16.10.2023

Digital Image Processing

Image Compression

Makarand Tapaswi

Center for Visual Information Technology (CVIT), IIIT Hyderabad

Most slides borrowed from Ravi Kiran @CVIT!

14.10.2022

Digital Image Processing (CSE/ECE 478)

Lecture-17: Image Compression

Ravi Kiran

Center for Visual Information Technology (CVIT), IIIT Hyderabad

Some slides borrowed from Vineet Gandhi @CVIT!

amazon prime video

hotstar

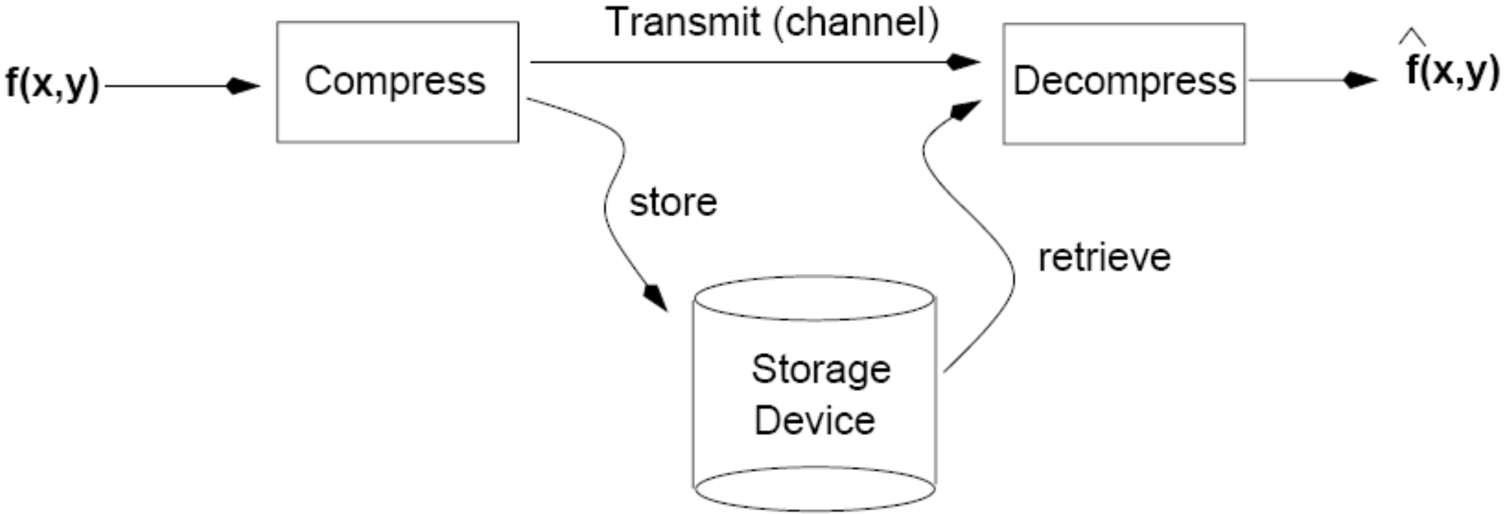
NETFLIX

Data Compression

- Data compression aims to reduce the amount of data while preserving as much information as possible.

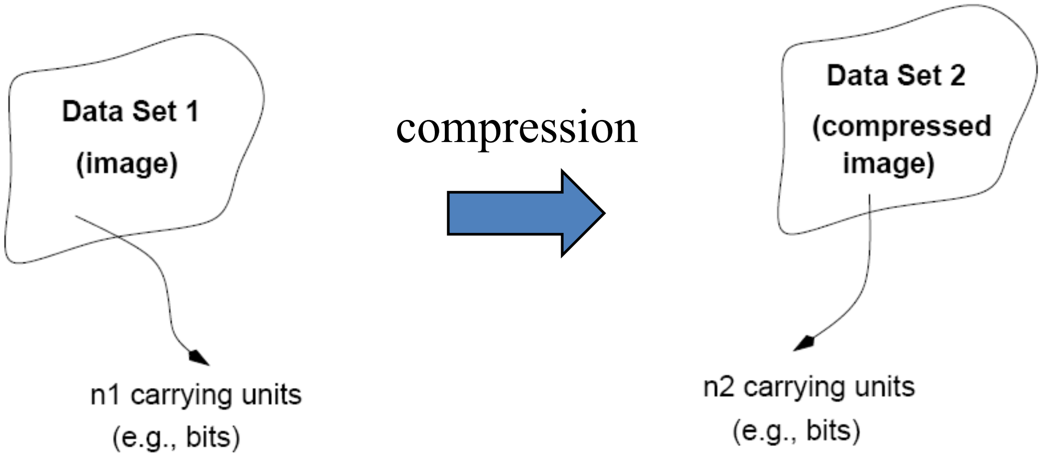
Image Compression

- Goal: Reduce amount of data required to represent a digital <u>image</u> (c.f. signal).



.. By exploiting redundancies in image data

Compression Ratio



Compression ratio: $C_R = \dfrac{n_1}{n_2}$

Relevant Data Redundancy

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

Example:

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10} = 0.9$

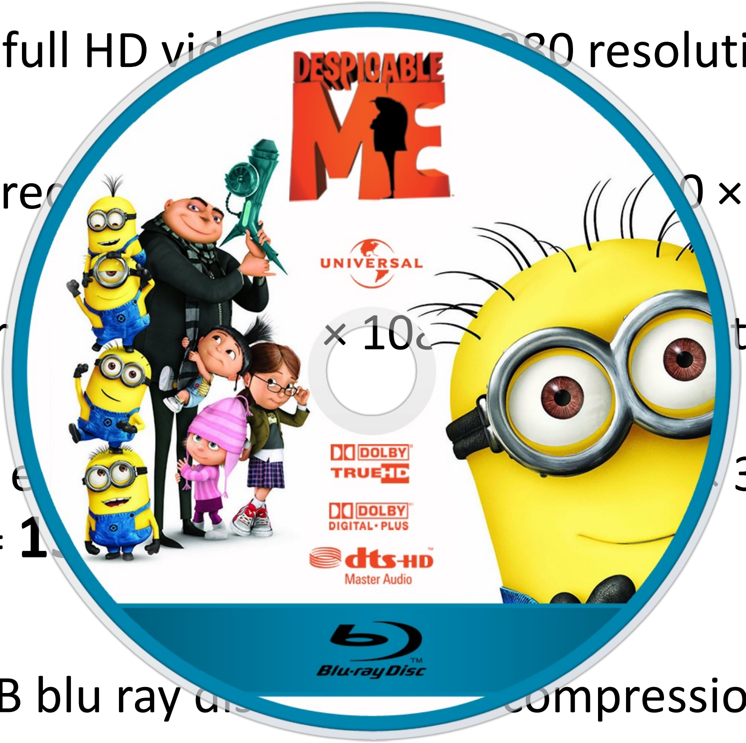
(90% of the data in dataset 1 is redundant)

if $n_2 = n_1$, then C_R=1, R_D=0

if $n_2 \ll n_1$, then $C_R \to \infty$, $R_D \to 1$

Motivation

- Consider a 2 hour, full HD vi ... 80 resolution@30fps)
- The storage space re ... 0 $\times$ 24 bits = 6.22 MB
- Space required per ... $\times$ 10 ... ts = 186.6 MB
- Space required for e ... 30 $\times$ 2 $\times$ 60 $\times$ 60 bits = 1.34×10^{12} bytes= **1**



- To put it on a 25 GB blu ray d ... ompression factor = **53.7**
- To put it in a 1GB mp4 file: required compression factor = **1343**

Diversion: bits and bytes, kilo vs kibi

- Previous convention: divide by 1024
- New convention: divide by 1000
- SI unit: kB (kilo = 1000)
- This is what causes a 4TB disk to have 3.7TiB!
- Also, note: B is not the same as b

Types of Redundancy

(1) Coding Redundancy

(2) Spatial/Temporal Redundancy

(3) Psychovisual Redundancy

- Image compression attempts to reduce one or more of these redundancy types.

Types of Redundancy

(1) Coding Redundancy

(2) Spatial/Temporal Redundancy

(3) Psychovisual Redundancy

- Image compression attempts to reduce one or more of these redundancy types.

Coding - Definitions

- Code: a list of symbols (letters, numbers, bits etc.)
- Code word: a sequence of symbols used to represent some information (e.g., gray levels).
- Code word length: number of symbols in a code word.

Example: (binary code, symbols: 0,1, length: 3)

0: 000	4: 100
1: 001	5: 101
2: 010	6: 110
3: 011	7: 111

Coding - Definitions (cont'd)

N x M image

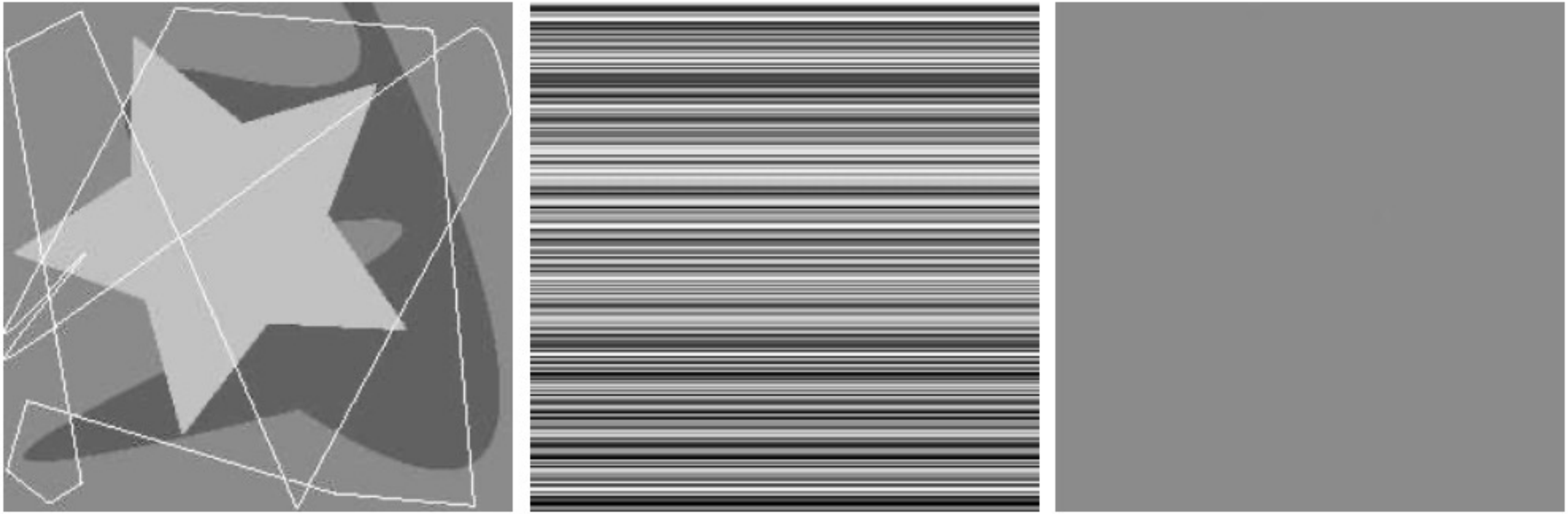
$\mathbf{r_k}$: k-th gray level

$\mathbf{l(r_k)}$: # of bits for r_k

$\mathbf{P(r_k)}$: probability of $\underbrace{r_k}_{\text{Discrete r.v. in the range [0,L-1]}}$

Average # of bits: $L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k)P(r_k)$

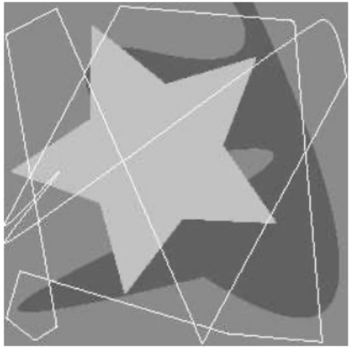
Total # of bits: NML_{avg}



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.)

Coding Redundancy



- <u>Case 1:</u> $l(r_k)$ = constant length

Example:

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

Average # of bits: $L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k)P(r_k)$

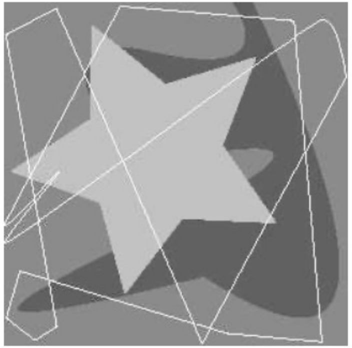
Total # of bits: NML_{avg}

Assume an image with $L = 8$

Assume $l(r_k) = 3$, $L_{avg} = \sum_{k=0}^{7} 3P(r_k) = 3\sum_{k=0}^{7} P(r_k) = 3$ bits

Total number of bits: $3NM$

Coding Redundancy (cont'd)



- <u>Case 2:</u> $l(r_k)$ = variable length

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

Table 6.1 Variable-Length Coding Example variable length

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

$$L_{avg} = \sum_{k=0}^{7} l(r_k)P(r_k) = 2.7 \text{ bits}$$

Total number of bits: 2.7NM

$$C_R = \frac{3}{2.7} = 1.11 \text{ (about 10\%)}$$

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

Types of Redundancy

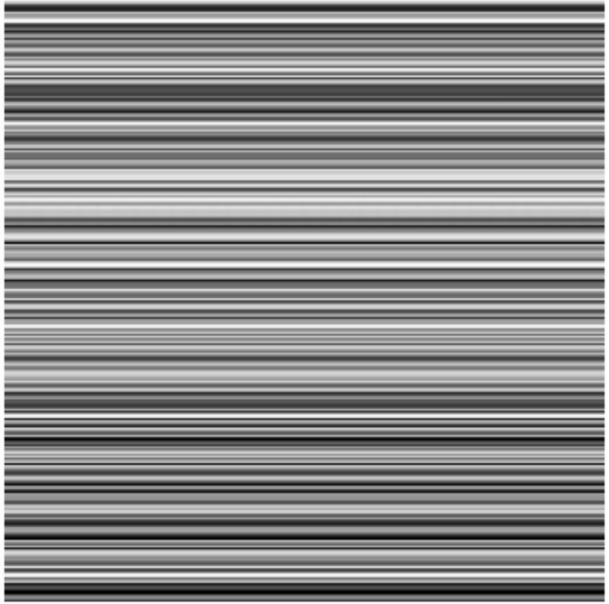
(1) Coding Redundancy

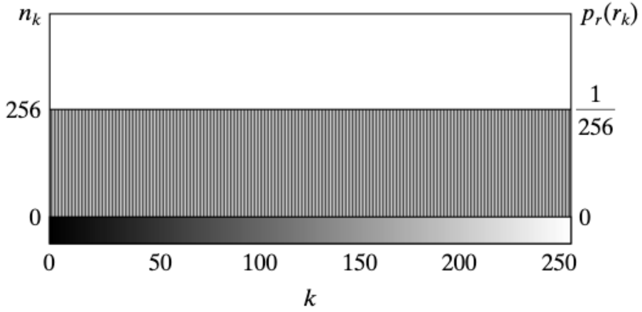
(2) Spatial/Temporal Redundancy

(3) Psychovisual Redundancy

- Image compression attempts to reduce one or more of these redundancy types.

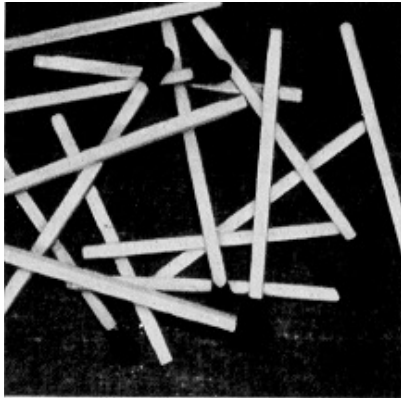
Spatial Redundancy

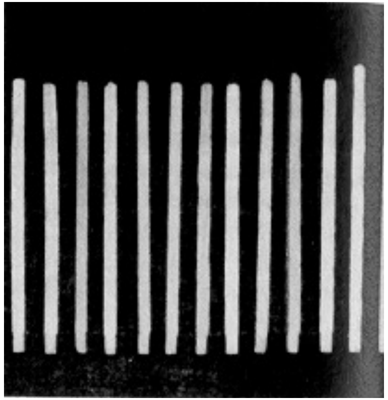




Spatial redundancy

- Interpixel redundancy exists ➔ pixel values are correlated
- i.e., a pixel value can be reasonably predicted by its neighbors





histograms

auto-correlation

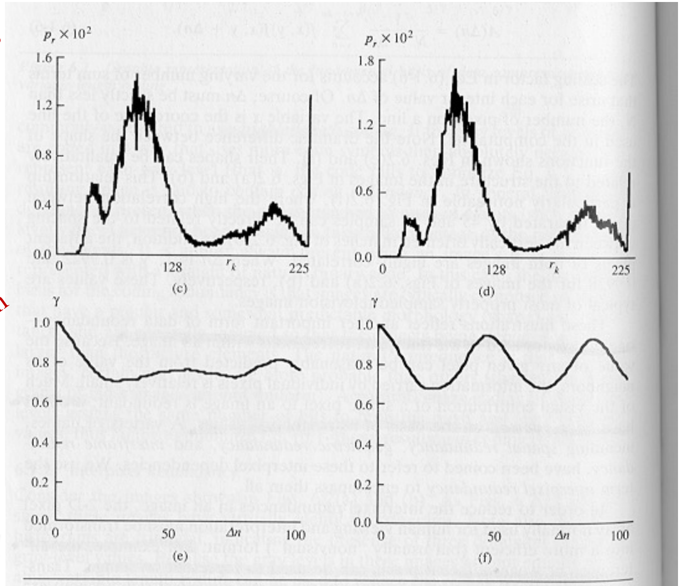


Figure 6.2 *Two images and their gray-level histograms and normalized autocorrelation*

$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f(x) g(x+a) da$$

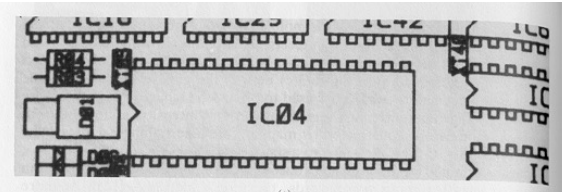
auto-correlation: f(x)=g(x)

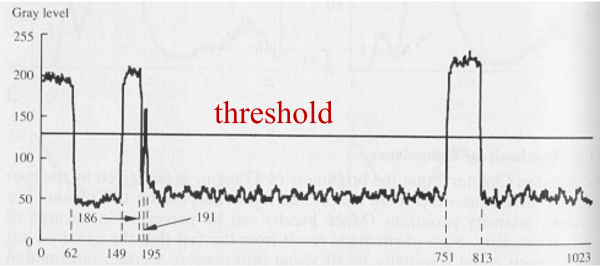
Interpixel redundancy (cont'd)

- To reduce interpixel redundancy, some kind of transformation must be applied on the data (e.g., thresholding, DFT, DWT)

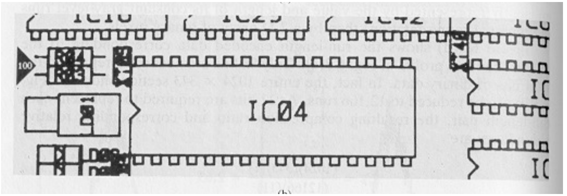
Example:

original





thresholded



11 ……………0000……………………...11…..000…..

Run-length encoding:

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

Using 11 bits/pair: (1+10) bits/pair

88 bits are required (compared to 1024 !!)

Spatial and temporal redundancy

frame t

frame t+1

Spatial and temporal redundancy

Types of Redundancy

(1) Coding Redundancy

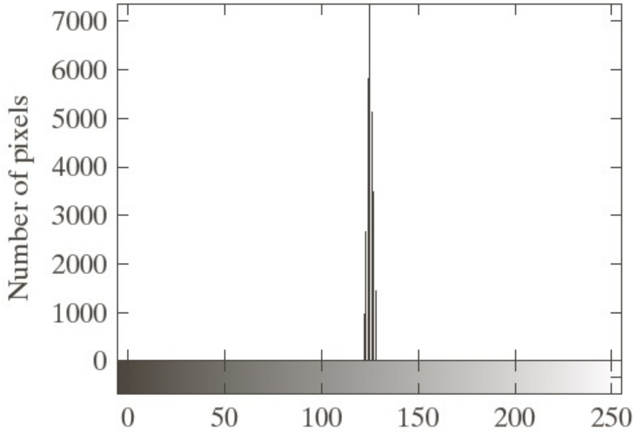
(2) Spatial/Temporal Redundancy

(3) Psychovisual Redundancy

- Image compression attempts to reduce one or more of these redundancy types.

Irrelevant information or perceptual redundancy

- Not all visual information is perceived by eye/brain, so throw away those that are not



Psychovisual redundancy (cont'd)

Example: quantization

256 gray levels

16 gray levels

16 gray levels + random noise

C=8/4 = 2:1

add a small pseudo-random number
to each pixel prior to quantization

Information theory

- Basic Premise: Generation of information can be treated as a probabilistic process defined over symbols.

- Symbol - carrier of information
- Consider a symbol with an occurrence probability p.
- The amount of information contained in the symbol is defined as:

$$I = \log_2 \frac{1}{p} \text{ bits} \quad \text{or} \quad I = -\log_2 p$$

Information theory: Entropy

- Consider a source that contains L possible symbols {s,i=0,1,2,…,L-1}
- With corresponding occurrence probabilities defined as {p_i, i=0,1,2,…,L−1}

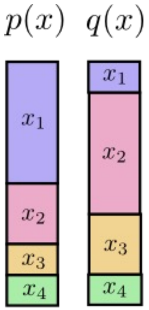
- **Entropy**

$$H = -\sum_{i=0}^{L-1} p_i \log_2 p_i$$

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

log(0.47) = -1.09
log(0.03) = -5.06

Slight Detour: Cross-entropy



Cross-Entropy: $H_p(q)$

Average Length
of message from q(x)
using code for p(x).

$$\hat{\mathbf{y}} = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix} \qquad D(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_j y_j \ln \hat{y}_j \qquad \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$H = -\sum_{i=0}^{L-1} p_i \log_2 p_i$$

http://colah.github.io/posts/2015-09-Visual-Information/

Information theory: Shannon's theorem

- Shannon's lossless source coding theorem: For a discrete, memoryless, stationary information source, the minimum bit rate required to encode a symbol on average is equal to the entropy of the source.

- In other words: we can't do better than the entropy

- Let's understand with an example

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

Types of Redundancy

(1) Coding Redundancy

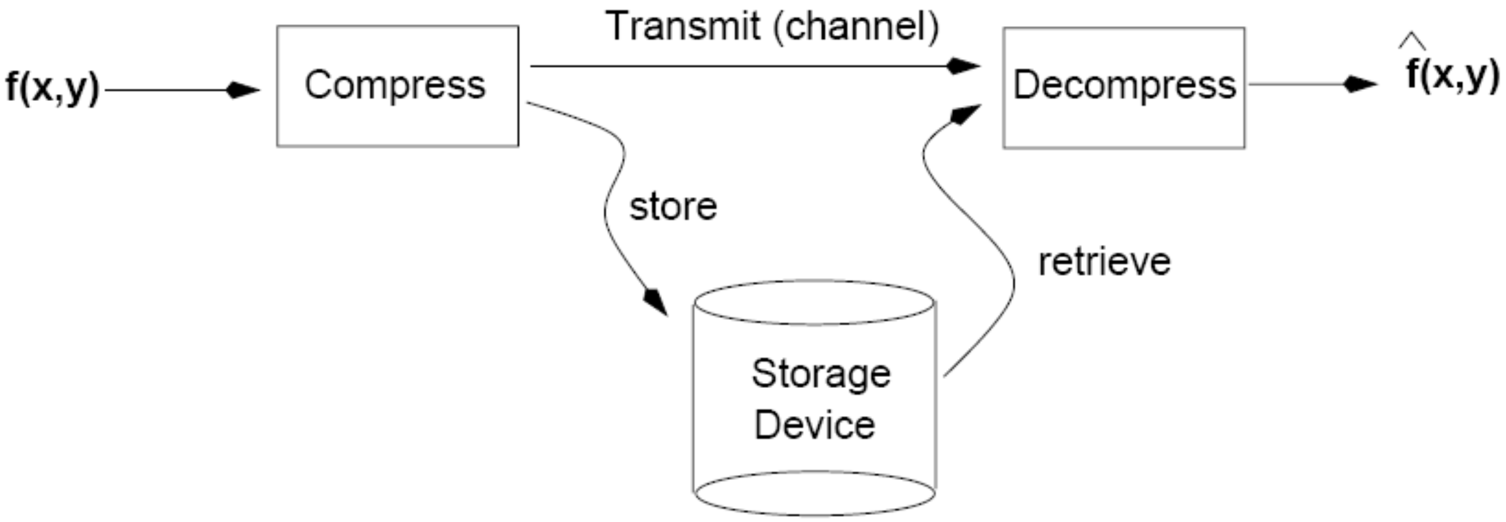
(2) Spatial/Temporal Redundancy

(3) Psychovisual Redundancy

- Image compression attempts to reduce one or more of these redundancy types.

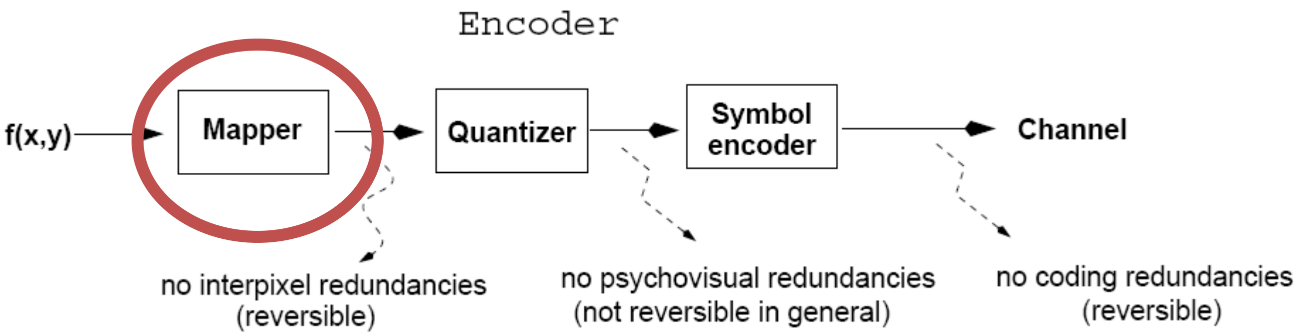
Image Compression

- Goal: Reduce amount of data required to represent a digital image (c.f. signal).



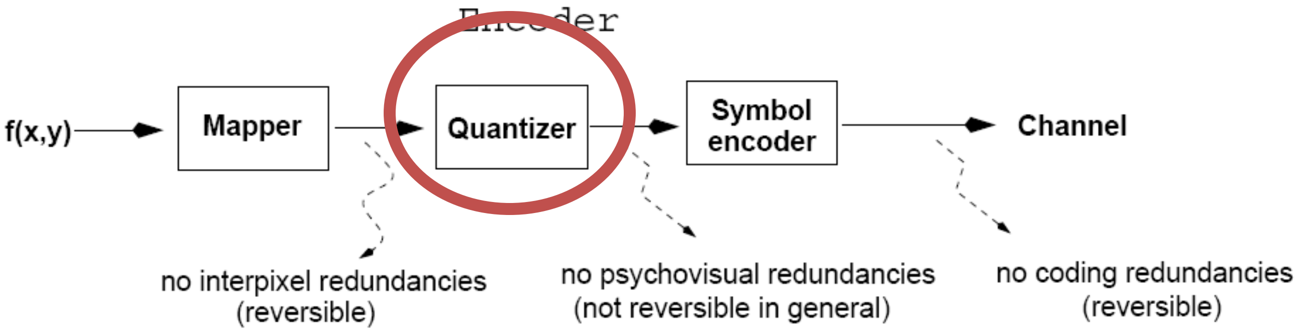
.. By exploiting redundancies in image data

Image Compression Model



- **Mapper:** transforms data to account for interpixel redundancies.

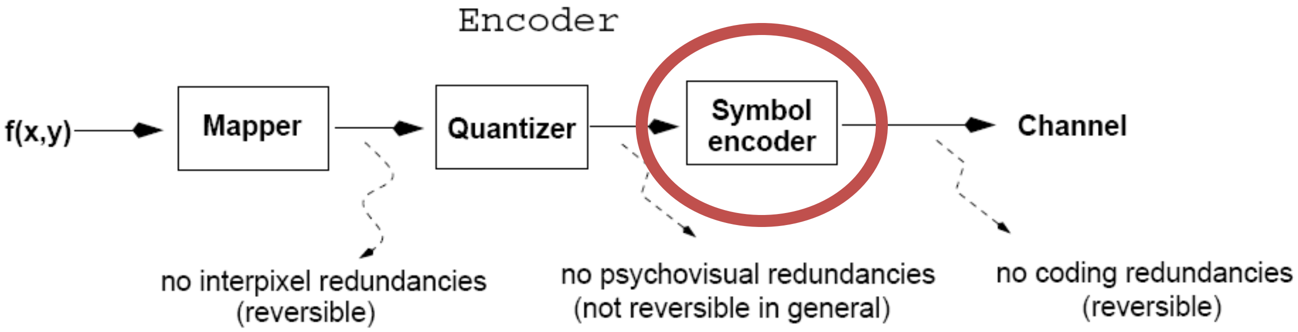
Image Compression Model (cont'd)



-

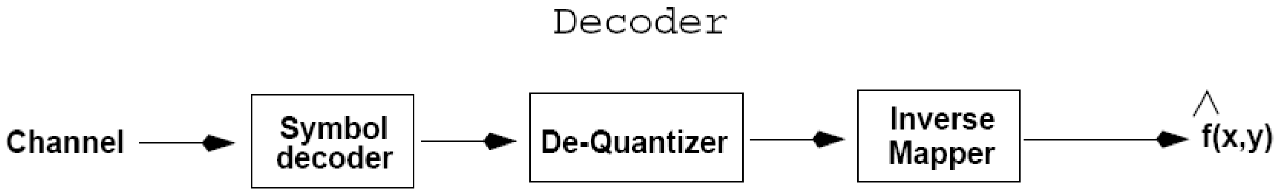
- **Quantizer:** quantizes the data to account for psychovisual redundancies.

Image Compression Model (cont'd)



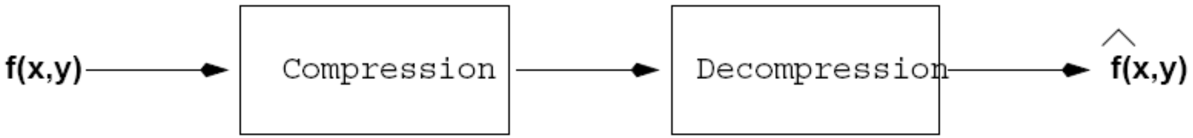
- **Symbol encoder:** encodes the data to account for coding redundancies.

Image Compression Models (cont'd)



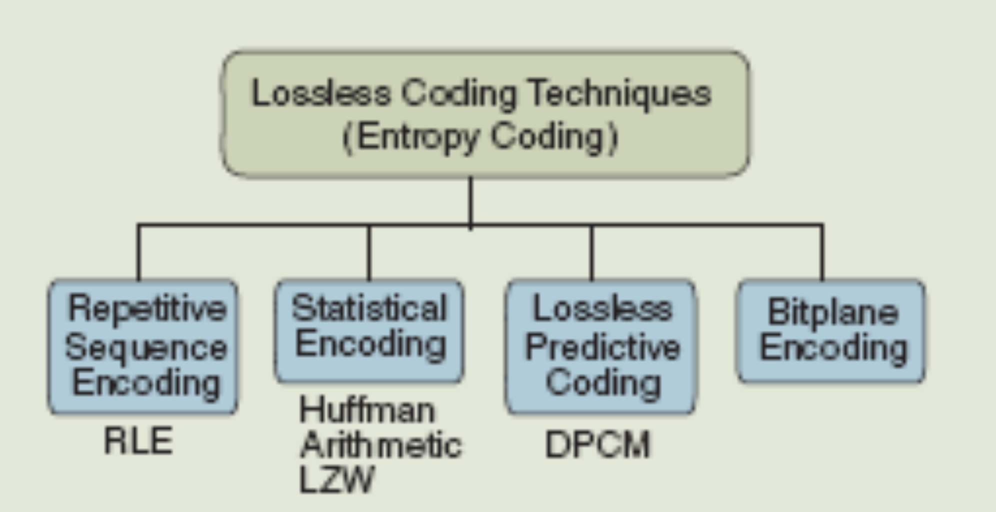
- The decoder applies the inverse steps.
- Note that quantization is irreversible in general.

Lossless Compression



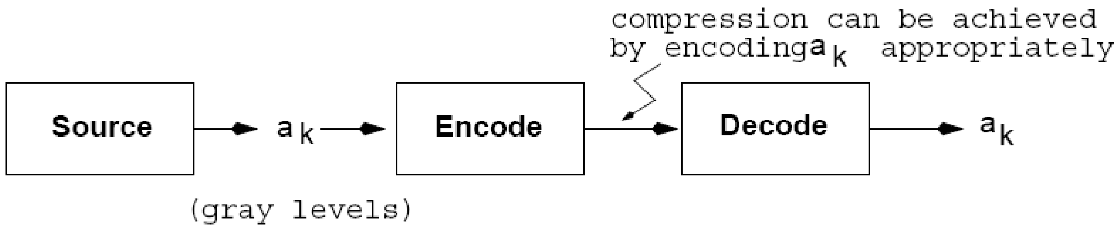
$$e(x, y) = \hat{f}(x, y) - f(x, y) = 0$$

Taxonomy of Lossless Methods



Huffman Coding

(addresses coding redundancy)



- A variable-length coding technique.
- Source symbols are encoded one at a time!
  - There is a one-to-one correspondence between source symbols and code words.
- Optimal code - minimizes code word length per source symbol.

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				

- ***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} assuming Huffman coding:

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

3x0.1 + 1x0.4 + 5x0.06 + 4x0.1 + 5x0.04 + 2x0.3=2.2 bits/symbol

- 6 symbols, we need a 3-bit code

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

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

https://youtu.be/CKbgVHg0w1U : Good video to understand this

Huffman Coding/Decoding

- Coding/Decoding can be implemented using a look-up table.
- Decoding can be done unambiguously.

$$\underbrace{0\ 1\ 0\ 1\ 0}_{a_3}\ \underbrace{0\ 1\ 1}_{a_1}\ \underbrace{1}_{a_2}\ \underbrace{1}_{a_2}\ \underbrace{0\ 0}_{a_6}$$

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								

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
(addresses coding redundancy)

- The main weakness of Huffman coding is that it encodes source symbols one at a time.
- Arithmetic coding encodes sequences of source symbols together.
  - There is no one-to-one correspondence between source symbols and code words.
- Slower than Huffman coding but can achieve better compression.

Arithmetic Coding (cont'd)

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

$$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4 \xrightarrow{} \underset{\text{sub-interval}}{[0.06752,\ 0.0688)} \xrightarrow{} \underset{\text{arithmetic code}}{0.068}$$

- Start with the interval [0, 1) ; a sub-interval is chosen to represent the message which becomes smaller and smaller as the number of symbols in the message increases.

Arithmetic Coding (cont'd)

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)

0 1

2) Subdivide [0, 1) based on the probabilities of α_i

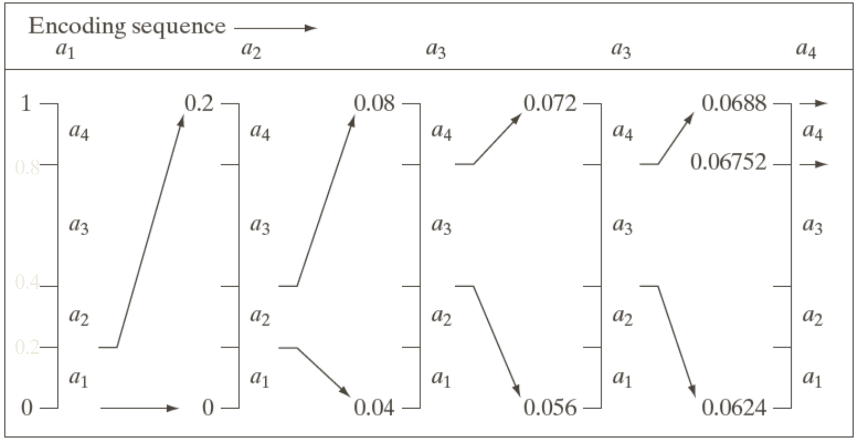
Initial Subinterval
[0.0, 0.2)
[0.2, 0.4)
[0.4, 0.8)
[0.8, 1.0)

3) Update interval by processing source symbols

Example

https://youtu.be/fZccqY-D2wU

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 sub-interval)

Example (cont'd)

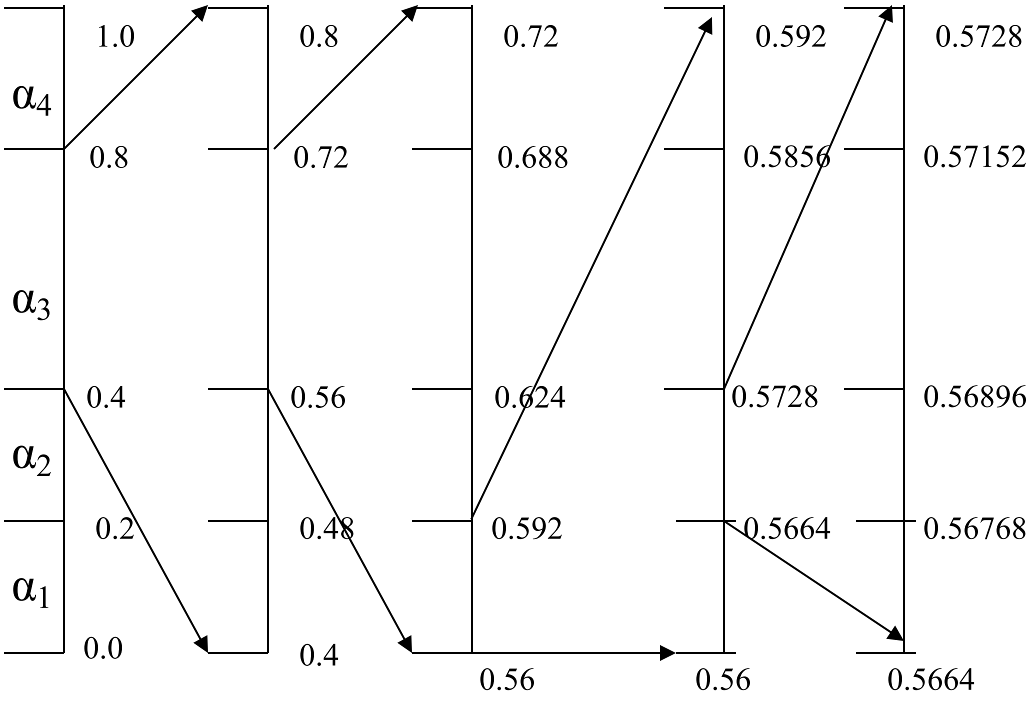
- The message $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$ is encoded using 3 decimal digits or 3/5 = 0.6 decimal digits per source symbol.

- The entropy of this message is: $H = -\sum_{k=0}^{3} P(r_k) log(P(r_k))$

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

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

Arithmetic Decoding



Decode 0.572

(code length=4)

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

Reference

- Ch 8, G&W textbook