Image compression

Radu Ranta

Ventspils University

Plan

1. Introduction
2. Information theory
3. Compression basics
4. Entropic coding
5. Inter-pixel coding
6. Quantization and thresholding
7. Color space transforms
8. Image transforms

Introduction

What is compression?

Avoid saying twice the same thing

Saying things differently, without changing the meaning

Keep to the essential, discard unimportant information

What is compression?

Avoid saying twice the same thing

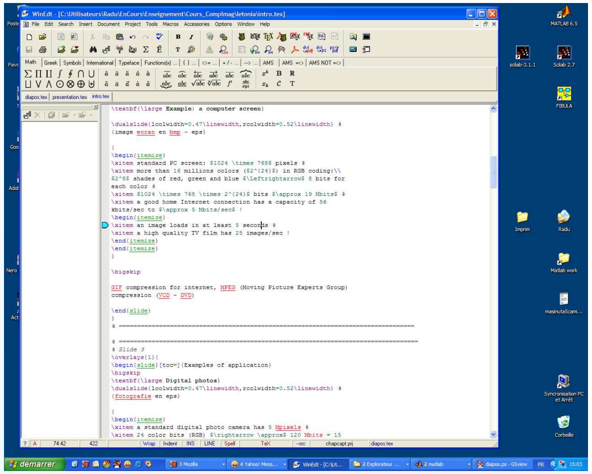
Saying things differently, without changing the meaning

Keep to the essential, discard unimportant information

Coding, Transforming, Approximating

Why compress ?

Example: a computer screen



- standard PC screen: 1024×768 pixels
- more than 16 millions colors (2^{24}) in RGB coding:
  2^8 shades of red, green and blue
  $\Leftrightarrow$ 8 bits for each color
- $1024 \times 768 \times 2^{24}$ bits $\approx$ 19 Mbits

A good home Internet connection has a capacity of 56 kbits/sec to ≈ 5 Mbits/sec !

- an image loads in at least 5 seconds
- a high quality TV film has 25 images/sec !

GIF compression for internet, MPEG (Moving Picture Experts Group) compression (VCD - DVD)

Examples of application

Digital photos

- a standard digital photo camera has 5 Mpixels
- 24 color bits (RGB) $\rightarrow \approx$ 120 Mbits = 15 Mbytes per photo

- standard SD cards have 64-512 Mbytes and transfer rate of 2 -10 Mbytes/sec

JPEG (Joint Photographic Experts Group) compression
JPEG 2000 compression

Examples of application

- a standard FBI fingerprint scanned at 500 dpi, $2^8 = 256$ grayscales $\rightarrow\approx$ 80 Mbits = 10 MBytes of data

- fingerprint cards since 1924, $\rightarrow\approx$ 200 million cards $\rightarrow\approx$ 2000 Terrabytes
- 30000-50000 new cards PER DAY, to send by network connection and to compare with . . .
- $\approx$ 29 million records of “usual suspects” !

Wavelet compression

Compression types

- 1.000.000.000 $\rightarrow 10^9$

Compression types

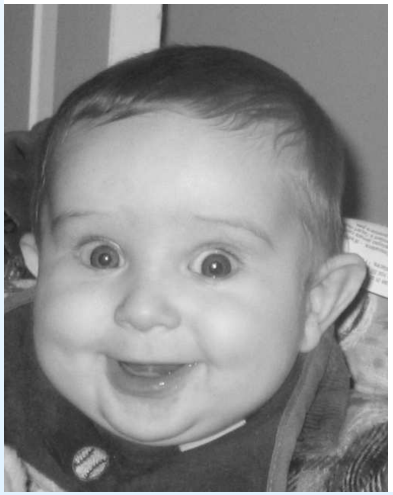
- 1.000.000.000 $\longrightarrow 10^9$
- Thank you $\longrightarrow$ *Merci* (9 characters $\longrightarrow$ 5 characters)
  Save Our Souls $\longrightarrow$ *S.O.S.* $\longrightarrow \cdots - - - \cdots$

Turn right $\longrightarrow$ 

Compression types

- 1.000.000.000 $\longrightarrow 10^9$
- Thank you $\longrightarrow$ *Merci* (9 characters $\longrightarrow$ 5 characters)
  Save Our Souls $\longrightarrow$ *S.O.S.* $\longrightarrow \cdot\cdot\cdot - - - \cdot\cdot\cdot$

Turn right $\longrightarrow$ 

-  $\longrightarrow$ 

Data and information

Information – The useful part of a message

- the color of a sheet of paper
- the frequency and the duration of a sound
- the length and the position of a straight line

Data and information

Information – The useful part of a message

- the color of a sheet of paper
- the frequency and the duration of a sound
- the length and the position of a straight line

Message – The coding of the information

- the character string {r-e-d} in English ({r-o-u-g-e} in French), the numbers 255 - 0 - 0 in RGB, . . .
- the musical partition, Short Time Fourier Transform, . . .
- the coordinates of (x_1, y_1) and (x_2, y_2), the origin, module and phase of the vector

Data and information

Information – The useful part of a message

- the color of a sheet of paper
- the frequency and the duration of a sound
- the length and the position of a straight line

Message – The coding of the information

- the character string {r-e-d} in English ({r-o-u-g-e} in French), the numbers 255 - 0 - 0 in RGB, . . .
- the musical partition, Short Time Fourier Transform, . . .
- the coordinates of (x_1, y_1) and (x_2, y_2), the origin, module and phase of the vector

Data – The physical support of the information

- the binarized image of the word "red", the ASCII codes of r, e, and d, the binary codes of 255 - 0 - 0, . . .
- the binarized recording and the STFT algorithm, the binarized image of the musical partition, . . .
- the binary codes of the coordinates values, . . .

Compression challenge

Keep as much information as possible and diminish data

Changing data (physical compression)

Smaller amount of data by code changing

Characteristics:

- lossless compression
- diminishes the amount of data
- perfect reconstruction

Compression challenge

Keep as much information as possible and diminish data

Changing messages (logical compression)

Smaller messages using *transforms*. Problem: the transform

- must be reversible
- must be understood by the user
- must be safe (for transmission, storage, security)

Characteristics:

- lossless compression
- can diminish or increase the amount of data

Compression challenge

Keep as much information as possible and diminish data

Changing information (approximation)

Keeping only essential information

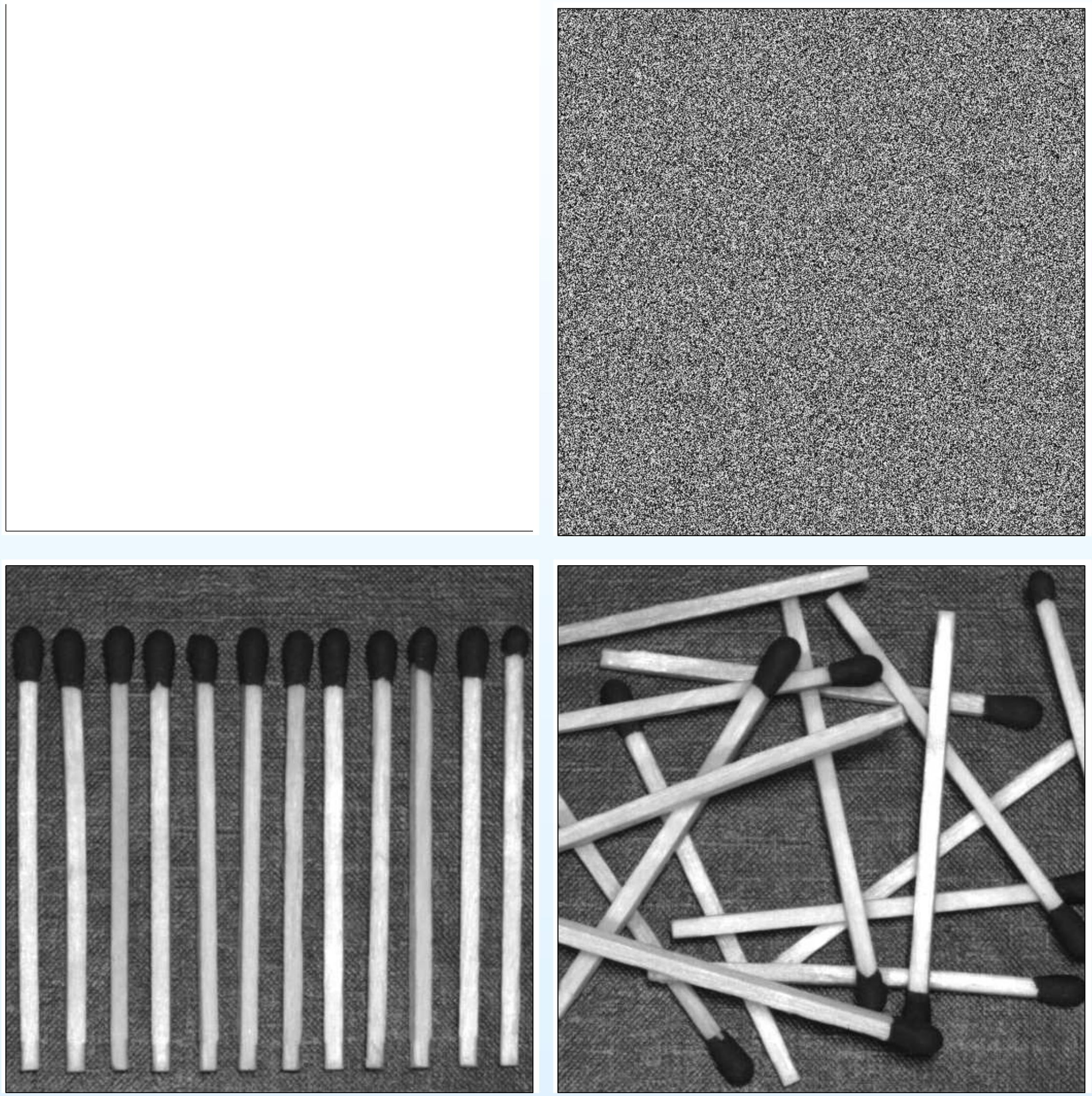
Characteristics:

- lossy compression
- diminishes the amount of data
- unrecoverable (approximate reconstruction)

Information Theory

Image and information

Similar images 100×100 pixels, 256 greylevels (10 kBytes)



Measuring information

The information is a measure of uncertainty : more an event is probable, less it is informative

Example (1)

- Consider a white image
- Chose a pixel (random) and name its color
- The probability of saying “white” is $p(white) = 1$
- The answer is “Obviously!” (the information is 0)

Measuring information

The information is a measure of uncertainty : more an event is probable, less it is informative

Example (1)

- Consider a white image
- Chose a pixel (random) and name its color
- The probability of saying “white” is $p(white) = 1$
- The answer is “Obviously!” (the information is 0)

Example (2)

- Consider a 10×10 pixels white image with a randomly positioned black pixel
- Chose a pixel (random) and name its color
- The probability of saying “white” is $p(white) = 0.99$
- The answer is “Well, I was quite sure” (not very informative)
- The probability of saying “black” is $p(black) = 0.01$
- The answer is “So you found it!” (that’s an information!)

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors
- The probability of saying “one is white” is $p(white) = 0.97$

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors
- The probability of saying “one is white” is $p(white) = 0.97$
- The probability of saying “one is red” is $p(red) = 0.01$

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors
- The probability of saying “one is white” is $p(white) = 0.97$
- The probability of saying “one is red” is $p(red) = 0.01$
- The probability of saying “one is blue” is $p(blue) = 0.02$

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors
- The probability of saying “one is white” is $p(white) = 0.97$
- The probability of saying “one is red” is $p(red) = 0.01$
- The probability of saying “one is blue” is $p(blue) = 0.02$
- The probability of the event “one is red and the other is blue” is $p(red, blue) = p(red) \cdot p(blue) = 2 \cdot 10^{-4}$

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors
- The probability of saying “one is white” is $p(white) = 0.97$
- The probability of saying “one is red” is $p(red) = 0.01$
- The probability of saying “one is blue” is $p(blue) = 0.02$
- The probability of the event “one is red and the other is blue” is $p(red, blue) = p(red) \cdot p(blue) = 2 \cdot 10^{-4}$
- You give more information (and the probability diminishes)

Measuring information

Example (3)

- Consider a 10×10 pixels white image with a randomly positioned red pixel and a two randomly positioned blue pixels
- Chose, independently, two pixels and name their colors
- The probability of saying “one is white” is $p(white) = 0.97$
- The probability of saying “one is red” is $p(red) = 0.01$
- The probability of saying “one is blue” is $p(blue) = 0.02$
- The probability of the event “one is red and the other is blue” is $p(red, blue) = p(red) \cdot p(blue) = 2 \cdot 10^{-4}$
- You give more information (and the probability diminishes)

The information (about an event) is a function of the probability (of the event): can we find a “good” function $f(p)$?

Measuring information

Which function ?

Characteristics:

- The two individual information should sum :

$$I(red, blue) = I(red) + I(blue) = f\left(p(red)\right) + f\left(p(blue)\right)$$

- For a sure event (see example 1), the information should be 0:

$$I_1(white) = f\left(p(white)\right) = f(1) = 0$$

Measuring information

Which function ?
Characteristics:

- The two individual information should sum :

$$I(red, blue) = I(red) + I(blue) = f\left(p(red)\right) + f\left(p(blue)\right)$$

- For a sure event (see example 1), the information should be 0:

$$I_1(white) = f\left(p(white)\right) = f(1) = 0$$

Definition:
For an event E, the information is

$$I(E) = -\lambda \log p(E)$$

Measuring information

Defining the unit of information:

Simplest case: flipping a coin

- two equiprobable events : $p(head) = p(tail) = \frac{1}{2}$
- the quantity of information about such un event will be defined as *a unit* of information:

$$I(tail) = I(head) = -\lambda \log \frac{1}{2} = 1$$

- take base two logarithm $\Rightarrow \lambda = 1$

Measuring information

Defining the unit of information:

Simplest case: flipping a coin

- two equiprobable events : $p(head) = p(tail) = \frac{1}{2}$
- the quantity of information about such un event will be defined as *a unit* of information:

$$I(tail) = I(head) = -\lambda \log \frac{1}{2} = 1$$

- take base two logarithm $\Rightarrow \lambda = 1$

For a binary equiprobable event, the information $= 1$

The measuring unit of the information is the bit

Information system

Shannon information diagram

- Source: image, sound, file, . . .
- Channel: radio, Ethernet, CDs, HDD, . . .
- User: human user, informatic system, industrail machine,. . .

Information system

Source:
- emits data
- first step: sampling and digitizing $\longrightarrow$ succession of binary digits (bits)
- groups bits in *symbols*, creating an *alphabet*
- optionally: codes symbols to form a optimal message
- source coding $\leftrightarrow$ compression

Information system

Source:
- emits data
- first step: sampling and digitizing $\longrightarrow$ succession of binary digits (bits)
- groups bits in *symbols*, creating an *alphabet*
- optionally: codes symbols to form a optimal message
- source coding $\leftrightarrow$ compression

Channel:
- transmits the the message as coded data
- can be noisy !
- optionally: the message can be re-coded to reduce the effects of noise
- channel coding $\leftrightarrow$ correction codes

Information system

Source:
- emits data
- first step: sampling and digitizing $\longrightarrow$ succession of binary digits (bits)
- groups bits in *symbols*, creating an *alphabet*
- optionally: codes symbols to form a optimal message
- source coding $\longleftrightarrow$ compression

Channel:
- transmits the the message as coded data
- can be noisy !
- optionally: the message can be re-coded to reduce the effects of noise
- channel coding $\longleftrightarrow$ correction codes

User:
- receives data
- decodes the message and extract the information

Symbols, Bits, Codes

Symbol: elementary part of a message

Bit: unit of information $\{0, 1\}$

Code: the expression of a symbol in bits

Symbols, Bits, Codes

Symbol: elementary part of a message

Bit: unit of information $\{0, 1\}$

Code: the expression of a symbol in bits

Example (1):

- Message: black and white image
- Symbol: color of a pixel = “black”, “white”
- Codes: “black”=0, “white”=1

Symbols, Bits, Codes

Symbol: elementary part of a message

Bit: unit of information $\{0, 1\}$

Code: the expression of a symbol in bits

Example (1):

- Message: black and white image
- Symbol: color of a pixel = “black”, “white”
- Codes: “black”=0, “white”=1

Example (2):

- Message: black and white image
- Symbol: colors of two consecutive pixels = “black-black”, “black-white”, . . .
- Codes:

Symbols, Bits, Codes

Symbol: elementary part of a message

Bit: unit of information $\{0, 1\}$

Code: the expression of a symbol in bits

Example (1):
- Message: black and white image
- Symbol: color of a pixel = “black”, “white”
- Codes: “black”=0, “white”=1

Example (2):
- Message: black and white image
- Symbol: colors of two consecutive pixels = “black-black”, “black-white”, . . .
- Codes:
  - 00, 01, 10, 11

Symbols, Bits, Codes

Symbol: elementary part of a message

Bit: unit of information $\{0, 1\}$

Code: the expression of a symbol in bits

Example (1):

- Message: black and white image
- Symbol: color of a pixel = “black”, “white”
- Codes: “black”=0, “white”=1

Example (2):

- Message: black and white image
- Symbol: colors of two consecutive pixels = “black-black”, “black-white”, . . .
- Codes:
  - 00, 01, 10, 11
  - 0, 10, 110, 1110

Symbols, Bits, Codes

Symbol: elementary part of a message

Bit: unit of information $\{0, 1\}$

Code: the expression of a symbol in bits

Example (1):

- Message: black and white image
- Symbol: color of a pixel = “black”, “white”
- Codes: “black”=0, “white”=1

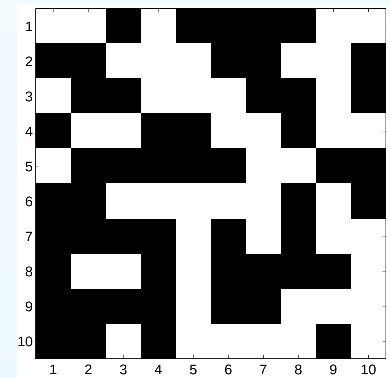
Example (2):

- Message: black and white image
- Symbol: colors of two consecutive pixels = “black-black”, “black-white”, . . .
- Codes:
  - 00, 01, 10, 11
  - 0, 10, 110, 1110
  - . . .

Entropy

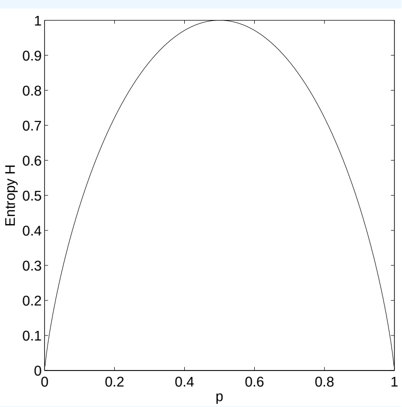
Consider a 10×10 black and white image (source) X:

- The message consists of *symbols* coded on 1 bit $x_i = \{0, 1\} \Rightarrow$ the source X emits 100 symbols
- The color of a pixel does not depend on the previous pixels $\rightarrow$ *zero-memory* source
- Then, for each symbol $x_i = \{0, 1\}$:



x_i	$p(x_i)$	$I(x_i) = -\log p(x_i)$
0	p	$-\log p$
1	$1-p$	$-\log(1-p)$

The mean information of the source is:



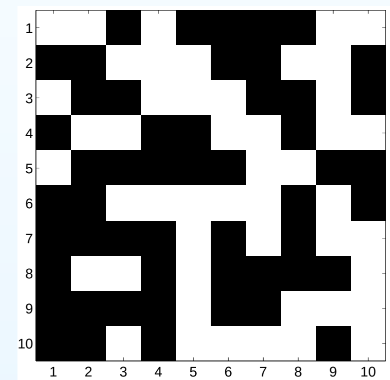
$$H(X) = \sum_{i=0}^{1} I(x_i)p(x_i) = -\sum_{i=0}^{1} p(x_i)\log p(x_i)$$

$$= -p\log p - (1-p)\log(1-p)$$

Entropy

Consider a 10×10 black and white image (source) X:

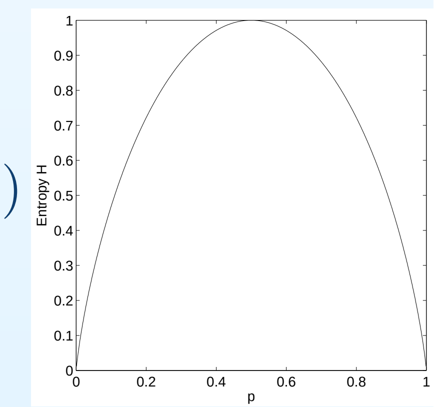
- The message consists of *symbols* coded on 1 bit $x_i = \{0, 1\} \Rightarrow$ the source X emits 100 symbols
- The color of a pixel does not depend on the previous pixels $\rightarrow$ *zero-memory* source
- Then, for each symbol $x_i = \{0, 1\}$:



x_i	$p(x_i)$	$I(x_i) = -\log p(x_i)$
0	p	$-\log p$
1	$1-p$	$-\log(1-p)$

The mean information of the source is:

$$H(X) = \sum_{i=0}^{1} I(x_i)p(x_i) = -\sum_{i=0}^{1} p(x_i)\log p(x_i)$$
$$= -p\log p - (1-p)\log(1-p)$$



The mean information of a source is called entropy.

Source entropy

Example

- Consider a 10×10 pixels white image with two randomly positioned black pixels

Source entropy

Example

- Consider a 10×10 pixels white image with two randomly positioned black pixels
- Then:

x_i	$p(x_i)$
1	0.98
0	0.02

$$I(0) = -\log p(0) = -\log 0.02 = 5.64$$

$$I(1) = -\log p(1) = -\log 0.98 = 0.029$$

Source entropy

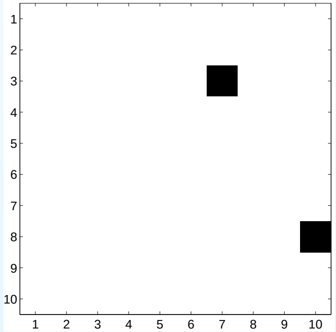
Example

- Consider a 10×10 pixels white image with two randomly positioned black pixels
- Then:

x_i	$p(x_i)$
1	0.98
0	0.02

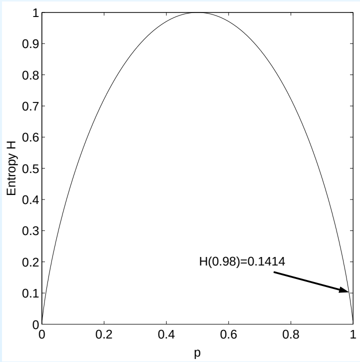
$$I(0) = -\log p(0) = -\log 0.02 = 5.64$$

$$I(1) = -\log p(1) = -\log 0.98 = 0.029$$



The entropy of the source is:

$$H(X) = \sum_{i=0}^{1} I(x_i)p(x_i) = -\sum_{i=0}^{1} p(x_i)\log p(x_i) = 0.1414$$



Mean information of 0.1414 bits/symbol
($\Leftrightarrow$ 0.1414 bits/pixel)

Source extensions

Example: same image, different alphabet

- Consider *symbols* representing sequences of two pixels

Source extensions

Example: same image, different alphabet

- Consider *symbols* representing sequences of two pixels
- The source S (zero-memory) emits 50 symbols

Source extensions

Example: same image, different alphabet

- Consider *symbols* representing sequences of two pixels
- The source S (zero-memory) emits 50 symbols
- Then, if we code each symbol using 2 bits $s_k = x_i x_j$:

s_k	$p(s_k)$	$I(s_k) = -\log p(s_k)$
00	$p(00) = p(0)p(0) = 0.9604$	0.0583
01	$p(01) = p(0)p(1) = 0.0196$	5.673
10	$p(10) = p(1)p(0) = 0.0196$	5.673
11	$p(11) = p(1)p(1) = 0.0004$	11.2877

Source extensions

Example: same image, different alphabet

- Consider *symbols* representing sequences of two pixels
- The source S (zero-memory) emits 50 symbols
- Then, if we code each symbol using 2 bits $s_k = x_i x_j$:

s_k	$p(s_k)$	$I(s_k) = -\log p(s_k)$
00	$p(00) = p(0)p(0) = 0.9604$	0.0583
01	$p(01) = p(0)p(1) = 0.0196$	5.673
10	$p(10) = p(1)p(0) = 0.0196$	5.673
11	$p(11) = p(1)p(1) = 0.0004$	11.2877

- The entropy is:

$$H(S) = \sum_{k=1}^{4} I(s_k)p(s_k) = -\sum_{k=1}^{4} p(s_k) \log p(s_k) = 0.2829 \text{ bits/symbol}$$

n-bits symbols $\Rightarrow$ *n-th extension of the source: H(S)=nH(X)*

Noiseless coding theorem

Shannon's First Theorem

- Consider a zero-memory source S who outputs 2^n symbols s_i $(i = 1..2^n)$
- The information unit is the bit (1/0)
- In natural coding, the length of a symbol in bits is $l_i = n$
- The information of each s_i is $I(s_i) = -\log p(s_i)$

Non-optimal solution, because *the number of bits per symbol should be proportional to the information*

Noiseless coding theorem

Shannon's First Theorem

- Consider a zero-memory source S who outputs 2^n symbols s_i $(i = 1..2^n)$
- The information unit is the bit (1/0)
- In natural coding, the length of a symbol in bits is $l_i = n$
- The information of each s_i is $I(s_i) = -\log p(s_i)$

Non-optimal solution, because *the number of bits per symbol should be proportional to the information*

Theorem: *The coding of a source can be modified, so symbols can be coded using different number of bits* $(l_i = n \to l'_i)$*. The average length of a symbol has an inferior bound given by the mean information of* S*:*

$$L' = \sum_{i=1}^{n} p(s_i) l'_i \geq -\sum_{i=1}^{n} p(s_i) \log p(s_i) = H(S) = nH(X)$$

$$L' \geq nH(X) = H(S)$$

Coding efficiency

When n increases ($n \to \infty$), the average symbol length decreases towards an optimal value:

$$L_O = \lim_{n\to\infty} \frac{L'}{n} \to H(S) \text{ bits/symbol}$$

In theory, one can choose symbols of average length $\ll n$!

Example:

- Consider a source S emitting symbols s_i of length $L = l_i = n$
  $\to s =$“two consecutive pixels”
  $\to$ natural coding $\{00, 01, 10, 11\}$
- Considering the first Shannon theorem, there exists a different coding s' with $L' \ll n$

Coding efficiency

When n increases ($n \to \infty$), the average symbol length decreases towards an optimal value:

$$L_O = \lim_{n \to \infty} \frac{L'}{n} \to H(S) \text{ bits/symbol}$$

In theory, one can choose symbols of average length $\ll n$!

Example:

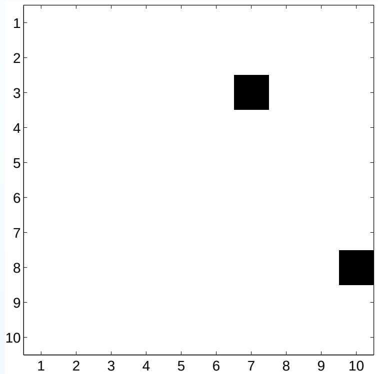
- Consider a source S emitting symbols s_i of length $L = l_i = n$
  $\to s =$"two consecutive pixels"
  $\to$ natural coding $\{00, 01, 10, 11\}$
- Considering the first Shannon theorem, there exists a different coding s' with $L' \ll n$

$\Rightarrow$ changing the code can diminish the number of bits used for an information $\Rightarrow$ Compression

$$\text{Compression ratio } C = \frac{L}{L'}$$

Coding efficiency

Example:



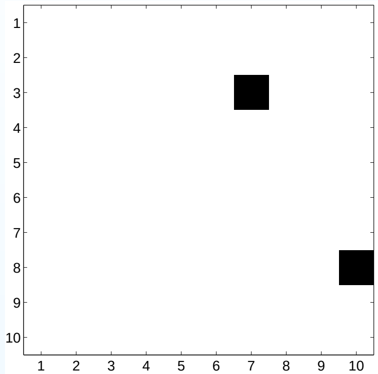
s	$p(s_i)$	Code A	Code B	l_A	l_B
s_1	0.9604	00	0	2	1
s_2	0.0196	01	10	2	2
s_3	0.0196	10	110	2	3
s_4	0.0004	11	1110	2	4

00	00	00	00	00
00	00	00	00	00
00	00	00	10	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	01
00	00	00	00	00
00	00	00	00	00

0	0	0	0	0
0	0	0	0	0
0	0	0	110	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	10
0	0	0	0	0
0	0	0	0	0

Coding efficiency

Example:



s	$p(s_i)$	Code A	Code B	l_A	l_B
s_1	0.9604	00	0	2	1
s_2	0.0196	01	10	2	2
s_3	0.0196	10	110	2	3
s_4	0.0004	11	1110	2	4

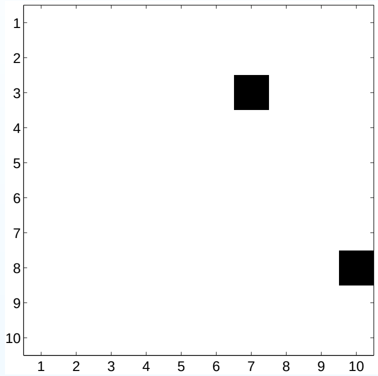
00	00	00	00	00
00	00	00	00	00
00	00	00	10	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	01
00	00	00	00	00
00	00	00	00	00

0	0	0	0	0
0	0	0	0	0
0	0	0	110	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	10
0	0	0	0	0
0	0	0	0	0

$$H(S) = -\sum_{i=1}^{4} p(s_i) \log p(s_i) = 0.2829$$

Coding efficiency

Example:



s	$p(s_i)$	Code A	Code B	l_A	l_B
s_1	0.9604	00	0	2	1
s_2	0.0196	01	10	2	2
s_3	0.0196	10	110	2	3
s_4	0.0004	11	1110	2	4

00	00	00	00	00
00	00	00	00	00
00	00	00	10	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	01
00	00	00	00	00
00	00	00	00	00

0	0	0	0	0
0	0	0	0	0
0	0	0	110	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	10
0	0	0	0	0
0	0	0	0	0

$$H(S) = -\sum_{i=1}^{4} p(s_i) \log p(s_i) = 0.2829$$

$$L_A = \sum_{i=1}^{4} p(s_i) l_{A,i} = 2 \qquad L_B = \sum_{i=1}^{4} p(s_i) l_{B,i} = 1.06$$

Coding efficiency

$$
\begin{aligned}
L_A &= 2 \text{ bits/symbol} \gg H(S) = 0.2829 \\
L_{A,p} &= 1 \text{ bit/pixel} \gg H(X) = 0.1414 \\
L_B &= 1.06 \text{ bits/symbol} > H(S) = 0.2829 \\
L_{B,p} &= 0.53 \text{ bit/pixel} > H(X) = 0.1414
\end{aligned}
$$

The optimal coding solution should be:

$$L_O \rightarrow H(S) \Leftrightarrow L_{O,p} \rightarrow H(X)$$

For each symbol s_i, consider $I(s_i) \leq \widehat{l_{O,i}} \leq I(s_i) + 1$

Coding efficiency

$$L_A = 2 \text{ bits/symbol} \gg H(S) = 0.2829$$

$$L_{A,p} = 1 \text{ bit/pixel} \gg H(X) = 0.1414$$

$$L_B = 1.06 \text{ bits/symbol} > H(S) = 0.2829$$

$$L_{B,p} = 0.53 \text{ bit/pixel} > H(X) = 0.1414$$

The optimal coding solution should be:

$$L_O \rightarrow H(S) \Leftrightarrow L_{O,p} \rightarrow H(X)$$

For each symbol s_i, consider $I(s_i) \leq \widehat{l_{O,i}} \leq I(s_i) + 1$

Then:

s_i	$p(s_i)$	$I(s_i)$	Code A	Code B	$l_{A,i}$	$l_{B,i}$	$\widehat{l_{O,i}}$
s_1	0.9604	0.0583	00	0	2	1	1
s_2	0.0196	5.673	01	10	2	2	6
s_3	0.0196	5.673	10	110	2	3	6
s_4	0.0004	11.2877	11	1110	2	4	12
		0.2829			2	1.06	1.2
		0.1414			1	0.53	0.6

Coding efficiency

$$L_A = 2 \text{ bits/symbol} \gg H(S) = 0.2829$$
$$L_{A,p} = 1 \text{ bit/pixel} \gg H(X) = 0.1414$$
$$L_B = 1.06 \text{ bits/symbol} > H(S) = 0.2829$$
$$L_{B,p} = 0.53 \text{ bit/pixel} > H(X) = 0.1414$$

The optimal coding solution should be:

$$L_O \rightarrow H(S) \Leftrightarrow L_{O,p} \rightarrow H(X)$$

For each symbol s_i, consider $I(s_i) \leq \widehat{l_{O,i}} \leq I(s_i) + 1$

Then:

s_i	$p(s_i)$	$I(s_i)$	Code A	Code B	$l_{A,i}$	$l_{B,i}$	$\widehat{l_{O,i}}$
s_1	0.9604	0.0583	00	0	2	1	1
s_2	0.0196	5.673	01	10	2	2	6
s_3	0.0196	5.673	10	110	2	3	6
s_4	0.0004	11.2877	11	1110	2	4	12
		0.2829			2	1.06	1.2
		0.1414			1	0.53	0.6

Optimal length : $L_{O,p} > L_{B,p}$???

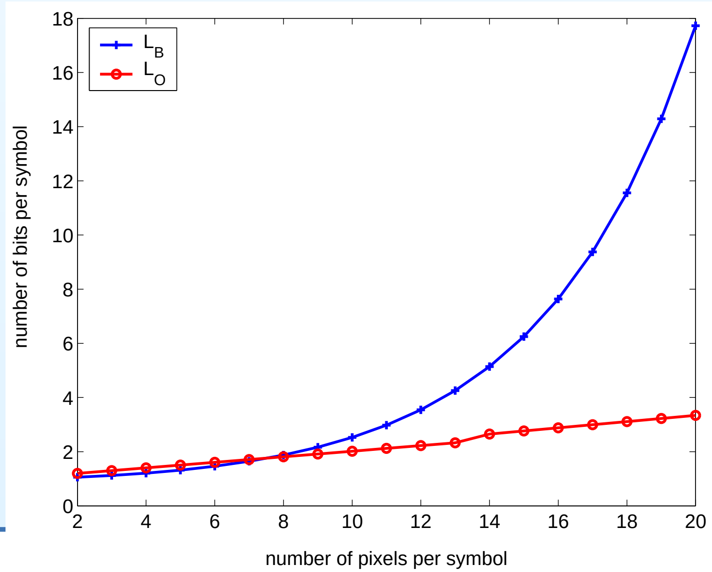
Coding efficiency

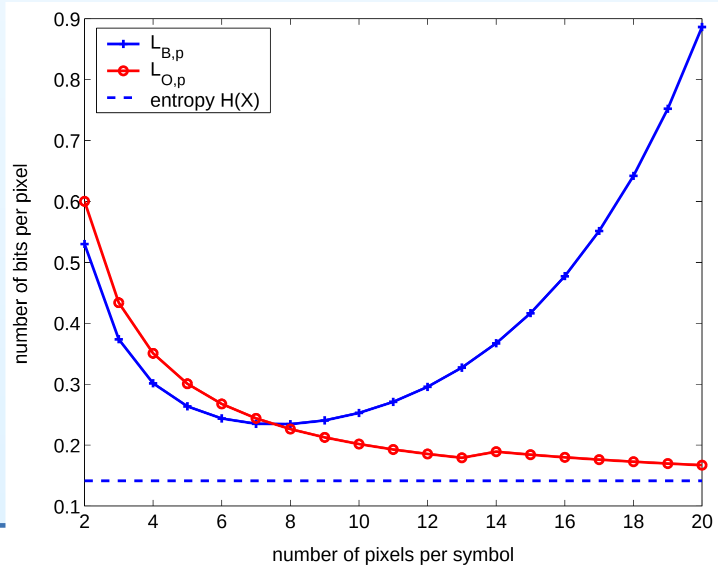
Optimal length : $L_{O,p} > L_{B,p}$?

Consider a source emitting n-length symbols and make $n \rightarrow \infty$.

- $L_{O,p} \rightarrow H(X)$ bits/pixel(=0.1414)
- $L_{B,p} = \sum_{i=1}^{2^n} l_{B,i} p(s_i) = \sum_{i=1}^{2^n} i p(s_i) = ?$
- *Coding efficiency (code B):*

$$\eta_B = \frac{H(S)}{L_B} = \frac{H(X)}{L_{B,p}}$$





Redundancy

Redundant coding $\leftrightarrow$ saying twice the same thing

Coding redundancy:

$$R_C = 1 - \eta = 1 - \frac{H(S)}{L} = 1 - \frac{H(X)}{L_p}$$

Coding redundancy $\rightarrow 0$ when

- L decreases ($L_p \rightarrow H(X)$):
  $\Rightarrow$ when the average length per bit is greater than the bit-entropy, *compression can be achieved by reducing the code length*
- $H(S)$ increases ($H(X) \rightarrow H_{max}(X)$):
  $\Rightarrow$ when the bits are almost equally probable (the entropy is close to its maximum), *compression cannot improve by reducing the code length*

Redundancy

Redundant coding $\leftrightarrow$ saying twice the same thing

Coding redundancy:

$$R_C = 1 - \eta = 1 - \frac{H(S)}{L} = 1 - \frac{H(X)}{L_p}$$

Coding redundancy $\rightarrow 0$ when

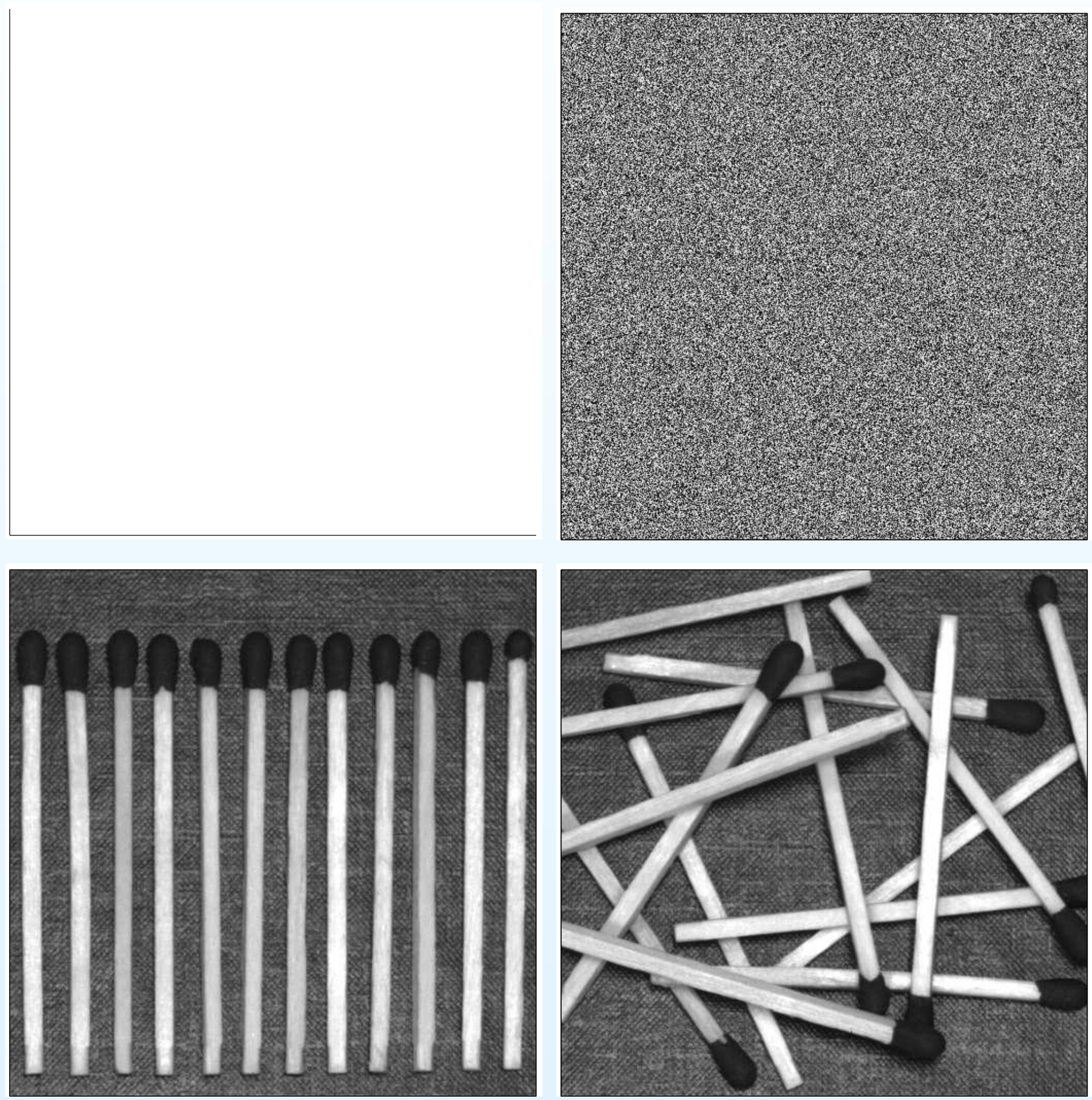
- L decreases ($L_p \rightarrow H(X)$):
  $\Rightarrow$ when the average length per bit is greater than the bit-entropy, *compression can be achieved by reducing the code length*
- $H(S)$ increases ($H(X) \rightarrow H_{max}(X)$):
  $\Rightarrow$ when the bits are almost equally probable (the entropy is close to its maximum), *compression cannot improve by reducing the code length*

Is the coding reduction the only way to compress?
Is the coding redundancy similar to information redundancy?

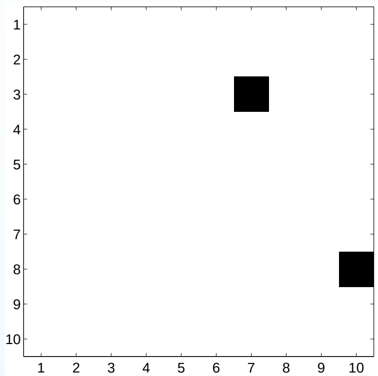
Compression basics

Image and information

Similar images 100×100 pixels, 256 greylevels (10 kBytes)



Coding redundancy

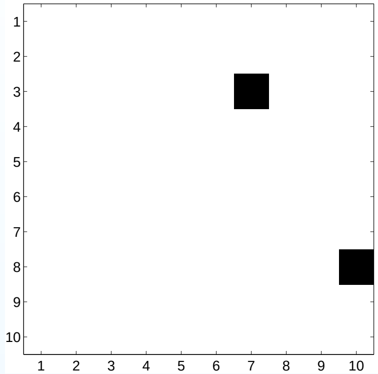


s	$p(s_i)$	Code A	Code B	l_A	l_B
s_1	0.9604	00	0	2	1
s_2	0.0196	01	10	2	2
s_3	0.0196	10	110	2	3
s_4	0.0004	11	1110	2	4

00	00	00	00	00
00	00	00	00	00
00	00	00	10	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	01
00	00	00	00	00
00	00	00	00	00

0	0	0	0	0
0	0	0	0	0
0	0	0	110	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	10
0	0	0	0	0
0	0	0	0	0

Coding redundancy



s	$p(s_i)$	Code A	Code B	l_A	l_B
s_1	0.9604	00	0	2	1
s_2	0.0196	01	10	2	2
s_3	0.0196	10	110	2	3
s_4	0.0004	11	1110	2	4

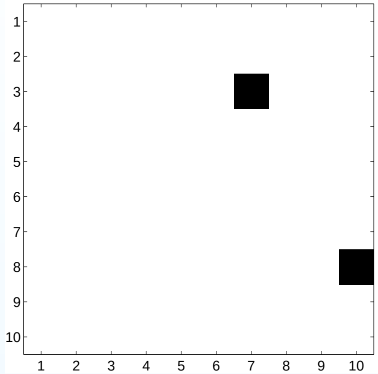
00	00	00	00	00
00	00	00	00	00
00	00	00	10	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	01
00	00	00	00	00
00	00	00	00	00

0	0	0	0	0
0	0	0	0	0
0	0	0	110	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	10
0	0	0	0	0
0	0	0	0	0

$$L_A = \sum_{i=1}^{4} p(s_i) l_{A,i} = 2$$

$$L_B = \sum_{i=1}^{4} p(s_i) l_{B,i} = 1.06$$

Coding redundancy



s	$p(s_i)$	Code A	Code B	l_A	l_B
s_1	0.9604	00	0	2	1
s_2	0.0196	01	10	2	2
s_3	0.0196	10	110	2	3
s_4	0.0004	11	1110	2	4

00	00	00	00	00
00	00	00	00	00
00	00	00	10	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	00
00	00	00	00	01
00	00	00	00	00
00	00	00	00	00

0	0	0	0	0
0	0	0	0	0
0	0	0	110	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	10
0	0	0	0	0
0	0	0	0	0

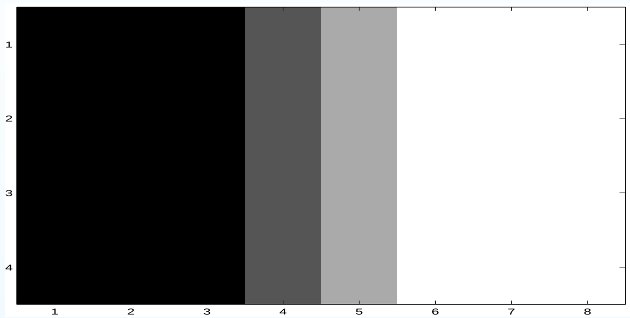
$$L_A = \sum_{i=1}^{4} p(s_i) l_{A,i} = 2$$

$$L_B = \sum_{i=1}^{4} p(s_i) l_{B,i} = 1.06$$

Compression rate C=2/1,06=1,89:1

Inter-pixel redundancy

Consider the image:



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

representative of a source emitting 4 gray-levels:

Gray Level	Count	Probability
21	12	$3/8$
95	4	$1/8$
169	4	$1/8$
243	12	$3/8$

How much compression by reducing coding redundancy?
Entropy estimate:

$$H(X) = -\sum_{1}^{4} p(i) \log p(i) = 1,25 \Rightarrow C = 8/1,25 = 6,4 : 1$$

Spatial redundancy

Consider the *difference* image:

$$\begin{matrix} 21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\ 21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\ 21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \\ 21 & 0 & 0 & 74 & 74 & 74 & 0 & 0 \end{matrix}$$

which is representative of a source emitting 3 gray-levels:

Gray Level	Count	Probability
0	12	$1/2$
21	12	$1/8$
74	4	$3/8$

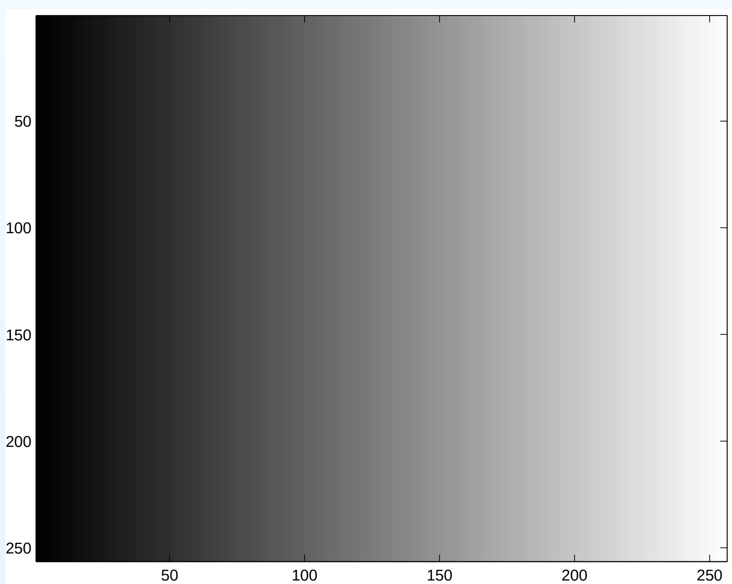
How much compression by reducing coding redundancy?
Entropy estimate:

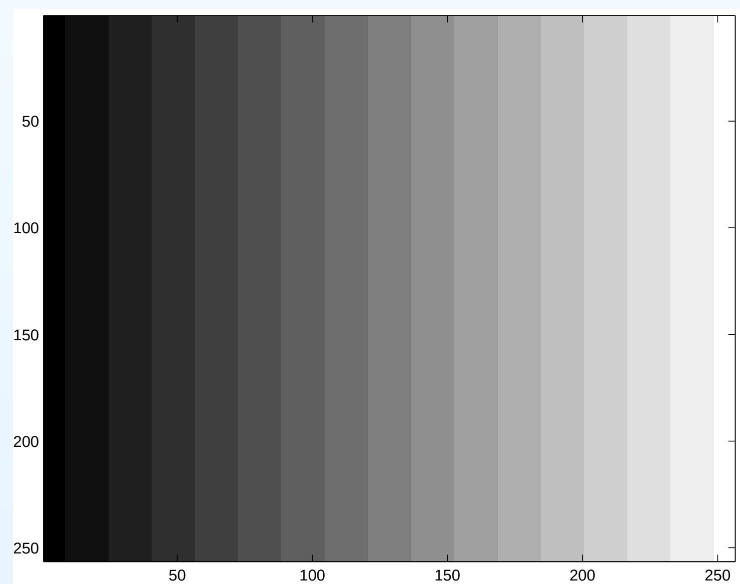
$$H(X_d) = -\sum_{1}^{3} p(i) \log p(i) = 0,97 \Rightarrow C_d = 8/0,97 = 8,2:1$$

Psycho-visual redundancy

Are the 256 gray levels of an image necessary?
Can we see 16 millions colors?

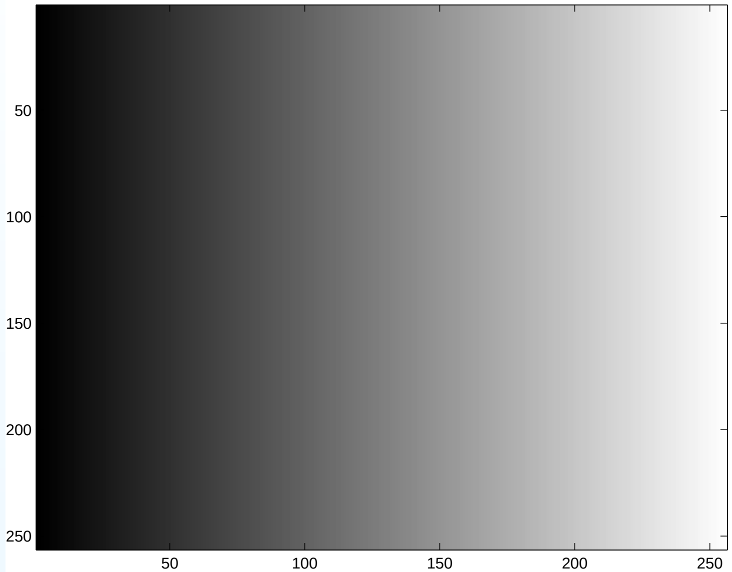
Reducing the number of gray levels
$\Leftrightarrow$ reducing the number of bits per pixel
$\Leftrightarrow$ *Quantizing*: 8 bits/pixels $\longrightarrow$ 4 bits/pixel $\longrightarrow$ C=2:1

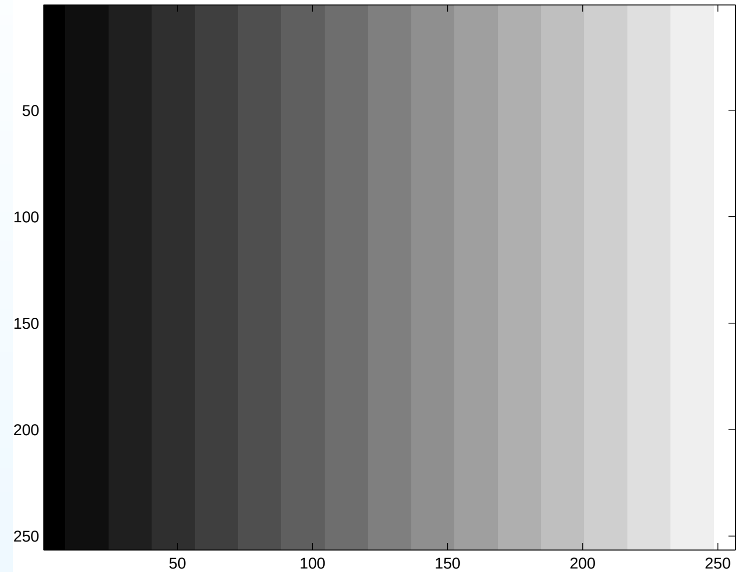


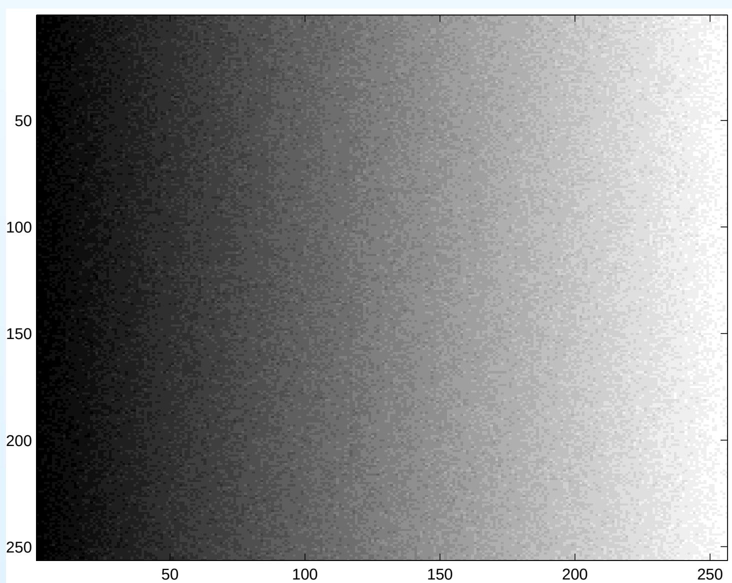


Visual artifacts $\longleftrightarrow$ false contouring

Psycho-visual redundancy







The human eye is sensible to contours

- each pixel is modified by adding a pseudo-random number generated from the neighboring pixels
- noisier image, but visually closer to the original

Psycho-visual redundancy

Quantization = taking the most significant bits (MSB)

$\Rightarrow$ Improved gray scale quantization IGS

Introduction
Information Theory
Compression basics
- Coding redundancy
- Inter-pixel redundancy
- Psycho-visual redundancy
- Image transforms
- Fidelity criteria
- Compression chain
Entropic coding
Inter-pixel coding
Quantizing and thresholding
Color space transforms
Image transforms

Psycho-visual redundancy

Improved gray scale quantization IGS algorithm:
- modifying the MSB by adding a pseudo-random value

1. Initialize a virtual pixel $New = \underbrace{0000}_{MSB}\underbrace{0000}_{LSB}$
2. Change[a] the gray level of the current pixel as:

$$New = Old + New_{LSB}$$

3. Take the New_{MSB} as the quantized value (IGS code)
4. Go to next pixel.

Pixel	Old	New	IGS code
$i-1$	?	00000000	?
i	01101100	01101100	0110
$i+1$	10001011	10010111	1001
$i+2$	10000111	10001110	1000
$i+3$	11110100	11110100	1111

[a] If the MSB of the actual gray level are 1111, left unchanged.

Redundancy types

Redundant information:

$\Leftrightarrow$ Coding redundancy (statistic redundancy)
$\longrightarrow$ lossless data compression
$\longrightarrow$ *Huffman, Shannon-Fano, arithmetic*

$\Leftrightarrow$ Spatio-temporal redundancy (inter-pixel / inter-frame redundancy)
$\longrightarrow$ lossless transforms
$\longrightarrow$ *predictive coding, LZW coding, run-length coding*

$\Leftrightarrow$ Psycho-visual redundancy (approximation)
$\longrightarrow$ lossy transforms
$\longrightarrow$ *color space transforms*
$\longrightarrow$ *thresholding, quantizing*

Redundancy types

Redundant information:

$\Leftrightarrow$ Coding redundancy (statistic redundancy)
$\longrightarrow$ lossless data compression
$\longrightarrow$ *Huffman, Shannon-Fano, arithmetic*

$\Leftrightarrow$ Spatio-temporal redundancy (inter-pixel / inter-frame redundancy)
$\longrightarrow$ lossless transforms
$\longrightarrow$ *predictive coding, LZW coding, run-length coding*

$\Leftrightarrow$ Psycho-visual redundancy (approximation)
$\longrightarrow$ lossy transforms
$\longrightarrow$ *color space transforms*
$\longrightarrow$ *thresholding, quantizing*

Compression = Redundancy reduction

Transforms: changing the point of view

All described approaches $\longrightarrow$ spatial domain
$\longrightarrow$ data changing (entropic coding, interpixel coding)
$\longrightarrow$ information changing (quantization, approximation)

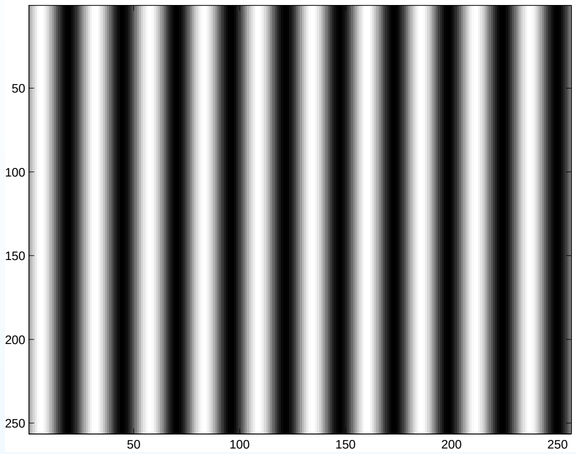
How about translating to another domain ?
- Thank you $\longrightarrow$ Merci (*message changing*)

1. Color space transforms: RGB $\longrightarrow$ YCbCr

2. Image transforms: Fourier, DCT, wavelets, . . .

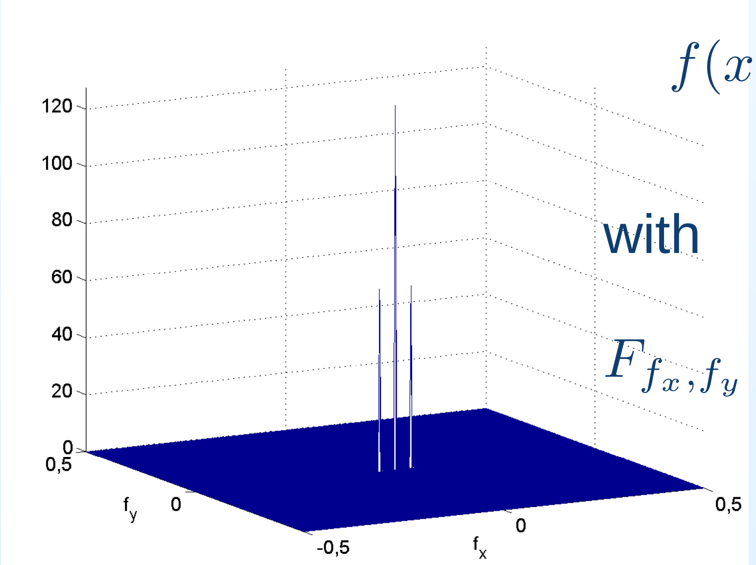
- *Idea:* describing a function (an image, a signal) using simple, elementary basis functions
- *Method:* each image is written as a linear combination of basis functions
- *Result:* the coefficients of this linear combination describe the image

Fourier Transform: example



- each line of this 256 gray-levels 256×256 image: cosine oscillating 10 times
- image: $f(x,y) =$ fix $(128[\cos(2\pi f_x x) + 0,999])$, with $x, y = 1 \ldots 256$, $f_x = 10/256$

Fourier transform:



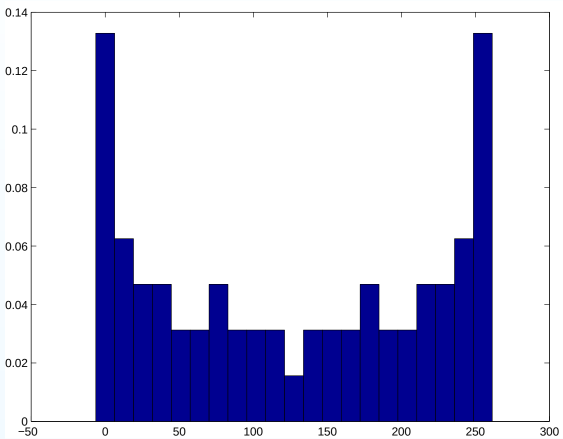
- $f(x,y)$ can be written as:

$$f(x,y) = \sum_{f_x} \sum_{f_y} F_{f_x,f_y} e^{2\pi j(f_x x + f_y y)}$$

with

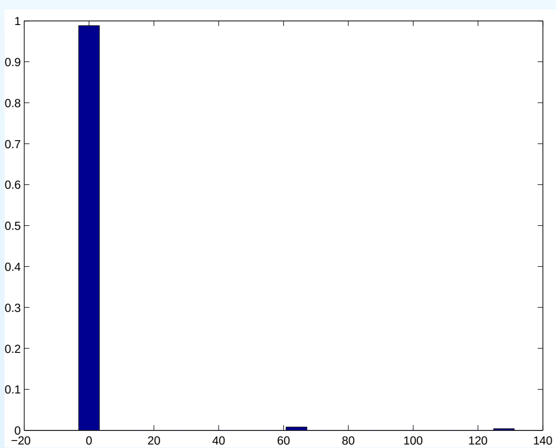
$$F_{f_x,f_y} = \frac{1}{256^2} \sum_x \sum_y f(x,y) e^{-j2\pi(f_x x + f_y y)}$$

Fourier Transform: example



- entropy estimation:
  $H(f) = -\sum p(f) \log p(f) = 4,17$
  $\rightarrow$ *low coding redundancy*
- *important spatial redundancy* (periodic function) but impossible differential predictive coding
- rather *low psycho-visual redundancy*, difficult to quantize

Fourier transform:



- entropy estimation:
  $H(F) = -\sum p(F) \log p(F) = 0,0008$
  $\rightarrow$ *important coding redundancy*
- *important spatial redundancy* (constant function)
- *important psycho-visual redundancy*, easy to quantize *and* to threshold

Fidelity criteria

Lossy compression = Approximation
Original image = $M \times N$ 256 gray-levels matrix:

$$f(x,y), x = 1 \dots M, y = 1 \dots N$$

Approximation image = $M \times N$ 256 gray-levels matrix:

$$\hat{f}(x,y), x = 1 \dots M, y = 1 \dots N$$

1. Mean square error:

$$MSE_f = \mathbb{E}\left[|f(x,y) - \hat{f}(x,y)|^2\right] = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\left[f(x,y) - \hat{f}(x,y)\right]^2$$

2. Signal to noise ratio:
   - $MSE \approx$ noise variance

$$SNR_f = \frac{\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\left[\hat{f}(x,y)\right]^2}{MSE}$$

Fidelity criteria

3. Peak signal to noise ratio:

$$PSNR_f = \frac{255^2}{MSE} = \frac{(2^b - 1)^2}{MSE}$$

with $2^b - 1$ the maximum possible value of a pixel

4. Psycho-visual subjective criterion:

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.

Fidelity criteria

Excellent	Marginal	Fine
	$MSE = 48,02$	$MSE = 45,97$
	$SNR = 10,25$	$SNR = 10,39$

Compression chain

Flowchart of a standard compression algorithm

Transforms
- color transforms
  - color space
  - bit-plane coding
- basis transform
  - Discrete Cosine
  - Wavelets

Message changing

Approximations
- quantizing
- thresholding

Information changing

Lossy compression

Coding
- inter-pixel
  - run-length
  - predictive
- entropic
  - Huffman
  - Shannon-Fano
  - LZW (inter-pix)

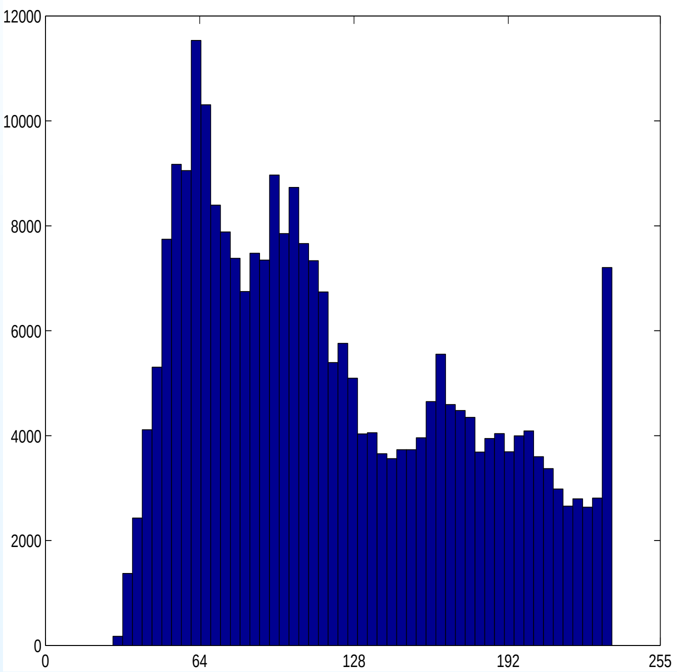
Data changing

Lossless compression

Entropic coding

Variable length

Natural image (monochrome): unequally probable gray levels
Natural coding (binary): 8 bits/pixel (0 . . . 255)



Idea:

Adapt the number of bits/pixel to the color of the pixel:
$\longrightarrow$ shorter codes for most probable colors

Variable length coding

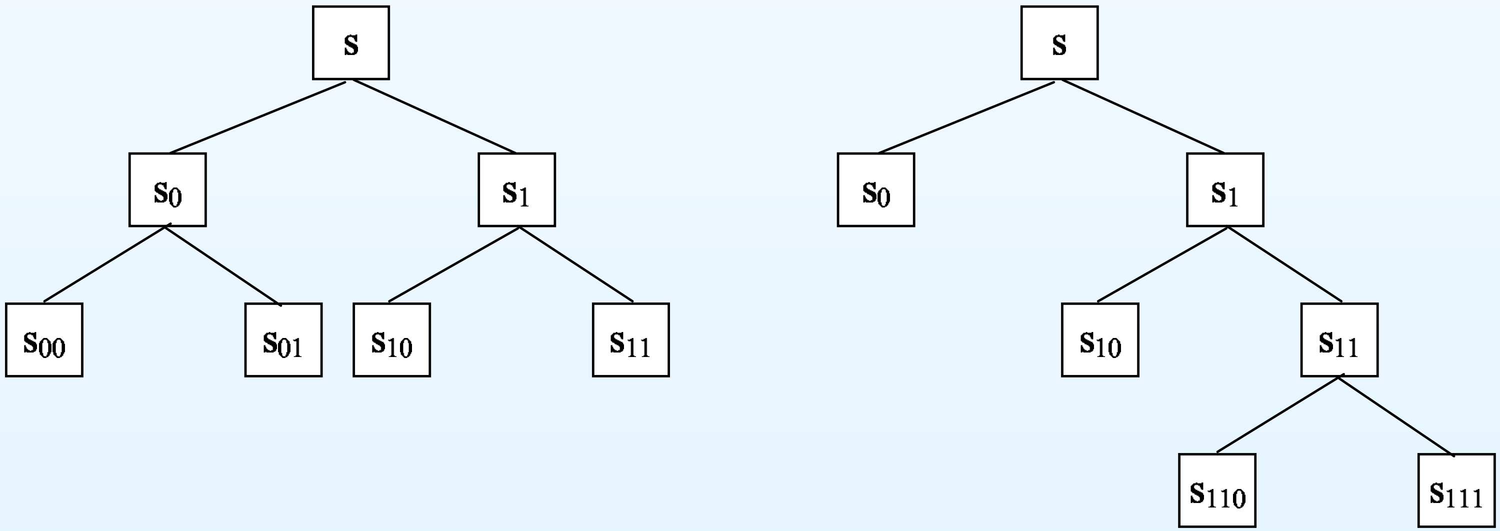
Efficiency condition for the new code:

$\longrightarrow$ no separation code between pixels $\Rightarrow$ prefix condition:

A color code must not be the beginning of another's color code.

Consider a source s emitting 4 symbols.

General coding method: binary tree



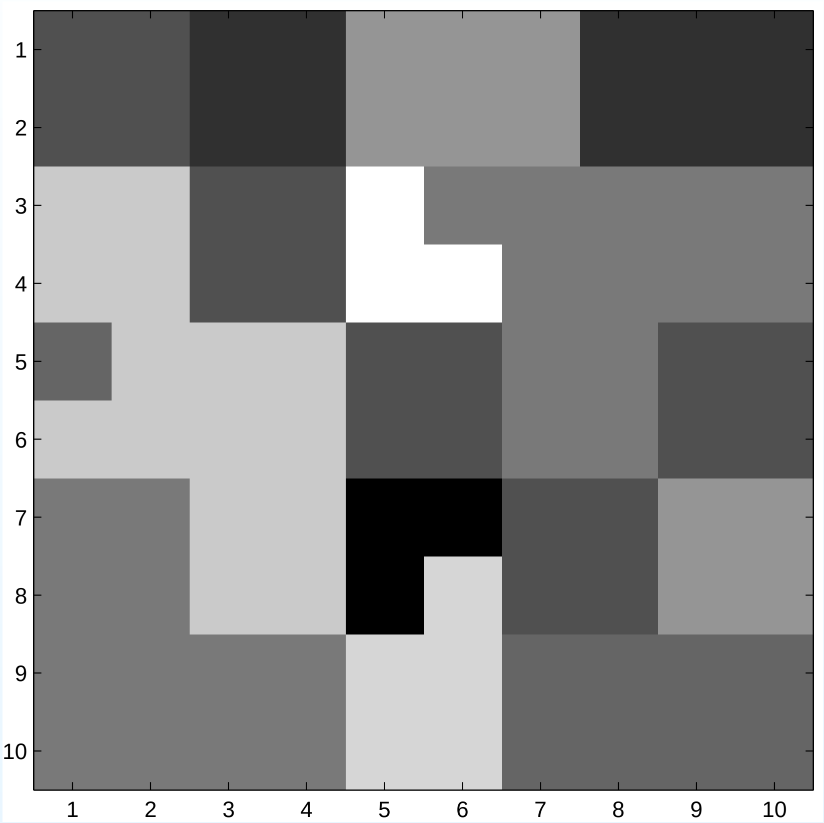
Shannon-Fano Coding

Algorithm:

1. Sort the symbols (gray level intensities) in decreasing order of probability and store the result in vector $\mathbf{s}$
2. Split the resulting vector $\mathbf{s}$ in two smaller vectors $\mathbf{s_0}$ and $\mathbf{s_1}$:
   - the first one ($\mathbf{s_0}$) contains the great probability symbols, their sum being $\leq$ 0,5
   - the second one ($\mathbf{s_1}$) the rest of the symbols
3. For each of the two vectors $\mathbf{s_0}$ and $\mathbf{s_1}$, go to step 2 and construct, if possible, vectors $\mathbf{s_{00}}$, $\mathbf{s_{01}}$,$\mathbf{s_{10}}$ and $\mathbf{s_{11}}$
4. Continue until arriving to individual symbols

The resulting indices $xxxx$ of the vectors $\mathbf{s_{xxxx}}$ are the new codes.

Shannon-Fano Coding: Example



s_k	$p(s_k)$
s_1	$0,25$
s_2	$0,20$
s_3	$0,15$
s_4	$0,10$
s_5	$0,10$
s_6	$0,09$
s_7	$0,05$
s_8	$0,03$
s_9	$0,03$

Introduction
Information Theory
Compression basics
Entropic coding
- Shannon-Fano
- Huffman
- LZW
Inter-pixel coding
Quantizing and thresholding
Color space transforms
Image transforms

Shannon-Fano Coding: Example

s_k	$p(s_k)$							Code c_k	Length l_k
s_1	$0,25$	0	0					00	2
s_2	$0,20$		1					01	2
s_3	$0,15$	1	0	0				100	3
s_4	$0,10$			1				101	3
s_5	$0,10$		1	0				110	3
s_6	$0,09$			1	0			1110	4
s_7	$0,05$				1	0		11110	5
s_8	$0,03$					1	0	111110	6
s_9	$0,03$						1	111111	6

- Source entropy: $H(S) = -\sum_k p(s_k) \log p(s_k) = 2,87$ bits/pixel

- Shannon-Fano coding: $L_p = \sum_k p(s_k) l_k = 2,92$ bits/pixel

- Natural binary coding: 8 bits/pixel

$\Rightarrow$ Compression $C = 8/2,92 = 2,74 : 1$

Shannon-Fano Coding

1. Upside - down algorithm
   - goes down from the whole set to individual symbols
   - constructs first the most probable codes
   - starts constructing the codes form the MSB

Shannon-Fano Coding

1. Upside - down algorithm
   - goes down from the whole set to individual symbols
   - constructs first the most probable codes
   - starts constructing the codes form the MSB
2. Very efficient if we can split the probabilities vector in exactly equally probable parts (1/2 - 1/2), so if individual probabilities are powers of 1/2
   $\rightarrow$ in this case we obtain total elimination of coding redundancy, $L_p = H(S)$!

Shannon-Fano Coding

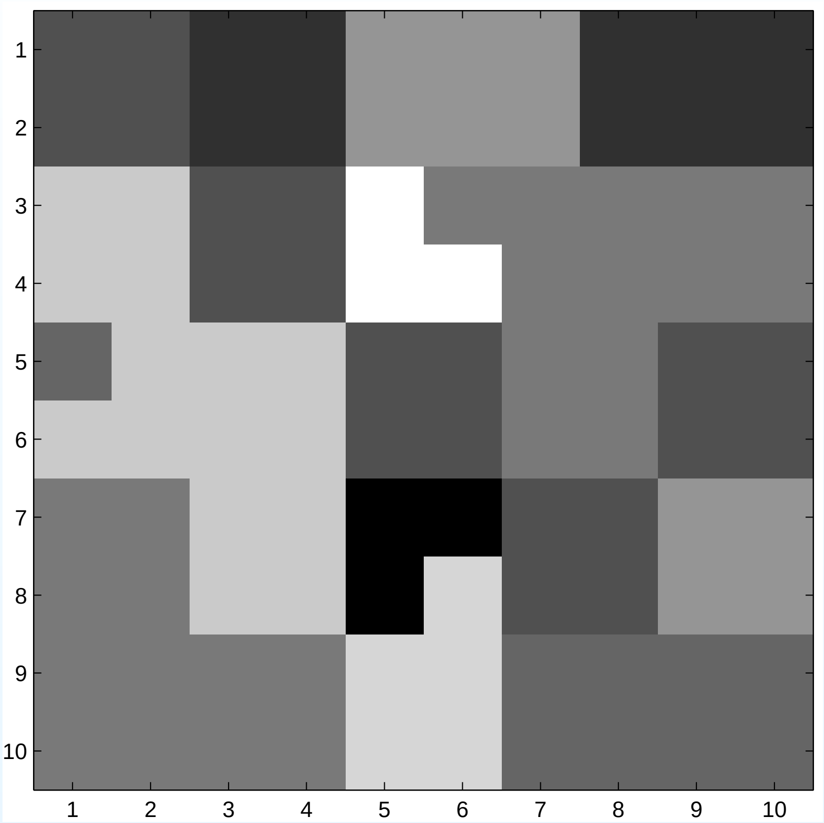
1. Upside - down algorithm
   - goes down from the whole set to individual symbols
   - constructs first the most probable codes
   - starts constructing the codes form the MSB
2. Very efficient if we can split the probabilities vector in exactly equally probable parts (1/2 - 1/2), so if individual probabilities are powers of 1/2
   $\rightarrow$ in this case we obtain total elimination of coding redundancy, $L_p = H(S)$!
3. Seldom used in practice.

Huffman Coding

Algorithm:

1. Sort the symbols (gray level intensities) in decreasing order of probability and store the result in vector $\mathbf{s}$
2. Initialize all the codes at $[]$ (void)
3. Associate the 2 smallest probabilities and modify the codes of the respective symbols:
   - increase their size by a bit placed in the most significant position
   - make this bit 0, respectively 1
4. Create a virtual temporary symbol having a probability equal to the the sum of the two probabilities from step 1
5. Create a new vector of probabilities associated to the new vector of symbols, replacing the two smallest by the their sum
6. Return to step 3. If one of the less probable symbols is a virtual one, modify the codes of the real symbols “inside”
7. Continue until the sum of 2 probabilities equals 1

Huffman Coding: Example



s_k	$p(s_k)$
s_1	$0,25$
s_2	$0,20$
s_3	$0,15$
s_4	$0,10$
s_5	$0,10$
s_6	$0,09$
s_7	$0,05$
s_8	$0,03$
s_9	$0,03$

Huffman Coding: Example

Introduction

Information Theory

Compression basics

Entropic coding
- Shannon-Fano
- Huffman
- LZW

Inter-pixel coding

Quantizing and thresholding

Color space transforms

Image transforms

s_k	$p(s_k)$									Code	l_k
s_1	$0,25$							1	0	01	2
s_2	$0,20$						1		1	11	2
s_3	$0,15$					1		0	0	001	3
s_4	$0,10$				1		0		1	101	3
s_5	$0,10$			0		0		0	0	0000	4
s_6	$0,09$			1						0001	4
s_7	$0,05$		1							1001	4
s_8	$0,03$	0	0		0		0		1	10000	5
s_9	$0,03$	1								10001	5
		$0,06$	$0,11$	$0,19$	$0,21$	$0,34$	$0,41$	$0,59$	1		

- Source entropy: $H(S) = 2,87$ bits/pixel
- Huffman coding: $L_p = 2,91$ bits/pixel

$\Rightarrow$ Compression $C = 8/2,91 = 2,75 : 1$

Huffman Coding

1. Bottom - up algorithm
   - goes up from individual symbols to the whole set
   - parallel construction of all codes
   - starts constructing the codes form the LSB

Huffman Coding

1. Bottom - up algorithm
   - goes up from individual symbols to the whole set
   - parallel construction of all codes
   - starts constructing the codes form the LSB
2. Optimal algorithm - shortest possible average length

Huffman Coding

1. Bottom - up algorithm
   - goes up from individual symbols to the whole set
   - parallel construction of all codes
   - starts constructing the codes form the LSB
2. Optimal algorithm - shortest possible average length
3. Most used in practice (JPEG, . . .).

Huffman Coding

1. Bottom - up algorithm
   - goes up from individual symbols to the whole set
   - parallel construction of all codes
   - starts constructing the codes form the LSB
2. Optimal algorithm - shortest possible average length
3. Most used in practice (JPEG, . . .).
4. Different alleged versions:
   - truncated Huffman
   - shifted Huffman

Introduction

Information Theory

Compression basics

Entropic coding
- Shannon-Fano
- Huffman
- LZW

Inter-pixel coding

Quantizing and thresholding

Color space transforms

Image transforms

Variable length coding

Source symbol	Probability	Binary Code	Huffman	Truncated Huffman	B_2-Code	Binary Shift	Huffman Shift
Block 1							
a_1	0.2	00000	10	11	C00	000	10
a_2	0.1	00001	110	011	C01	001	11
a_3	0.1	00010	111	0000	C10	010	110
a_4	0.06	00011	0101	0101	C11	011	100
a_5	0.05	00100	00000	00010	C00C00	100	101
a_6	0.05	00101	00001	00011	C00C01	101	1110
a_7	0.05	00110	00010	00100	C00C10	110	1111
Block 2							
a_8	0.04	00111	00011	00101	C00C11	111 000	00 10
a_9	0.04	01000	00110	00110	C01C00	111 001	00 11
a_{10}	0.04	01001	00111	00111	C01C01	111 010	00 110
a_{11}	0.04	01010	00100	01000	C01C10	111 011	00 100
a_{12}	0.03	01011	01001	01001	C01C11	111 100	00 101
a_{13}	0.03	01100	01110	100000	C10C00	111 101	00 1110
a_{14}	0.03	01101	01111	100001	C10C01	111 110	00 1111
Block 3							
a_{15}	0.03	01110	01100	100010	C10C10	111 111 000	00 00 10
a_{16}	0.02	01111	010000	100011	C10C11	111 111 001	00 00 11
a_{17}	0.02	10000	010001	100100	C11C00	111 111 010	00 00 110
a_{18}	0.02	10001	001010	100101	C11C01	111 111 011	00 00 100
a_{19}	0.02	10010	001011	100110	C11C10	111 111 100	00 00 101
a_{20}	0.02	10011	011010	100111	C11C11	111 111 101	00 00 1110
a_{21}	0.01	10100	011011	101000	C00C00C00	111 111 110	00 00 1111
Entropy	4.0						
Average length		5.0	4.05	4.24	4.65	4.59	4.13

LZW Coding

- Shannon-Fano, Huffman $\longrightarrow$ variable length coding
  - need a previous probability estimation for each symbol
  - assign variable length codes to fixed length symbols
- Lempel-Ziv-Welch (LZW) $\longrightarrow$ fixed length coding
  - doesn't need a previous probability estimation of symbol apparition
  - assign a fixed length code to variable length symbols, ALWAYS created by concatenation of two previously defined symbols
  - constructs un dictionary of symbols adapted to the image

Applications:

- TIFF images
- GIF images
- PDF documents

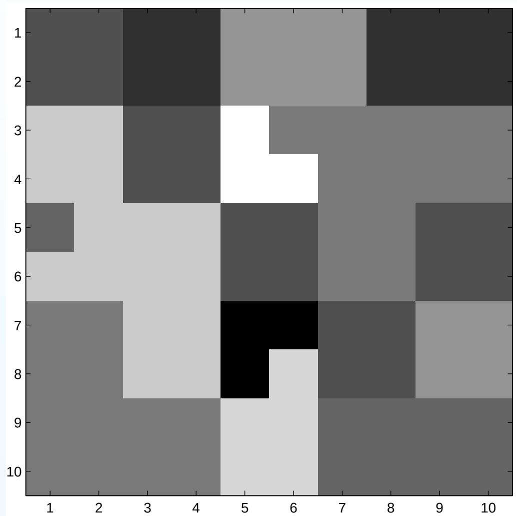
LZW Algorithm

1. Define the length $n > 8$ of the code (fixed) $\Leftrightarrow$ size of the dictionary = 2^n
2. Define the first 255 symbols of the dictionary as the normal gray levels
3. Read the first symbol (pixel gray level) to S_1
4. Read the next symbol to S_2
5. Concatenate S_1 and S_2 to form a new symbol $S_N = S_1 S_2$
6. If S_N is not in the dictionary
   - Output S_1
   - Add S_N to the dictionary
   - Make $S_1 = S_2$

   else
   - Make $S_1 = S_N$
7. Goto step 4 until end of file.

Introduction
Information Theory
Compression basics
Entropic coding
- Shannon-Fano
- Huffman
- LZW
Inter-pixel coding
Quantizing and thresholding
Color space transforms
Image transforms

LZW Coding: Example



S_1	S_2	Out (LZW code)	Dictionary

80	80	50	50	150	150	150	50	50	50
80	80	50	50	150	150	150	50	50	50
200	200	80	80	255	120	120	120	120	120
200	200	80	80	255	255	120	120	120	120
100	200	200	200	80	80	120	120	80	80
200	200	200	200	80	80	120	120	80	80
120	120	200	200	0	0	80	80	150	150
120	120	200	200	0	215	80	80	150	150
120	120	120	120	215	215	100	100	100	100
120	120	120	120	215	215	100	100	100	100

LZW Coding: Example

S_1	S_2	Dictionary	Out (LZW code)
80	80	$256 = 80\,80$	80
80	50	$257 = 80\,50$	80
50	50	$258 = 50\,50$	50
50	150	$259 = 50\,150$	50
150	150	$260 = 150\,150$	150
150	150		
260	50	$261 = 150\,150\,50$	260
50	50		
258	50	$262 = 50\,50\,50$	258
50	80	$263 = 50\,80$	50
80	80		
256	50	$264 = 80\,80\,50$	256
50	50		
258	150	$265 = 50\,50\,150$	258
150	150		
260	150	$266 = 150\,150\,150$	260
150	50	$267 = 150\,50$	150
50	50		
258	50		
262	200	$268 = 50\,50\,50\,200$	262
200	200	$269 = 200\,200$	200
...	...	...	...

9 bits/symbol $\rightarrow$ Compression ratio $C_R = \dfrac{100 \cdot 8}{57 \cdot 9} = 1,56 : 1$

LZW Decoding algorithm

1. Knowing n the fixed length of the code, define the first 255 symbols of the dictionary as the normal gray levels
2. Initialize $S_1 = LZW(1)$ (first symbol)
3. Output S_1
4. Read the next symbol to S_2
5. If S_2 is in the dictionary
   - Make $C = S_2(1)$ (the first element of S_2)
   - Concatenate S_1 and C to form a new symbol S_N
   - Make $S_1 = S_2$

   else
   - Make $C = S_1(1)$ (the first element of S_1)
   - Concatenate S_1 and C to form a new symbol S_N
   - Make $S_1 = S_N$
6. Add S_N to the dictionary
7. Output S_1
8. Goto step 4 until end of file.

LZW Decoding: Example

S_2(LZW code)	C	Dictionary	$S_1 =$ Out (Image)
			80
80	80	$256 = 80\ 80$	80
50	50	$257 = 80\ 50$	50
50	50	$258 = 50\ 50$	50
150	150	$259 = 50\ 150$	150
260	150	$260 = 150\ 150$	150 150
258	50	$261 = 150\ 150\ 50$	50 50
50	50	$262 = 50\ 50\ 50$	50
256	80	$263 = 50\ 80$	80 80
258	50	$264 = 80\ 80\ 50$	50 50
260	150	$265 = 50\ 50\ 150$	150 150
150	150	$266 = 150\ 150\ 150$	150
262	50	$267 = 150\ 50$	50 50 50
200	200	$268 = 50\ 50\ 50\ 200$	200
...	...	...	...

LZW Coding

- The dictionary is not transmitted (created from the file, both for encoding and for decoding)
- The number of symbols in the dictionary depends on the file
- The size of a symbol is predefined
  - if too big, inefficient compression: optimal number of oversized symbols
  - if too small, inefficient compression: not enough symbols to exploit all redundancies

Example:

Bits/symbol	Compression ratio
9	0.98:1
10	0.99:1
11	1.01:1
12	1.06:1
13	1.11:1

Inter-pixel coding

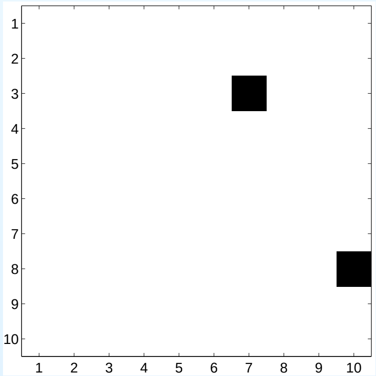
Black&white images

Run Length Coding

1. Create an empty output vector
2. If the first bit of the image is 1 (first pixel is white), put 0 in the output vector
3. Starting from the first pixel, count the number of pixels until the next change and place the result in the output vector

Output vector:

- odd elements: length of black sequences
- even elements: length of white sequences

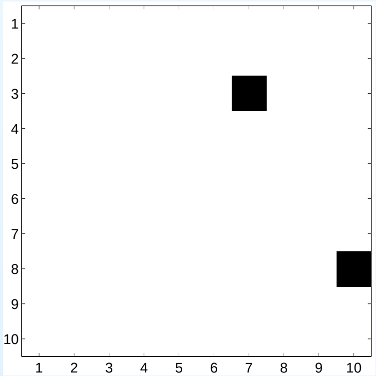


Black&white images

Run Length Coding

1. Create an empty output vector
2. If the first bit of the image is 1 (first pixel is white), put 0 in the output vector
3. Starting from the first pixel, count the number of pixels until the next change and place the result in the output vector

Output vector:
- odd elements: length of black sequences
- even elements: length of white sequences



- RLC=[0 26 1 52 1 20]
- Compression rate:

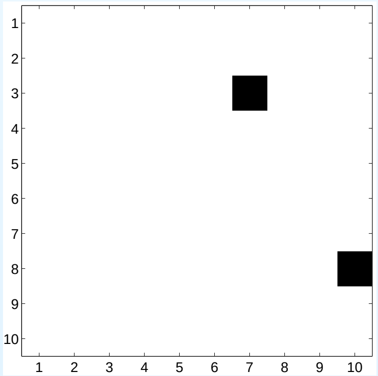
$$C_R = \frac{100}{6 \cdot 8} = 2.08 : 1$$

Black&white images

Constant Area Coding

1. Choose the size $p \times q$ of the coding block
2. Count the number of all-white, all-black and mixed blocks
3. Code the most probable as 0, followed by 10 and 11
4. Starting from the left up corner
   - read the $p \times q$ block
   - if monochrome
     output the correspondent code (0 or 10)

     else
     output 11 followed by the pq bits of the block



Black&white images

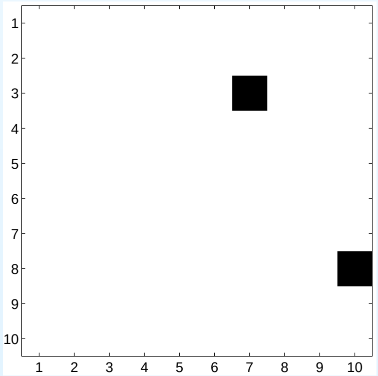
Constant Area Coding

1. Choose the size $p \times q$ of the coding block
2. Count the number of all-white, all-black and mixed blocks
3. Code the most probable as 0, followed by 10 and 11
4. Starting from the left up corner
   - read the $p \times q$ block
   - if monochrome

     output the correspondent code (0 or 10)

     else

     output 11 followed by the pq bits of the block



$CAC_{4\times 4}$

Black&white images

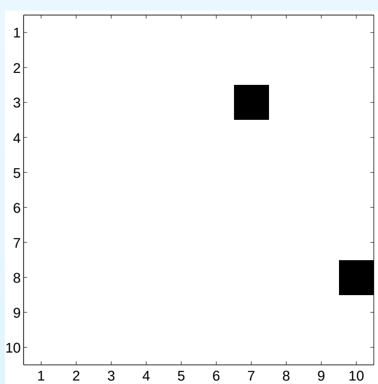
Constant Area Coding

1. Choose the size $p \times q$ of the coding block
2. Count the number of all-white, all-black and mixed blocks
3. Code the most probable as 0, followed by 10 and 11
4. Starting from the left up corner
   - read the $p \times q$ block
   - if monochrome

     output the correspondent code (0 or 10)

     else

     output 11 followed by the pq bits of the block



$$CAC_{4\times 4} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 110100 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 110001 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

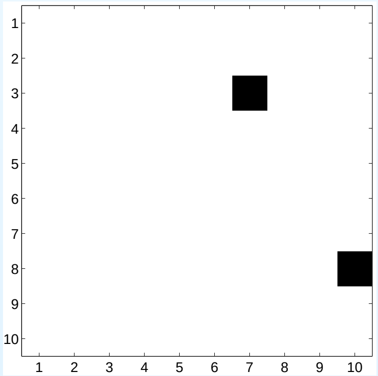
$$C_R = \frac{100}{35} = 2,86 : 1$$

Black&white images

Constant Area Coding

1. Choose the size $p \times q$ of the coding block
2. Count the number of all-white, all-black and mixed blocks
3. Code the most probable as 0, followed by 10 and 11
4. Starting from the left up corner
   - read the $p \times q$ block
   - if monochrome
     output the correspondent code (0 or 10)

     else
     output 11 followed by the pq bits of the block



$$CAC_{4\times4} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 110100 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 110001 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$C_R = \frac{100}{35} = 2,86:1$$

$$CAC_{10\times1}$$

Black&white images

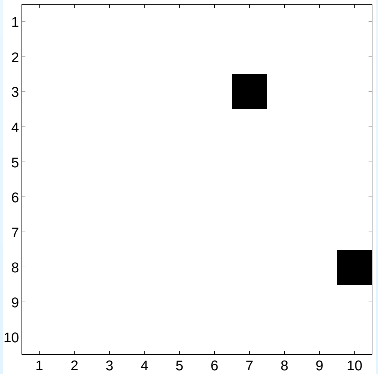
Constant Area Coding

1. Choose the size $p \times q$ of the coding block
2. Count the number of all-white, all-black and mixed blocks
3. Code the most probable as 0, followed by 10 and 11
4. Starting from the left up corner
   - read the $p \times q$ block
   - if monochrome

     output the correspondent code (0 or 10)

     else

     output 11 followed by the pq bits of the block



$$CAC_{4\times 4} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 110100 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 110001 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$C_R = \frac{100}{35} = 2,86:1$$

$$CAC_{10\times 1} = 0\ 0\ 0\ 0\ 0\ 0\ 110010000000\ 0\ 0\ 110000000100$$

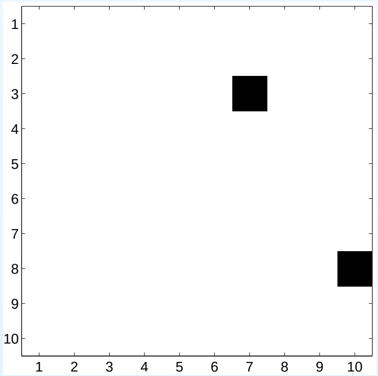
$$C_R = \frac{100}{32} = 3,12:1$$

Black&white images

White block skipping

For mostly white images:

1. Choose the size $p \times q$ of the coding block
2. Starting from the left up corner
   - read the $p \times q$ block
   - if all-white
     output 0
     else
     output 1 followed by the pq bits of the block

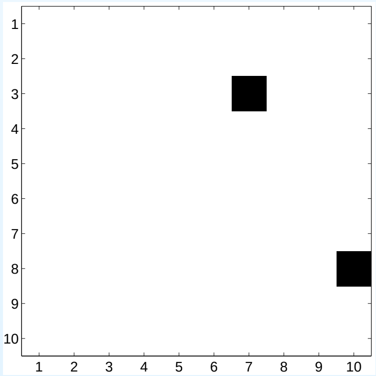


Black&white images

White block skipping

For mostly white images:

1. Choose the size $p \times q$ of the coding block
2. Starting from the left up corner
   - read the $p \times q$ block
   - if all-white
     output 0
     else
     output 1 followed by the pq bits of the block



$WBS_{4\times 4}$

Black&white images

White block skipping

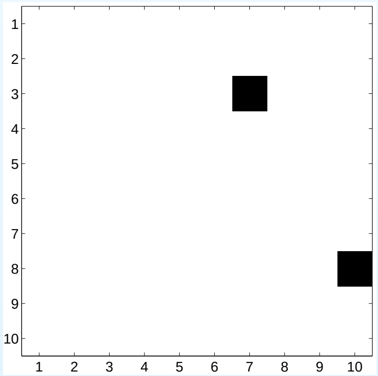
For mostly white images:

1. Choose the size $p \times q$ of the coding block
2. Starting from the left up corner
   - read the $p \times q$ block
   - if all-white

     output 0

     else

     output 1 followed by the pq bits of the block



$$WBS_{4\times4} = \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10100 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10001 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

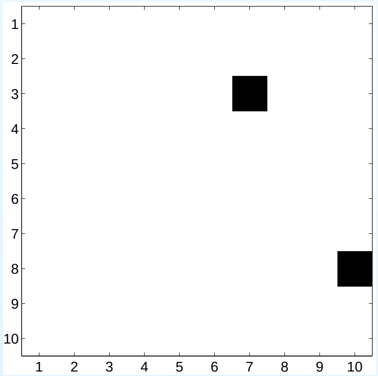
$$C_R = \frac{100}{33} = 3 : 1$$

Black&white images

White block skipping

For mostly white images:

1. Choose the size $p \times q$ of the coding block
2. Starting from the left up corner
   - read the $p \times q$ block
   - if all-white
     output 0
     else
     output 1 followed by the pq bits of the block



$$WBS_{4\times 4} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10100 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10001 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$C_R = \frac{100}{33} = 3:1$$

$$WBS_{1\times 10}$$

Black&white images

White block skipping

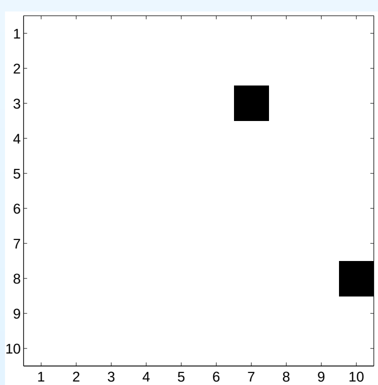
For mostly white images:

1. Choose the size $p \times q$ of the coding block
2. Starting from the left up corner
   - read the $p \times q$ block
   - if all-white

     output 0

     else

     output 1 followed by the pq bits of the block



$$WBS_{4\times 4} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10100 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10001 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$C_R = \frac{100}{33} = 3 : 1$$

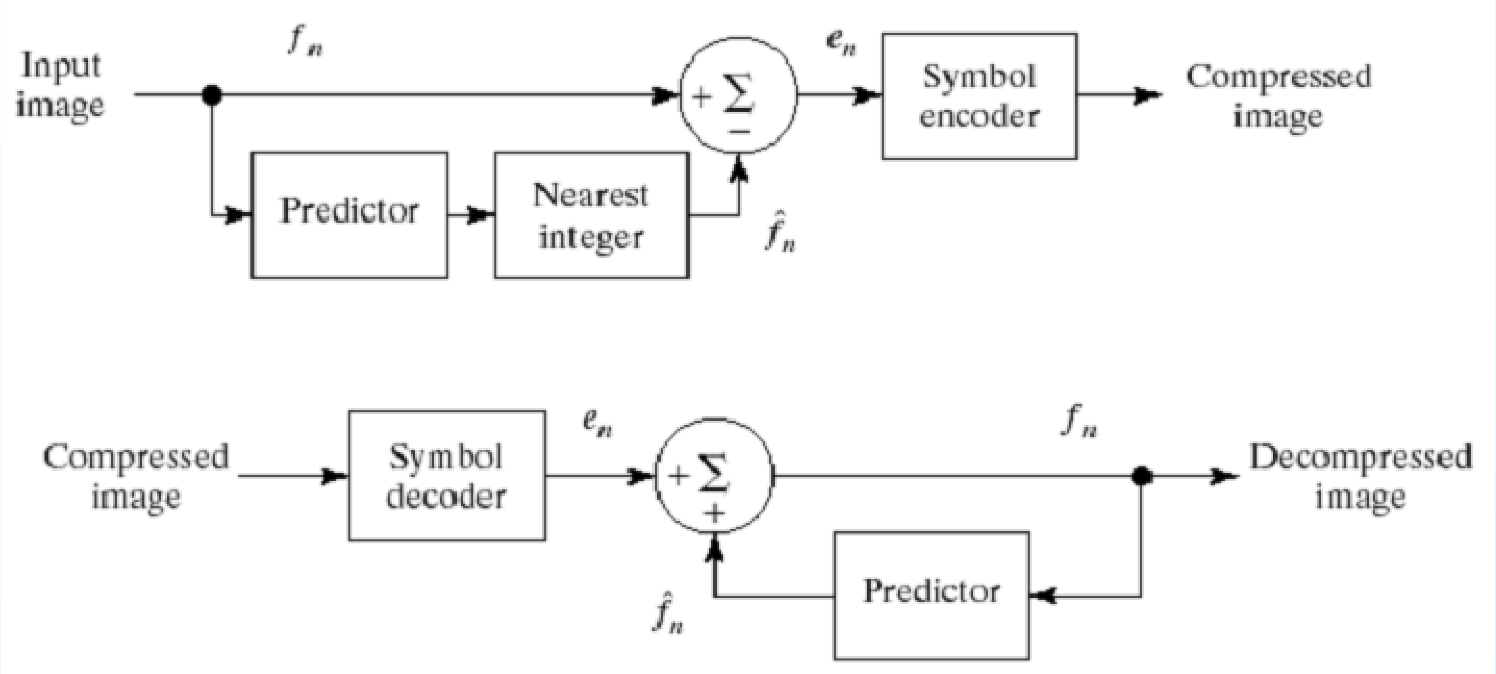
$$WBS_{1\times 10} = [0\ \ 0\ \ 10000001000\ \ 0\ \ 0\ \ 0\ \ 0\ \ 10000000001\ \ 0\ \ 0]^T$$

$$C_R = \frac{100}{30} = 3,33 : 1$$

Grayscale images

Predictive coding

- Idea: coding only the *new* information in each pixel
- What is "new"?
  $\rightarrow$ new= difference between the actual gray level value and the *predicted* value
- What is "predicted"?
  $\rightarrow$ predicted= probable value of the gray level, knowing the preceding pixels



Prediction

General formula for the m^{th} order predictor:

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

$\rightarrow$ the n^{th} value is predicted as a linear combination of previous values ($\sum_i \alpha_i = 1$)

Previous values:

- in time
  $\rightarrow$ successive values of a measured signal
  $\rightarrow$ pixel (x, y) values in successive frames
- in space
  $\rightarrow$ previous values on the same line
  $\rightarrow$ neighboring values in the same block
  $\rightarrow$ same position pixel in previous blocks

Encoder output:

$$e_n = f_n - \hat{f}_n$$

Predictive coding algorithm

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

Encoding algorithm

1. Choose the predictor order m and coefficients α_i
2. Initialization: Error image e = input image f
3. For each line x
   - For all pixels $(x,y), \quad y > m$

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

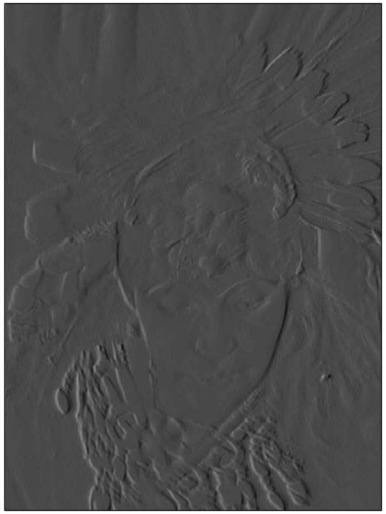
Decoding algorithm

1. Fix the predictor order m and coefficients α_i
2. Initialization: Reconstructed image f = error image e
3. For each line x
   - For all pixels $(x,y), \quad y > m$

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

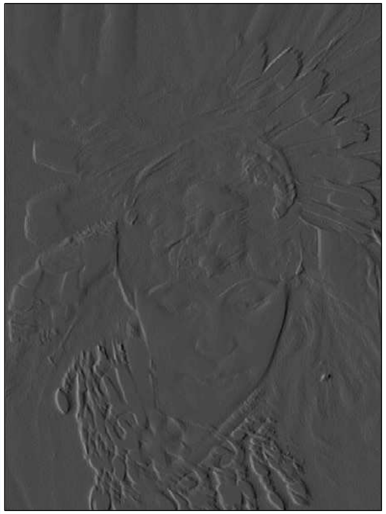
Example

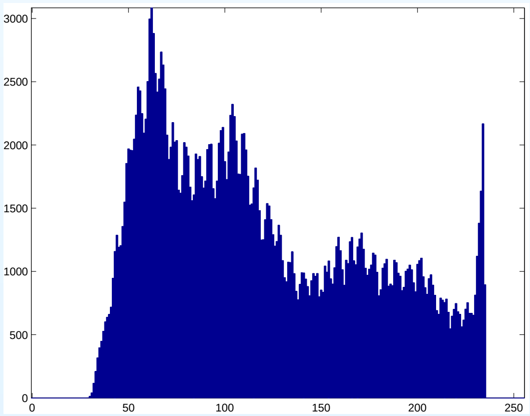
Observation: for $m = 1$ and $\alpha_1 = 1 \rightarrow$ differential coding

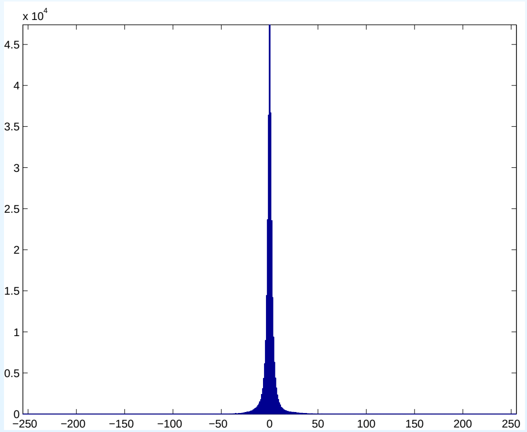


Example

Observation: for $m = 1$ and $\alpha_1 = 1 \rightarrow$ differential coding

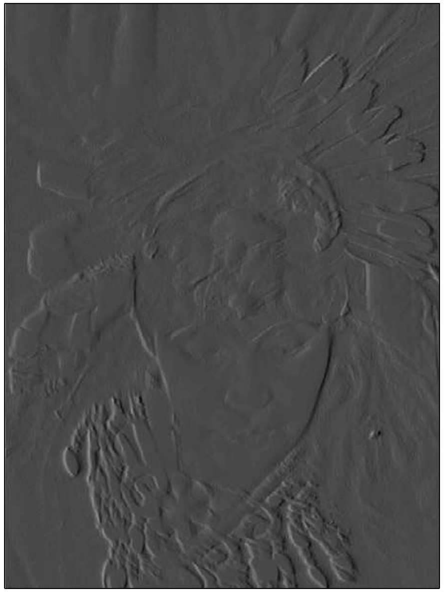






$$H_M = -\sum_{i=0}^{255} p(i) \log p(i) = 7.53 \qquad H_E = -\sum_{i=-255}^{255} p(i) \log p(i) = 4.14$$

Example



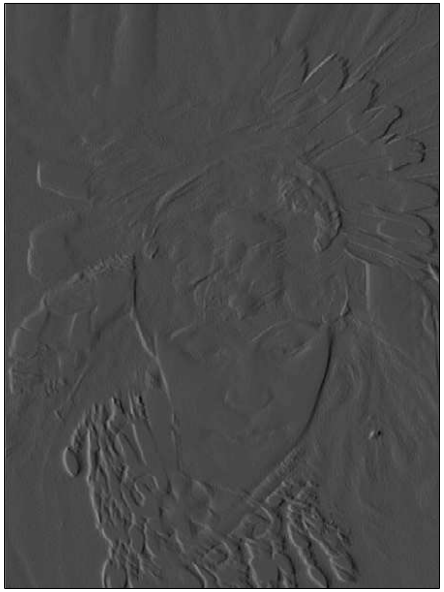
Example

Entropy:

$$H_M = 7,53$$

Maximum compression by entropic coding:

$$C_O = 8/7,53 = 1,062 : 1$$



Entropy:

$$H_E = 4.14$$

Maximum compression by entropic coding:

$$C_O = 8/4,14 = 1,93 : 1$$

Example

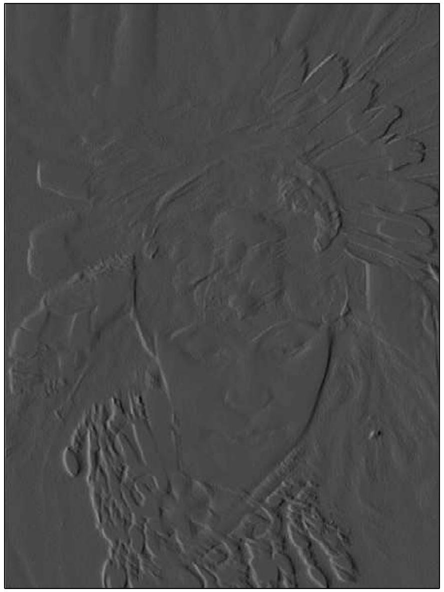
Entropy:

$$H_M = 7,53$$

Maximum compression by entropic coding:

$$C_O = 8/7,53 = 1,062 : 1$$

Huffman coding: $C_H = 1,058 : 1$



Entropy:

$$H_E = 4.14$$

Maximum compression by entropic coding:

$$C_O = 8/4,14 = 1,93 : 1$$

Huffman coding: $C_H = 1,91 : 1$

Optimal prediction – DPCM

Objective: obtain error images $f - \hat{f}$ of low entropy
$\longrightarrow$ low variance
$\Leftrightarrow$ low $MSE = \mathbb{E}\left[(f - \hat{f})^2\right]$

Differential Pulse Code Modulation:

- Model the f image as a random autocorrelated process:
  - variance: $\mathbb{E}\left[f^2\right] = \sigma^2$
  - horizontal autocorrelation: $\mathbb{E}\left[f(x,y)f(x,y-1)\right] = \sigma^2 \rho_h$
  - vertical autocorrelation: $\mathbb{E}\left[f(x,y)f(x-1,y)\right] = \sigma^2 \rho_v$
  - diagonal autocorr.: $\mathbb{E}\left[f(x,y)f(x-1,y-1)\right] = \sigma^2 \rho_{vh}$
- Consider a third order predictor based on previous neighboring pixels:

$$\begin{aligned}
\hat{f}(x,y) &= \alpha_1 f(x,y-1) + \alpha_2 f(x-1,y-1) + \alpha_3 f(x-1,y) \\
&= \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} \begin{bmatrix} f(x,y-1) \\ f(x-1,y-1) \\ f(x-1,y) \end{bmatrix} = \alpha \mathbf{f}
\end{aligned}$$

Problem: which are the optimal (low MSE) values for α_i?

Optimal prediction – DPCM

$$\begin{aligned} MSE &= \mathbb{E}\left[(f-\hat{f})^2\right] = \mathbb{E}\left[f^2\right] + \mathbb{E}\left[\hat{f}^2\right] - 2\mathbb{E}\left[f\hat{f}\right] \\ &= \sigma^2 + \sigma^2\alpha \begin{bmatrix} 1 & \rho_v & \rho_{vh} \\ \rho_v & 1 & \rho_h \\ \rho_{vh} & \rho_h & 1 \end{bmatrix} \alpha^{\mathbf{T}} - 2\sigma^2\alpha \begin{bmatrix} \rho_h \\ \rho_{vh} \\ \rho_v \end{bmatrix} \end{aligned}$$

Under certain conditions (separable autocorrelation $\rho_{vu} = \rho_v\rho_h$),

$$\alpha_1 = \rho_h, \;\; \alpha_2 = -\rho_v\rho_h, \;\; \alpha_3 = \rho_v,$$

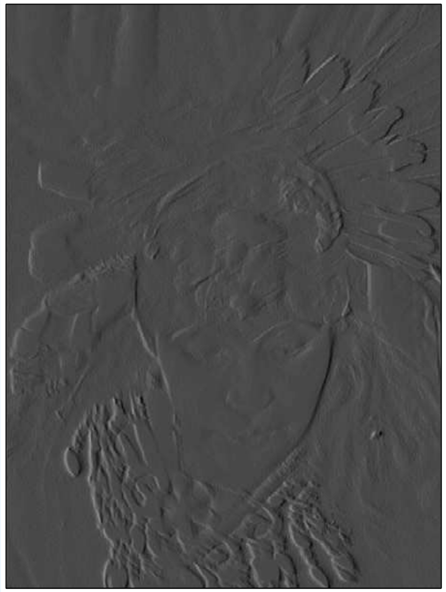
In practice, to avoid autocorrelation calculus, DCPM predictor:

$$\alpha_1 = 0,75, \;\; \alpha_2 = -0,5, \;\; \alpha_3 = 0,75$$

Observation: for second order predictors

$$\hat{f}(x,y) = \alpha_1 f(x,y-1) + \alpha_2 f(x-1,y), \;\; \alpha_1 = \alpha_2 = 0,5$$

Predictor comparisons

First order	Second order	Third order DPCM
		

Standard deviation:

$\sigma_{1o} = 7.18$ $\sigma_{2o} = 5.39$ $\sigma_{DPCM} = 3.39$

Entropy:

$H_{1o} = 4.14$ $H_{2o} = 3.78$ $H_{DPCM} = 3.28$

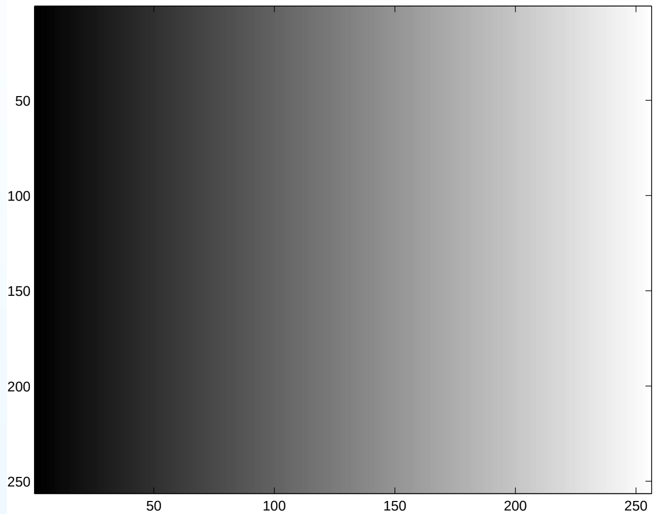
Variable length coding maximal compression rate:

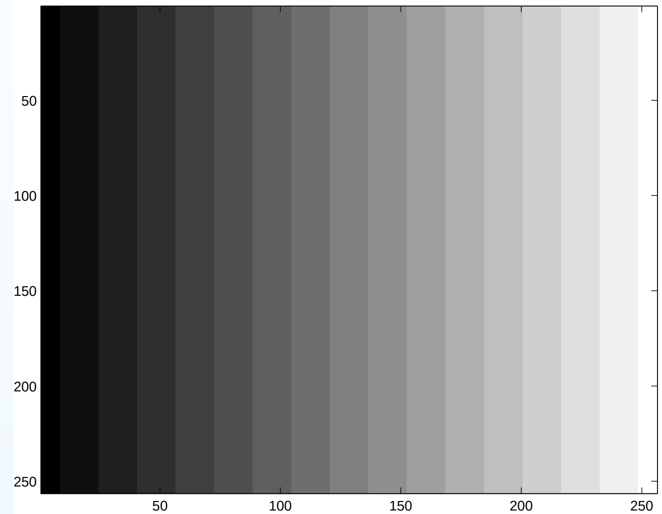
$C_{1o} = 1,93 : 1$ $C_{2o} = 2,11 : 1$ $C_{DPCM} = 2,44 : 1$

Quantizing and thresholding

Quantization

- replacing continuous functions by discrete values functions





$$\forall f \in [f_{i-1}, f_i], f \rightarrow v_i : g(f) = v_i$$

- default quantization: $v_i = f_{i-1}$
- excess quantization: $v_i = f_i$
- round quantization: $v_i = \frac{f_i + f_{i-1}}{2}$

Quantization = making two dictionaries:

- *interval* = *code* (compression, coding)
- *code* = *value* (decompression, decoding)

Quantization levels

Example: quantifying [0...255] graylevels $\rightarrow$ 3 bits:

- intervals:

  $\rightarrow$ 3 bits $\Rightarrow 2^3$ intervals (equal size?!)

  $\rightarrow [0..31], [32..63], \ldots, [224..255]$

- codes:

  $[0..31] \rightarrow 000, [32..63] \rightarrow 001, \ldots, [224..255] \rightarrow 111$

- values
  - default coding: $000 \rightarrow 0, \quad 001 \rightarrow 32, \quad \ldots, \quad 111 \rightarrow 224$
  - excess coding: $000 \rightarrow 31, \quad 001 \rightarrow 63, \quad \ldots, \quad 111 \rightarrow 255$
  - round coding: $000 \rightarrow 15, \quad 001 \rightarrow 47, \quad \ldots, \quad 111 \rightarrow 239$

Quantization levels

Example: quantifying [0. . . 255] graylevels $\rightarrow$ 3 bits:

- intervals:

  $\rightarrow$ 3 bits $\Rightarrow 2^3$ intervals (equal size?!)
  $\rightarrow [0..31], [32..63], \ldots, [224..255]$

- codes:

  $[0..31] \rightarrow 000, [32..63] \rightarrow 001, \ldots, [224..255] \rightarrow 111$

- values
  - default coding: $000 \rightarrow 0, \quad 001 \rightarrow 32, \quad \ldots, \quad 111 \rightarrow 224$
  - excess coding: $000 \rightarrow 31, \quad 001 \rightarrow 63, \quad \ldots, \quad 111 \rightarrow 255$
  - round coding: $000 \rightarrow 15, \quad 001 \rightarrow 47, \quad \ldots, \quad 111 \rightarrow 239$

$\Rightarrow$ Different approximations $\hat{f}(x) \Leftrightarrow$ *quantization* errors

$$q(x) = f(x) - \hat{f}(x)$$

$$MSE = \mathbb{E}\left[q^2\right] = \int_{-\infty}^{\infty} q^2 p(q) dq \quad \Leftrightarrow \quad SNR = \frac{\mathbb{E}\left[f^2\right]}{\mathbb{E}\left[q^2\right]}$$

Uniform probability

- take an $[f_{i-1}, f_i]$ interval
  $\longrightarrow$ quantization: $\forall f \in [f_{i-1}, f_i], f \longrightarrow \hat{f} = v_i$
- suppose f having a uniform probability on $[f_{i-1}, f_i]$
  $\longrightarrow p(f) = \frac{1}{\Delta}$, with $\Delta = f_i - f_{i-1}$
- then:
  - quantification error $q_i(x) = f(x) - v_i$
  - probability $p(q_i) = \frac{1}{\Delta}$ for $q_i \in [f_{i-1} - v_i, f_i - v_i]$
  - $MSE_i = \mathbb{E}\left[q_i^2\right] = \frac{1}{\Delta}\int_{f_{i-1}-v_i}^{f_i - v_i} q_i^2 dq$

Uniform probability

- take an $[f_{i-1}, f_i]$ interval
  $\rightarrow$ quantization: $\forall f \in [f_{i-1}, f_i], f \rightarrow \hat{f} = v_i$
- suppose f having a uniform probability on $[f_{i-1}, f_i]$
  $\rightarrow p(f) = \frac{1}{\Delta},$ with $\Delta = f_i - f_{i-1}$
- then:
  - quantification error $q_i(x) = f(x) - v_i$
  - probability $p(q_i) = \frac{1}{\Delta}$ for $q_i \in [f_{i-1} - v_i, f_i - v_i]$
  - $MSE_i = \mathbb{E}\left[q_i^2\right] = \frac{1}{\Delta} \int_{f_{i-1}-v_i}^{f_i - v_i} q_i^2 dq$
- for $v_i = f_{i-1}$ (default coding - keeping MSB):

Uniform probability

- take an $[f_{i-1}, f_i]$ interval
  $\rightarrow$ quantization: $\forall f \in [f_{i-1}, f_i], f \rightarrow \hat{f} = v_i$
- suppose f having a uniform probability on $[f_{i-1}, f_i]$
  $\rightarrow p(f) = \frac{1}{\Delta}$, with $\Delta = f_i - f_{i-1}$
- then:
  - quantification error $q_i(x) = f(x) - v_i$
  - probability $p(q_i) = \frac{1}{\Delta}$ for $q_i \in [f_{i-1} - v_i, f_i - v_i]$
  - $MSE_i = \mathbb{E}\left[q_i^2\right] = \frac{1}{\Delta}\int_{f_{i-1}-v_i}^{f_i-v_i} q_i^2 dq$
- for $v_i = f_{i-1}$ (default coding - keeping MSB):

$$MSE_i = \frac{1}{\Delta}\int_0^{\Delta} q_i^2 dq = \frac{\Delta^2}{3}$$

mean error $\mathbb{E}\left[q_i\right] = \frac{1}{\Delta}\int_0^{\Delta} q_i dq = \frac{\Delta^2}{2}$

Uniform probability

For $v_i = \frac{f_{i-1}+f_i}{2}$ (round coding):

Uniform probability

For $v_i = \frac{f_{i-1}+f_i}{2}$ (round coding):

- mean error:

$$\mathbb{E}\left[q_i\right] = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q_i dq = 0$$

- mean square error:

$$MSE_i = \text{var}(q_i) = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q_i^2 dq = \frac{\Delta^2}{12}$$

Uniform probability

For $v_i = \frac{f_{i-1}+f_i}{2}$ (round coding):

- mean error:

$$\mathbb{E}\left[q_i\right] = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q_i dq = 0$$

- mean square error:

$$MSE_i = \text{var}(q_i) = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q_i^2 dq = \frac{\Delta^2}{12}$$

Round quantization:

- optimal approximation ⇔ "optimal" error (noise)
  - zero mean error
  - minimal energy (variance) error (for uniformly distributed signals)

Uniform probability

- let f be a gray level image, having uniform probability in $[0, A] : p(f) = \frac{1}{A}$

Uniform probability

- let f be a gray level image, having uniform probability in $[0, A] : p(f) = \frac{1}{A}$
- image energy:

$$\mathbb{E}\left[f^2\right] = \int_0^A f^2 p(f) df = \frac{A^2}{3}$$

- image variance:

$$\begin{aligned} \text{var}(f) &= \mathbb{E}\left[|f - \mathbb{E}\left[f\right]|^2\right] = \mathbb{E}\left[f^2\right] - \mathbb{E}\left[f\right]^2 \\ &= \int_0^A f^2 p(f) df - \left[\int_0^A f p(f) df\right]^2 = \frac{A^2}{3} - \frac{A^2}{4} = \frac{A^2}{12} \end{aligned}$$

Uniform probability

- let f be a gray level image, having uniform probability in $[0, A] : p(f) = \frac{1}{A}$
- image energy:

$$\mathbb{E}\left[f^2\right] = \int_0^A f^2 p(f) df = \frac{A^2}{3}$$

- image variance:

$$\begin{aligned} \text{var}(f) &= \mathbb{E}\left[|f - \mathbb{E}[f]|^2\right] = \mathbb{E}\left[f^2\right] - \mathbb{E}[f]^2 \\ &= \int_0^A f^2 p(f) df - \left[\int_0^A f p(f) df\right]^2 = \frac{A^2}{3} - \frac{A^2}{4} = \frac{A^2}{12} \end{aligned}$$

- take round quantization on n bits (2^n intervals)
- take equal intervals $\Rightarrow$ size $\Delta = A/2^n$
- f has also a uniform probability on each $[f_{i-1}, f_i]$

Uniform probability

Then:

- mean square error = error variance

$$\text{var}(q) = MSE = \frac{\Delta^2}{12} = \frac{A^2}{12 \cdot 2^{2n}}$$

Uniform probability

Then:

- mean square error = error variance

$$\text{var}(q) = MSE = \frac{\Delta^2}{12} = \frac{A^2}{12 \cdot 2^{2n}}$$

- signal to noise (error) (relative) ratio

$$SNR = \frac{\text{var}(f)}{\text{var}(q)} = 2^{2n}$$

Uniform probability

Then:

- mean square error = error variance

$$\text{var}(q) = MSE = \frac{\Delta^2}{12} = \frac{A^2}{12 \cdot 2^{2n}}$$

- signal to noise (error) (relative) ratio

$$SNR = \frac{\text{var}(f)}{\text{var}(q)} = 2^{2n}$$

- SNR in decibels

$$SNR_{dB} = 10 \log_{10} 2^{2n} = 20n \log_{10} 2 \approx 6n$$

Uniform probability

Then:

- mean square error = error variance

$$\text{var}(q) = MSE = \frac{\Delta^2}{12} = \frac{A^2}{12 \cdot 2^{2n}}$$

- signal to noise (error) (relative) ratio

$$SNR = \frac{\text{var}(f)}{\text{var}(q)} = 2^{2n}$$

- SNR in decibels

$$SNR_{dB} = 10 \log_{10} 2^{2n} = 20n \log_{10} 2 \approx 6n$$

- Each bit adds to the precision of the approximation 6 dB !
- If the quantified image (the approximation) is seen as the original *plus/minus* the quantization error (noise), a 8 bits quantization implies 48 dB SNR !
$\Leftrightarrow$ practically no difference between continuous gray levels and 256 gray levels images!

General probability

In general, the pixel's gray level distribution is NOT uniform.

- take an $[f_{i-1}, f_i]$ interval
  $\rightarrow$ quantization: $\forall f \in [f_{i-1}, f_i], f \longrightarrow \hat{f} = v_i$
- suppose f having a *unknown* probability $p(f)$ on $[f_{i-1}, f_i]$
- then:
  - quantification error $q_i(x) = f(x) - v_i$
  - mean error:

$$\mathbb{E}\left[q_i\right] = \int_{f_{i-1}}^{f_i} (f - v_i) p(f) df$$

$$MSE_i = \mathbb{E}\left[q_i^2\right] = \int_{f_{i-1}}^{f_i} (f - v_i)^2 p(f) df$$

Optimality conditions:

- mean error = 0 $\Rightarrow v_i$ not necessarily $= \frac{f_{i-1}+f_i}{2}$
- MSE minimal $\Rightarrow [f_{i-1}, f_i]$ intervals not necessarily equal

Lloyd-Max algorithm

Estimation of v_i values and $[f_{i-1}, f_i]$ intervals for a given image.

1. Choose the number of bits n (2^n quantization levels)
2. Randomly initialize v_i values
3. Create quantized image by attributing to each pixel the closest value $v_i \Leftrightarrow$ implicitly creates intervals $[f_{i-1}, f_i]$ with

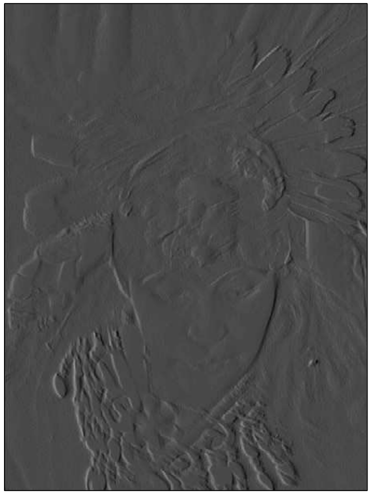
$$f_i = \frac{v_i + v_{i+1}}{2}$$

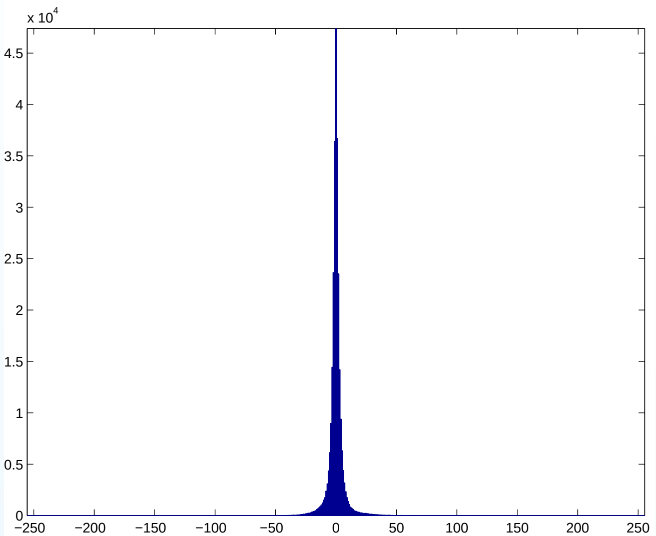
4. Compute the empirical mean value for each interval $[f_{i-1}, f_i]$ and make v_i equal to this value
5. Goto step 3 until convergence

Optimal Lloyd-Max quantifier

Particular interest: $\longrightarrow$ Laplacian probability distributions
$\longrightarrow$ prediction error images

Laplacian probability





Laplace law:

$$p(x) = \frac{1}{\sigma_x\sqrt{2}} e^{-\frac{\sqrt{2}|x|}{\sigma_x}}$$

Example: error image

$$e(x,y) = f(x,y) - f(x,y-i)$$

Almost Laplacian, with

- mean value $\mu_e \approx 0$
- standard deviation $\sigma_e \approx 7,18$

Laplacian probability

For $\sigma_x = 1$:

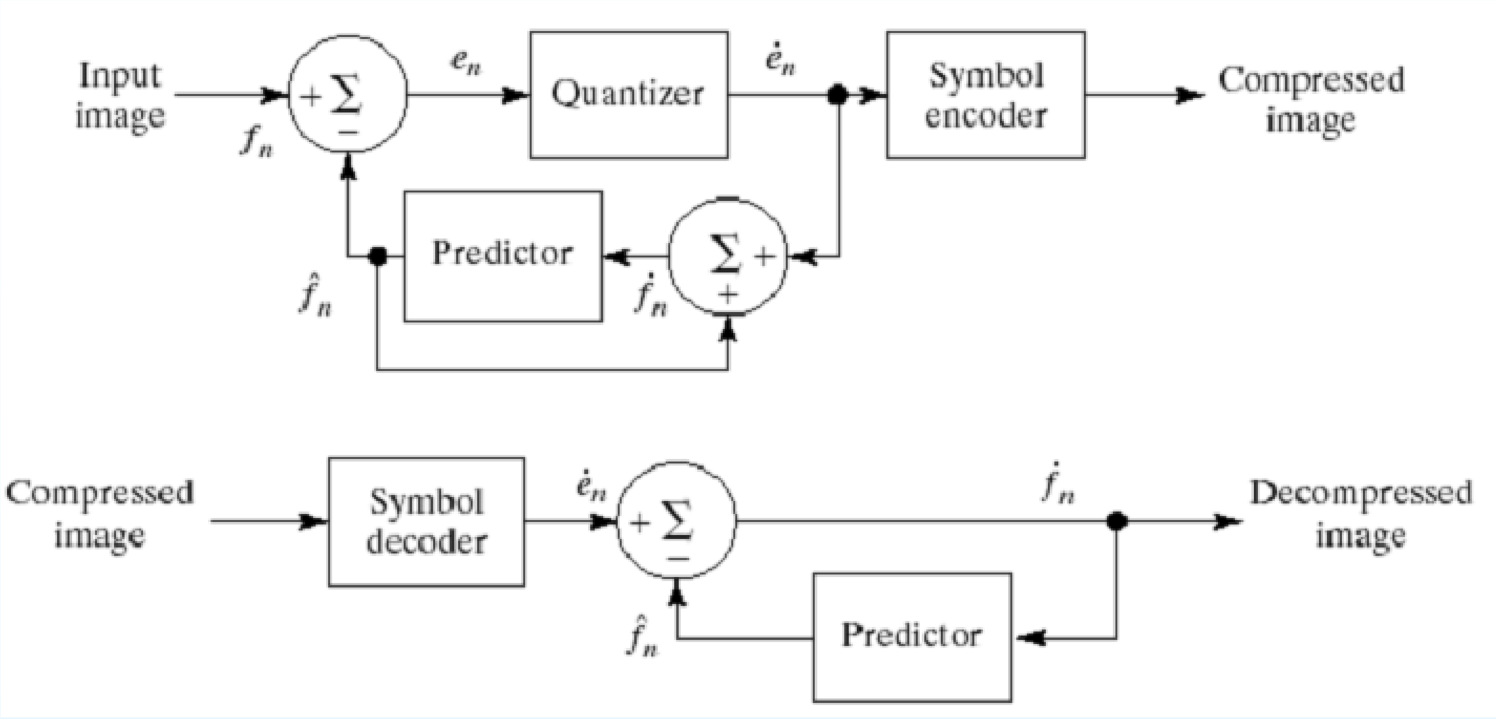
	Nb. of quantization bits					
	1		2		3	
Code	f_i	v_i	f_i	v_i	f_i	v_i
0	$-\infty$	-0,707	$-\infty$	-1,810	$-\infty$	-2,994
1	0	0,707	-1,102	-0,395	-2,285	-1,576
2	∞		0	0,395	-1,181	-0,785
3			1,102	1,810	-0,504	-0,222
4			∞		0	0,222
5					0,504	0,785
6					1,181	1,576
7					2,285	2,994
					∞	

For real images, $[f_{i-1}, f_i]$ intervals and v_i values are obtained by multiplying by the real σ and rounding.

Prediction error quantization

Combination of predictors and quantizers

Lossy predictive coding:



Prediction error quantization

Combination of predictors and quantizers

Lossy predictive coding:

- Delta Modulation (DM):
  - differential predictor (1 pixel, by line) + 1 bit quantization
    - if error < 0 ($e(x,y) = f(x,y) - \hat{f}(x,y) \in (-\infty, 0]$)
      $v_i = -$value (Lloyd-Max, uniform, empiric, ...)
    - else
      $v_i = +$value
- Lossy Differential Pulse Code Modulation (DPCM):
  - optimal DPCM predictor (3 preceding neighboring pixels)
  - n bit quantization, with $[f_{i-1}, f_i]$ intervals and v_i values given by Lloyd-Max or uniform quantization
- Adaptive predictive quantization:
  - optimal predictor + n bit quantization *by sub-image*

Example of DM

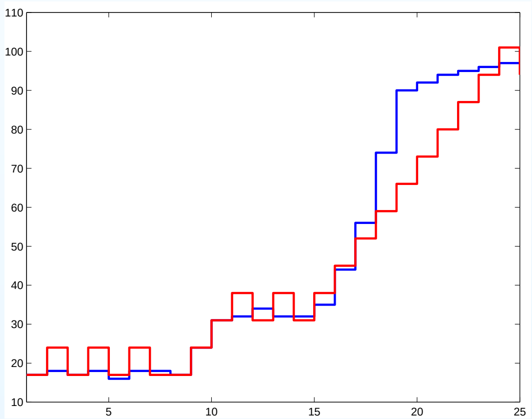
- Error image: Laplace distribution with $\sigma_e = 10$
- Delta modulation with Lloyd-Max quantizer: $v_{0,1} = \pm 7$
- Current image line $f_x(y)$:

$$[17, 18, 17, 18, 16, 18, 18, 17, 24, 31, 32, 34, 32, 32, 35, 44, 56, 74, 90, 92, 94, 95, 96, 97, 97]$$

Example of DM

- Error image: Laplace distribution with $\sigma_e = 10$
- Delta modulation with Lloyd-Max quantizer: $v_{0,1} = \pm 7$
- Current image line $f_x(y)$:

$$[17, 18, 17, 18, 16, 18, 18, 17, 24, 31, 32, 34, 32, 32, 35, 44, 56, 74, 90, 92, 94, 95, 96, 97, 97]$$

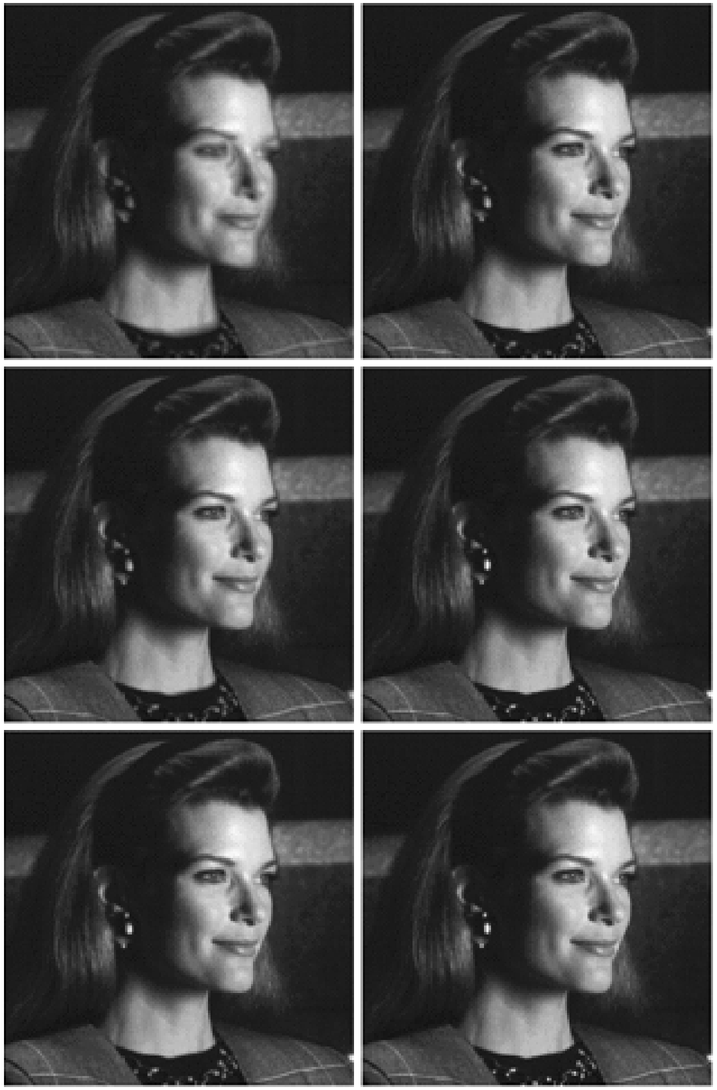


Encoder					Decoder		Error
f	$\hat{f}$	e	$\dot{e}$	$\dot{f}$	$\hat{f}$	$\dot{f}$	$f - \dot{f}$
17	-	-	-	17	-	17	0
18	17	1	7	24	17	24	-6
17	24	-7	-7	17	24	17	0
18	17	1	7	24	17	24	-6
16	24	-8	-7	17	24	17	-1
.	.	.	.	.	.	.	.
35	31	4	7	38	31	38	-3
44	38	6	7	45	38	45	-1
56	45	11	7	52	45	52	4
74	52	22	7	59	52	59	15
90	59	31	7	66	59	66	24
92	66	26	7	73	66	73	19
94	73	21	7	80	73	80	14
95	80	15	7	87	80	87	8
96	87	9	7	94	87	94	2
97	94	3	7	101	94	101	-4
97	101	-4	-7	94	101	94	3

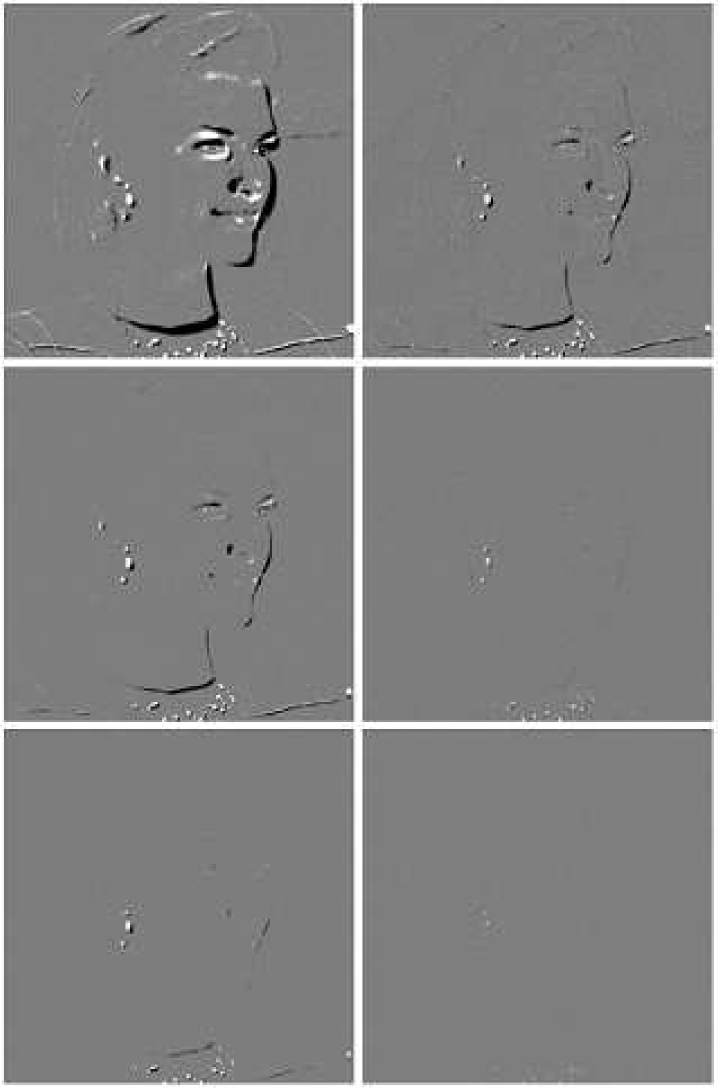
- granular noise
- slope overload

Example of lossy DPCM (Lloyd-Max)

Compressed images

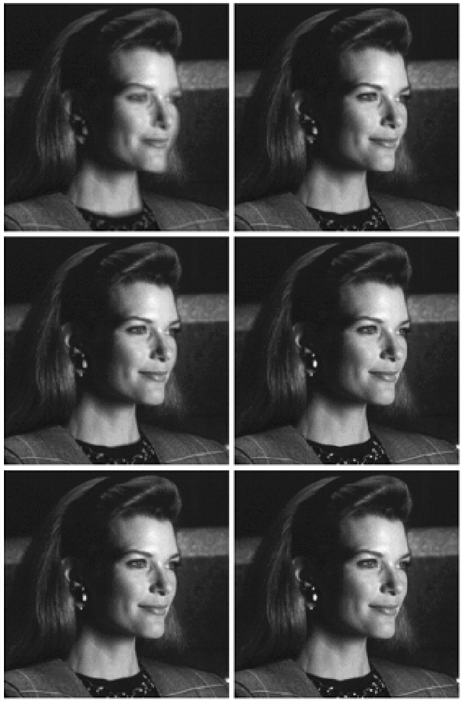


Error images



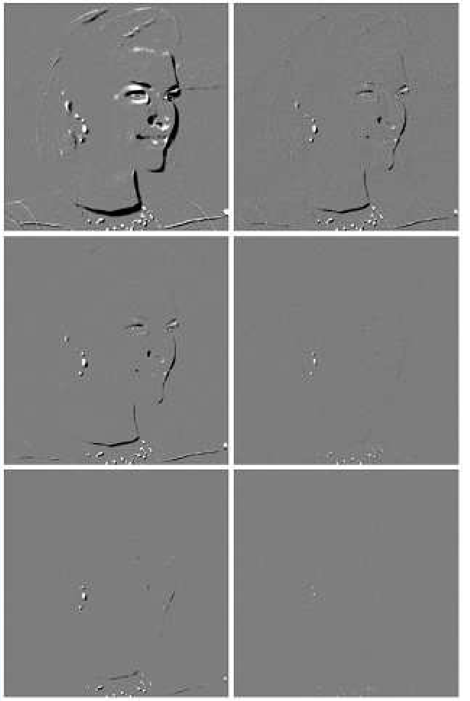
Example of lossy DPCM (Lloyd-Max)

(a) (b) (c)



(d) (e) (f)

(a) (b) (c)



(d) (e) (f)

	global			adaptive (4×4 pixels)		
	(a)	(b)	(c)	(d)	(e)	(f)
Bits/pixel	1	2	3	1,125	2,125	3,125
Compression ratios	8:1	4:1	2,66:1	7,11:1	3,77:1	2,56:1
MSE	9,90	4,30	2,31	4,61	1,70	0,76

Thresholding

- Idea:
  keeping only significant values while making all others 0
- Problem: what is significant?
- Difficult to say on original images but:
  - on prediction error images
    - low prediction error ⇔ small variations in the original image ⇔ imperceptible for human eye
  - on transformed images
    - small value coefficients (Fourier) ⇔ small variations in the original image ⇔ imperceptible for human eye

Mainly on transformed images. Adapted by sub-images.

Algorithm:

1. Split the original image in sub-images (optional!)
2. Transform each sub-image
3. Threshold (make small values 0)
4. Code 0 values on 1 bit and Run Length

Thresholding&Quantizing

- Idea:
  quantizing only significant values while making all others 0
- Method:
  - change the original values by multiplication (mask)
  - quantize&threshold (small values $\rightarrow 0$ by quantizing)
  $\Leftrightarrow$ non-uniform quantization !
- On transformed images
  - small value coefficients rounded to 0, all others quantified depending on their psycho-visual importance

Algorithm:

1. Split the original image in sub-images (optional!)
2. Transform each sub-image
3. Multiply the transform's coefficients by a mask
4. Quantize the result
5. Code 0 values on 1 bit and Run Length

Color space transforms

Color to grayscale images

All colors → mixtures of primary colors R,G,B

- Standard form of a color image → three superposed images (pure red, green and blue)
- RGB coding: 8 bits (256 levels) for each color
- each of the three images can be seen as a gray level image

Is this representation optimal from a compression point of view?

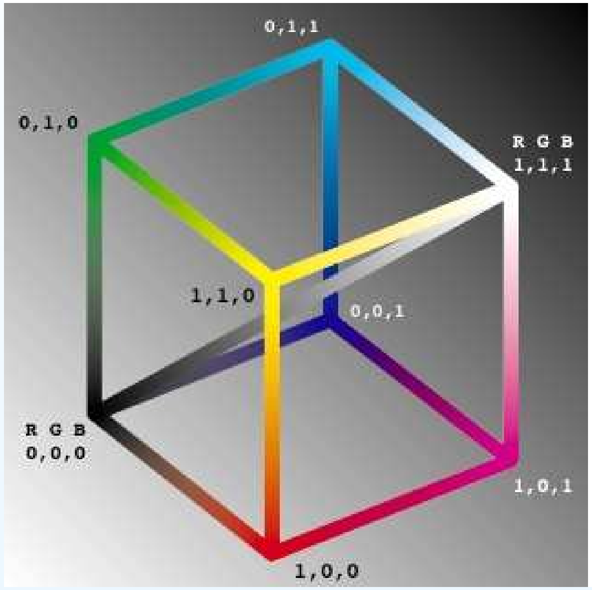
- human vision has different sensibilities for the three colors
- human eye is more sensible to luminance than to colors
- luminance ∼ gray level image (black&white television)

Compression idea:

- change RGB coding to luminance plus pseudo-color
- keep most of the luminance information
- discard the less visible pseudo-color information

RGB to YCbCr

Y = luminance **Cb** = chrominance blue **Cr**=chrominance red



Approximate transform:

- diagonal gray level (intensity, luminance): ≈**Y**
- eye different sensibility to red, green and blue, so:

$$Y \approx 0,3R + 0,6G + 0,1B$$

- chrominances:

$$Cb = \frac{B-Y}{2} + 128 = -0,15R - 0,3G + 0,45B + 128$$

$$Cr = \frac{R-Y}{1.6} + 128 = 0,44R - 0,38G - 0,06B + 128$$

RGB to YCbCr

Usually implemented color transform (JPEG/JPEG2000):

$$
\begin{aligned}
Y &= 0.2568R + 0.5041G + 0.0979B + 16 \\
Cb &= -0.1482R - 0.2910G + 0.4392B + 128 \\
Cr &= 0.4392R - 0.3678G - 0.0714B + 128
\end{aligned}
$$

Original image	G & B subsampling	Cb & Cr subsampling
R:8, G:8, B:8	R:8, G:2, B:2	Y:8, Cb:2, Cr:2
1:1	2:1	2:1



RGB to YCbCr

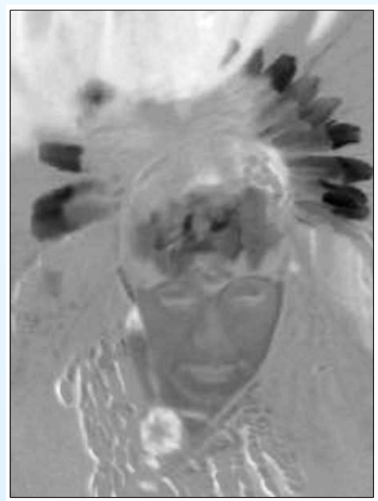
Red image

Green image

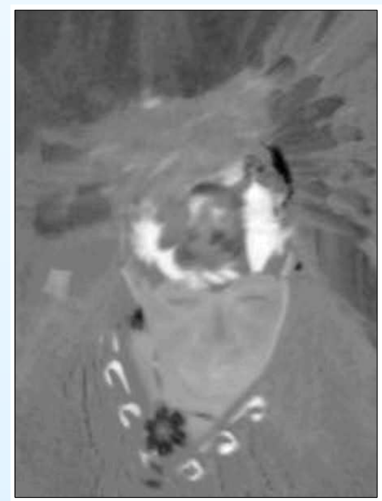
Blue image

Y image

Cb image



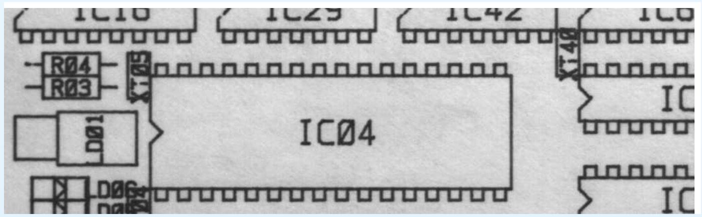
Cr image

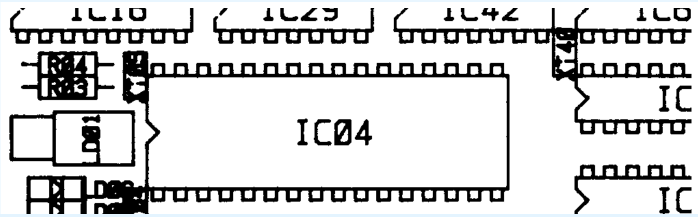


Grayscale to black&white images

Binarizing

- Most "brutal" quantizing
- Adapted for simple gray scale images:
  → 2 modes distribution (lot of clear pixels, lot of dark pixels)
  → Scanned documents, drawings, handwriting, . . .
- Fixed threshold algorithm:
  For each pixel:
  - if *gray level > threshold*
    pixel value=1 (white)
    else
    pixel value=0 (black)
- Binary image compression methods (RLC, CAC, . . .)





Grayscale to black&white images

Bit-Plane Coding

- Each image (color or monochrome) is a superposition of at most 3 monochrome (gray level) images
- Each pixel gray level $l \rightarrow$ 8 bits (MSB $\rightarrow$ LSB):

$$l = b_7 2^7 + b_6 2^6 + \cdots + b_0 2^0 = \sum_{i=0}^{7} b_i 2^i$$

  with $b_i = \{0, 1\}$
- Bit-plane i= black&white image constructed with b_i values of each pixel
- Compression techniques:
  - Quantizing = keeping only the MS bits $b_i, i >$ level $\Leftrightarrow$ only i bit-planes
  - Gray code conversion
  - Binary image compression methods (run-length, constant area, . . .)

Bit-Plane Coding

Bit 7

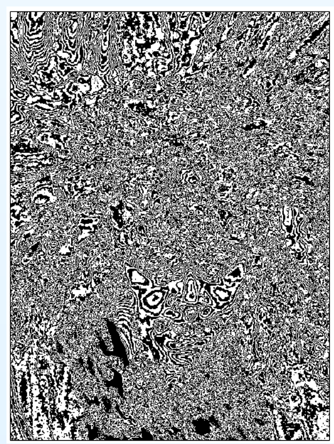
Bit 6

Bit 5

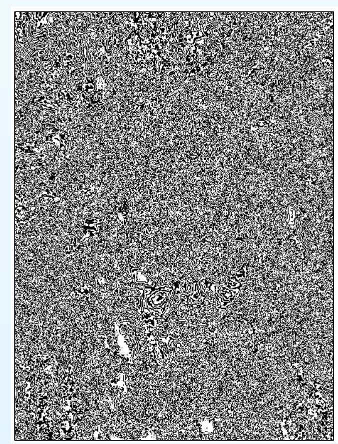
Bit 4

Bit 3

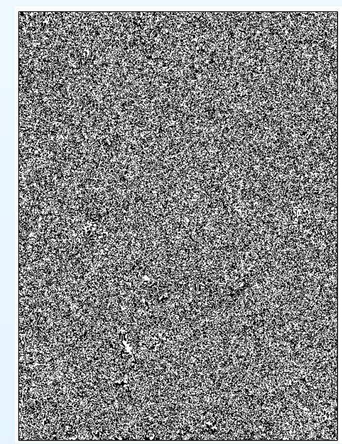
Bit 2



Bit 1

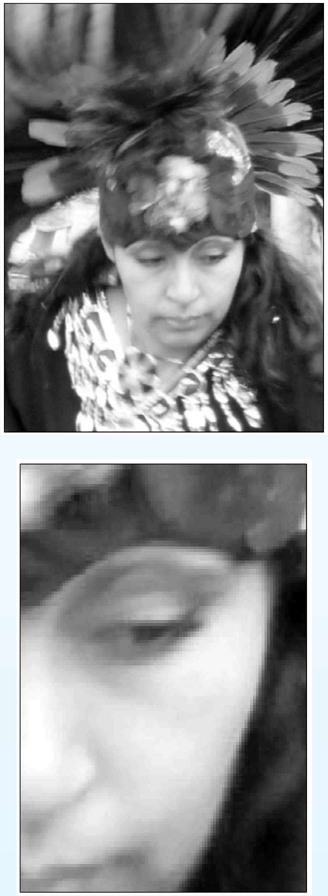


Bit 0

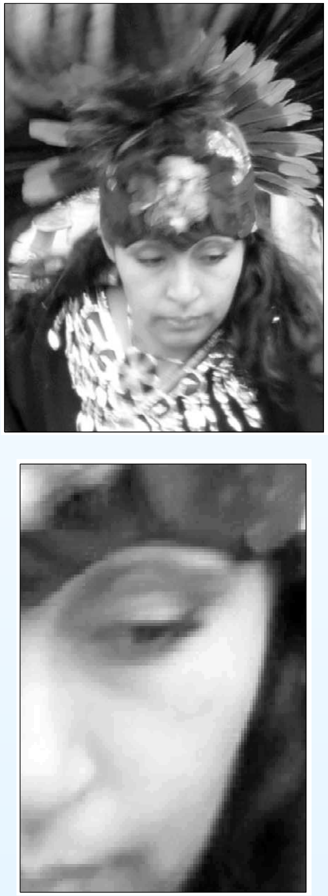


Quantizing

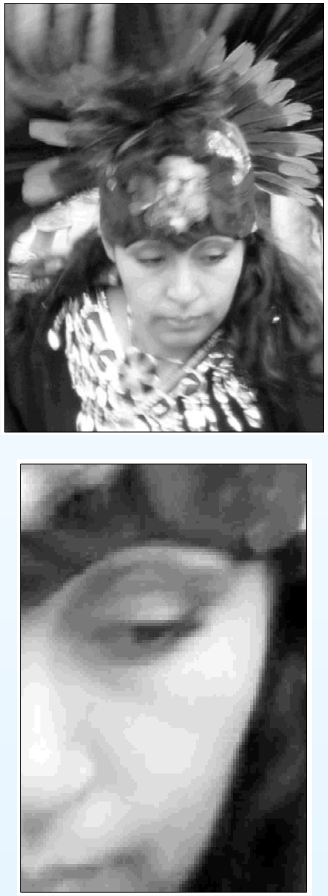
C=8:7=1,14:1



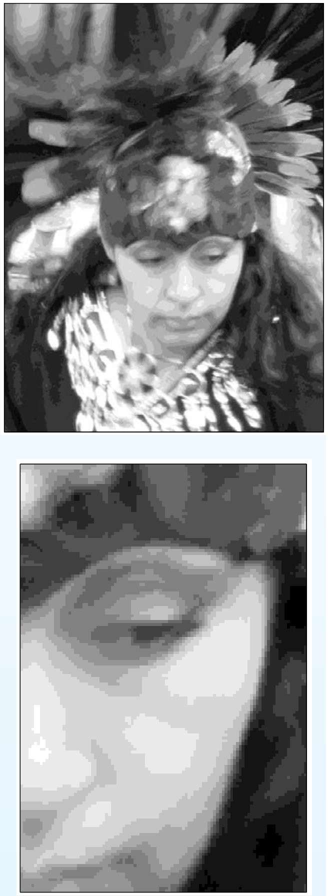
C=8:6=1,33:1



C=8:5=1,6:1



C=8:4=2:1



IGS Quantizing

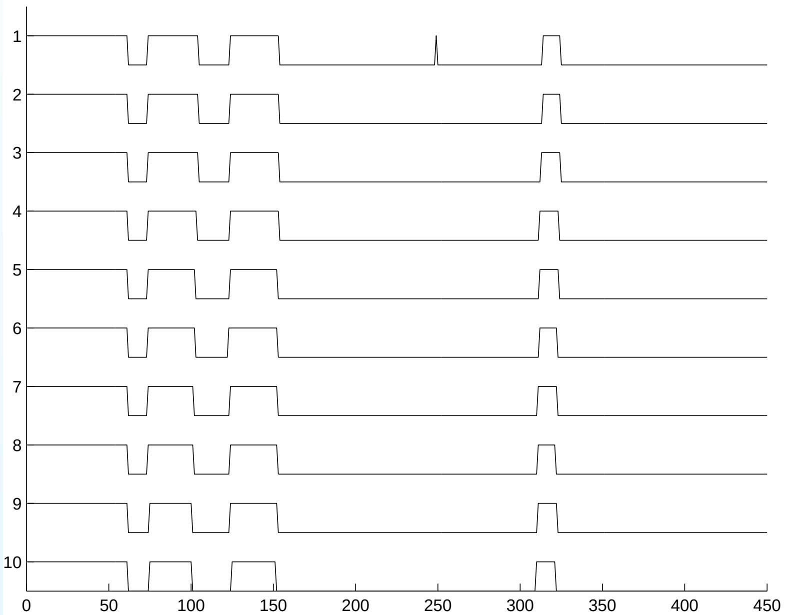
Original | 4 bits quantizing | 4 bits IGS quantizing

Run-length coding

- quantization = keeping only significant bit-planes
- RLC = coding each bit-plane individually

Bit-plane 7

lines 1-10



RL Coding

- line 1: 61 12 31 19 30 95 1 64 11 126
- line 2: 61 12 31 19 30 160 11 126
- . . .

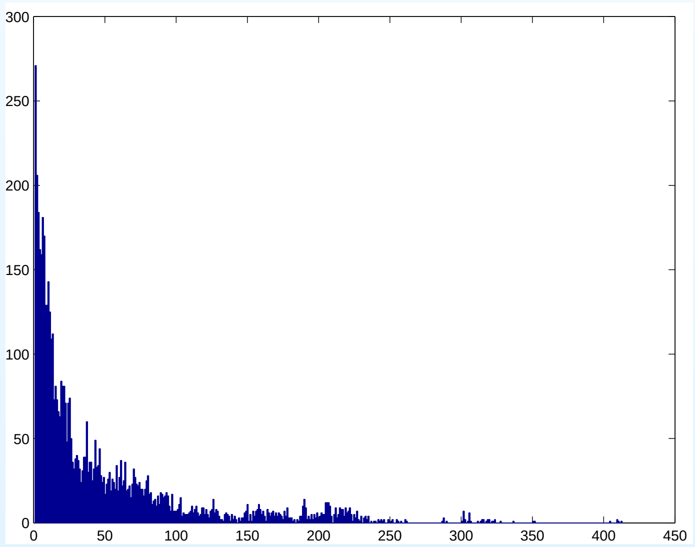
Run-length coding

RL Coding (bits/pixel)

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	Total	C_R
0.19	0.52	0.96	1.00	1.00	1.00	1.00	1.00	$6,67$	$1,2:1$

- No compression for bit-planes $i \leq 4$ (9 bit/length)
- Possible further compression by entropic coding (Huffman)

$$H(B_7) = 6,8 \text{bits/pixel}$$



- Possible further compression by quantizing

BCD to Gray code

- BCD coding: each pixel gray level $l \rightarrow$ 8 bits:

$$l = b_7 2^7 + b_6 2^6 + \cdots + b_0 2^0 = \sum_{i=0}^{7} b_i 2^i \text{ with } b_i = \{0, 1\}$$

- Sensible to small variations in gray level values

$$\begin{array}{ccc} 127 & \rightarrow & 01111111 \\ 128 & \rightarrow & 10000000 \end{array}$$

$\rightarrow$ a visually imperceptible change $\Rightarrow$ changing all bit-planes!

- Gray code $\rightarrow$ only one bit changes for each gray level unit
- BCD $\rightarrow$ Gray code conversion

$$\begin{array}{ll} \text{MSB} & g_7 = b_7 \\ \text{bits 0-6} & g_i = b_i \oplus b_{i+1} \end{array}$$

with $\oplus$ the XOR (exclusive OR) symbol.

Bit-Plane Coding - Gray Code

Bit 7

Bit 6

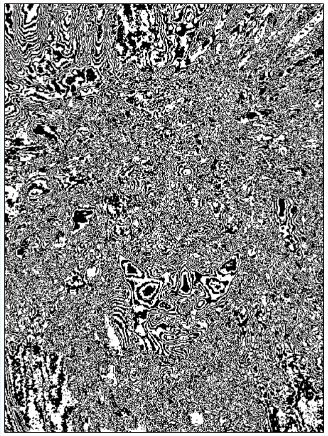
Bit 5

Bit 4

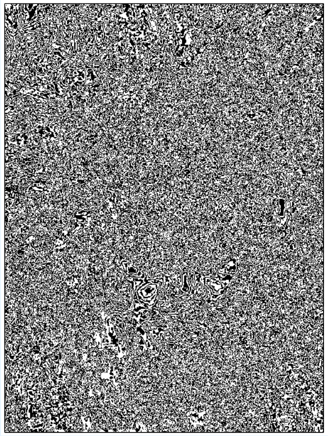
Bit 3

Bit 2

Bit 1



Bit 0



Bit-Plane Coding - BCD / Gray Code

BCD 7

BCD 6

BCD 5

BCD 4

Gray 7

Gray 6

Gray 5

Gray 4

Run-length coding

BCD bit-planes RL Coding (bits/pixel)

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	Total	C_R
0.19	0.52	0.96	1.00	1.00	1.00	1.00	1.00	$6,67$	$1,2:1$

Gray coded bit-planes RL Coding (bits/pixel)

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	Total	C_R
0.19	0.34	0.47	0.92	1.00	1.00	1.00	1.00	$5,92$	$1,4:1$

- No compression for bit-planes $i \leq 3$ (9 bit/length)
- Possible further compression by entropic coding (Huffman)
- Possible further compression by quantizing (worse result for the same quantization!)

Image transforms

Changing the point of view

All described approaches $\longrightarrow$ spatial domain

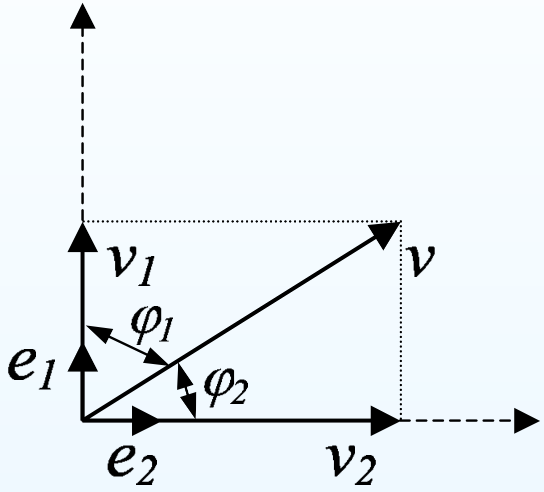
How about translating to another domain ?
- Thank you $\longrightarrow$ Merci

Image transforms: Fourier, DCT, wavelets, . . .

- **Idea:** describing a function (an image, a signal) using simple, elementary basis functions
- **Method:** each image is written as a linear combination of basis functions
- **Result:** the coefficients of this linear combination describe the image

Geometric analogy

Let v be a vector in $\mathbb{R}^2$

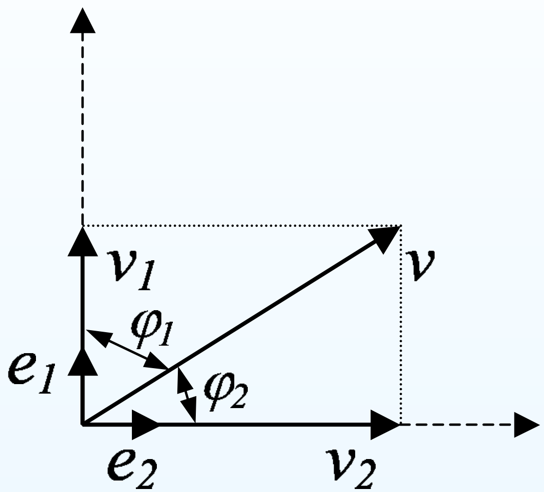


- $\overrightarrow{v} = \overrightarrow{v_1} + \overrightarrow{v_2} = a_1\overrightarrow{e_1} + a_2\overrightarrow{e_2}$
- $\overrightarrow{e_1} \perp \overrightarrow{e_2}$ basis vectors:
  - null scalar product: $\langle \overrightarrow{e_1}, \overrightarrow{e_2} \rangle = 0$
  - unit length: $|e_1| = \sqrt{\langle \overrightarrow{e_1}, \overrightarrow{e_1} \rangle} = 1$
- $a_1 = \langle \overrightarrow{v}, \overrightarrow{e_1} \rangle = |v||e_1| \cos\phi_1$
  $a_2 = \langle \overrightarrow{v}, \overrightarrow{e_2} \rangle = |v||e_2| \cos\phi_2$

Within a predefined orthonormal basis $\overrightarrow{e_i}$, the coefficients a_i perfectly describe the vector of interest

Geometric analogy

Let v be a vector in $\mathbb{R}^2$



- $\overrightarrow{v} = \overrightarrow{v_1} + \overrightarrow{v_2} = a_1\overrightarrow{e_1} + a_2\overrightarrow{e_2}$
- $\overrightarrow{e_1} \perp \overrightarrow{e_2}$ basis vectors:
  - null scalar product: $\langle \overrightarrow{e_1}, \overrightarrow{e_2} \rangle = 0$
  - unit length: $|e_1| = \sqrt{\langle \overrightarrow{e_1}, \overrightarrow{e_1} \rangle} = 1$
- $a_1 = \langle \overrightarrow{v}, \overrightarrow{e_1} \rangle = |v||e_1| \cos\phi_1$
  $a_2 = \langle \overrightarrow{v}, \overrightarrow{e_2} \rangle = |v||e_2| \cos\phi_2$

Within a predefined orthonormal basis $\overrightarrow{e_i}$, the coefficients a_i perfectly describe the vector of interest

Pythagoras Theorem:

$$\begin{aligned} |v|^2 &= \langle \overrightarrow{v}, \overrightarrow{v} \rangle = \langle \overrightarrow{v_1} + \overrightarrow{v_2}, \overrightarrow{v_1} + \overrightarrow{v_2} \rangle \\ &= \langle \overrightarrow{v_1}, \overrightarrow{v_1} \rangle + 2\langle \overrightarrow{v_1}, \overrightarrow{v_2} \rangle + \langle \overrightarrow{v_2}, \overrightarrow{v_2} \rangle \\ &= a_1^2 + a_2^2 \end{aligned}$$

Geometric analogy

Let $\overrightarrow{v}$ be a vector in $\mathbb{R}^n$

- decomposition: $\overrightarrow{v} = \sum_i^n \overrightarrow{v_i} = \sum_i^n a_i \overrightarrow{e_i}$
- coefficients: $a_i = \langle \overrightarrow{v}, \overrightarrow{e_i} \rangle$
- Pythagoras: $|v|^2 = \sum_i^n a_i^2$

Imagine an algorithm that rearranges the coefficients a_i by their absolute value: $|a_1| \geq |a_2| \geq |a_3| \geq ...|a_n|$

- perfect reconstruction: $\overrightarrow{v} = \sum_i^n \overrightarrow{v_i} = \sum_i^n a_i \overrightarrow{e_i}$
  $\rightarrow$ the vector can be perfectly reconstructed if we know the coefficients
- approximation: $\overrightarrow{v_K} = \sum_i^{K<n} \overrightarrow{v_i} = \sum_i^{K<n} a_i \overrightarrow{e_i}$
  $\rightarrow$ the vector can be approximatively reconstructed using only the greatest coefficients
- amelioration: $\overrightarrow{v_{K+1}} = \sum_i^{K+1<n} \overrightarrow{v_i} = \overrightarrow{v_K} + a_{k+1} \overrightarrow{e_{k+1}}$
  $\rightarrow$ better approximation can be obtained, if needed, by iteration

Functional analysis

A basis of functions:

- **complete:** *all* functions $f(x)$ (vectors, signals, images) can be written as a weighted sum of basis functions $\psi_u(x)$) (vectors, signals, images):

$$f(x) = \sum_u F_u \psi_u(x) \tag{1}$$

- **orthonormal:** the basis functions $\psi_u(x)$ are orthogonal and have unit norm (length)

$$\langle \psi_u(x), \psi_u(x) \rangle = \delta_u \tag{2}$$

Functional analysis

A basis of functions:

- **complete:** *all* functions $f(x)$ (vectors, signals, images) can be written as a weighted sum of basis functions $\psi_u(x)$) (vectors, signals, images):

$$f(x) = \sum_u F_u \psi_u(x) \tag{1}$$

- **orthonormal:** the basis functions $\psi_u(x)$ are orthogonal and have unit norm (length)

$$\langle \psi_u(x), \psi_u(x) \rangle = \delta_u \tag{2}$$

Then:

- the equation (1) is the expression of a function (signal, image) as a sum of its projections on the basis functions
- the coefficients F_u are the scalar values of these projections:

$$F_u = \langle f(x), \psi_u(x) \rangle$$

Physical interpretation of F_u

The scalar product of a two discrete signals (images) having N samples is defined as:

$$\langle f(x), g(x) \rangle = \sum_{1}^{N} f(x) g(x) \qquad (3)$$

Physical interpretation of F_u

The scalar product of a two discrete signals (images) having N samples is defined as:

$$\langle f(x), g(x) \rangle = \sum_{1}^{N} f(x)g(x) \qquad (3)$$

The same equation (3) define also **the correlation** between the signals f and g (up to the multiplicative factor N).

Physical interpretation of F_u

The scalar product of a two discrete signals (images) having N samples is defined as:

$$\langle f(x), g(x)\rangle = \sum_{1}^{N} f(x)g(x) \qquad (3)$$

The same equation (3) define also **the correlation** between the signals f and g (up to the multiplicative factor N).

Observations:

- important correlation between f and $\psi_u \iff f$ similar to $\psi_u \iff F_u$ grand
- decorrelation $\iff f$ orthogonal to $\psi \iff F_u \to 0$

Physical interpretation of F_u

The scalar product of a two discrete signals (images) having N samples is defined as:

$$\langle f(x), g(x) \rangle = \sum_{1}^{N} f(x) g(x) \tag{3}$$

The same equation (3) define also **the correlation** between the signals f and g (up to the multiplicative factor N).

Observations:

- important correlation between f and $\psi_u \;\Leftrightarrow f$ similar to $\psi_u \;\Leftrightarrow\; F_u$ grand
- decorrelation $\Leftrightarrow\; f$ orthogonal to $\psi \;\Leftrightarrow\; F_u \rightarrow 0$

Parseval Theorem:

$$||f||^2 = \sum_x |f(x)|^2 = \sum_u |F_u|^2. \tag{4}$$

$\Leftrightarrow$**Pythagoras Theorem**

Physical interpretation of F_u

Observation:

- the norm of a signal $||f||^2 = \langle f, f \rangle$ is the energy!
- according to Parseval, $||f||^2 = \sum_u |F_u|^2$
- F_u are the weights (coefficients) of the unitary norm (*energy*) basis functions

$\Rightarrow$ each coefficient F_u is a measure of the energy contributed by the basis function ψ_u to the signal of interest !

Physical interpretation of F_u

Observation:

- the norm of a signal $||f||^2 = \langle f, f \rangle$ is the energy!
- according to Parseval, $||f||^2 = \sum_u |F_u|^2$
- F_u are the weights (coefficients) of the unitary norm (*energy*) basis functions

$\Rightarrow$ each coefficient F_u is a measure of the energy contributed by the basis function ψ_u to the signal of interest !

Consequently:

- F_u coefficients represent the original function $f(x)$
- F_u are a measure of energy of the basis function ψ_u
- F_u are a measure of similarity between the signal of interest and the basis function ψ_u

Physical interpretation of F_u

Observation:

- the norm of a signal $||f||^2 = \langle f, f \rangle$ is the energy!
- according to Parseval, $||f||^2 = \sum_u |F_u|^2$
- F_u are the weights (coefficients) of the unitary norm (*energy*) basis functions

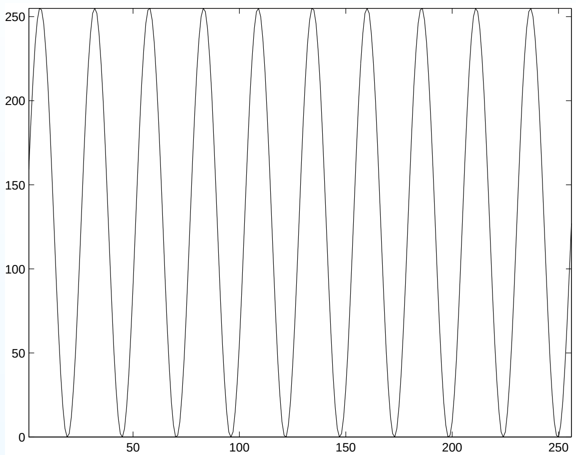
$\Rightarrow$ each coefficient F_u is a measure of the energy contributed by the basis function ψ_u to the signal of interest !

Consequently:

- F_u coefficients represent the original function $f(x)$
- F_u are a measure of energy of the basis function ψ_u
- F_u are a measure of similarity between the signal of interest and the basis function ψ_u

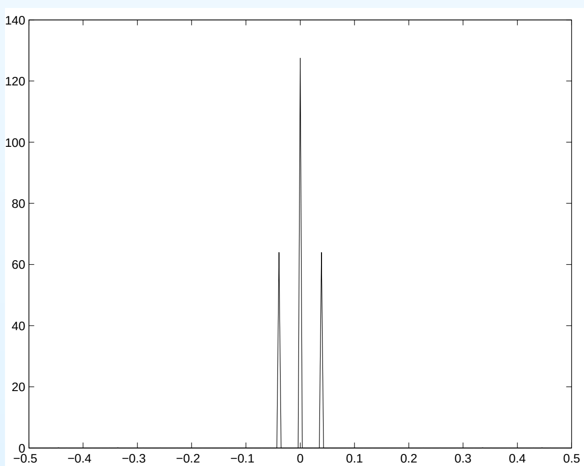
How do we choose the coefficients?
How do we choose the basis?

Fourier Transform: example



- line of a 256 gray-levels 256×256 image
- $f(x) =$ fix $(128[\sin(2\pi f x) + 0,999])$, with $x = 1 \ldots 256$, $f = 10/256$
- sine oscillating 10 times

Fourier transform:



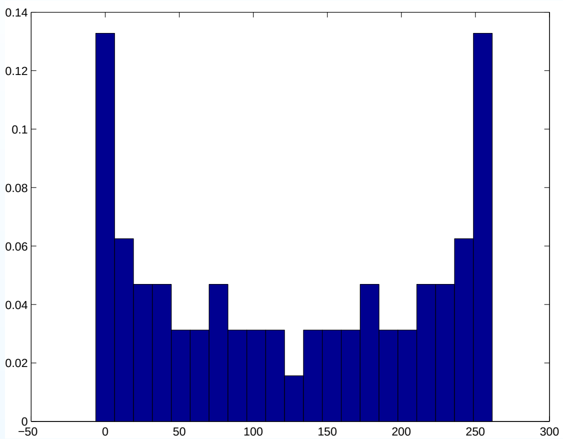
- $f(x)$ can be written as:

$$f(x) = \sum_u a_u \psi_u(x) = \sum_f F(f) e^{j2\pi f x}$$

with $F(f) = \frac{1}{256} \sum_x f(x) e^{-j2\pi f x}$

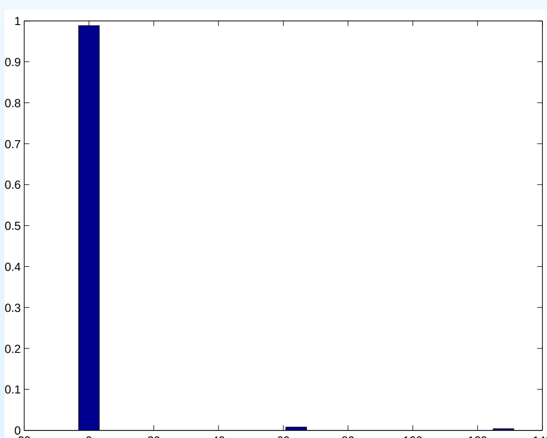
- basis functions $\psi_f(x)$:
  $\longrightarrow$ complex exponentials $e^{j2\pi f x}$
- coefficients a_f:
  $\longrightarrow$ Fourier transform $F(f)$

Fourier Transform: example



- entropy estimation:
  $H(f) = -\sum p(f) \log p(f) = 4,17$
  $\rightarrow$ *low coding redundancy*
- *important spatial redundancy* (periodic function) but impossible differential predictive coding
- rather *low psycho-visual redundancy*, difficult to quantize

Fourier transform:



- entropy estimation:
  $H(F) = -\sum p(F) \log p(F) = 0,10$
  $\rightarrow$ *important coding redundancy*
- *important spatial redundancy* (constant function)
- *important psycho-visual redundancy*, easy to quantize *and* to threshold

Choosing coefficients. Partial reconstruction

As we have seen:

- according to Parseval, an important coefficient $F_u \Rightarrow$ a great amount of the energy of the function (signal) $f(x)$ is contributed by the basis function (signal) ψ_u

Choosing coefficients. Partial reconstruction

As we have seen:

- according to Parseval, an important coefficient $F_u \Rightarrow$ a great amount of the energy of the function (signal) $f(x)$ is contributed by the basis function (signal) ψ_u
- an important coefficient $F_u \Rightarrow$ the signal $f(x)$ is similar to the basis signal ψ_u

Choosing coefficients. Partial reconstruction

As we have seen:

- according to Parseval, an important coefficient $F_u \Rightarrow$ a great amount of the energy of the function (signal) $f(x)$ is contributed by the basis function (signal) ψ_u
- an important coefficient $F_u \Rightarrow$ the signal $f(x)$ is similar to the basis signal ψ_u

Idea:

1. *An approximated signal $\hat{f}(x)$ reconstructed considering only the most important (in absolute value) coefficients F_u will preserve most of the energy of the original signal $f(x)$*

Choosing coefficients. Partial reconstruction

As we have seen:

- according to Parseval, an important coefficient $F_u \Rightarrow$ a great amount of the energy of the function (signal) $f(x)$ is contributed by the basis function (signal) ψ_u
- an important coefficient $F_u \Rightarrow$ the signal $f(x)$ is similar to the basis signal ψ_u

Idea:

1. *An approximated signal $\hat{f}(x)$ reconstructed considering only the most important (in absolute value) coefficients F_u will preserve most of the energy of the original signal $f(x)$*
2. *The approximated signal $\hat{f}(x)$ reconstructed as described here will be “similar” to the original signal $f(x)$*

1. Energy conservation

- $f(x), x = 1 \dots N$: a signal of finite energy and $F_u, u = 1 \dots N$ his N correspondent coefficients
  - the energy of the signal

$$E = ||f(x)||^2 = \sum_{x=1}^{N} f(x)^2 = \sum_{u=1}^{N} F_u^2$$

1. Energy conservation

- $f(x), x = 1 \dots N$: a signal of finite energy and $F_u, u = 1 \dots N$ his N correspondent coefficients
  - the energy of the signal

$$E = ||f(x)||^2 = \sum_{x=1}^{N} f(x)^2 = \sum_{u=1}^{N} F_u^2$$

- $\hat{f}(x)$: a partial reconstruction of $f(x)$, using P coefficients
  - the energy of the approximation

$$\hat{E} = ||\hat{f}(x)||^2 = \sum_{x=1}^{N} \hat{f}(x)^2 = \sum_{u=1}^{N} (g(F_u) \cdot F_u)^2 ,$$

with $g(F_u) = 1$ a mask being 1 for the P retained coefficients and 0 elsewhere

1. Energy conservation

- $f(x), x = 1 \dots N$: a signal of finite energy and $F_u, u = 1 \dots N$ his N correspondent coefficients
  - the energy of the signal

$$E = ||f(x)||^2 = \sum_{x=1}^{N} f(x)^2 = \sum_{u=1}^{N} F_u^2$$

- $\hat{f}(x)$: a partial reconstruction of $f(x)$, using P coefficients
  - the energy of the approximation

$$\hat{E} = ||\hat{f}(x)||^2 = \sum_{x=1}^{N} \hat{f}(x)^2 = \sum_{u=1}^{N} \left(g(F_u) \cdot F_u\right)^2 ,$$

with $g(F_u) = 1$ a mask being 1 for the P retained coefficients and 0 elsewhere

Criterion to minimize: $E - \hat{E} = ||f(x)||^2 - ||\hat{f}(x)||^2$

$\Leftrightarrow$ maximizing the energy of the approximation$\hat{E} = ||\hat{f}(x)||^2$

$\Leftrightarrow$ retaining the P greatest coefficients in absolute value

2. Signal similarity

- $\hat{f}(x)$: a partial reconstruction of $f(x)$, using P coefficients
- $r(x) = f(x) - \hat{f}(x) = \sum_{k=1}^{N-P} F_k \psi_k$: the residual error and its decomposition

Criterion to minimize: Mean Square Error (MSE)

$$\begin{aligned} MSE &= ||f(x) - \hat{f}(x)||^2 = ||r(x)||^2 \\ &= \sum_{k=1}^{N-P} F_k^2 \end{aligned}$$

The MSE is minimized when it is constructed from the smallest (in absolute value) coefficients F_k
$\Leftrightarrow$ the P retained coefficients for the approximation are the greatest.

2. Signal similarity

- $\hat{f}(x)$: a partial reconstruction of $f(x)$, using P coefficients
- $r(x) = f(x) - \hat{f}(x) = \sum_{k=1}^{N-P} F_k \psi_k$: the residual error and its decomposition

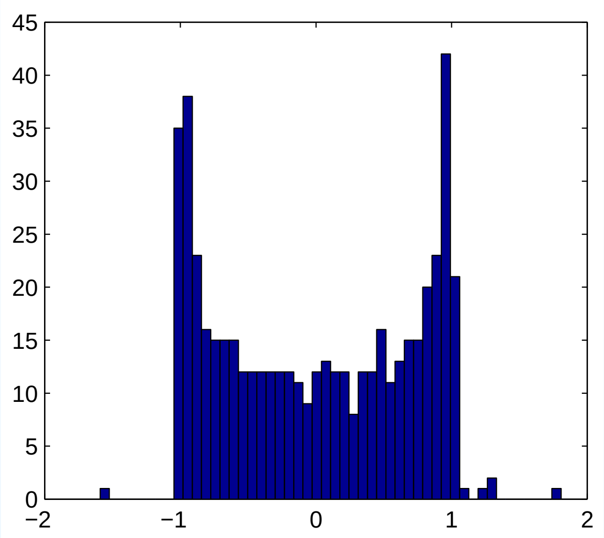
Criterion to minimize: Mean Square Error (MSE)

$$\begin{aligned} MSE &= ||f(x) - \hat{f}(x)||^2 = ||r(x)||^2 \\ &= \sum_{k=1}^{N-P} F_k^2 \end{aligned}$$

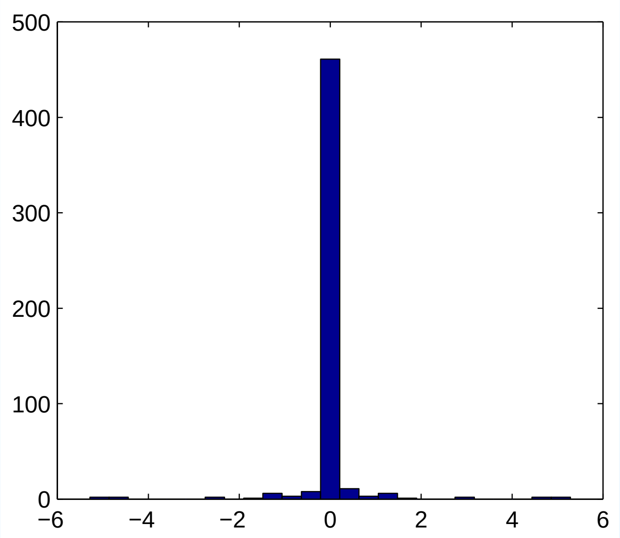
The MSE is minimized when it is constructed from the smallest (in absolute value) coefficients F_k
$\Leftrightarrow$ the P retained coefficients for the approximation are the greatest.

$\Rightarrow$ *The two criteria (energy difference and MSE) are minimized in the same time by choosing the greatest coefficients in absolute value.*

Histograms



Original signal



Transform coefficients

- “Dirac coefficients”
- the energy is distributed upon all coefficients (samples)

- redistribution of the energy upon a small number of great value coefficients
- a lot of coefficients ≈ 0

Choosing coefficients. Conclusion

Principle of lossy compression by approximation

- Decomposing a signal $f(x)$ on an orthonormal basis of dimension N implies a redistribution of the energy of the signal upon the basis functions (signals).
- If the basis is well chosen, the energy concentrates upon few coefficients, which correspond to the basis functions that contribute the most to the reconstruction of a good approximation of the original signal.
- If we want to reconstruct a good approximation (i.e. minimize the MSE and maximize the retained energy), we should retain the greatest coefficients. The approximation will improve with every new term added, but the benefit could be unimportant considering the amount of data.

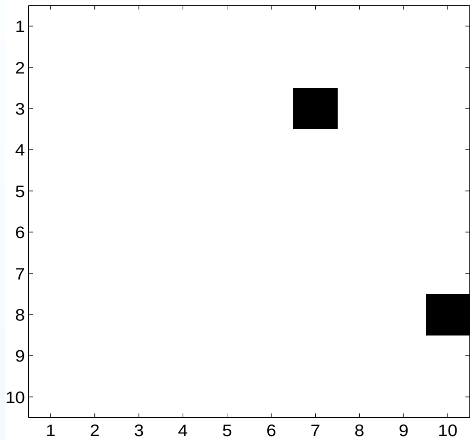
Choosing coefficients. Conclusion

Principle of lossy compression by approximation

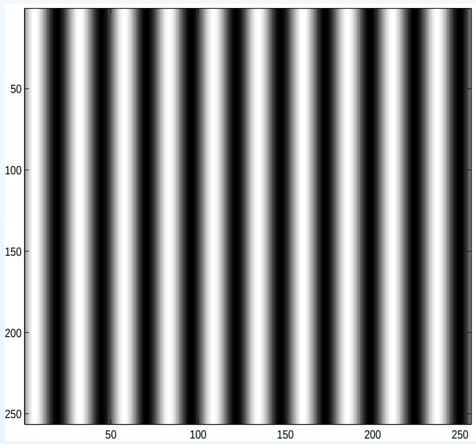
- Decomposing a signal $f(x)$ on an orthonormal basis of dimension N implies a redistribution of the energy of the signal upon the basis functions (signals).
- If the basis is well chosen, the energy concentrates upon few coefficients, which correspond to the basis functions that contribute the most to the reconstruction of a good approximation of the original signal.
- If we want to reconstruct a good approximation (i.e. minimize the MSE and maximize the retained energy), we should retain the greatest coefficients. The approximation will improve with every new term added, but the benefit could be unimportant considering the amount of data.

How do we choose the coefficients? ✓
How do we choose the basis?

Choosing the basis

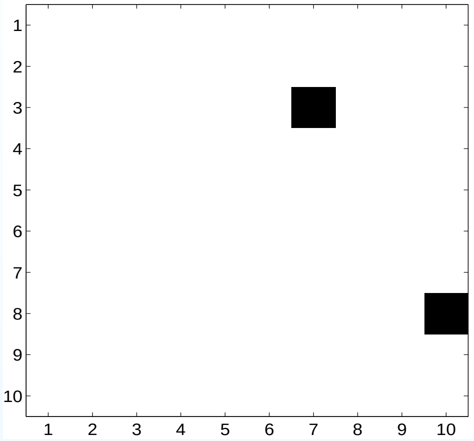


- 10×10 B&W image $f(x, y)$
- two isolated black pixels
  $\sim$ two Dirac pulses
- optimal basis : Dirac (natural image)
  - very low entropy
  - only two non-zero coefficients

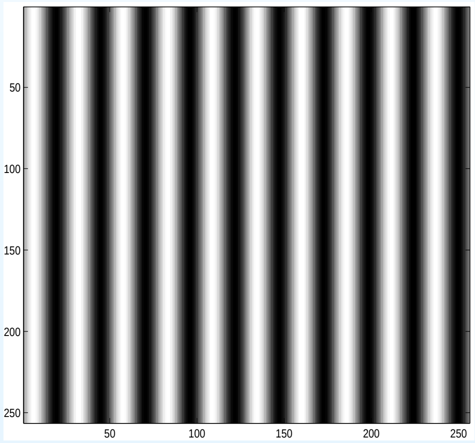


- 256×256 gray level image $f(x, y)$
- sinusoidal pattern by line
- optimal basis : Fourier
  - very low entropy
  - only three (two significant) non-zero coefficients

Choosing the basis



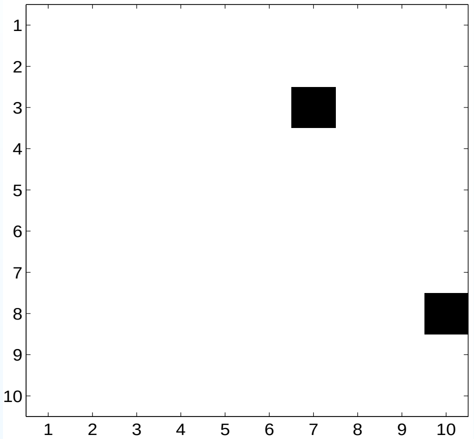
- 10×10 B&W image $f(x, y)$
- two isolated black pixels
  $\sim$ two Dirac pulses
- optimal basis : Dirac (natural image)
  - very low entropy
  - only two non-zero coefficients



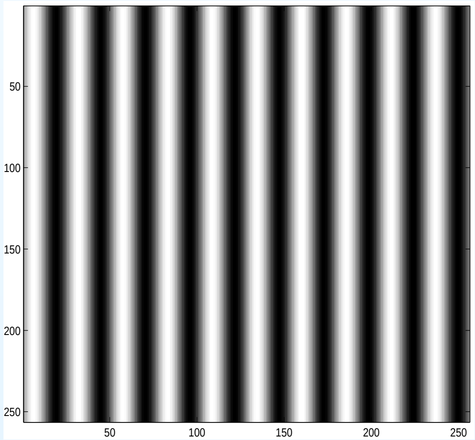
- 256×256 gray level image $f(x, y)$
- sinusoidal pattern by line
- optimal basis : Fourier
  - very low entropy
  - only three (two significant) non-zero coefficients

Is there a unique optimal basis for all images?   No.

Choosing the basis



- 10×10 B&W image $f(x, y)$
- two isolated black pixels
  $\sim$ two Dirac pulses
- optimal basis : Dirac (natural image)
  - very low entropy
  - only two non-zero coefficients



- 256×256 gray level image $f(x, y)$
- sinusoidal pattern by line
- optimal basis : Fourier
  - very low entropy
  - only three (two significant) non-zero coefficients

Is there a unique optimal basis for all images? No.
Are there sub-optimal basis acceptable for all images? Yes.

Transform = Change of basis

Two dimensional transforms formalism:

- $M \times N$ image of interest (function, signal):

$$f(x,y), x = 1 \ldots M, y = 1 \ldots N$$

- family of u, v basis images (functions):

$$\psi_{u,v}(x,y)$$

- coefficients (weights), obtained by the direct transform:

$$F_{u,v} = \langle f(x,y), \psi_{u,v}(x,y) \rangle = \sum_x \sum_y f(x,y) \overline{\psi_{u,v}(x,y)}$$

- inverse transform:

$$f(x,y) = \sum_u \sum_v F_{u,v} \psi_{u,v}(x,y)$$

Separable transforms

Theoretical two dimensional global transforms
$\longrightarrow$ work *simultaneously* on lines and columns (in x and y):

- $\psi_{u,v}(x,y)$ basis functions have 4 parameters:
  - domain definition $x = 1 \dots M, y = 1 \dots N$, as they are 2D images
  - shape (gray levels pattern) definition: u, v (frequency, location, scale, …)

- $F_{u,v}$ coefficients are computed as projections of the whole image upon the $\psi_{u,v}(x,y)$ basis function

$$F_{u,v} = \langle f(x,y), \psi_{u,v}(x,y) \rangle = \sum_x \sum_y f(x,y) \overline{\psi_{u,v}(x,y)}$$

Separable transforms

In practice, most of the transforms are separable:
$\longrightarrow$ work *successively* on lines and columns (in x and y):

- $\psi_{u,v}(x,y) = \psi_u(x)\psi_v(y)$

  Fourier basis:

$$e^{2\pi j(ux/M+vy/N)} = e^{2\pi jux/M} e^{2\pi jvy/N}$$

- $F_{u,v}$ coefficients are computed in two steps:
  1. projections of the image upon $\psi_u(x)$ basis functions
  2. projections of the result upon $\psi_v(y)$ basis functions

  Fourier coefficients:

$$F_{u,v} = \sum_x e^{-2\pi jux/M} \sum_y f(x,y) e^{-2\pi jvy/N}$$

Changing the basis

A given image $f(x, y)$ can be represented:

Natural basis: Dirac

- sum of weighted ($F_{i,j} =$ gray level) Dirac pulses ($\psi_{i,j}(x, y) = \delta_{i,j}(x, y)$)
- the image is described by the gray level values of each pixel $\Leftrightarrow$ by the coefficients (weights,amplitudes) of each Dirac pulse

Fourier basis

- sum of weighted ($F_{i,j}$) complex exponentials ($\psi_{i,j}(x, y)$)
- the image is described by its spectrum $\Leftrightarrow$ by the coefficients (weights, amplitudes) of each exponential

How do we change the basis?

- Fourier transform
- Practically: matrix multiplication

Changing the basis

The image $f(x, y)$ is a $M \times N$ matrix $\mathbf{f}$

Pre-multiplying by a $M \times M$ matrix $\mathbf{T_1}$ gives:

$$\mathbf{F_1} = \mathbf{T_1}\mathbf{f},$$

where the element (i, j) of $\mathbf{F_1}$ is:

$$F_1(i, j) = \sum_k T_1(i, k) f(k, j) = \langle T_1(i, :), f(:, j) \rangle$$

$\Leftrightarrow$ the scalar product between the line i of the transform matrix $\mathbf{T_1}$ and the column j of the image.

- *If the lines of $\mathbf{T_1}$ are basis functions, the elements of $\mathbf{F_1}$ are the coefficients of the transform of the columns of $\mathbf{f}$.*

Changing the basis

The image $f(x, y)$ is a $M \times N$ matrix $\mathbf{f}$

Pre-multiplying by a $M \times M$ matrix $\mathbf{T_1}$ gives:

$$\mathbf{F_1} = \mathbf{T_1}\mathbf{f},$$

where the element (i, j) of $\mathbf{F_1}$ is:

$$F_1(i, j) = \sum_k T_1(i, k) f(k, j) = \langle T_1(i, :), f(:, j) \rangle$$

$\Leftrightarrow$ the scalar product between the line i of the transform matrix $\mathbf{T_1}$ and the column j of the image.

- *If the lines of $\mathbf{T_1}$ are basis functions, the elements of $\mathbf{F_1}$ are the coefficients of the transform of the columns of $\mathbf{f}$.*
- *Post-multiplying $\mathbf{f}$ by a $N \times N$ matrix $\mathbf{T_2}$ having basis functions as columns, we obtain the transform coefficients of the image lines.*

Changing the basis

The image $f(x, y)$ is a $M \times N$ matrix $\mathbf{f}$

Pre-multiplying by a $M \times M$ matrix $\mathbf{T_1}$ gives:

$$\mathbf{F_1} = \mathbf{T_1}\mathbf{f},$$

where the element (i, j) of $\mathbf{F_1}$ is:

$$F_1(i, j) = \sum_k T_1(i, k) f(k, j) = \langle T_1(i, :), f(:, j) \rangle$$

$\Leftrightarrow$ the scalar product between the line i of the transform matrix $\mathbf{T_1}$ and the column j of the image.

- *If the lines of $\mathbf{T_1}$ are basis functions, the elements of $\mathbf{F_1}$ are the coefficients of the transform of the columns of $\mathbf{f}$.*
- *Post-multiplying $\mathbf{f}$ by a $N \times N$ matrix $\mathbf{T_2}$ having basis functions as columns, we obtain the transform coefficients of the image lines.*
- *Separable transforms are performed by two successive matrix multiplications of the original image.*

Changing the basis, practical approach

1. In practical applications:

- the full-size image is divided in square sub-images
- the remaining rectangular sub-images are padded with zeros to become square

$\Rightarrow$ Transform matrices $\mathbf{T_1}, \mathbf{T_2}$ are square and have the size of the image (sub-image) to transform $N \times N$

2. Transform matrices $\mathbf{T_1}, \mathbf{T_2}$ contain the same basis functions on their lines (respectively columns):

- $T_1 = T_2^T = T$ transform matrix
- basis functions are orthonormal, so $T^T = T^{-1}$

Changing the basis, practical approach

1. In practical applications:
   - the full-size image is divided in square sub-images
   - the remaining rectangular sub-images are padded with zeros to become square

   $\Rightarrow$ Transform matrices $\mathbf{T_1}, \mathbf{T_2}$ are square and have the size of the image (sub-image) to transform $N \times N$

2. Transform matrices $\mathbf{T_1}, \mathbf{T_2}$ contain the same basis functions on their lines (respectively columns):
   - $T_1 = T_2^T = T$ transform matrix
   - basis functions are orthonormal, so $T^T = T^{-1}$

Choosing the lines of the transform matrix T means choosing the transform.

Transform of $\mathbf{f}$ image: $\mathbf{F} = \mathbf{T}\mathbf{f}\mathbf{T}^{\mathbf{T}}$

Fourier Transform

Fourier transform:

- basis functions: complex exponentials

$$\psi_u(x) = e^{j2\pi ux/N}, \quad u = -N/2..N/2-1$$

- for a 8×8 matrix (sub-image) $\mathbf{f}$:

$$\mathbf{T} = \begin{pmatrix} e^{-j2\pi(-4)\cdot 0/8} & e^{-j2\pi(-4)\cdot 1/8} & \dots & e^{-j2\pi 0\cdot 7/8} \\ e^{-j2\pi(-3)\cdot 0/8} & e^{-j2\pi(-3)\cdot 1/8} & \dots & e^{-j2\pi(-3)\cdot 7/8} \\ . & . & \dots & . \\ e^{-j2\pi 3\cdot 0/8} & e^{-j2\pi 3\cdot 1/8} & \dots & e^{-j2\pi 3\cdot 7/8} \end{pmatrix}$$

- $\mathbf{T}$ matrix $\rightarrow$ complex conjugate, because basis functions are complex!

Discrete Cosine Transform

Discrete cosine transform:

- *real* basis functions

$$\psi_u(x) = c_u \cos\left(\frac{(2x+1)u\pi}{2N}\right)$$

with

$$c_u = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1 \dots N-1 \end{cases}$$

- for a 8×8 matrix (sub-image) $\mathbf{f}$:

$$\mathbf{T} = \begin{pmatrix} \frac{1}{\sqrt{8}} \cos\frac{(2\cdot 0+1)0\pi}{16} & \frac{1}{\sqrt{8}} \cos\frac{(2\cdot 1+1)0\pi}{16} & \dots & \frac{1}{\sqrt{8}} \cos\frac{(2\cdot 7+1)0\pi}{16} \\ \frac{1}{2} \cos\frac{(2\cdot 0+1)1\pi}{16} & \frac{1}{2} \cos\frac{(2\cdot 1+1)1\pi}{16} & \dots & \frac{1}{2} \cos\frac{(2\cdot 7+1)1\pi}{16} \\ . & . & \dots & . \\ \frac{1}{2} \cos\frac{(2\cdot 0+1)7\pi}{16} & \frac{1}{2} \cos\frac{(2\cdot 1+1)7\pi}{16} & \dots & \frac{1}{2} \cos\frac{(2\cdot 7+1)7\pi}{16} \end{pmatrix}$$

Discrete Cosine Transform

Introduction

Information Theory

Compression basics

Entropic coding

Inter-pixel coding

Quantizing and thresholding

Color space transforms

Image transforms

- Geometric analogy
- Functional analysis
- FT
- DCT
- DCT approximations
- DCT zonal coding
- DCT threshold coding
- JPEG
- WT

$$\mathbf{T} = \begin{pmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \cdots & \frac{1}{\sqrt{8}} \\ \frac{1}{2}\cos\frac{\pi}{16} & \frac{1}{2}\cos\frac{3\pi}{16} & \cdots & \frac{1}{2}\cos\frac{15\pi}{16} \\ \cdot & \cdot & \cdots & \cdot \\ \frac{1}{2}\cos\frac{7\pi}{16} & \frac{1}{2}\cos\frac{21\pi}{16} & \cdots & \frac{1}{2}\cos\frac{105\pi}{16} \end{pmatrix}$$

$$\mathbf{f} = \begin{pmatrix} 140 & 144 & 147 & 140 & 140 & 155 & 179 & 175 \\ 144 & 152 & 140 & 147 & 140 & 148 & 167 & 179 \\ 152 & 155 & 136 & 167 & 163 & 162 & 152 & 172 \\ 168 & 145 & 156 & 160 & 152 & 155 & 136 & 160 \\ 162 & 148 & 156 & 148 & 140 & 138 & 147 & 162 \\ 147 & 167 & 140 & 155 & 155 & 140 & 136 & 162 \\ 136 & 156 & 123 & 167 & 162 & 144 & 140 & 147 \\ 148 & 155 & 136 & 155 & 152 & 147 & 147 & 136 \end{pmatrix}$$

$$\mathbf{F} = \mathbf{T}\mathbf{A}\mathbf{T}^{\mathbf{T}} = \begin{pmatrix} 1210 & -18 & 15 & -9 & 23 & -9 & -14 & -19 \\ 20 & -34 & 26 & -9 & -11 & 11 & 14 & 7 \\ -11 & -23 & -2 & 6 & -18 & 3 & -21 & 0 \\ -8 & -5 & 14 & -14 & -8 & -3 & -3 & 8 \\ -3 & 9 & 8 & 2 & -11 & 18 & 19 & 15 \\ 4 & -2 & -18 & 8 & 9 & -4 & 0 & -7 \\ 9 & 1 & -3 & 3 & -1 & -7 & -1 & -2 \\ 0 & -8 & -3 & 2 & 1 & 4 & -6 & 0 \end{pmatrix}$$

Inside the matrix F

- What represent $F(u, v)$ elements?

Inside the matrix $\mathbf{F}$

- What represent $F(u, v)$ elements?
- Coefficients of the DC decomposition
  $\Leftrightarrow$ projection coefficients on the DC basis
  $\Leftrightarrow$ weights of the basis functions in image decomposition

Inside the matrix F

- What represent $F(u, v)$ elements?
- Coefficients of the DC decomposition
  $\Leftrightarrow$ projection coefficients on the DC basis
  $\Leftrightarrow$ weights of the basis functions in image decomposition
- $F(0, 0)$ (first line, first column)?

Inside the matrix F

- What represent $F(u,v)$ elements?
- Coefficients of the DC decomposition
  $\Leftrightarrow$ projection coefficients on the DC basis
  $\Leftrightarrow$ weights of the basis functions in image decomposition
- $F(0,0)$ (first line, first column)?
- proportional to the mean value of $\mathbf{f}$

$$
\begin{aligned}
F(0,0) &= \frac{1}{\sqrt{8}} \sum_y \frac{1}{\sqrt{8}} \sum_x f(x,y) \\
&= \frac{1}{8} \sum_{x,y} f(x,y) = 8 \cdot \text{mean}(\mathbf{f})
\end{aligned}
$$

DCT basis

Consider $F(0,0) = 8 \cdot \text{mean}(\mathbf{f})$ and *all* other $F(u,v) = 0$

$$\overline{\mathbf{F}} = \begin{pmatrix} F(0,0) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ . & . & . & . \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

and reconstruct a “simplified” image

$$\overline{\mathbf{f}} = \mathbf{T}^{\mathbf{T}}\overline{\mathbf{F}}\mathbf{T} = F(0,0) \begin{pmatrix} 1/8 & 1/8 & \dots & 1/8 \\ 1/8 & 1/8 & \dots & 1/8 \\ . & . & . & . \\ 1/8 & 1/8 & \dots & 1/8 \end{pmatrix}$$

DCT basis

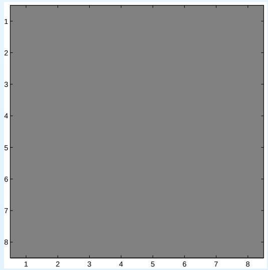
Consider $F(0,0) = 8 \cdot \text{mean}(\mathbf{f})$ and *all* other $F(u,v) = 0$

$$\overline{\mathbf{F}} = \begin{pmatrix} F(0,0) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ . & . & . & . \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

and reconstruct a “simplified” image

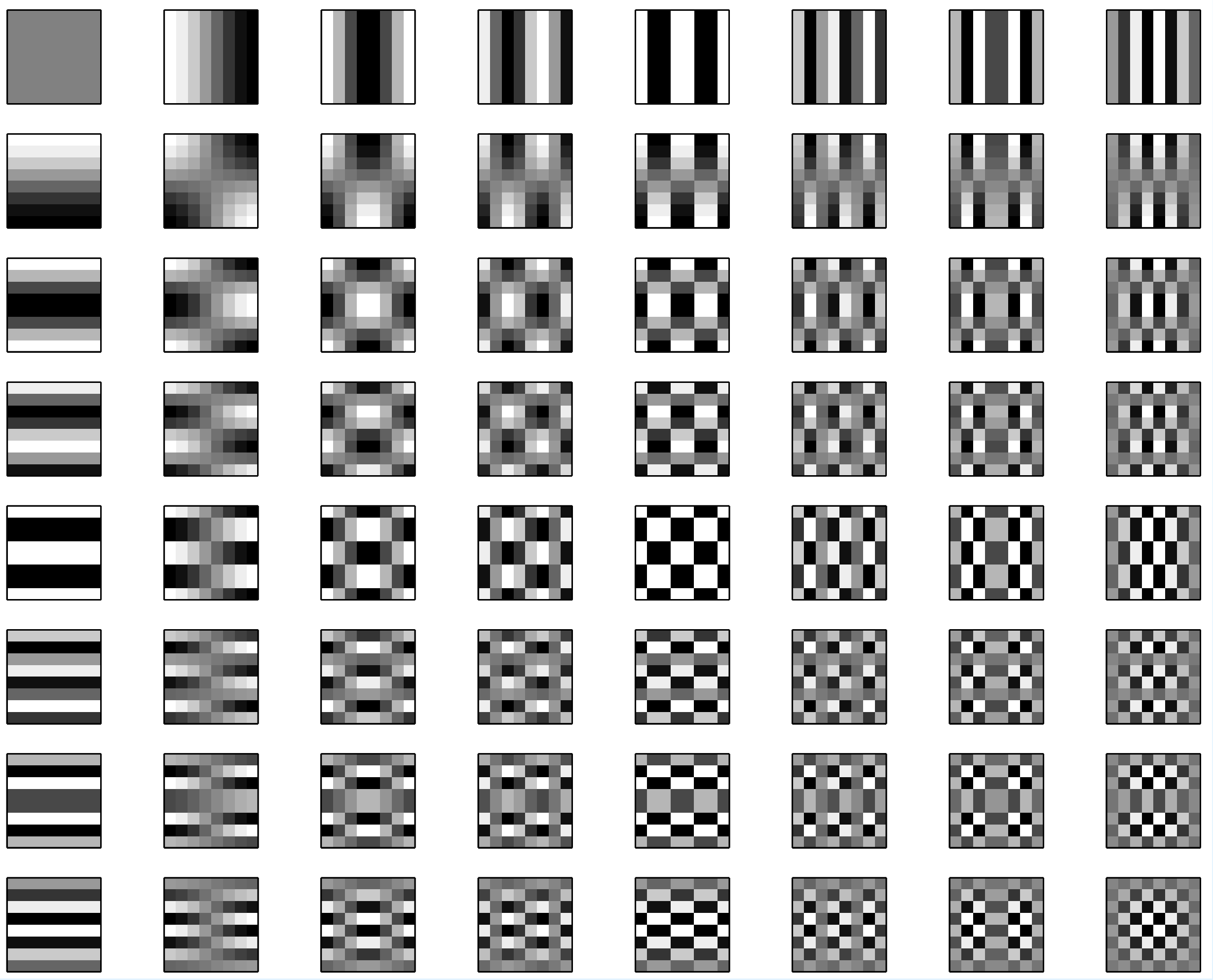
$$\overline{\mathbf{f}} = \mathbf{T}^{\mathbf{T}}\overline{\mathbf{F}}\mathbf{T} = F(0,0) \begin{pmatrix} 1/8 & 1/8 & \dots & 1/8 \\ 1/8 & 1/8 & \dots & 1/8 \\ . & . & . & . \\ 1/8 & 1/8 & \dots & 1/8 \end{pmatrix}$$

First basis function (image) $\longrightarrow$

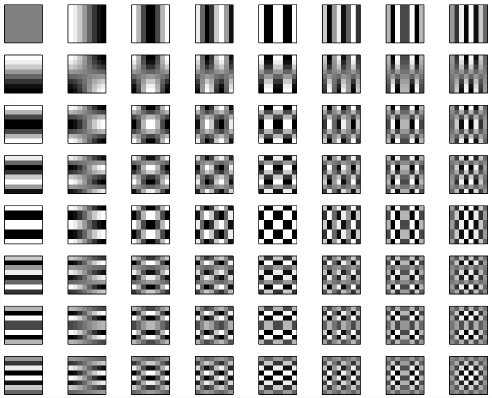


DCT basis

In general, considering only one $F(u, v)$ non null, one can obtain by inverse DCT the corresponding basis image.



DCT basis insight

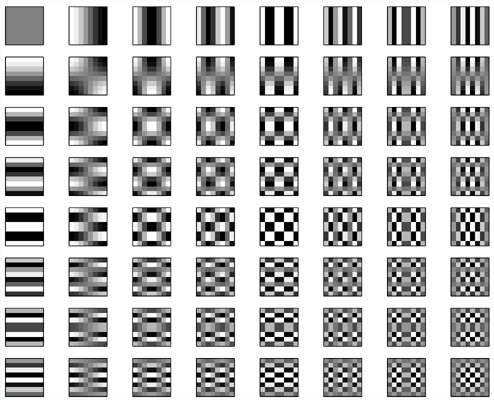


- $F(0,0)$: DC coefficient (mean value)
- $F(u,v)$: AC coefficients (variations)
  - line 1: vertical variations, low to high frequency
  - column 1: horizontal variations
  - others : angular variations (diagonal)

Human eye:

- sensible to luminance (mean gray level) (DC coefficient)
- more sensible to horizontal and vertical variations than to angular variations
- more sensible to low frequencies than to high frequencies

DCT basis insight



- $F(0,0)$: DC coefficient (mean value)
- $F(u,v)$: AC coefficients (variations)
  - line 1: vertical variations, low to high frequency
  - column 1: horizontal variations
  - others : angular variations (diagonal)

Human eye:

- sensible to luminance (mean gray level) (DC coefficient)
- more sensible to horizontal and vertical variations than to angular variations
- more sensible to low frequencies than to high frequencies

Keeping only the upper-left corner of the $\mathbf{F}$ *matrix (i.e reconstructing using only continuous and low frequencies mainly vertical and horizontal basis functions) leads to good approximation and compression*

DCT improvement

$F(0,0)$: high value (8 times the mean gray level)

$\rightarrow$ common solution: subtract 128 from the original image

$$\begin{pmatrix} 140 & 144 & 147 & 140 & 140 & 155 & 179 & 175 \\ 144 & 152 & 140 & 147 & 140 & 148 & 167 & 179 \\ 152 & 155 & 136 & 167 & 163 & 162 & 152 & 172 \\ 168 & 145 & 156 & 160 & 152 & 155 & 136 & 160 \\ 162 & 148 & 156 & 148 & 140 & 138 & 147 & 162 \\ 147 & 167 & 140 & 155 & 155 & 140 & 136 & 162 \\ 136 & 156 & 123 & 167 & 162 & 144 & 140 & 147 \\ 148 & 155 & 136 & 155 & 152 & 147 & 147 & 136 \end{pmatrix} \quad \begin{pmatrix} 12 & 16 & 19 & 12 & 12 & 27 & 51 & 47 \\ 16 & 24 & 12 & 19 & 12 & 20 & 39 & 51 \\ 24 & 27 & 8 & 39 & 35 & 34 & 24 & 44 \\ 40 & 17 & 28 & 32 & 24 & 27 & 8 & 32 \\ 34 & 20 & 28 & 20 & 12 & 10 & 19 & 34 \\ 19 & 39 & 12 & 27 & 27 & 12 & 8 & 34 \\ 8 & 28 & -5 & 39 & 34 & 16 & 12 & 19 \\ 20 & 27 & 8 & 27 & 24 & 19 & 19 & 8 \end{pmatrix}$$

$$\begin{pmatrix} 1210 & -18 & 15 & -9 & 23 & -9 & -14 & -19 \\ 20 & -34 & 26 & -9 & -11 & 11 & 14 & 7 \\ -11 & -23 & -2 & 6 & -18 & 3 & -21 & 0 \\ -8 & -5 & 14 & -14 & -8 & -3 & -3 & 8 \\ -3 & 9 & 8 & 2 & -11 & 18 & 19 & 15 \\ 4 & -2 & -18 & 8 & 9 & -4 & 0 & -7 \\ 9 & 1 & -3 & 3 & -1 & -7 & -1 & -2 \\ 0 & -8 & -3 & 2 & 1 & 4 & -6 & 0 \end{pmatrix} \quad \begin{pmatrix} 186 & -18 & 15 & -9 & 23 & -9 & -14 \\ 20 & -34 & 26 & -9 & -11 & 11 & 14 \\ -11 & -23 & -2 & 6 & -18 & 3 & -21 \\ -8 & -5 & 14 & -14 & -8 & -3 & -3 \\ -3 & 9 & 8 & 2 & -11 & 18 & 19 \\ 4 & -2 & -18 & 8 & 9 & -4 & 0 \\ 9 & 1 & -3 & 3 & -1 & -7 & -1 \\ 0 & -8 & -3 & 2 & 1 & 4 & -6 \end{pmatrix}$$

DCT approximations

- Previous approach ("keep or kill") $\longrightarrow$ Thresholding $\Leftrightarrow$ multiplying by a $\{0, 1\}$ mask:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

DCT approximations

- Previous approach ("keep or kill") $\longrightarrow$ Thresholding $\Leftrightarrow$ multiplying by a $\{0,1\}$ mask:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 0 & \ldots & 0 \\ 1 & 1 & 0 & 0 & \ldots & 0 \\ 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

- reconstructed image $\hat{\mathbf{f}} = \mathbf{T}^{\mathbf{T}}\hat{\mathbf{F}}\mathbf{T} = \mathbf{T}^{\mathbf{T}}(\mathbf{M} \odot \mathbf{F})\mathbf{T}$ (with $\odot =$ element by element multiplication

DCT approximations

- Previous approach (“keep or kill”) $\longrightarrow$ Thresholding $\Leftrightarrow$ multiplying by a $\{0, 1\}$ mask:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

- reconstructed image $\hat{\mathbf{f}} = \mathbf{T}^{\mathbf{T}}\hat{\mathbf{F}}\mathbf{T} = \mathbf{T}^{\mathbf{T}}(\mathbf{M} \odot \mathbf{F})\mathbf{T}$ (with $\odot =$ element by element multiplication
- Problem: optimal threshold ? $\Leftrightarrow$ mask matrix $\mathbf{M}$?

DCT approximations

- Previous approach (“keep or kill”) $\rightarrow$ Thresholding $\Leftrightarrow$ multiplying by a $\{0, 1\}$ mask:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

- reconstructed image $\hat{\mathbf{f}} = \mathbf{T}^{\mathbf{T}}\hat{\mathbf{F}}\mathbf{T} = \mathbf{T}^{\mathbf{T}}(\mathbf{M} \odot \mathbf{F})\mathbf{T}$ (with $\odot =$ element by element multiplication
- Problem: optimal threshold ? $\Leftrightarrow$ mask matrix $\mathbf{M}$?

Two approaches:

- maximum energy (zonal coding) $\rightarrow$ global
- maximum magnitude (threshold coding) $\rightarrow$ adaptive

DCT zonal coding

Statistic global approach: each DCT coefficient $F(u, v)$ is seen as a particular measure of a random process

Algorithm:

1. Split the image in $n \times n$ sub-images of size $N \times N$ pixels
2. Compute the DCT coefficients $F(u_i, v_i)$ for each sub-image i
3. Compute the energy $E_{u,v} = \sum_i (F(u_i, v_i))^2$
4. Choose the mask according to a strategy:
   - keep a *fixed number* of coefficients of maximum energy (fixed ratio compression)
   - keep the maximum energy coefficients that preserve a *fixed proportion* of the total energy (variable ratio compression)

DCT zonal coding

Statistic global approach: each DCT coefficient $F(u,v)$ is seen as a particular measure of a random process

Algorithm:

1. Split the image in $n \times n$ sub-images of size $N \times N$ pixels
2. Compute the DCT coefficients $F(u_i, v_i)$ for each sub-image i
3. Compute the energy $E_{u,v} = \sum_i (F(u_i, v_i))^2$
4. Choose the mask according to a strategy:
   - keep a *fixed number* of coefficients of maximum energy (fixed ratio compression)
   - keep the maximum energy coefficients that preserve a *fixed proportion* of the total energy (variable ratio compression)

- Previous mask $\mathbf{M} \rightarrow$ keep 6 coefficients out of 64
- Compression ratio: $C_R = 10,66 : 1$ (+ the coordinates $C_R = 6,1 : 1$)

DCT zonal coding

Improve the compression ratio by changing the “keep or kill” approach:

- quantize the remaining coefficients according to their importance
- bit allocation mask:

$$\mathbf{B} : \begin{pmatrix} 8 & 7 & 6 & 0 & \dots & 0 \\ 7 & 5 & 0 & 0 & \dots & 0 \\ 6 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix} \qquad \mathbf{M_B} : \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{4} & 0 & \dots & 0 \\ \frac{1}{2} & \frac{1}{8} & 0 & 0 & \dots & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

DCT zonal coding

Improve the compression ratio by changing the “keep or kill” approach:

- quantize the remaining coefficients according to their importance
- bit allocation mask:

$$\mathbf{B}: \begin{pmatrix} 8 & 7 & 6 & 0 & \dots & 0 \\ 7 & 5 & 0 & 0 & \dots & 0 \\ 6 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix} \qquad \mathbf{M_B}: \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{4} & 0 & \dots & 0 \\ \frac{1}{2} & \frac{1}{8} & 0 & 0 & \dots & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

- Original sub-image: $\mathbf{f} \rightarrow 64$ pixels $\times 8$ bits = 512 bits
- Compressed & transformed sub-image:
$\hat{\mathbf{F}} = \text{round}(\mathbf{F} \odot \mathbf{M_B}) \rightarrow 8+7+7+6+5+6 = 39$ bits
- Compression ratio: $C_R = 13,13:1$ (+ the coordinates $C_R = 6,8:1$)

DCT threshold coding

Adaptive approach

Algorithm:

1. Split the original image in $n \times n$ sub-images of size $N \times N$ pixels
2. Compute the DCT coefficients $F(u_i, v_i)$ for each sub-image i
3. For each sub-image, choose the mask according to a strategy:
   - keep the coefficients $F(u_i, v_i)$ greater than a global threshold T (variable ratio compression)
   - keep a *fixed number* of coefficients of maximum magnitude $\Leftrightarrow$ image adapted threshold T_i (fixed ratio compression)
   - keep the coefficients if they exceed a local threshold $T_i(u, v)$ (variable ratio compression)

DCT threshold coding

Improve the compression ratio by changing the “keep or kill” approach:

- merge the thresholding and the quantization procedures:
- typical bit allocation mask:

$$\mathbf{M}_{\mathbf{JPEG}} = \begin{pmatrix} \frac{1}{16} & \frac{1}{11} & \frac{1}{10} & \frac{1}{16} & \frac{1}{24} & \frac{1}{40} & \frac{1}{51} & \frac{1}{61} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{14} & \frac{1}{19} & \frac{1}{26} & \frac{1}{58} & \frac{1}{60} & \frac{1}{55} \\ \frac{1}{14} & \frac{1}{13} & \frac{1}{16} & \frac{1}{24} & \frac{1}{40} & \frac{1}{57} & \frac{1}{69} & \frac{1}{56} \\ \frac{1}{14} & \frac{1}{17} & \frac{1}{22} & \frac{1}{29} & \frac{1}{51} & \frac{1}{87} & \frac{1}{80} & \frac{1}{62} \\ \frac{1}{18} & \frac{1}{22} & \frac{1}{37} & \frac{1}{56} & \frac{1}{68} & \frac{1}{109} & \frac{1}{103} & \frac{1}{77} \\ \frac{1}{24} & \frac{1}{35} & \frac{1}{55} & \frac{1}{64} & \frac{1}{81} & \frac{1}{104} & \frac{1}{113} & \frac{1}{92} \\ \frac{1}{49} & \frac{1}{64} & \frac{1}{78} & \frac{1}{87} & \frac{1}{103} & \frac{1}{121} & \frac{1}{120} & \frac{1}{101} \\ \frac{1}{72} & \frac{1}{92} & \frac{1}{95} & \frac{1}{98} & \frac{1}{112} & \frac{1}{100} & \frac{1}{103} & \frac{1}{99} \end{pmatrix}$$

DCT threshold coding

Improve the compression ratio by changing the “keep or kill” approach:

- merge the thresholding and the quantization procedures:
- typical bit allocation mask:

$$\mathbf{M}_{\mathbf{JPEG}} = \begin{pmatrix} \frac{1}{16} & \frac{1}{11} & \frac{1}{10} & \frac{1}{16} & \frac{1}{24} & \frac{1}{40} & \frac{1}{51} & \frac{1}{61} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{14} & \frac{1}{19} & \frac{1}{26} & \frac{1}{58} & \frac{1}{60} & \frac{1}{55} \\ \frac{1}{14} & \frac{1}{13} & \frac{1}{16} & \frac{1}{24} & \frac{1}{40} & \frac{1}{57} & \frac{1}{69} & \frac{1}{56} \\ \frac{1}{14} & \frac{1}{17} & \frac{1}{22} & \frac{1}{29} & \frac{1}{51} & \frac{1}{87} & \frac{1}{80} & \frac{1}{62} \\ \frac{1}{18} & \frac{1}{22} & \frac{1}{37} & \frac{1}{56} & \frac{1}{68} & \frac{1}{109} & \frac{1}{103} & \frac{1}{77} \\ \frac{1}{24} & \frac{1}{35} & \frac{1}{55} & \frac{1}{64} & \frac{1}{81} & \frac{1}{104} & \frac{1}{113} & \frac{1}{92} \\ \frac{1}{49} & \frac{1}{64} & \frac{1}{78} & \frac{1}{87} & \frac{1}{103} & \frac{1}{121} & \frac{1}{120} & \frac{1}{101} \\ \frac{1}{72} & \frac{1}{92} & \frac{1}{95} & \frac{1}{98} & \frac{1}{112} & \frac{1}{100} & \frac{1}{103} & \frac{1}{99} \end{pmatrix}$$

$\Leftrightarrow$ third thresholding strategy: individual threshold depending on the position (u, v) !

$$\hat{\mathbf{F}} = \text{round}(\mathbf{F} \odot \mathbf{M}_{\mathbf{JPEG}}) \Leftrightarrow \hat{\mathbf{f}} = \mathbf{T}^{\mathbf{T}}\text{round}(\mathbf{F} \odot \mathbf{M}_{\mathbf{JPEG}})\mathbf{T}$$

DCT compression method

Coding:

1. Subtract 128 from the gray level image
2. Split the results in 8×8 sub-images $\mathbf{f}$
3. Transform each sub-image by DCT to obtain $\mathbf{F}$
4. Quantize $\mathbf{F}$ to obtain $\hat{\mathbf{F}} = \text{round}(\mathbf{F} \odot \mathbf{M}_{\mathbf{JPEG}})$

Decoding:

1. Reconstruct $\tilde{\mathbf{F}} = \hat{\mathbf{F}} \oslash \mathbf{M}_{\mathbf{JPEG}}$
2. Reconstruct approximations of each sub-image by inverse DCT $\hat{\mathbf{f}} = \text{round}(\mathbf{T}^{\mathbf{T}} \tilde{\mathbf{F}} \mathbf{T})$
3. Reassemble the complete image
4. Add 128

DCT coding example

$$
\begin{pmatrix}
140 & 144 & 147 & 140 & 140 & 155 & 179 & 175 \\
144 & 152 & 140 & 147 & 140 & 148 & 167 & 179 \\
152 & 155 & 136 & 167 & 163 & 162 & 152 & 172 \\
168 & 145 & 156 & 160 & 152 & 155 & 136 & 160 \\
162 & 148 & 156 & 148 & 140 & 138 & 147 & 162 \\
147 & 167 & 140 & 155 & 155 & 140 & 136 & 162 \\
136 & 156 & 123 & 167 & 162 & 144 & 140 & 147 \\
148 & 155 & 136 & 155 & 152 & 147 & 147 & 136
\end{pmatrix}
\quad
\begin{pmatrix}
12 & 16 & 19 & 12 & 12 & 27 & 51 & 47 \\
16 & 24 & 12 & 19 & 12 & 20 & 39 & 51 \\
24 & 27 & 8 & 39 & 35 & 34 & 24 & 44 \\
40 & 17 & 28 & 32 & 24 & 27 & 8 & 32 \\
34 & 20 & 28 & 20 & 12 & 10 & 19 & 34 \\
19 & 39 & 12 & 27 & 27 & 12 & 8 & 34 \\
8 & 28 & -5 & 39 & 34 & 16 & 12 & 19 \\
20 & 27 & 8 & 27 & 24 & 19 & 19 & 8
\end{pmatrix}
$$

$$
\begin{pmatrix}
186 & -18 & 15 & -9 & 23 & -9 & -14 & -19 \\
20 & -34 & 26 & -9 & -11 & 11 & 14 & 7 \\
-11 & -23 & -2 & 6 & -18 & 3 & -21 & 0 \\
-8 & -5 & 14 & -14 & -8 & -3 & -3 & 8 \\
-3 & 9 & 8 & 2 & -11 & 18 & 19 & 15 \\
4 & -2 & -18 & 8 & 9 & -4 & 0 & -7 \\
9 & 1 & -3 & 3 & -1 & -7 & -1 & -2 \\
0 & -8 & -3 & 2 & 1 & 4 & -6 & 0
\end{pmatrix}
\quad
\begin{pmatrix}
12 & -2 & 1 & -1 & 1 & 0 & 0 & 0 \\
2 & -3 & 2 & 0 & 0 & 0 & 0 & 0 \\
-1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

Compression ratio: $C_R = \frac{64}{12} = 5,33$

(+ position coding $\rightarrow C_R = \frac{64 \cdot 8}{12 \cdot 8 + 12 \cdot 6} = 3,04$)

DCT decoding example

$$\begin{pmatrix} 12 & -2 & 1 & -1 & 1 & 0 & 0 & 0 \\ 2 & -3 & 2 & 0 & 0 & 0 & 0 & 0 \\ -1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

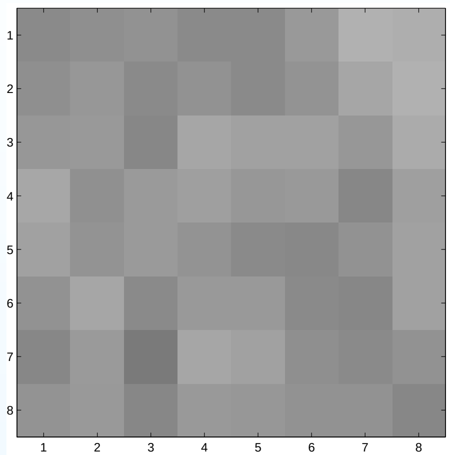
$$\begin{pmatrix} 192 & -22 & 10 & -16 & 24 & 0 & 0 & 0 \\ 24 & -36 & 28 & 0 & 0 & 0 & 0 & 0 \\ -14 & -26 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 18 & 11 & 8 & 13 & 17 & 23 & 41 & 60 \\ 20 & 16 & 17 & 23 & 25 & 26 & 38 & 52 \\ 24 & 22 & 26 & 33 & 32 & 27 & 32 & 42 \\ 30 & 26 & 28 & 33 & 30 & 22 & 25 & 34 \\ 32 & 26 & 25 & 26 & 22 & 15 & 19 & 29 \\ 27 & 22 & 21 & 22 & 18 & 11 & 15 & 25 \\ 18 & 16 & 21 & 27 & 23 & 14 & 13 & 21 \\ 10 & 13 & 22 & 33 & 30 & 18 & 13 & 17 \end{pmatrix}$$

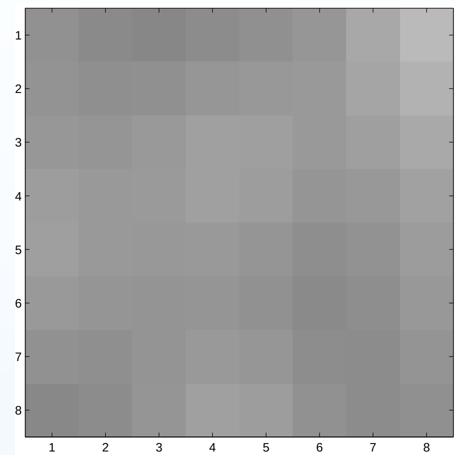
$$\begin{pmatrix} 146 & 139 & 136 & 141 & 145 & 151 & 169 & 188 \\ 148 & 144 & 145 & 151 & 153 & 154 & 166 & 180 \\ 152 & 150 & 154 & 161 & 160 & 155 & 160 & 170 \\ 158 & 154 & 156 & 161 & 158 & 150 & 153 & 162 \\ 160 & 154 & 153 & 154 & 150 & 143 & 147 & 157 \\ 155 & 150 & 149 & 150 & 146 & 139 & 143 & 153 \\ 146 & 144 & 149 & 155 & 151 & 142 & 141 & 149 \\ 138 & 141 & 150 & 161 & 158 & 146 & 141 & 145 \end{pmatrix}$$

Mean square error : $MSE = 72,65$
Compression ratio: $C_R = 3,04$

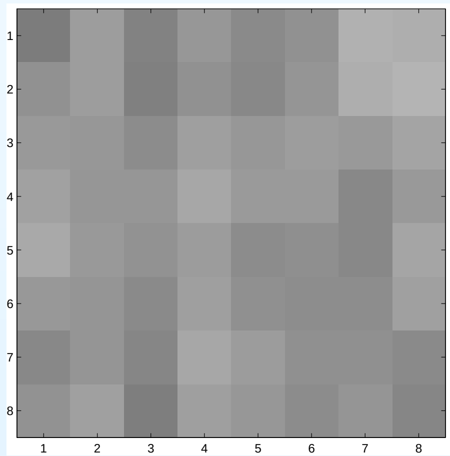
Different quantizations M_{JPEG}



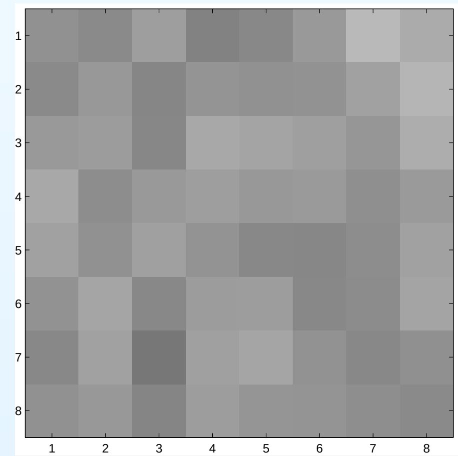
Original f



Reconstructed using $\mathbf{M_{JPE}}$
$(MSE = 72, C_R = 3,04)$



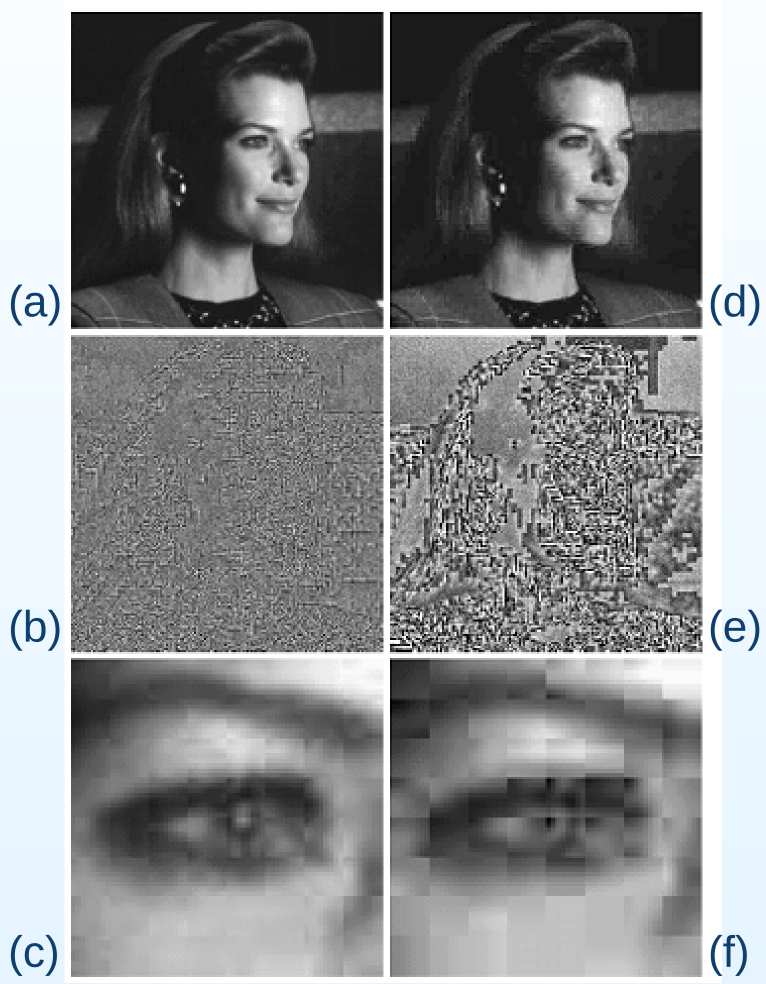
. . . using $2\mathbf{M_{JPEG}}$
$(MSE = 46, C_R = 1,66)$



. . . using $4\mathbf{M_{JPEG}}$
$(MSE = 14, C_R = 0,96)$

Comparisons

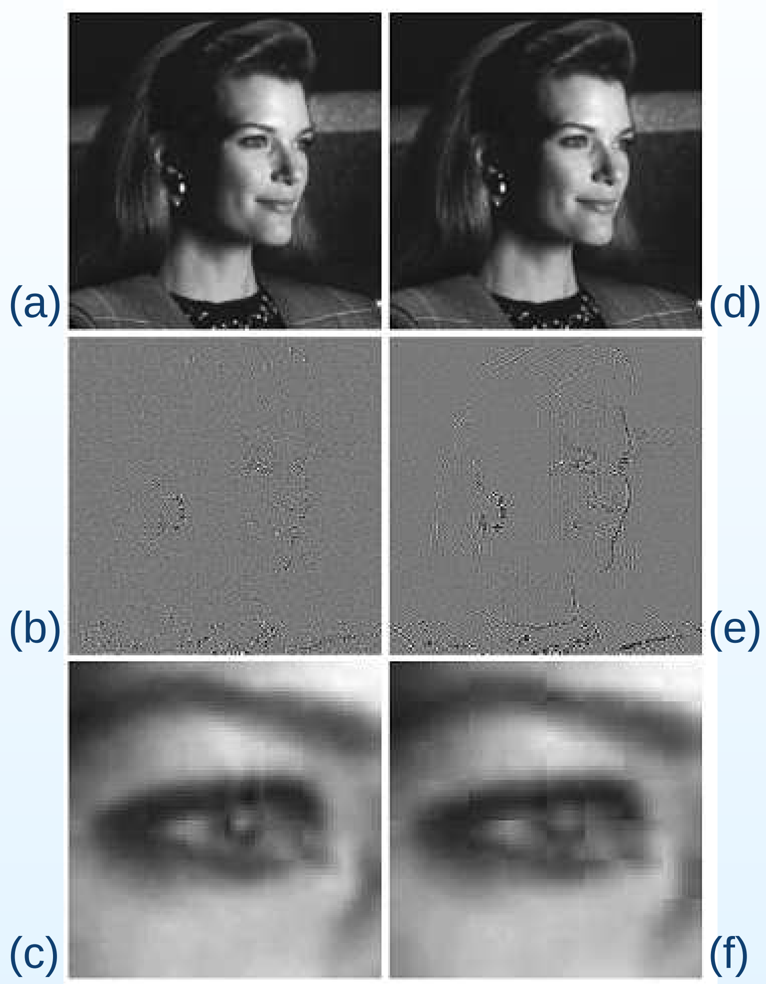
Different quantization threshold coding:
$\rightarrow$ dividing $\mathrm{M}_{\mathbf{JPEG}}$ $\Rightarrow$ rougher quantization



(a), (b), (c) - threshold coding using $\mathrm{M}_{\mathbf{JPEG}}$ mask
(d), (e), (f) - threshold coding using $\frac{1}{4}\mathrm{M}_{\mathbf{JPEG}}$ mask

Comparisons

Fixed ratio coding : keep 8 coefficients out of 64



(a), (b), (c) - threshold coding
(d), (e), (f) - zonal coding

Further compression

Threshold coding:

$\longrightarrow$ keeps (quantized) great magnitude coefficients

$\longrightarrow$ change to 0 small magnitude coefficients

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

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Further compression

Threshold coding:
$\longrightarrow$ keeps (quantized) great magnitude coefficients
$\longrightarrow$ change to 0 small magnitude coefficients

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

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

1. Specify the positions of the preserved coefficients:
   $12 \cdot 6 = 72$ bits

Introduction
Information Theory
Compression basics
Entropic coding
Inter-pixel coding
Quantizing and thresholding
Color space transforms
Image transforms
- Geometric analogy
- Functional analysis
- FT
- DCT
- DCT approximations
- DCT zonal coding
- DCT threshold coding
- JPEG
- WT

Further compression

Threshold coding:
$\longrightarrow$ keeps (quantized) great magnitude coefficients
$\longrightarrow$ change to 0 small magnitude coefficients

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

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

1. Specify the positions of the preserved coefficients: $12 \cdot 6 = 72$ bits
2. Run-length code horizontally the mask: $[\underline{2}, 1, \underline{2}, 3, \underline{4}, 4, \underline{2}, 6, \underline{1}, 16, \underline{1}, 22] \rightarrow 11 \cdot 6 = 66$ bits (last run-length optional!)

Introduction
Information Theory
Compression basics
Entropic coding
Inter-pixel coding
Quantizing and thresholding
Color space transforms
Image transforms
- Geometric analogy
- Functional analysis
- FT
- DCT
- DCT approximations
- DCT zonal coding
- DCT threshold coding
- JPEG
- WT

Further compression

Threshold coding:
$\longrightarrow$ keeps (quantized) great magnitude coefficients
$\longrightarrow$ change to 0 small magnitude coefficients

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

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

1. Specify the positions of the preserved coefficients: $12 \cdot 6 = 72$ bits
2. Run-length code horizontally the mask: $[\underline{2}, 1, \underline{2}, 3, \underline{4}, 4, \underline{2}, 6, \underline{1}, 16, \underline{1}, 22] \rightarrow 11 \cdot 6 = 66$ bits (last run-length optional!)
3. Run-length code the mask using a zig-zag pattern: $[\underline{5}, 1, \underline{4}, 3, \underline{2}, 7, \underline{1}, 41] \rightarrow 7 \cdot 6 = 42$ bits (last run-length optional!)

Further compression

Previous example:
-quantized DCT

$$\begin{pmatrix} 192 & -22 & 10 & -16 & 24 & 0 & 0 & 0 \\ 24 & -36 & 28 & 0 & 0 & 0 & 0 & 0 \\ -14 & -26 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Run length (zig-zag): $[\underline{10}, 4, \underline{1}, 3, \underline{1}, 46] \rightarrow 5 \cdot 6 = 30$ bits

- Possible coding of the sequence:
$[192, -22, 24, -14, -36, 10, -16, 28, -26, 14, 24, 22, 256, 10, 4, 1, 3, 1]$

- Lengths in bits:
$[8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 9, 6, 6, 6, 6, 6]$

- Total length: 124 bits

- Compression: $C_R = 64 \cdot 8/124 = 4,13 : 1$

Further compression

- Possible coding of the sequence:

$$[192, -22, 24, -14, -36, 10, -16, 28, -26, 14, 24, 22, 256, 10, 4, 1, 3, 1]$$

- Lengths in bits:

$$[8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 9, 6, 6, 6, 6, 6]$$

Ideas:

- separate DC and AC coefficients
- code DC coefficients differentially (between sub-images)
- code AC coefficients according to their psycho-visual importance ($\Leftrightarrow$ JPEG mask)
- combine AC coefficients coding with their position (include run-lengths of previous zeros)
- Huffman code (including the separation character)

Lookup tables in standard JPEG for Huffman and RLC

JPEG Standard

Lookup tables

TABLE 8.17
JPEG coefficient coding categories.

Range	DC Difference Category	AC Category
0	0	N/A
−1, 1	1	1
−3, −2, 2, 3	2	2
−7, . . . , −4, 4, . . . , 7	3	3
−15, . . . , −8, 8, . . . , 15	4	4
−31, . . . , −16, 16, . . . , 31	5	5
−63, . . . , −32, 32, . . . , 63	6	6
−127, . . . , −64, 64, . . . , 127	7	7
−255, . . . , −128, 128, . . . , 255	8	8
−511, . . . , −256, 256, . . . , 511	9	9
−1023, . . . , −512, 512, . . . , 1023	A	A
−2047, . . . , −1024, 1024, . . . , 2047	B	B
−4095, . . . , −2048, 2048, . . . , 4095	C	C
−8191, . . . , −4096, 4096, . . . , 8191	D	D
−16383, . . . , −8192, 8192, . . . , 16383	E	E
−32767, . . . , −16384, 16384, . . . , 32767	F	N/A

Lookup tables

TABLE 8.18
JPEG default DC code (luminance).

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Introduction
Information Theory
Compression basics
Entropic coding
Inter-pixel coding
Quantizing and thresholding
Color space transforms
Image transforms
- Geometric analogy
- Functional analysis
- FT
- DCT
- DCT approximations
- DCT zonal coding
- DCT threshold coding
- JPEG
- WT

Lookup tables

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23

TABLE 8.19 JPEG default AC code (luminance) *(continues on next page)*.

Lookup tables

2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110010010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26

JPEG overflow

1. Color space transform (RGB$\rightarrow$YCrCb) (optional)
2. Downsampling Cr and Cb planes (8:2:2, 4:1:1) (optional)
3. Divide image planes in 8×8 pixels blocks
4. Perform DCT on each block
5. Quantize and threshold the DCT by masks (compression quality)
6. Predictive code DC coefficients between blocks
7. Run-length code AC coefficients in zig-zag pattern
8. Huffman coding (combine RLC and AC values)
9. Construct header, mask information, . . .

Wavelet Transform

- Global transform (DCT = block by block)
  - inherent local because of wavelets
  - avoids blocking artefacts (FBI fingerprints)
- Multi-resolution nature
  - permits progressive compression
  - permits progressive restitution of the image
- Very good compression quality