Filters in Image Processing

Image Compression

David Svoboda

email: svoboda@fi.muni.cz
Centre for Biomedical Image Analysis
Faculty of Informatics, Masaryk University, Brno, CZ

October 25, 2019

Outline

1. Introduction
2. Lossless Compression
   - Variable-Length Coding (Huffman, arithmetic, . . .)
   - LZW Coding
   - Bit-plane Coding
   - Lossless Predictive Coding
3. Lossy Compression
   - Lossy Predictive Coding
   - Transform Coding

Image Compression

Introduction

What is an image compression?

- reduction of the amount of data required to represent a digital image

Application

- TV conferencing
- remote sensing (satellite imagery)
- medical imaging
- facsimile transmission (fax)
- . . .

Image Compression

Definitions

Some theory

⇒ Information – what we want to store or transmit

⇒ Data – the mean by which information is conveyed

⇒ Data redundancy – two distinct sources use a different type of data to give the same information

There are three main data redundancy types

- coding redundancy
- interpixel redundancy
- psychovisual redundancy

Coding Redundancy

Average number of bits

Let h be a intensity histogram of an inspected image I. The probability of the occurrence of level i:

$$p(i) = \frac{h(i)}{n},$$

where n is total number of image pixels.

If $len(k)$ is number of bits needed to represent the value k then

$$L_{avg} = \sum_{i=0}^{2^{\text{bit depth}}-1} len(i)p(i)$$

is the *average number of bits* required to represent each pixel in image I.

Notice: The total number of bits needed to code $M \times N$ image is MNL_{avg}.

Coding Redundancy

An example

We have 8-level image with 3-bit binary code

i	$p(i)$	3-bit code	$len(i)$	new code	*new* $len(i)$
$I_0 = 0$	0.19	000	3	11	2
$I_1 = 1$	0.25	001	3	01	2
$I_2 = 2$	0.21	010	3	10	2
$I_3 = 3$	0.16	011	3	001	3
$I_4 = 4$	0.08	100	3	0001	4
$I_5 = 5$	0.06	101	3	00001	5
$I_6 = 6$	0.03	110	3	000001	6
$I_7 = 7$	0.02	111	3	000000	6

Using new code brings better (lower) average number of bits per pixel:

$$\begin{aligned} L_{avg} &= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08) + \\ &\quad 5(0.06) + 6(0.03) + 6(0.02) = 2.7 \text{ bits} \end{aligned}$$

Coding Redundancy

The meaning:

- coding redundancy . . . the phenomenon when we use more code symbols (bits) than it is necessary

The aim:

- the functions "bit length" and "appearance probability" are inversely proportional
- the assignment of fewer bits to the more probable gray level and vice versa leads to image compression

Interpixel Redundancy

- correlation of pixels within an image
- the value of certain pixel in the image can be reasonably predicted from the values of group of other pixels in the image
- the gray levels of neighboring pixels are roughly the same, for example

An example: chessboard

$$\begin{matrix} 1&1&1&1&1&0&0&0&0&0&1&1&1&1&1 \\ 1&1&1&1&1&0&0&0&0&0&1&1&1&1&1 \\ 1&1&1&1&1&0&0&0&0&0&1&1&1&1&1 \\ 1&1&1&1&1&0&0&0&0&0&1&1&1&1&1 \\ 1&1&1&1&1&0&0&0&0&0&1&1&1&1&1 \\ 0&0&0&0&0&1&1&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&1&1&1&1&0&0&0&0&0 \end{matrix}$$

Interpixel Redundancy

Sample solution

Repetitious pixels may be grouped together:

$$
\begin{array}{l}
1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \rightarrow (5,1)(5,0)(5,1) \\
1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \rightarrow (5,1)(5,0)(5,1) \\
1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \rightarrow (5,1)(5,0)(5,1) \\
1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \rightarrow (5,1)(5,0)(5,1) \\
1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \rightarrow (5,1)(5,0)(5,1) \\
0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0 \rightarrow (5,0)(5,1)(5,0) \\
0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0 \rightarrow (5,0)(5,1)(5,0)
\end{array}
$$

Psychovisual Redundancy

⇒ The eye does not respond with equal sensitivity to the whole visual information.

⇒ Small intensity variations can be perceived in an area of constant intensity.

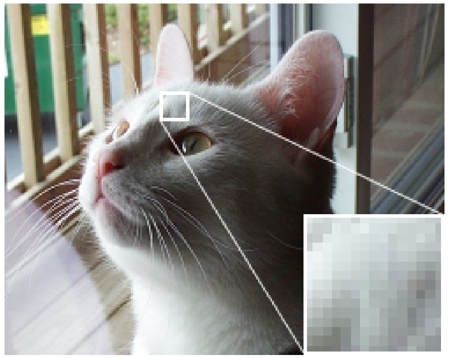
⇒ Certain information has less relative importance than other information → is psychovisually redundant.

Some examples:

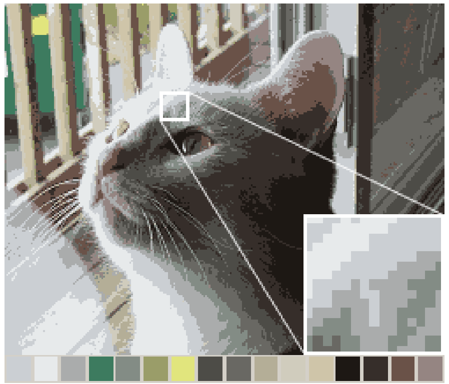
- We are more sensitive to differences between dark intensities than bright ones.
- We are more sensitive to differences of intensity in green than red or blue.

Psychovisual Redundancy

An example



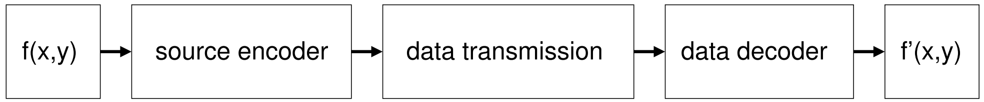
24-bit RGB color image



4-bit color image

Notice: Elimination of psychovisual redundant data results in a loss of (nonimportant) information → quantization → lossy compression methods.

General Compression Model



- $f(x,y)$ ...original input image
- $f'(x,y)$ ...input image after compression followed by decompression
- $f(x,y) \neq f'(x,y)$ in general
- encoder ...image filter responsible for image compression
- decoder ...image filter responsible for image decompression

Lossless Compression

Request for error-free compression:

- archival documents
- medical imaging
- business documents
- digital radiography

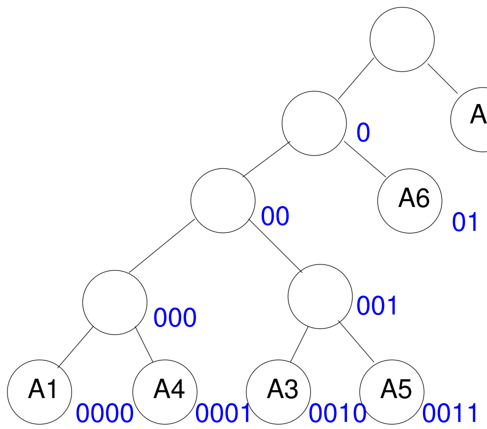
Common error-free compression methods:

- variable-length coding (Huffman, arithmetic)
- LZW coding
- bit-plane coding
- Lossless predictive coding
- run length encoding (RLE)

Huffman coding

An example

symbol	A_1	A_2	A_3	A_4	A_5	A_6
probability	0.1	0.4	0.07	0.1	0.04	0.29



Huffman coding

Variable-length coding $\approx$ reduction of coding redundancy

Algorithm

1. order the source symbol probabilities of appearance
2. reduce/merge the two lowest probable symbols into a new single symbol
3. repeat the reductions until it is possible
4. get the coding tree
5. mark the tree nodes with binary code (0 – left, 1 – right)
6. use the path code to code the individual symbol in the leaves

Huffman coding

Huffman coding creates the optimal code for a set of source symbols but it is difficult to construct Huffman code tree for larger sets.

Alternative solutions:

- Truncated Huffman – using Huffman coding only for the most probable source symbols. A prefix code followed by a suitable fixed-length code is used to represent all lower probable symbols.
- B_2-Code
- Binary shift
- Huffman shift

Notice: Alternative solution reduces the computational complexity with sacrificing coding efficiency.

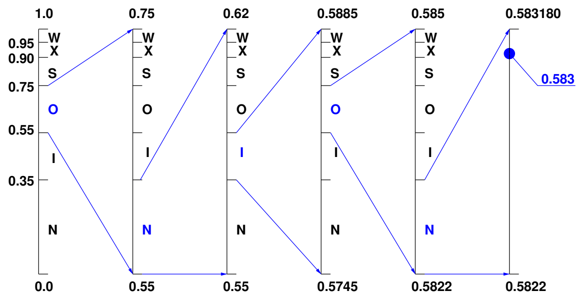
Arithmetic Coding

An example of encoding process

Table of available symbols:

symbol	N	I	O	S	X	W
probability	0.35	0.2	0.2	0.15	0.05	0.05

Input sequence: `'ONION'`



Output code word: `0.583`

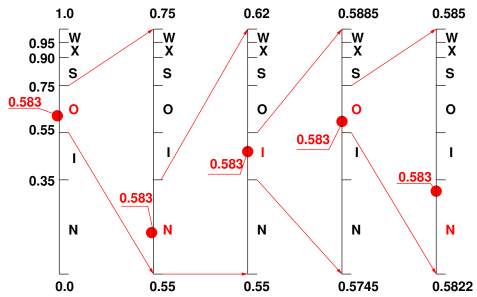
Arithmetic Coding

An example of decoding process

Table of available symbols:

symbol	N	I	O	S	X	W
probability	0.35	0.2	0.2	0.15	0.05	0.05

Input code word: `0.583`



Output sequence: `'ONION'`

Arithmetic Coding

In arithmetic coding the whole sequence of source symbols is assigned a single arithmetic code word.

- The sequence is usually very short. The data block is therefore split into several sequences.
- Code word itself defines an interval of real number between 0 and 1.
- The number of symbols increases → the code interval is smaller → number of bits required becomes larger.

LZW Coding

An example

Given this sequence for encoding

`TOBEORNOTTOBEORTOBEORNOT#`

and simple dictionary containing only single characters:

```
Dictionary (5-bit) Entries:
  ---------------------------
     0:  # = 00000
     1:  A = 00001
     2:  B = 00010
     3:  C = 00011
     ...
     27: Z = 11011
```

The length of sequence $=$ 25 symbols $\times$ 5b $=$ 125 b

LZW Coding

An example of encoding process

```
Input:   Bit Code (Output):   New Dictionary Entry:
---------------------------------------------------
  T        20 =  10100
  O        15 =  01111               28: TO
  B         2 =  00010               29: OB
  E         5 =  00101               30: BE
  O        15 =  01111               31: EO (full dictionary!)
  R        18 = 010010               32: OR
  N        14 = 001110               33: RN
  O        15 = 001111               34: NO
  T        20 = 010100               35: OT
  TO       28 = 011100               36: TT
  BE       30 = 011110               37: TOB
  OR       32 = 100000               38: BEO
  TOB      37 = 100101               39: ORT
  EO       31 = 011111               40: TOBE
  RN       33 = 100001               41: EOR
  OT       35 = 100011               42: RNO
  #         0 = 000000               43: OT#
```

Coded length $= 5 \times 5b + 12 \times 6b = 97b$.

LZW Coding

An example of decoding process

Before we start decoding, the only original dictionary (A,B,...,Z) is available!

```
Input (BitCode):   Output:   New Dictionary Entry:
---------------------------------------------------
  20 ( 10100)        T
  15 ( 01111)        O          28: TO
   2 ( 00010)        B          29: OB
   5 ( 00101)        E          30: BE
  15 ( 01111)        O          31: EO (full dictionary!)
  18 (010010)        R          32: OR
  14 (001110)        N          33: RN
  15 (001111)        O          34: NO
  20 (010100)        T          35: OT
  28 (011100)        TO         36: TT
  30 (011110)        BE         37: TOB
  32 (100000)        OR         38: BEO
  37 (100101)        TOB        39: ORT
  31 (011111)        EO         40: TOBE
  33 (100001)        RN         41: EOR
  35 (100011)        OT         42: RNO
   0 (000000)        #          43: OT#
```

Output: TOBEORNOTTOBEORTOBEORNOT#

LZW Coding

Covers the problem of coding as well as interpixel redundancy.

LZW = Lempel-Ziv-Welch:

- assigns fixed length code words to variable length sequences
- is dictionary (codebook)-based coding
- its dictionary is built during the coding process
- does not require any apriori knowledge of the probability of occurrence of the source symbols
- integrated in GIF, TIFF, PDF
- when the dictionary is full, we flush it and start with empty one

Bit-plane coding

Bit-plane decomposition ... gray level of an m-bit pixel in gray scale image can be represented in the form of the base 2 polynomial:

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

Therefore, the image can be simply decomposed into m 1-bit bit planes:

- zeroth-order bit plane ... a_0 bits of each pixel
- first-order bit plane ... a_1 bit of each pixel

  ⋮
- $(m-1)$st-order bit plane ... a_{m-1} bit each pixel

Question: Is it a good representation? Imagine neighbouring graylevels 127 and 128.

Bit-plane coding

An alternative decomposition approach: *m*-bit Gray code

$$\begin{aligned} g_i &= a_i \oplus a_{i+1} \quad 0 \le i \le m-2 \\ g_{m-1} &= a_{m-1} \end{aligned}$$

where $\oplus$ denotes the exclusive OR operation:

- small changes in gray level are less likely to affect all m bit planes
- neighbours 127 and 128 in Gray code are 11000000 and 01000000

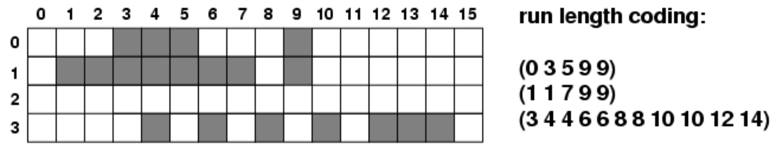
Bit-plane coding

Bit-plane properties:

- high-order bit-plane . . . large uniform areas
- low-order bit-plane . . . quite complex

Bit-plane compression:

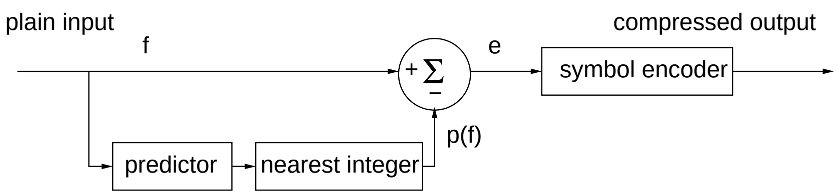
- Constant Area Coding (CAC) . . . binary image is split into tiles
- White Block Skipping (WBS) . . . only black area is stored
- Run-Length Coding (RLE) . . . codes the length and the value of each uniform run



Lossless Predictive Coding

Intended to eliminate the interpixel redundancy of closely spaced pixels.

An idea: encoded value of the pixel is the difference between the actual and predicted value of that pixel.



$$e(n) = f(n) - p(f(n))$$

Lossless Predictive Coding

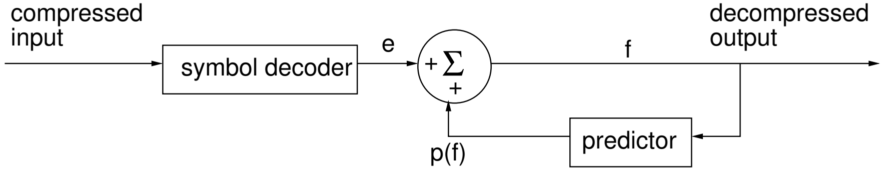
Some types of predictors:

- $p(f(n)) = \text{round}\left[\sum_{i=1}^{M} \alpha(i) f(n-i)\right]$
- $p(f(m,n)) = \text{round}\left[\sum_{i=1}^{M} \alpha(i) f(m,n-i)\right]$
- $p(f(m,n)) = \text{round}\left[\alpha f(m,n-1)\right]$

Notice: Predictor is typically linear combination of few previous pixels.

Lossless Predictive Coding

Decompression



Lossy compression

By now $f(x, y) = f'(x, y)$ was valid.

Now:

$$f(x, y) \neq f'(x, y)$$

Why lossy compression?

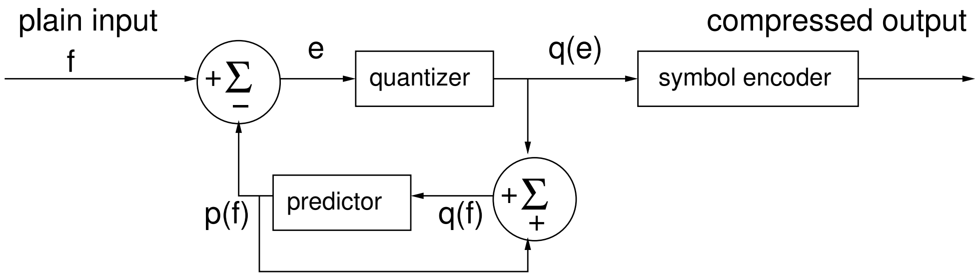
- compression ratio is higher . . . 10:1 to 50:1

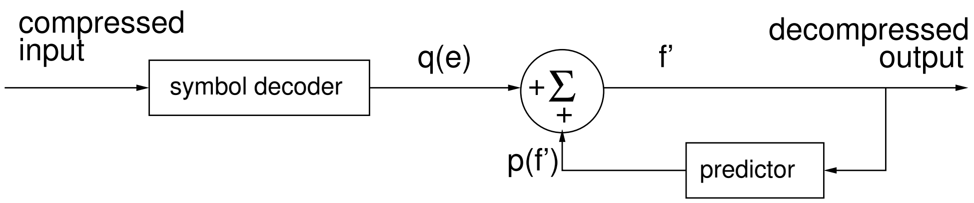
What is the main difference?

- quantizer (see psychovisual redundancy) is present

(Lossy Predictive Coding)

Encoder & Decoder





(Lossy Predictive Coding)

An example

Delta modulation (DM)

$$p(f(n)) = \alpha f(n-1)$$

$$q(e(n)) = \begin{cases} +\xi & \text{for} \quad e(n) > 0 \\ -\xi & \text{otherwise} \end{cases}$$

where:

- α . . . prediction coefficient (≤ 1)
- ξ . . . positive constant

(Lossy Predictive Coding)

An example

Delta modulation (DM)

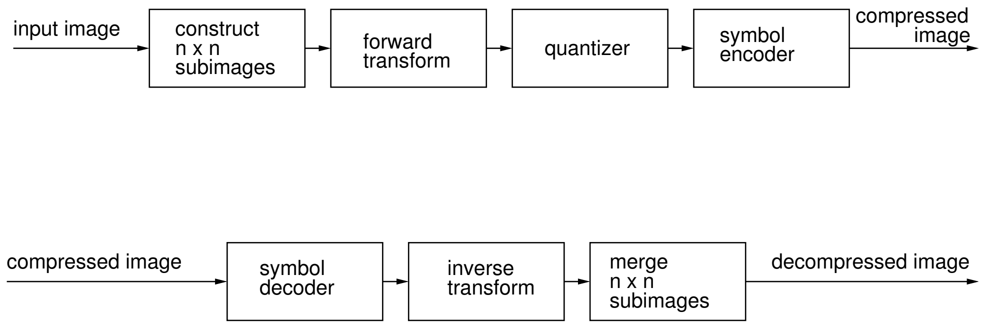
$$\alpha = 1; \quad \xi = 6.5$$

n	f	p(f)	e	q(e)	q(f)
0	14	–	–	–	14.0
1	15	14.0	1.0	6.5	20.5
2	14	20.5	-6.5	-6.5	14.0
3	15	14.0	1.0	6.5	20.5
⋮	⋮	⋮	⋮	⋮	⋮
14	29	20.5	8.5	6.5	27.0
15	37	27.0	10.0	6.5	33.5
⋮	⋮	⋮	⋮	⋮	⋮

Transform Coding

All the previous compression techniques operate directly on the pixels of an image → are spatial domain methods.

Transform coding . . . compression is realized in another domain (e.g. Fourier domain)



Transform Coding

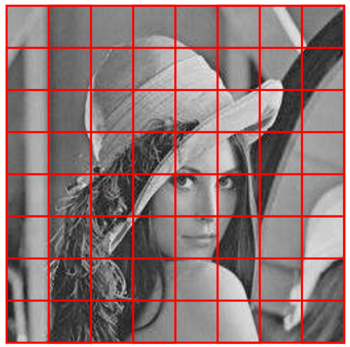
Three main issues:

- subimage size
  dimensions are typically power of 2, i.e. 8×8 or 16×16
- transform selection
  (Fourier, DCT, Walsh-Hadamard, KLT, Wavelet, . . .)
- bit allocation
  select which part of transformed domain is less important (redundant) and hence can be eliminated

Transform Coding

Subimage Size

The individual transforms are supposed to be applied to subimages.



The image size affects:

- transform coding error
- computational complexity

Transform Coding

Transform Selection

The selected transform should be

- able to decorrelate the input signal (≈ energy compaction)
- easy to implement and fast (computational complexity)
- orthogonal (reversible)

The most common transforms

- Optimal decorrelation
  - PCA transform
- Approximations
  - Discrete Fourier transform
  - DCT
  - Walsh-Hadamard

Transform Coding

Bit Allocation ≈ Quantization

An idea

- remove/suppress less important (redundant) part of transform domain

The process of less important pixels removal is called bit allocation:

- zonal coding . . . based on variance
  coefficients of maximum variance carry the most image information and should be retained
- threshold coding . . . based on magnitude
  coefficients which are high enough are retained – very simple for evaluation

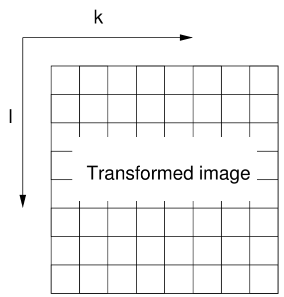
Notice: In Fourier domain, the less important pixels correspond to high frequency coefficients.

Transform Coding

Bit Allocation

Example – Zonal coding

- The transformed coefficients are masked (multiplied by 0/1) with zonal mask.
- Coefficients of maximum variance usually are located around the low frequencies.
- An example of subimage 8×8 and corresponding zonal mask:



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

Transform Coding

Bit Allocation

Example – Threshold coding

- The transformed coefficient are quantized using point-wise division with normalization array Z.
- The least important coefficients are suppressed:

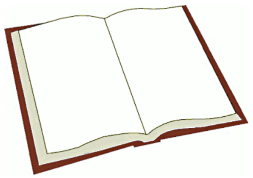
$$\overline{T}(k,l) = round\left[\frac{T(k,l)}{Z(k,l)}\right]$$

- An example of normalization array:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Bibliography

- Salomon D. Data Compression, The Complete Reference, 4th edition, Springer, London, 2007, ISBN-1846286025
- Gonzalez, R. C., Woods, R. E. Digital image processing / 2nd ed., Upper Saddle River: Prentice Hall, c2002, pages 793, ISBN 0201180758
- Kuhn M. Information Theory and Coding Image, video and audio compression – slides from lectures
- Welch, T. A. A technique for high-performance data compression, Computer. Vol. 17, pp. 8-19. June 1984



You should know the answers . . .

- Explain the difference among *coding redundancy*, *interpixel redundancy*, and *psychovisual redundancy*.
- What does quantizer do?
- Which type of data cannot be compressed by using lossy compression methods?
- Show the construction of Huffman coding tree.
- How does the arithmetic decoder know when it should stop decoding one codeword?
- How does the encoder deliver the dictionary to the receiver's decoder in LZW compression scheme?
- Explain the meaning of *predictor* and *error* in lossless predictive coding scheme.
- Why do we split the 2D images into tiles of size 8×8 or 16×16 pixels?
- Which compression scheme would you use for coding of chessboard image of size 64×64 pixels. Assume that each tile is defined as a homogeneous square area.