Chapter 6: Image Compression

Andre L. C. Barczak

Computer Science
Massey University, Albany

Contents

Introduction

Redundant information

Three types of redundancies to be explored by compressors:

- coding: minimise the length of the representation
  - e.g., if you have AAAAAAAABB, it could be compressed as 8A2B
- inter-pixel: minimise the repetitions (redundancies)
  - e.g., represent regions (or blobs) as vectors
- psycho-visual: lossy, disregard information not "visible"
  - Humans do not see subtle changes in illuminat

Introduction

Compression Ratio

$$C_r = \frac{d_1}{d_2} \tag{1}$$

where: d_1 and d_2 are the amounts of data in the original and compressed images Average compression ratios rarely go over 30 for normal coding.

Lossy x Loss-less compression

- Loss-less (non-lossy): recover the exact input
- Lossy: the image recovered looks the same to users, but it is not

Shannon's Information Theory and Entropy

Entropy

$$H = -\sum_{i=0}^{N} p_i \, log(p_i) \tag{2}$$

Where: p_i is the probability of a symbol to appear in a stream.

Example

8 symbols (000, 001, 010, 011, 100, 101, 110, and 111) with equal probabilities would yield maximum Entropy:

$$H = -\sum_{i=0}^{N} p_i \, log_2(p_i) = -8 \, \frac{1}{8} \, log_2(\frac{1}{8}) = 3$$

More examples on Entropy

For example, suppose that in the 3 bits system we have symbols A, B, C, D, E, F, G and H. The probabilities are: A(0.4) B(0.3) C(0.1) D(0.1) E(0.06) F(0.04) G(0) and H(0).

$$H = -(0.4log_2(0.4)+0.3log_2(0.3)+20.1log_2(0.1)+0.06log_2(0.06)+0.04log$$

$$H = 2.1435$$

Another example, A and B in equal quantities (probabilities are 0.5). All other symbols have probabilities equals zero.

$$H = -2\,(0.5)\,log_2(0.5) = 1$$

The entropy expresses the average number of bits that would take to represent the same stream using a *variable-length code*.

Loss-less compression

Huffman

- Huffman coding (1952)
- So-called Entropy coder, it converts the probabilities into a sequence of bits
- Entropy: $H = -\sum p_i \log_2(p_i)$
- Max Entropy for a 1 byte code: 8
- When a file has Entropy 8, it is not compressible using Huffman's code

LZW

- LZW (Lempel-Ziv-Welch, 1984)
- Builds a dynamic dictionary of repeated sequences
- Plagued by patents (Unisys and IBM)

Huffman

The method

- Create an ordered list for each symbol
- Symbol can be of any length
- Each arbitrary size symbol is represented by an increasing number of bits
- *Compression Ratio* is calculated based on the total number of bits

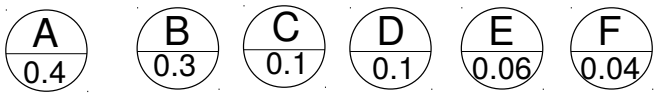
Huffman pseudo-code

Require: Heap (minHeap, or a priority queue) P
Require: Tree node T_{huf} and N tree nodes T_i, temporary tree nodes T_a, T_b and T
1: Create a leaf T_i for each symbol (using the probability as a key)
2: Insert all T_i in the heap P
3: **while** there is more than one node in the Heap P **do**
4: delete the two nodes from P (the two with lowest probability) and copy to T_a and T_b
5: create a **new** tree node T
6: make the probability of T equals to the sum of the probabilities of $T_a + T_b$
7: make T_a and T_b children (smallest on left) of the new node
8: insert the new tree node back into the heap P
9: **end while**
10: the remaining node in P (with probability 1) is the root of tree T_{huf}.
11: Traverse T_{huf}, marking 0 if going left and 1 if going right.
12: When reaching a leaf, output the code.

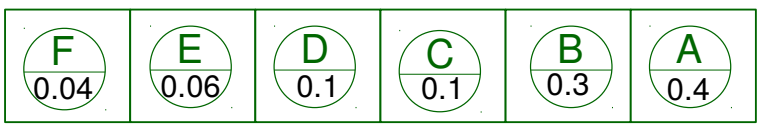
Huffman

Initialisation and step 1

Symbol	A	B	C	D	E	F
Probability	0.4	0.3	0.1	0.1	0.06	0.04



Heap



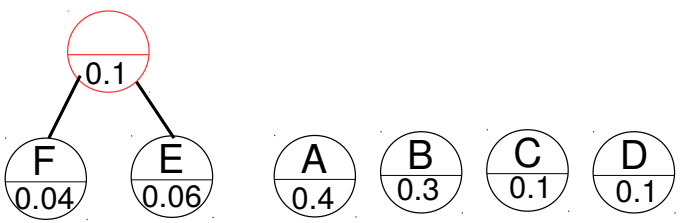


Figure 1: Huffman Example.

Huffman

steps 2 and 3

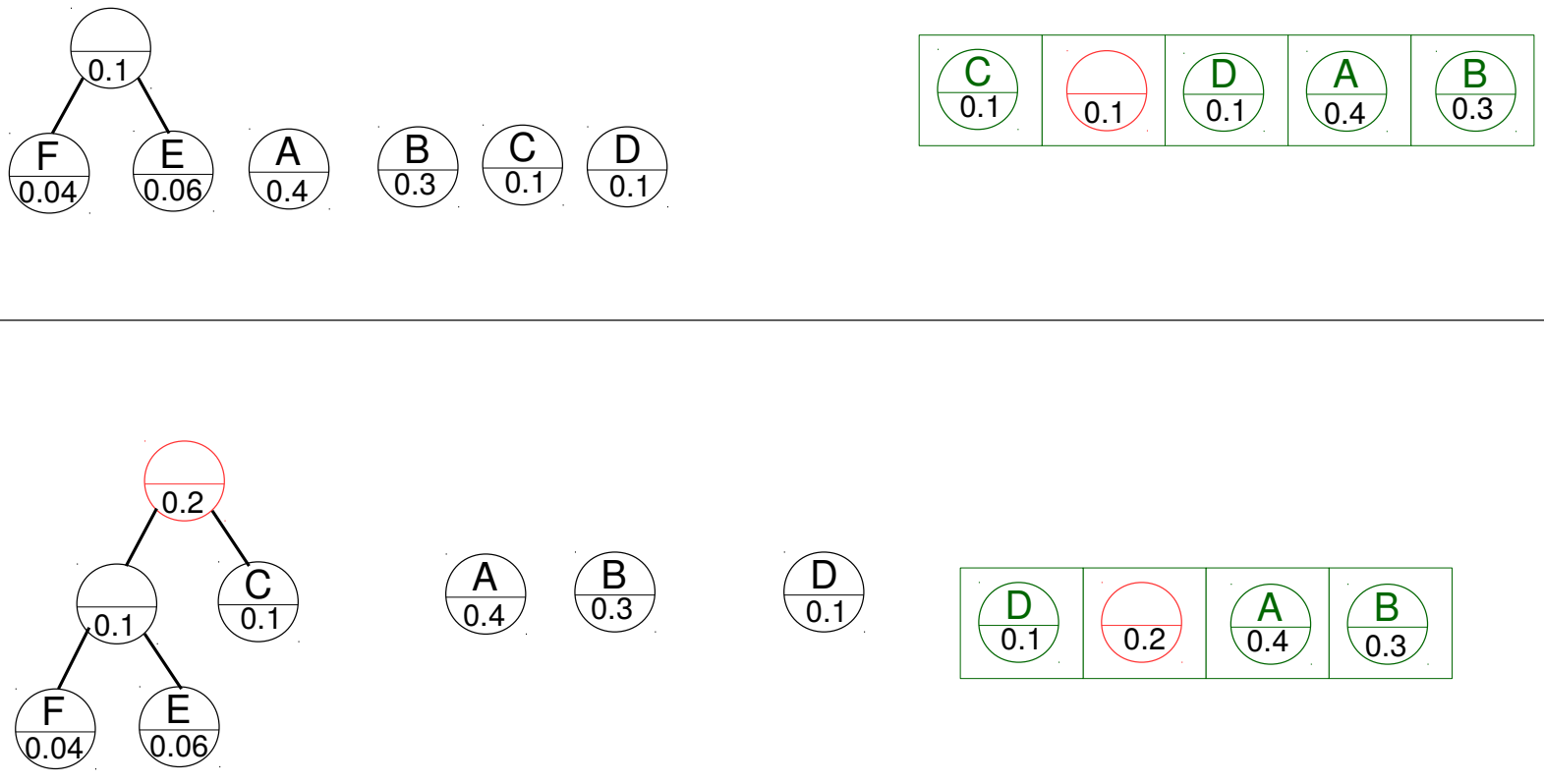
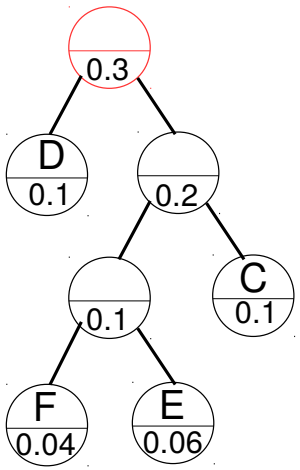
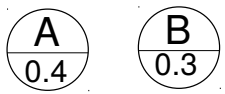


Figure 2: Huffman Example.

Huffman

step 4





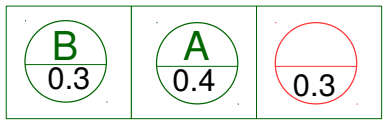
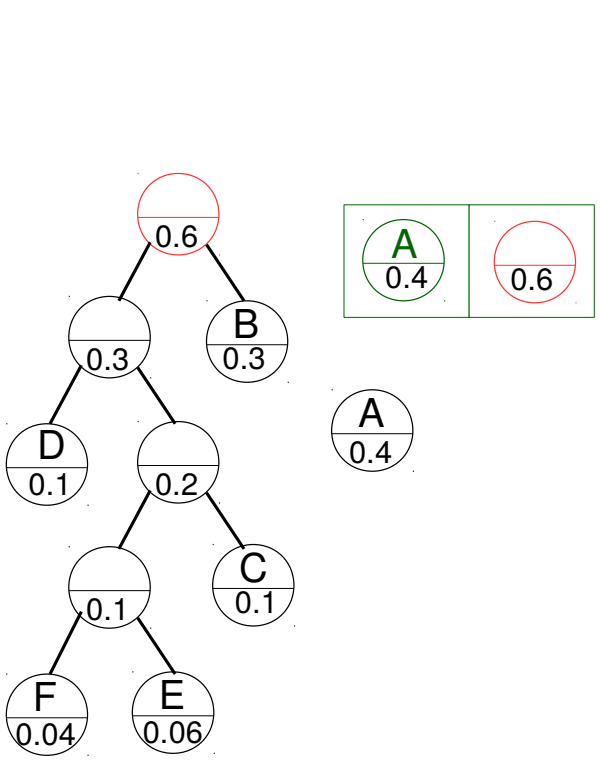


Figure 3: Huffman Example.

Huffman

final steps



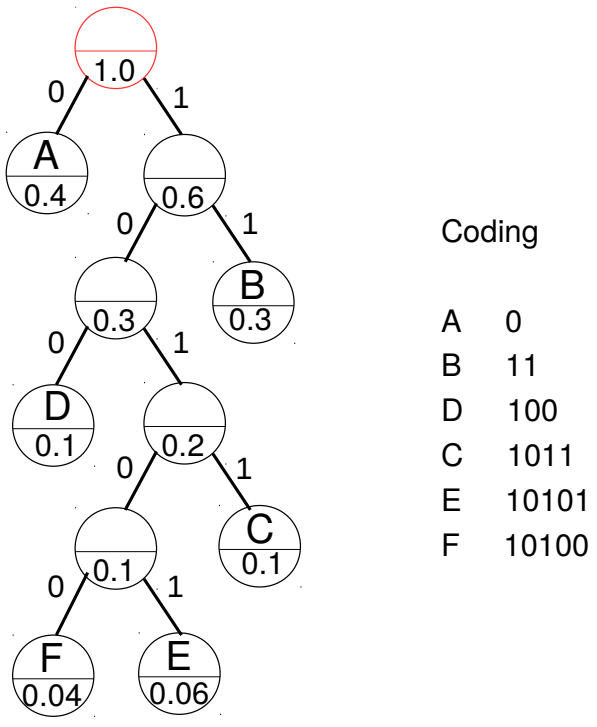


Figure 4: Huffman Example.

Huffman

Compression ratio

$$C_r = \frac{3}{(0.4 * 1 + 0.3 * 2 + 0.1 * 3 + 0.1 * 4 + 0.06 * 5 + 0.04 * 5)} = 1.36$$

The average number of bits predicted by the Entropy of the uncompressed file is:

$$H = -\sum p_i \log_2(p_i) = 2.1435$$

so the compression predicted by the number of bits is $\frac{3}{2.1435} = 1.39$, very close to the real result.

LZW

Introduction

- GIF and TIF formats can use this method
- Patents created controversy: PNG format was created
- Patents expired in 2003 and 2004
- Many variants: LZ77, LZ78, LZMW (1985), LZAP (1988), LZWL etc

LZW

The method

- Create a dictionary
- Uses inter-code redundancies: i.e. repeated 'letters' are used once
- In images, repeated sequences of pixels are encoded this in this way

LZW

The algorithms

- Input: file1 (or image1)
- Output: file2 (or image2)
- Strings S, O and character C.
- Array T[N] (N=number of entries)
- + means append

LZW

Compression

1. Initialise a table with a single value strings (say from 0 to 255).
2. Initialise S = pixel[0]
3. WHILE there are pixels in *image* DO

   C = get input character

   IF S+C is in the string table then

   $S = S$+C

   ELSE

   output the code for S

   add S+C to the string table

   $S = C$

   END of IF
4. END of WHILE
5. output the code for S

LZW

Decompression

1. Initialise a table with single value strings (say from 0 to 255).
2. Read O = pixel[0] of *image*2
3. output O
4. WHILE there are pixels in *image*2 DO
   Read N = pixel[i]
   S = get translation of N
   output S
   C = first character in S
   add $O + C$ to the translation table T
   $O = N$
5. END of WHILE

LZW

Example

Let us use a 16 byte sequence:

39 39 126 126 39 39 126 126 39 39 126 126 39 39 126 126

LZW

Example

Table 1: LZW compression.

S	C	Output	Code	Dictionary String
39				
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

LZW

Compression ratio

- Original sequence with 8 bits: 39 39 126 126 39 39 126 126 39 39 126 126 39 39 126 126
- Compressed with 9 bits: 39 39 126 126 256 258 260 259 257 126
- $C_r = (16 * 8bits)/(10 * 9bits) = 1.42$

LZW

Example

Table 2: LZW decompression.

N	O	C	S	Dictionary String	Code
	39				
39	39	39	39	39-39	256
126	39	126	126	39-126	257
126	126	126	126	126-126	258
256	126	39	39-39	126-39	259
258	256	126	126-126	39-39-126	260
260	258	39	39-39-126	126-126-39	261
259	260	126	126-39	39-39-126-126	262
257	259	39	39-126	126-39-39	263
126	257	126	126	39-126-126	264

Lossy compression

Modelling the data to increase redundancies

- Cannot recover the exact original image
- However, "good enough" quality
- Use of a model for the data,i.e., use specific properties of the data:
  - Quantisation
  - Finite coefficients of frequency domain (DCT, FFT etc)
  - Colour space

Quantisation

- Staircase most commonly used
- Humans do not perceive subtle changes in intensities, specially if not neighbours
- Can be applied to colour spaces (e.g., remember YUV411)
- Non-linear quantisation (e.g., Lloyd-Max quantisers, with different quantisation for different intervals)

DCT

Introduction

- Remember Fourier Series?
- Use Cosines only
- Discrete Cosine Transform uses NxN number of coefficients
- Redundancies yield coefficients with low or zero values.
- On decompressing, we delete a number of low value coefficients.

DCT

Equations

A Discrete Cosine Transform for a 2D square sized image (NxN) $i(x, y)$ is:

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} i(x, y).cos \left[\frac{(2x+1)u\pi}{2N} \right] .cos \left[\frac{(2y+1)v\pi}{2N} \right] \quad (3)$$

Where:

$\alpha(0) = \sqrt{1/N}$

$\alpha(u) = \sqrt{2/N} for\ u = 1, 2, ..., N-1$

DCT

Equations

To recover the image given the coefficients $C(u, v)$, the following equation is used:

$$i(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u, v).cos\left[\frac{(2x+1)u\pi}{2N}\right].cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (4)$$

DCT

Basis functions visualised

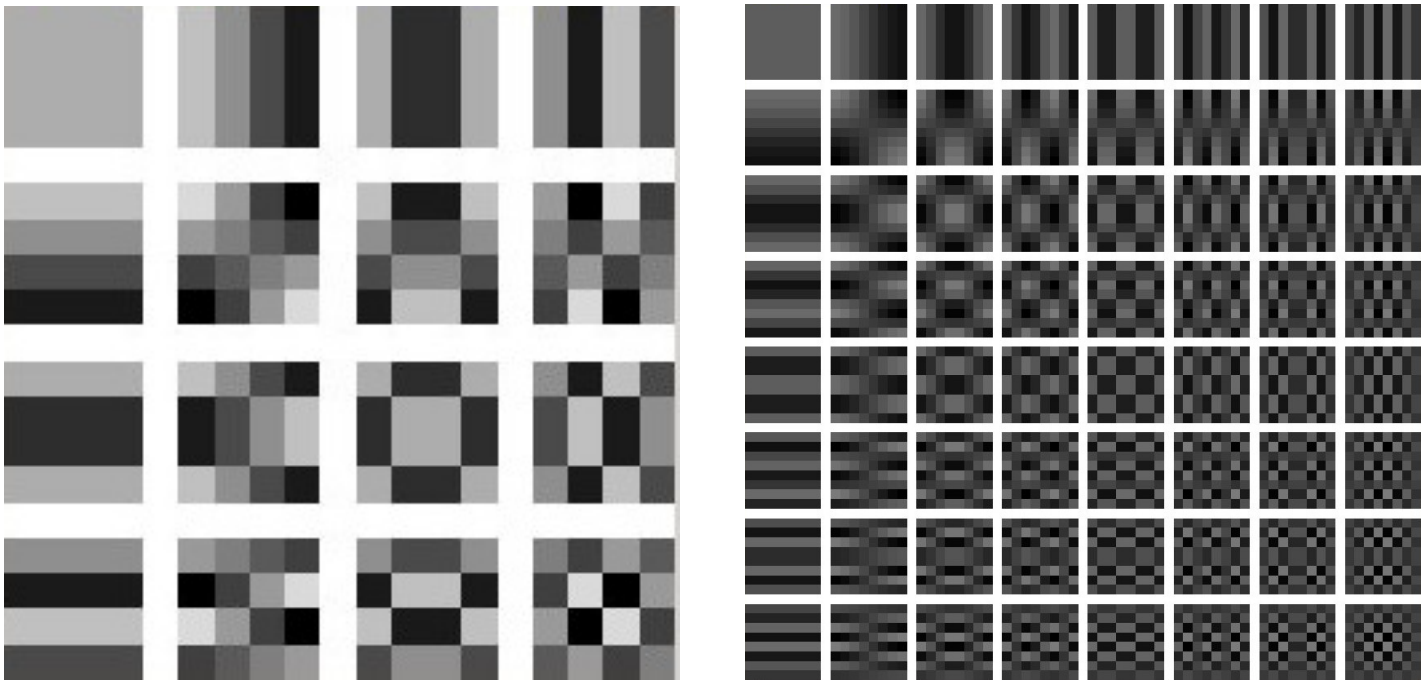


Figure 5: Basis functions for a DCT transform. a) 4x4 transform b) 8x8 transform.

DCT

Example

Image 8x8

1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23
1	127	127	120	120	123	124	23

DCT result

765	−24	−28	1	−32	−23	7	−20
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	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 6: DCT example.

DCT

Observations:

- Loss-less DCT compression: encode with all coefficients
- Lossy DCT compression: encode with the largest coefficients
- e.g., JPEG (DCT, quantise, encode)

Video Compression

Introduction

- In video, we can compress every frame.
- Are there Inter-frame redundancies?
- How to explore them?

Video Compression

Mpeg

- "Infinite" ways to write codecs (coder-decoder)
- Attempts to standardise them: MPEG-1, MPEG-2, MPEG-4, MPEG-7, MPEG-21...
- The ultimate goal is to compress and decompress efficiently

Video Compression

Mpeg

- If only objects move, we could keep the background in one frame...
- However, in practise there are problems with that.
- E.g., occlusion, too many things moving etc.
- noisy videos take more space...

Video Compression

Mpeg-2

- Three frame types
- I frames: intra-coded (jpeg).
- P frames: predicted frames, use previous I or other P frames with few blocks changes.
- B frames: Bi-directional frames, predicted frames using past and future frames.

Video Compression

Mpeg-2

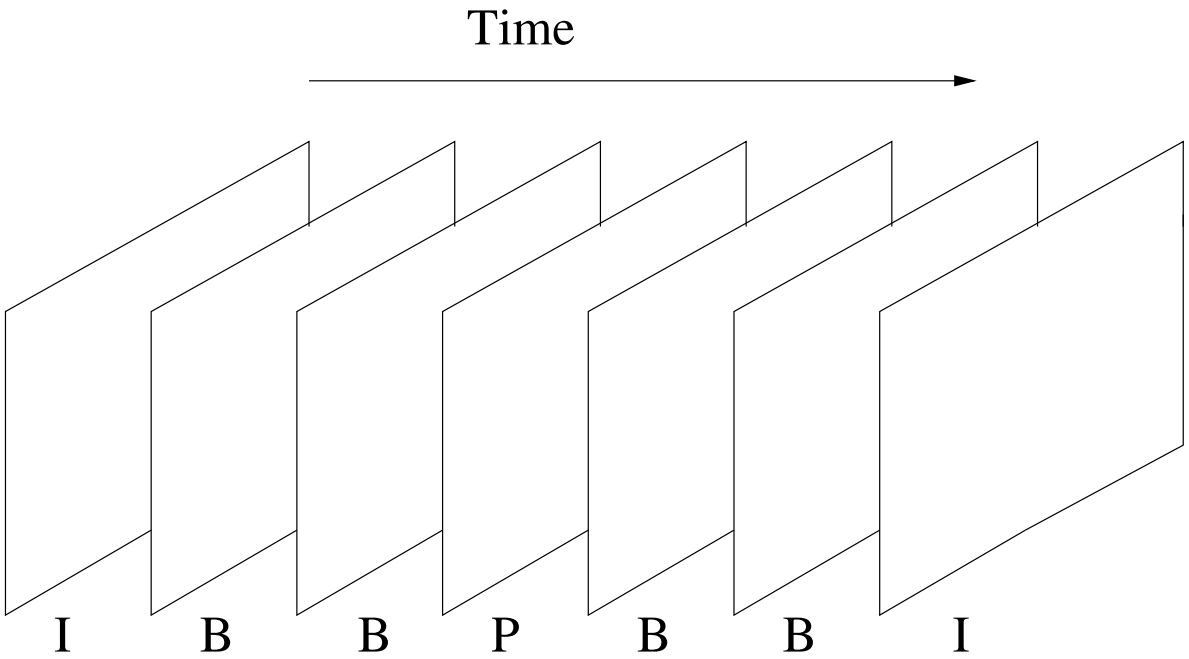


Figure 7: MPEG frame types.

Exercises 1 and 2

- 1) Write a simple version of Huffman code that is able to compress a grey-scale image in OpenCV. How does your compression rate fare against jpg? (to check that, you can export a jpg image in GIMP, after transforming it into grey-scale.)
- 2) Using the DCT transform above, write a program to compute the cosine transform coefficients for an 8x8 block. Compress it using Huffman code.

Exercises 3 and 4

- 3) Using the DCT transform above, write a program to compute the cosine transform coefficients for an 16x16 block. Compress it using Huffman code. Any difference in the compression rate?
- 4) Modify the program to compute the inverse cosine transform. Analyse the errors in the resulting images by simple rounding (to integers) and quantisation of the coefficients compared to the use of floating point numbers to represent the coefficients.

Bibliography
(partial)

 R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2002.

 M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis and Machine Vision*, PWS Publishing, 1999