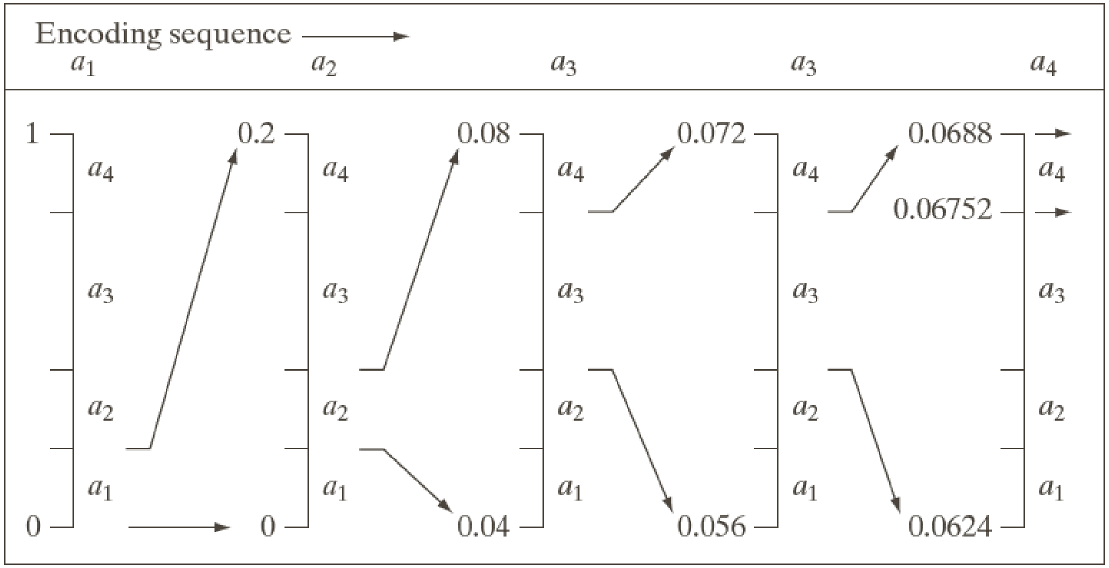

2) Subdivide [0, 1) based on the probabilities of $\alpha_i$


| Initial Subinterval |
|---|
| [0.0, 0.2) |
| [0.2, 0.4) |
| [0.4, 0.8) |
| [0.8, 1.0) |

3) Update interval by processing source symbols

# Example

| 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) |




Encode

$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

[0.06752, 0.0688)

or

0.068

(must be inside interval)

# Example (cont.)

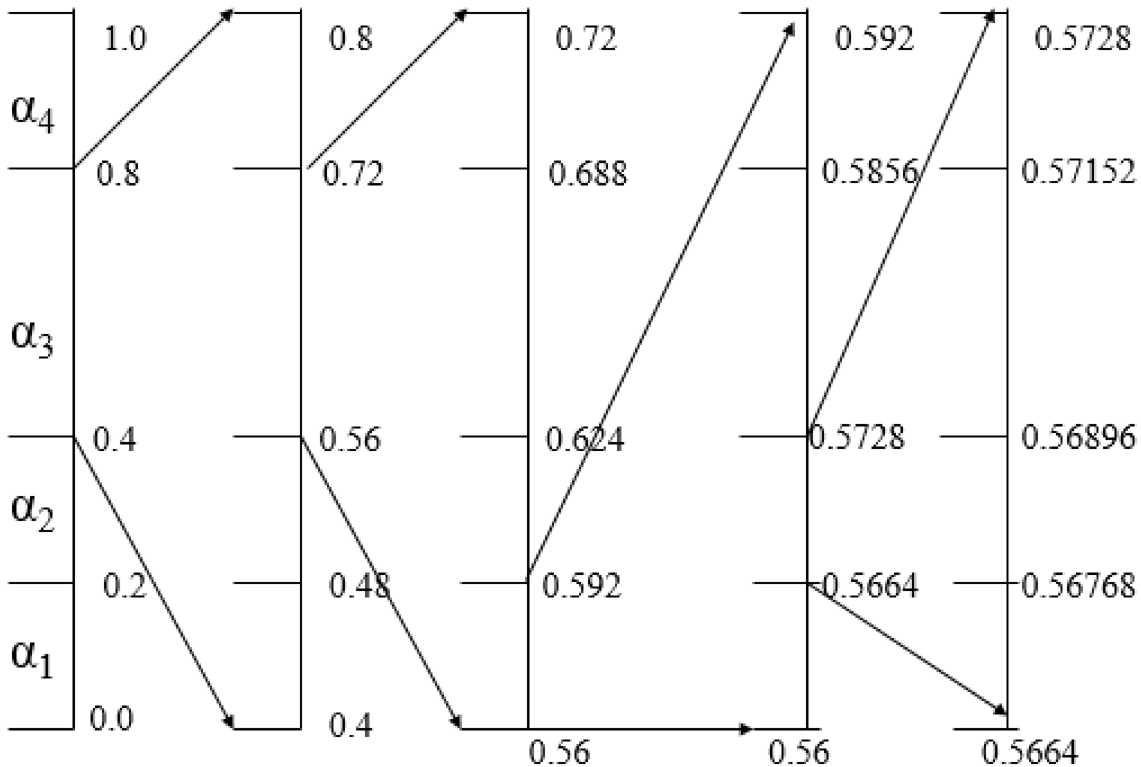
$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

- The message is encoded using 3 decimal digits or 3/5 = 0.6 decimal digits per source symbol.
- The entropy of this message is: $$H = -\sum_{k=0}^{3} P(r_k) log(P(r_k))$$

$-(3 \text{ x } 0.2\log_{10}(0.2)+0.4\log_{10}(0.4))=0.5786$ digits/symbol

**Note:** finite precision arithmetic might cause problems due to truncations!

# Arithmetic Decoding




Decode 0.572

$\alpha_3\ \alpha_3\ \alpha_1\ \alpha_2\ \alpha_4$