Image compression

Jan Kybic, Václav Hlaváč, Tomáš Svoboda

http://cmp.felk.cvut.cz/~kybic

kybic@fel.cvut.cz

December 2010

Overview

- **Introduction** — definition, motivation, classification, basic principles, simple methods
- **General methods (1D)** — entropy coding, run length encoding (RLL), Lempel-Ziv (Deflate)
- **Image specific coding** — predictive coding (DPCM), transform coding, KL transformation, DCT, JPEG, PNG
- **Conclusions**

Resources

- Anil Jain: "Fundamentals of Digital Image Processing", 1989.
- M. Sonka, V. Hlaváč, R. Boyle R.: "Image Processing, Analysis, and Machine Vision", 2007.
- T. Svoboda, J. Kybic, V. Hlaváč: "Image Processing, Analysis, and Machine Vision, A MATLAB Companion", 2007. http://visionbook.felk.cvut.cz

Image compression

- **Image compression** =
  minimizing the number of bits to represent an image.
- **Lossy / lossless compression** — distortion versus size
- **Reasons to compress**
  - to save *storage space*
  - to shorten *transmission time* and conserve bandwidth

Example

$5184 \times 3456 \times 14$ bits $= 17.9$ Mp $= 251$ Mb $=$ **31.3** MB.
Raw file is **22** MB (lossless compression).
A high quality JPEG (shown) is **5.1** MB.

Example

JPEG, quality parameter 75, 847 kB.

Example

JPEG, quality parameter 50, 452 kB.

Example

JPEG, quality parameter 30, 309 kB.
Acceptable quality (for this resolution), compression ratio 1 : 100.

Example

JPEG, quality parameter 10, 198 kB.

What makes compression possible?

- Images are not noise. Images are redundant and predictable.
- Intensities are distributed non-uniformly.
- Colour channels are correlated.
- Pixel values are spatially correlated.
- Properties of human visual perception.

Downsampling

- Reduce the size (spatial resolution) of the image
- Lossy, simple, often appropriate (limited monitor resolution, web)
- High-quality interpolation (B-splines) helps

Downsampling

Original size, 3456×5184, 859 kB (stored as JPEG with quality 75)

Downsampling

Scaling factor 0.5, 1728 × 2592, 237 kB

Downsampling

Scaling factor 0.25, 864×1296, 75 kB

Downsampling

Scaling factor 0.125, 432×648, 27 kB

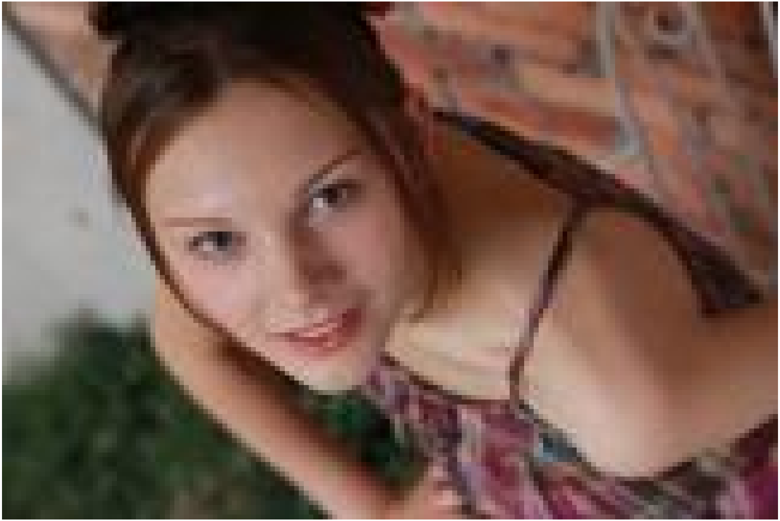
Downsampling

Scaling factor 0.0625, 216 × 324, 10 kB

Downsampling

Scaling factor 0.0625, 216×324, 10 kB, bicubic interpolation

Downsampling



Scaling factor 0.03125, 108×162, 4.2 kB

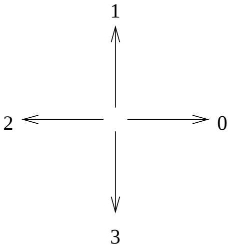
Downsampling

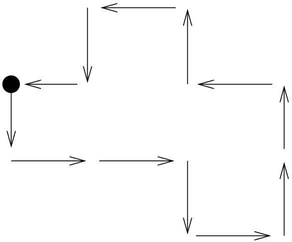
Scaling factor 0.03125, 108 × 162, 4.2 kB, bicubic interpolation

Chain code

(Freeman, 1961)

- Lossless encoding of binary images.
- Encode the boundary of each connected component.





Chain code:

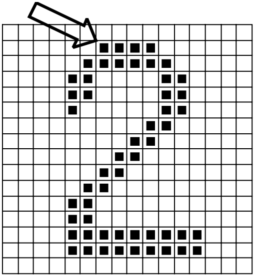
initial position,3, 0, 0, 3, 0, 1, 1, 2, 1, 2, 3, 2.

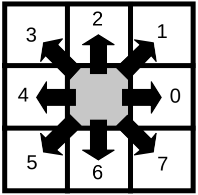
Derivative code:

initial position, initial direction, 1, 0, 3, 1, 1, 0, 1, 3, 1, 1, 3, 1.

Chain code

(Freeman, 1961)



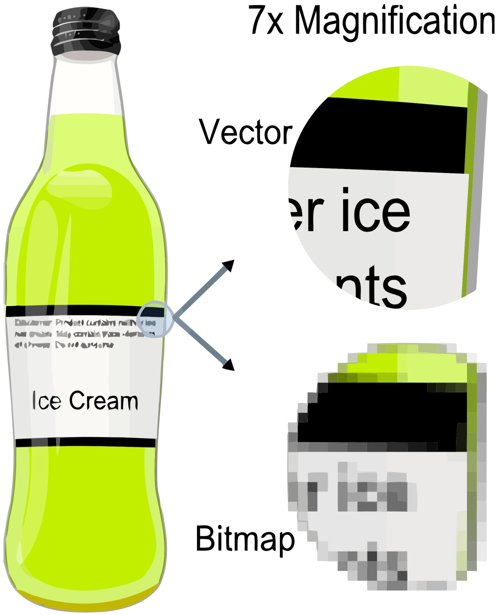


Code (8-neighborhood):
00077665555556600000006444444442221111112234445652211

Vector image formats

- Use geometric primitives (points, lines, polygons, spline curves)
- Very efficient (small files), allows unlimited zoom, avoids interpolation when rescaling.
- PostScript, PDF, SVG, ...
- Vector to raster conversion is easy.
- Raster to vector conversion (*vectorization*) is very difficult.
- Unsuitable for natural images (photographs)

Vector image formats



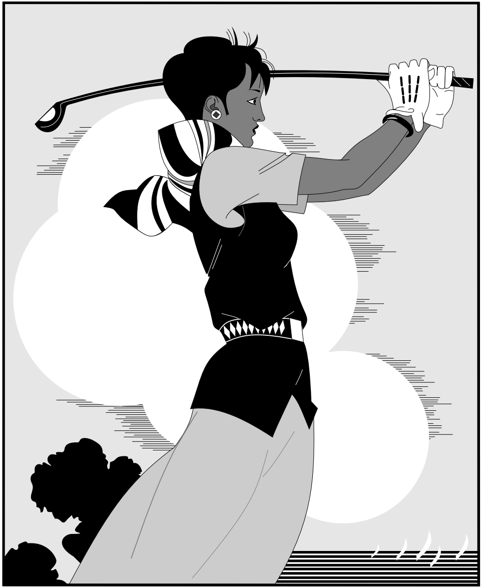
Vector image formats

PDF, 34 kB

Vector image formats

JPG, 1146 × 1183, 197 kB

Vector image formats



PDF, 12 kB

Vector image formats

JPG, 1188 × 1448, 105 kB

Pixel coding

- Each pixel is processed independently
- **Quantization**
  - Reduce the number of quantization (intensity) levels
  - **Lossy** compression
  - Applicable to other signals (e.g. sound)
- **Entropy coding**
  - Use short codewords for frequently occurring input symbols
  - Lossless compression
  - Applicable to any data stream
  - Huffman coding, arithmetic coding

Quantization

- **Quantizer**: maps an input u to a quantized output u_q
- u_q takes values from a discrete set

Quantization

- **Quantizer**: maps an input u to a quantized output u_q
- u_q takes values from a discrete set
- Piecewise constant (staircase function)

$$u \to u_q : \quad t_k \leq u < t_{k+1} \Rightarrow u_q(u) = r_k$$

t_k — decision levels (thresholds)
r_k — reconstruction levels (outputs)

Quantization

- **Quantizer**: maps an input u to a quantized output u_q
- u_q takes values from a discrete set
- Piecewise constant (staircase function)

$$u \rightarrow u_q : \quad t_k \leq u < t_{k+1} \Rightarrow u_q(u) = r_k$$

  t_k — decision levels (thresholds)
  r_k — reconstruction levels (outputs)
- Uniform quantization
- Adaptive/optimal quantization

Quantization

- **Quantizer**: maps an input u to a quantized output u_q
- u_q takes values from a discrete set
- Piecewise constant (staircase function)

$$u \rightarrow u_q : \quad t_k \leq u < t_{k+1} \Rightarrow u_q(u) = r_k$$

  t_k — decision levels (thresholds)
  r_k — reconstruction levels (outputs)

- Uniform quantization
- Adaptive/optimal quantization

Note: Color quantization leads to *vector quantization*

Quantization example

PNG, 813 × 897 pixels, 256 levels, 315 kB

Quantization example

PNG, 813 × 897 pixels, 128 levels, 256 kB

Quantization example

PNG, 813 × 897 pixels, 64 levels, 202 kB

Quantization example

PNG, 813 × 897 pixels, 32 levels, 147 kB

Quantization example

PNG, 813 × 897 pixels, 16 levels, 89 kB

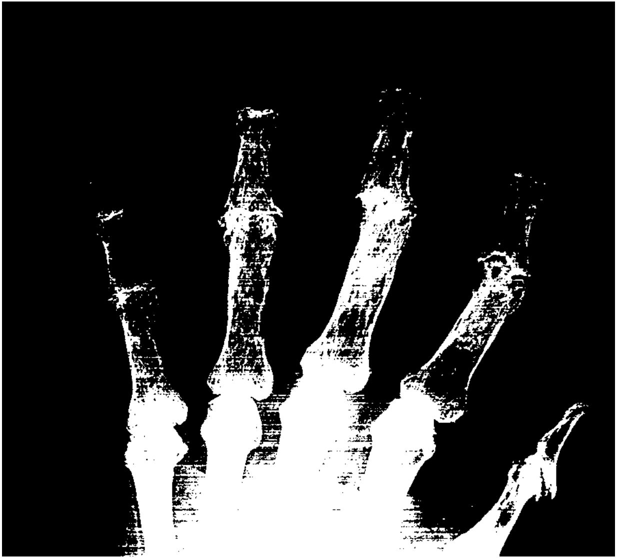
Quantization example

PNG, 813 × 897 pixels, 8 levels, 53 kB

Quantization example

PNG, 813 × 897 pixels, 4 levels, 36 kB

Quantization example



PNG, 813 × 897 pixels, 2 levels, 20 kB

Optimal quantization*

Lloyd-Max

- Minimize the mean squared error (MSE) $J = \mathrm{E}\{(u - u_q)^2\}$
- We know the probability distribution (histogram) $p(u)$

$$J = \int \big(u - u_q(u)\big)^2 p(u)\,\mathrm{d}u = \sum_i \int_i^{i+1} (u - r_i)^2 p(u)\,\mathrm{d}u$$

$$u \to u_q : \quad t_k \leq u < t_{k+1} \Rightarrow u_q(u) = r_k$$

Optimal quantization*

Lloyd-Max

- Minimize the mean squared error (MSE) $J = \mathrm{E}\{(u - u_q)^2\}$
- We know the probability distribution (histogram) $p(u)$

$$J = \int \big(u - u_q(u)\big)^2 p(u)\,\mathrm{d}u = \sum_i \int_i^{i+1} (u - r_i)^2 p(u)\,\mathrm{d}u$$

$$u \to u_q : \quad t_k \leq u < t_{k+1} \Rightarrow u_q(u) = r_k$$

- **Optimality conditions:**

$$t_k = (r_k + r_{k+1})/2$$

$$r_k = \mathrm{E}\{u | t_k \leq u < t_{k+1}\} = \frac{\int_{t_k \leq u < t_{k+1}} u\, p(u)\,\mathrm{d}u}{\int_{t_k \leq u < t_{k+1}} p(u)\,\mathrm{d}u}$$

Optimal quantization*

Lloyd-Max

- Minimize the mean squared error (MSE) $J = \mathrm{E}\{(u - u_q)^2\}$
- We know the probability distribution (histogram) $p(u)$

$$J = \int \big(u - u_q(u)\big)^2 p(u)\,\mathrm{d}u = \sum_i \int_i^{i+1} (u - r_i)^2 p(u)\,\mathrm{d}u$$

$$u \to u_q : \quad t_k \le u < t_{k+1} \Rightarrow u_q(u) = r_k$$

- **Optimality conditions:**

$$t_k = (r_k + r_{k+1})/2$$

$$r_k = \mathrm{E}\{u | t_k \le u < t_{k+1}\} = \frac{\int_{t_k \le u < t_{k+1}} u\, p(u)\,\mathrm{d}u}{\int_{t_k \le u < t_{k+1}} p(u)\,\mathrm{d}u}$$

- No closed form solution for general $p(u)$.
  Can be solved iteratively.

Optimal quantization*

Lloyd-Max

- Minimize the mean squared error (MSE) $J = \mathrm{E}\{(u - u_q)^2\}$
- We know the probability distribution (histogram) $p(u)$

$$J = \int \big(u - u_q(u)\big)^2 p(u)\,\mathrm{d}u = \sum_i \int_i^{i+1} (u - r_i)^2 p(u)\,\mathrm{d}u$$

$$u \to u_q : \quad t_k \le u < t_{k+1} \Rightarrow u_q(u) = r_k$$

- **Optimality conditions:**

$$t_k = (r_k + r_{k+1})/2$$

$$r_k = \mathrm{E}\{u | t_k \le u < t_{k+1}\} = \frac{\int_{t_k \le u < t_{k+1}} u\,p(u)\,\mathrm{d}u}{\int_{t_k \le u < t_{k+1}} p(u)\,\mathrm{d}u}$$

- No closed form solution for general $p(u)$.
  Can be solved iteratively.
- Has been precomputed for frequently occurring distributions (normal, Laplace,...)

Entropy

X — random variable with possible values $x_1, x_2, \ldots, x_L$
p_k — probability of a symbol x_k

Information content of x_i is $I_k = -\log_2 p_i$ $[bits]$

Entropy of X is

$$H = \mathrm{E}\big[I_k\big] = -\sum_{k=1}^{L} p_k \log_2 p_k$$

Maximum entropy is $H = \log_2 L$ for $p_k = \frac{1}{L}$

Notes:

- For 8 bit images, $L = 256$ and maximum entropy is 8 bits.
- Probabilities p_k can be estimated from a histogram
- This is Shannon discrete entropy. There are other entropy types (differential, thermodynamic, ...).

Shannon coding theorem

Noiseless

Theorem (informal version)

An i.i.d. source with entropy H bits/symbol can be coded using $H + \epsilon$ bits/symbol on the average as the size of the message tends to infinity, where $\epsilon > 0$ can be arbitrarily small.

Shannon coding theorem

Noiseless

Theorem (informal version)

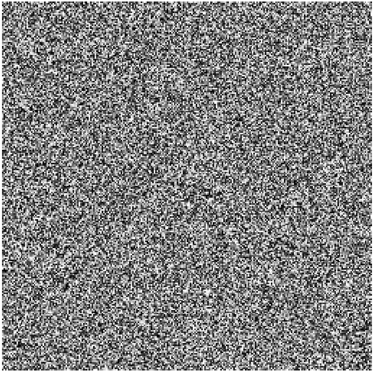
An i.i.d. source with entropy H bits/symbol can be coded using $H + \epsilon$ bits/symbol on the average as the size of the message tends to infinity, where $\epsilon > 0$ can be arbitrarily small.

Notes:

- For images, entropy of the pixel intensities gives an estimate how many bits per pixel we will need.
- Less bits than H might be needed thanks to pixel dependencies.
- More bits might be needed if the coder is suboptimal.
- **Entropy coders** assymptotically approach rate H
  - Huffman coding
  - Arithmetic coding
- Preprocessing the image to decrease H improves compression.

Image entropy examples

Entropy H=8.00



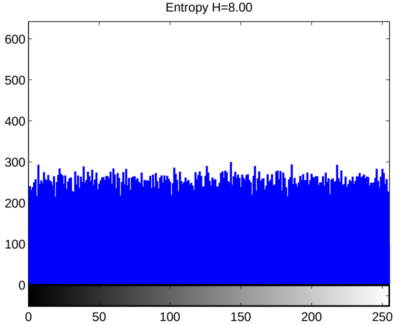
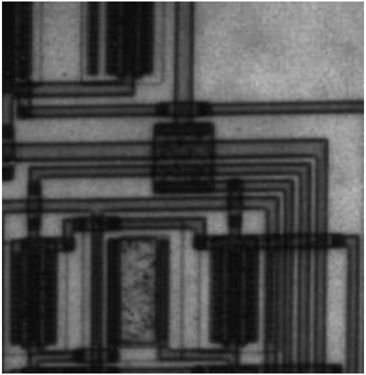


Image entropy examples

Entropy H=6.94



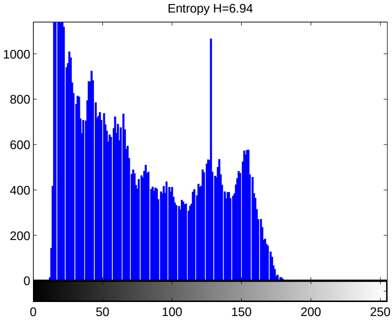


Image entropy examples

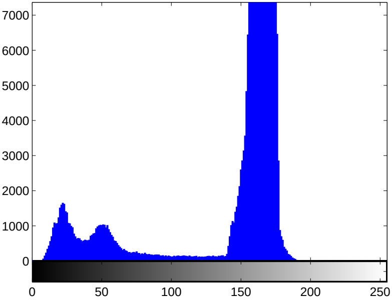
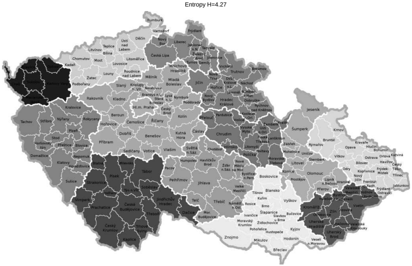


Image entropy examples



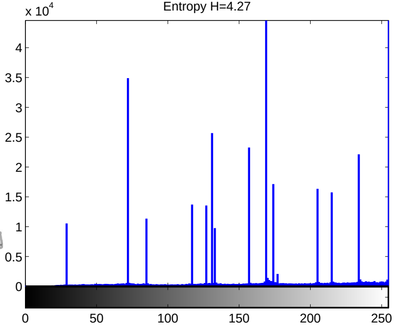
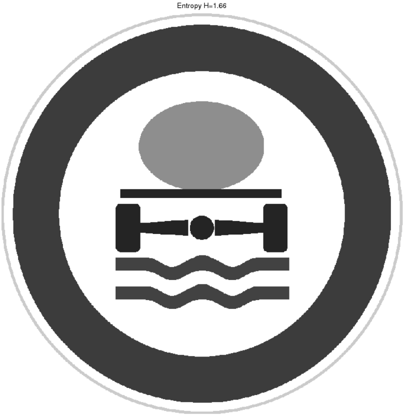


Image entropy examples



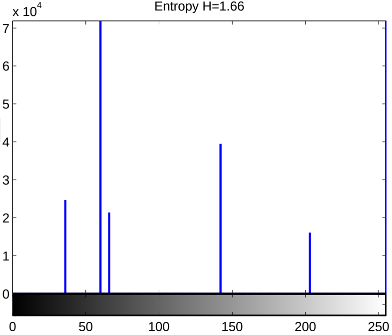
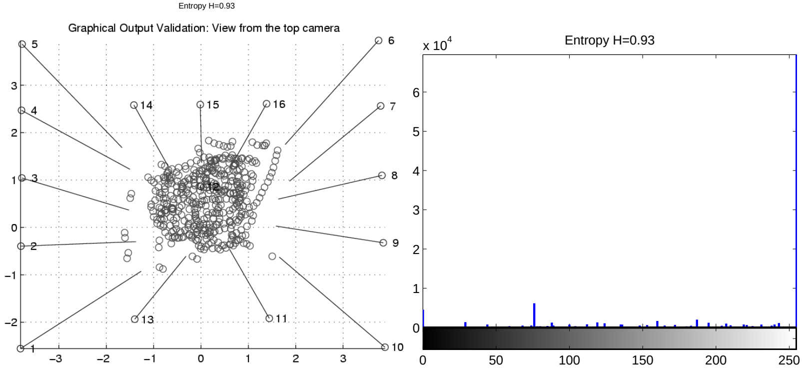


Image entropy examples



Huffman coding*

Input: Symbols $a_1, \ldots, a_L$ with probabilities $p_1, p_2, \ldots, p_L$
Output: Prefix-free binary code $c_1, c_2, \ldots, c_L$ with minimum expected codeword length $\sum_{k=1}^{L} p_k \operatorname{length}(c_k)$.

Algorithm:

- Create a leaf node for each symbol.
- While there are more than two orphan nodes, merge the two orphan nodes with the smallest p_i, p_j, creating a new parent node with probability $p_i + p_j$. The new edges are labeled 0 and 1.
- The code is the sequence of edge labels from root to leafs.

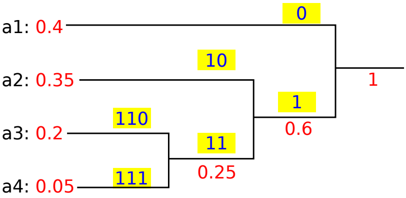
The algorithm uses a priority queue sorted by p. Its complexity is $O(L \log L)$.
If p_k are already sorted, there is an $O(L)$ method using two queues.

Huffman coding example

- Encode symbols a_1, a_2, a_3, a_4 with probabilities $p_1 = 0.4$, $p_2 = 0.35$, $p_3 = 0.2$, $p_4 = 0.05$.
- Naive code **2** bits/symbol.
- Entropy is $H = -\sum_{k=1}^{4} p_k \log_2 p_k \approx \mathbf{1.74}$ bits/symbol

Huffman coding example

- Encode symbols a_1, a_2, a_3, a_4 with probabilities $p_1 = 0.4,\ p_2 = 0.35,\ p_3 = 0.2,\ p_4 = 0.05$.
- Naive code **2** bits/symbol.
- Entropy is $H = -\sum_{k=1}^{4} p_k \log_2 p_k \approx \mathbf{1.74}$ bits/symbol



- The code is $c_1 = 0,\ c_2 = 10,\ c_3 = 110,\ c_4 = 111$.
- Average length is $\sum_{k=1}^{4} p_k \operatorname{length}(c_k)a \approx \mathbf{1.85}$ bits/symbol
- **Decoder:** walk from the root and stop in a leaf.

Arithmetic coding

A quick overview

- Fractional number of bits per symbol.
- Can produce near optimal rates ($\approx H$) for long messages.

Arithmetic coding

A quick overview

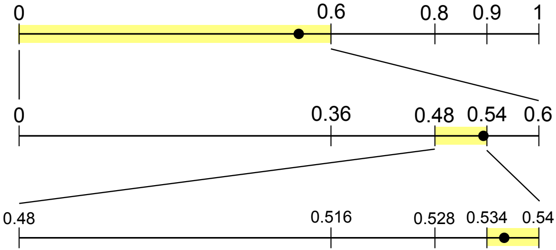
- Fractional number of bits per symbol.
- Can produce near optimal rates ($\approx H$) for long messages.
- Message is a single binary rational number r from $[0, 1)$.

Algorithm (decoding):

- Divide the interval to unequal parts according to p_k.
- The first symbol is the interval r falls in.
- Consider this interval and repeat to obtain further symbols.

Arithmetic coding (2)

Symbols a_1, a_2, a_3, a_4
with probabilities $p_1 = 0.6$, $p_2 = 0.2$, $p_3 = 0.1$, $p_4 = 0.1$
Input is $r = 0.10001010_B \approx 0.5390$



Output is: $a_1 a_3 a_4$.

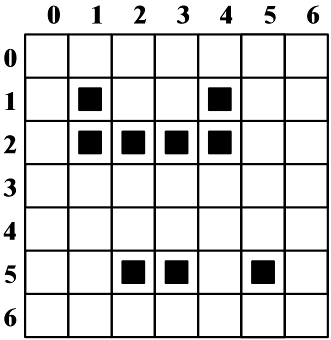
Arithmetic coding (2)

- The shortest possible binary string representing a number falling into the interval is used.
- Message length must be known.
- Coding can be adaptive (on the fly).
- Arithmetic coding is covered by US patents. Affected bzip2 and JPG. Now probably expired.

Dictionary coders

- Lossless
- Not image specific, works for any (1D) sequence of bytes
- Based on detecting and efficiently encoding *repeated sequences*

Run length encoding
(RLE)



Bit string:

0000000010010001111000000000000000000110100000000

Can be coded as:

$(7\times\mathbf{0})$ $(1\times\mathbf{1})$ $(2\times\mathbf{0})$ $(1\times\mathbf{0})$ $(3\times\mathbf{0})$ $(4\times\mathbf{1})$ $(18\times\mathbf{0})$ $(2\times\mathbf{1})$
$(1\times\mathbf{0})$ $(1\times\mathbf{1})$ $(8\times\mathbf{0})$

Run length encoding (2)

Encoding runs into a bit stream

000000001001000111100000000000000000000110100000000

$(7\times\mathbf{0})$ $(1\times\mathbf{1})$ $(2\times\mathbf{0})$ $(1\times\mathbf{0})$ $(3\times\mathbf{0})$ $(4\times\mathbf{1})$ $(18\times\mathbf{0})$ $(2\times\mathbf{1})$
$(1\times\mathbf{0})$ $(1\times\mathbf{1})$ $(8\times\mathbf{0})$

- One-bit flag to signal repetitions:
  $0\,1\,\boxed{7},1\,0\,,0\,1\,\boxed{2},0\,0\,,0\,1\,\boxed{3},1\,1\,\boxed{4},0\,1\,\boxed{18},1\,1\,\boxed{2},0\,0\,,1\,0\,,0\,1\,\boxed{8}$

Run length encoding (2)

Encoding runs into a bit stream

00000001001000111100000000000000000011010000000

$(7\times\mathbf{0})$ $(1\times\mathbf{1})$ $(2\times\mathbf{0})$ $(1\times\mathbf{0})$ $(3\times\mathbf{0})$ $(4\times\mathbf{1})$ $(18\times\mathbf{0})$ $(2\times\mathbf{1})$
$(1\times\mathbf{0})$ $(1\times\mathbf{1})$ $(8\times\mathbf{0})$

- One-bit flag to signal repetitions:
  0 1 $\boxed{7}$,1 0 ,0 1 $\boxed{2}$,0 0 ,0 1 $\boxed{3}$,1 1 $\boxed{4}$,0 1 $\boxed{18}$,1 1 $\boxed{2}$,0 0 ,1 0 ,0 1 $\boxed{8}$
- Two repeated symbols signal a run:
  00 $\boxed{7}$ 100 $\boxed{2}$ 100 $\boxed{3}$ 11 $\boxed{4}$ 00 $\boxed{18}$ 11 $\boxed{2}$ 0100 $\boxed{8}$

Run length encoding (2)

Encoding runs into a bit stream

0000000010010001111000000000000000000110100000000

$(7\times\mathbf{0})\ (1\times\mathbf{1})\ (2\times\mathbf{0})\ (1\times\mathbf{0})\ (3\times\mathbf{0})\ (4\times\mathbf{1})\ (18\times\mathbf{0})\ (2\times\mathbf{1})$
$(1\times\mathbf{0})\ (1\times\mathbf{1})\ (8\times\mathbf{0})$

- One-bit flag to signal repetitions:
  $0\,1\,\boxed{7}\,,1\,0\,,0\,1\,\boxed{2}\,,0\,0\,,0\,1\,\boxed{3}\,,1\,1\,\boxed{4}\,,0\,1\,\boxed{18}\,,1\,1\,\boxed{2}\,,0\,0\,,1\,0\,,0\,1\,\boxed{8}$
- Two repeated symbols signal a run:
  $00\boxed{7}100\boxed{2}100\boxed{3}11\boxed{4}00\boxed{18}11\boxed{2}0100\boxed{8}$
- Code run length using a variable-length prefix code (Huffman)
  This is used for fax transmission, (CCITT Group 3).

Run length encoding (2)

Encoding runs into a bit stream

000000010010001111000000000000000000110100000000

$(7\times\mathbf{0})$ $(1\times\mathbf{1})$ $(2\times\mathbf{0})$ $(1\times\mathbf{0})$ $(3\times\mathbf{0})$ $(4\times\mathbf{1})$ $(18\times\mathbf{0})$ $(2\times\mathbf{1})$
$(1\times\mathbf{0})$ $(1\times\mathbf{1})$ $(8\times\mathbf{0})$

- One-bit flag to signal repetitions:
  0 1 [7] ,1 0 ,0 1 [2] ,0 0 ,0 1 [3] ,1 1 [4] ,0 1 [18] ,1 1 [2] ,0 0 ,1 0 ,0 1 [8]
- Two repeated symbols signal a run:
  00 [7] 100 [2] 100 [3] 11 [4] 00 [18] 11 [2] 0100 [8]
- Code run length using a variable-length prefix code (Huffman)
  This is used for fax transmission, (CCITT Group 3).

Note: RLE can work with larger set of symbols (e.g. bytes).

Lempel-Ziv coding

(Abraham Lempel, Jacob Ziv, 1977, 1978.)

- Often used general purpose compression methods (gzip, zlib, zip, compress, gif, pdf. . .)
- Take advantage of repeated sequences of symbols.
- Generalization of RLE (repeated sequences of length 1)
- **LZ77** — a "sliding window" algorithm
- Replace already seen subsequences by a *length-distance pair*, referring to past data in the sliding window.
- codeword = literal symbol *or* (length,distance)

LZ77 example

Output	History	Lookahead
		SIDVICIIISIDIDVI
S	S	IDVICIIISIDIDVI
I	SI	DVICIIISIDIDVI
D	SID	VICIIISIDIDVI
V	SIDV	ICIIISIDIDVI
I	SIDVI	CIIISIDIDVI
C	SIDVIC	IIISIDIDVI
I	SIDVICI	IISIDIDVI
I	SIDVICII	ISIDIDVI
I	**SID**VICIII	**SID**IDVI
	9	
(9,3)	SIDVICIIIS **ID**	**ID**VI
	2	
(2,2)	SID **VI**CIIISIDID	**VI**
	11	
(11,2)	SIDVICIIISIDIDVI	

LZ77 example

Input: `SIDVICIIISIDIDVI`

Output: `S I D V I C I I I` (9,3) (2,2) (11,2)

LZ77 example

Decoding:

History	Input
	S
S	I
SI	D
SID	V
SIDV	I
SIDVI	C
SIDVIC	I
SIDVICI	I
SIDVICII	I
SIDVICIII	$(9,3) \longrightarrow$ SID
SIDVICIIIS**ID**	$(2,2) \longrightarrow$ ID
SID**VI**CIIISIDID	$(11,2) \longrightarrow$ VI
SIDVICIIISIDIDVI	

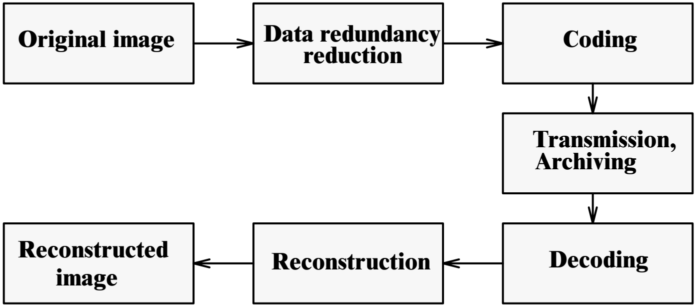
DEFLATE algorithm

- DEFLATE = LZ77 + Huffman coding
- Used in PKZIP, zlib, 7-zip, PNG
- LZ77: sliding window 32kB, match length 3–258 bytes
- Huffman coding: efficiently encode LZ77 output. 288 basic symbols = 256 literal bytes + 32 match length codes (indicating the number of extra bits to describe distance)

LZ78 and LZW

- maintaining a dictionary of frequently occurring substrings
- **LZW** (Lempel, Ziv, Welch) is an improvement of **LZ78**
- **encoding** = find the longest input string matching a dictionary entry, output its index and the unmatched character. Create a new dictionary entry.
- codeword = index to dictionary + next non-matching symbol
- in LZW originally 12 bit codewords (256 literal bytes+dictionary indices), clear table code.
- variable width code
- similar performance as DEFLATE, slightly better than raw LZ77, patent protected

Image compression flowchart

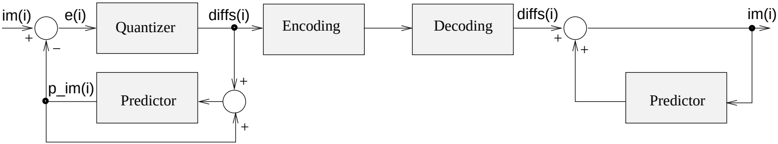


Predictive coding

Ideas:

- Preprocess image to reduce its entropy.
- Reduce redundancy between neighboring pixels
- Predict pixel values from previously encoded values.
- Code the prediction error.
- Entropy coding *and/or* quantization (lossy)
- Causal ordering

DPCM



- Feedback prediction = quantizer is in the encoding loop.
- Predictors in encoder and decoder work on the same inputs.
- Avoids error accumulation, better results than feedforward coding.
- Encoder can compensate prediction error due to quantization.

Portable Network Graphics

PNG file format

- popular lossless, license-less bitmap image format
- supports grayscale, color, indexed, $1 \sim 32$ bits/pixel
- no quantization
- PNG = prediction + DEFLATE

Predictor

Predict value $f(i,j)$ based on its neighbors $A = f(i, j-1)$, $B = f(i-1, j)$, $C = f(i-1, j-1)$.

None	0
Sub	A
Up	B
Average	$(A+B)/2$
Paeth	the closest of A,B,C from $A+B-C$

Each line can use a different predictor.

Optimal linear predictor*

Find the best linear predictor for 2×2 neighborhood

$$\begin{aligned}\hat{f}(i,j) &= a_1 f(i,j-1) + a_2 f(i-1,j-1) + a_3 f(i-1,j) \\ &= \mathbf{a}^T \mathbf{f}(i,j)\end{aligned}$$

$$\text{with} \quad \mathbf{a} = [a_1 \; a_2 \; a_3]^T$$
$$\mathbf{f}(i,j) = [f(i,j-1) \; f(i-1,j-1) \; f(i-1,j)]^T$$

minimizing the expected quadratic prediction error

$$J = \mathrm{E}\big[(\hat{f}(i,j) - f(i,j))^2\big]$$

assumming a stationary zero mean f with known covariance

$$R(k,l) = \mathrm{E}\big[f(i,j) f(i-k, j-l)\big]$$

Optimal linear predictor*

We write (omitting (i,j) for simplicity)

$$\begin{aligned} J &= \mathrm{E}\big[\big(\mathbf{a}^T\mathbf{f} - f\big)^2\big] = \mathrm{E}\big[\mathbf{a}^T\mathbf{f}\,\mathbf{f}^T\mathbf{a} + f^2 - 2f\mathbf{a}^T\mathbf{f}\big] \\ &= \mathbf{a}^T\mathbf{C}\mathbf{a} - 2\mathbf{a}^T\mathbf{b} + R(0,0) \end{aligned}$$

$$\text{with} \quad \mathbf{C} = \mathrm{E}\big[\mathbf{f}\,\mathbf{f}^T\big] = \begin{bmatrix} R(0,0) & R(0,1) & R(1,1) \\ R(0,1) & R(0,0) & R(1,0) \\ R(1,1) & R(1,0) & R(0,0) \end{bmatrix}$$

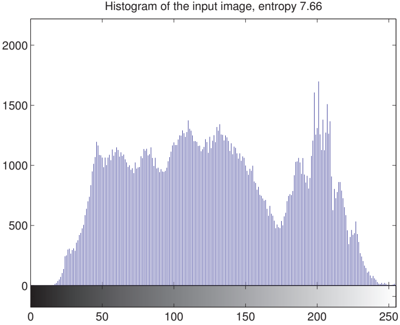
$$\mathbf{b} = \mathrm{E}\big[f\mathbf{f}\big] = \begin{bmatrix} R(1,0) & R(1,1) & R(0,1) \end{bmatrix}^T$$

The optimal **a** must satisfy

$$\mathbf{b} = \mathbf{C}\,\mathbf{a}$$

Predictive compression example

Input image



Predictive compression example

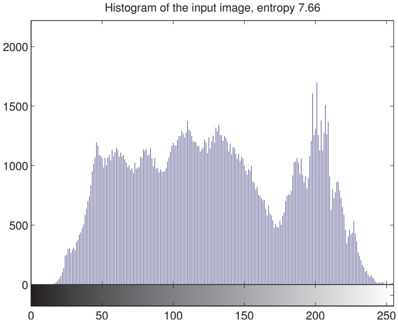
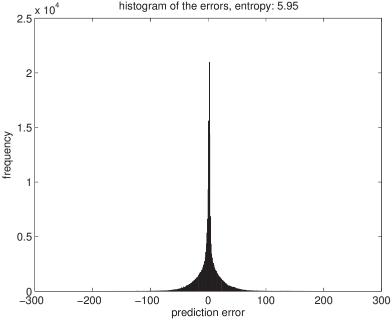


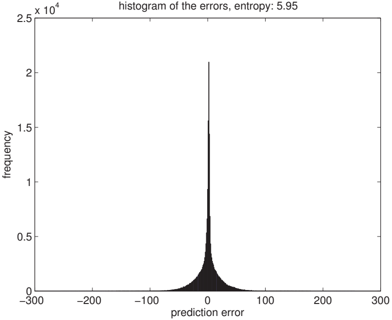
Image histogram



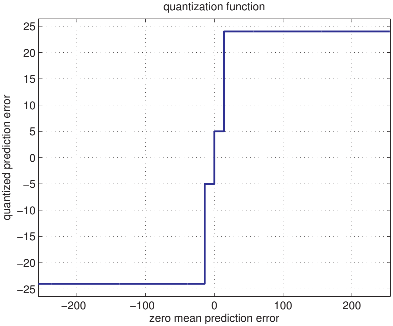
Prediction error histogram

Linear 1D predictor (columnwise) of order 3.

Predictive compression example

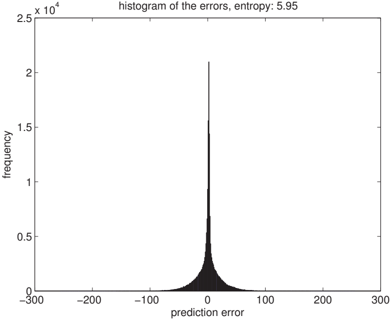


Prediction error histogram

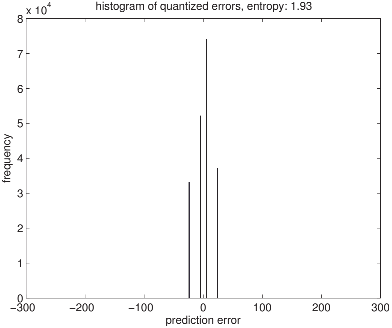


Quantization function, Lloyd-Max

Predictive compression example



Prediction error histogram



Quantized prediction error histogram

Predictive compression example

Input image

Original image

Predictive compression example

Reconstructed image
Compressed to 25% of the original size.

Transform coding

- The basis of most successful lossy image compression methods

Overview

- Apply an *energy compaction transform* to an image or a large block.
- Quantize the coefficients (lossy step)
- Apply entropy coding

Energy compaction transform

- An invertible linear transform, typically unitary.
- To get a few large coefficients and many small ones (for typical images).

Principal component analysis

Discrete Karhunen-Loève transform

Vector random variable X with $\mathrm{E}\big[\mathbf{x}\big] = \mathbf{0}$ and $\mathbf{R} = \mathrm{E}\big[\mathbf{x}\mathbf{x}^T\big]$

Eigenvalue decomposition

$$\mathbf{R}\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \text{or} \quad \mathbf{R}\mathbf{V} = \mathbf{V}\Lambda \quad \text{with} \quad \mathbf{V} = [\mathbf{v}_1 \ldots \mathbf{v}_N]$$

Karhunen-Loève transform:

$$\mathbf{y} = \mathbf{V}^T \mathbf{x}$$

Properties

- Diagonalization: $\mathrm{E}\big[\mathbf{y}\mathbf{y}^T\big] = \mathbf{V}^T\mathbf{R}\mathbf{V} = \mathbf{V}^T\mathbf{V}\Lambda = \Lambda$
- Variance maximization: If $\lambda_1 > \ldots \lambda_N$, then $\mathrm{E}\Big[\big(\sum_{k=1}^{M} y_k\big)^2\Big] = \sum_{k=1}^{M} \lambda_k^2$ is maximized over $\mathbf{V}$

Principal component analysis

Discrete Karhunen-Loève transform

Disadvantages:

- Data dependent transform (must be transmitted with the data)
- No fast algorithm.
- $\mathbf{R}$ must be estimated from the data.

Discrete cosine transform
DCT

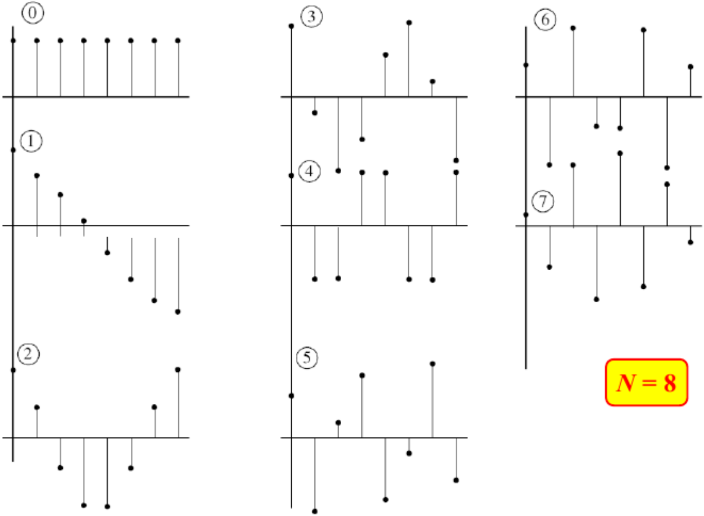
- Like Fourier transform but real numbers only.
- Fast DCT exists.
- DCT approximately diagonalizes an AR(1) process $x_{t+1} = \rho x_t + e_t$ as $\rho \to 1$

Definition: (DCT type II)

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1$$

$$X_k = \text{DCT}\left(x_n\right)$$

Discrete cosine transform
DCT
1D DCT basis functions



2D DCT

2D DCT is separable:

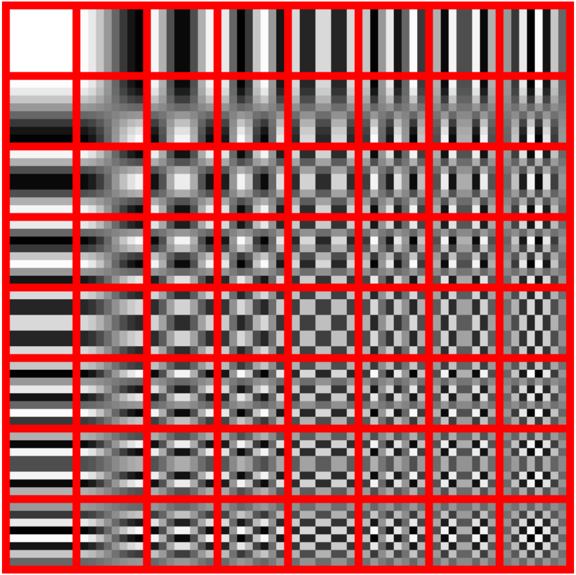
$$X_{jk} = \mathrm{DCT}_{mn}(x_{mn}) = \mathrm{DCT}_m\big(\mathrm{DCT}_n(x_{mn})\big)$$

Basis functions are:

$$\varphi_{jk} = \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right) j\right] \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right) k\right]$$

2D DCT

Basis functions are:



JPEG compression*

1992

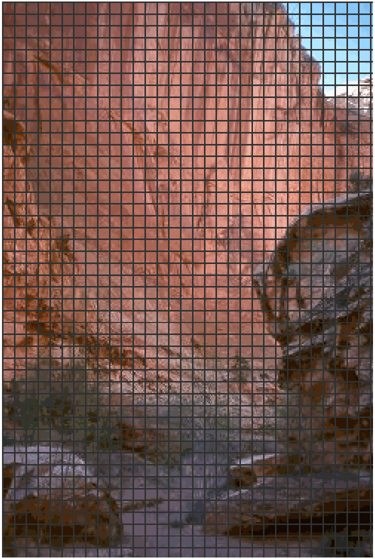
- *Joint Photographic Experts Group*,
  JPEG File Interchange Format
- Most widely used lossy still image format.
- Good compression, especially for natural images (photographs).
- Mostly for 8 bit RGB images.
- Can embed metadata (EXIF).

Compression overview

- Color transformation and subsampling.
- DCT of 8×8 blocks.
- Coefficient quantization (lossy step).
- DPCM on DC components.
- Run length encoding.
- Huffman encoding.

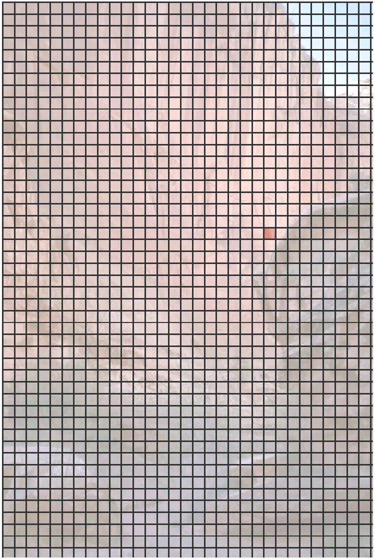
JPEG: Step 1 — Divide to blocks

JPEG: Step 1 — Divide to blocks



Usually 8×8, can be 16×16

JPEG: Step 1 — Divide to blocks



JPEG: Step 2 — Color conversion

Optional

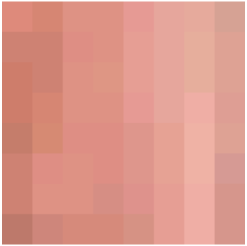
- Convert RGB to *luminance* and *chrominance*

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.168736 & -0.331264 & 0.5 \\ 0.5 & 0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Human vision less sensitive to chrominance
- Chrominance often more slowly varying
- Chrominance is downsampled by 2 (optional)

JPEG: Step 2 — Color conversion

Optional



original patch

Y

C_b

C_r

JPEG: Step 3 — DCT

image block

image

image intensities

	1	2	3	4	5	6	7	8
1	185	187	184	183	189	186	185	186
2	185	184	186	190	187	186	189	191
3	186	187	187	188	190	185	189	191
4	186	189	189	189	193	193	193	195
5	185	190	188	193	199	198	189	184
6	191	187	162	156	116	30	15	14
7	168	102	49	22	15	11	10	10
8	25	19	19	26	17	11	10	10

block intensities

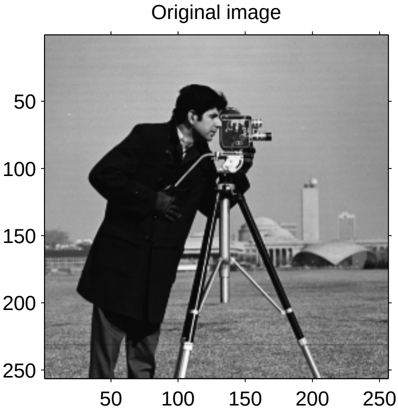
coefficients of the DCT2

	1	2	3	4	5	6	7	8
1	1117	114	10	7	19	–2	–7	2
2	459	–119	–20	–11	–16	–4	3	0
3	–267	–3	24	8	1	6	4	–1
4	50	107	–9	–1	11	–6	–7	3
5	52	–111	–22	–2	–16	–2	5	–3
6	–38	39	46	19	2	0	4	3
7	–17	39	–46	–26	8	–5	–10	2
8	30	–46	28	22	–9	2	7	–1

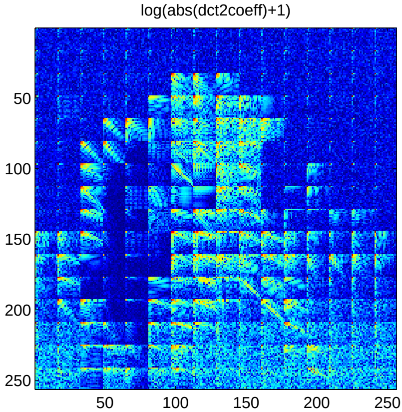
DCT coefficients

- Values shifted to $[-128, 127]$
- Energy compaction $=$ Low frequency coefficients predominate.

JPEG: Step 3 — DCT



image



DCT coefficients

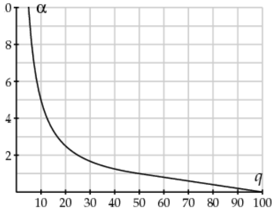
- Values shifted to $[-128, 127]$
- Energy compaction = Low frequency coefficients predominate.

JPEG: Step 4 — Quantization

of DCT coefficients

- Quality factor $q \in [1, 100]$
- Calculate α

$$\alpha = \begin{cases} 50/q & \text{if } 1 \leq q \leq 50 \\ 2 - q/50 & \text{if } 50 \leq q \leq 100 \end{cases}$$

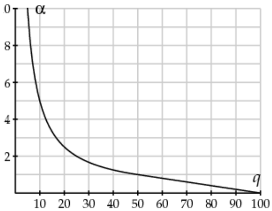


JPEG: Step 4 — Quantization

of DCT coefficients

- Quality factor $q \in [1, 100]$
- Calculate α

$$\alpha = \begin{cases} 50/q & \text{if } 1 \leq q \leq 50 \\ 2 - q/50 & \text{if } 50 \leq q \leq 100 \end{cases}$$



- Truncate the coefficients

$$F'_{jk} = \text{round}\left(\frac{F_{jk}}{\alpha Q_{jk}}\right)$$

JPEG: Step 4 — Quantization

of DCT coefficients

- Truncate the coefficients

$$F'_{jk} = \text{round}\left(\frac{F_{jk}}{\alpha Q_{jk}}\right)$$

- Matrix **Q** is recommended by the standard. For luminance:

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

It can be different for chrominance.

JPEG: Step 4 — Quantization

of DCT coefficients

DCT2 coefficients

	−275	663	231	77	−34	−54	−37	−16	14	2	2	−2	−6	7	5	4
2	882	−343	−301	−116	−5	55	51	22	7	−11	−2	1	−1	−1	−12	−11
	154	−294	70	101	77	13	−28	−32	−28	−8	−2	1	9	8	9	6
4	−90	16	160	−19	−77	−77	−21	33	38	39	5	−15	−13	−13	0	5
	−153	101	−1	−43	25	56	43	−2	−41	−36	−16	10	19	8	2	−4
6	4	137	−103	−39	−3	−4	−11	−6	18	14	16	8	−8	−6	−7	−3
	−1	−35	−64	77	39	2	−15	−6	13	−7	−8	−7	−6	9	7	9
8	29	−61	46	44	−34	−29	−10	1	0	2	11	10	7	−2	−11	−12
	34	−36	56	−50	−40	17	34	24	−8	−4	−9	−14	−5	1	9	4
10	−5	−13	21	−33	33	19	−20	−22	−24	7	16	12	9	−5	−3	0
	−16	28	−25	−8	28	−13	−18	15	20	9	−2	−10	−8	−9	−3	2
12	−22	32	−38	27	−5	−16	7	10	5	−21	−13	2	9	15	3	0
	−9	12	−15	29	−30	2	22	−10	0	−1	11	9	−6	−4	−15	−7
14	−1	−14	9	−1	−12	17	7	−13	−6	3	2	−9	−12	5	22	18
	14	−23	24	−20	18	−3	−21	5	−1	10	7	2	0	−12	−9	−8
16	20	−13	10	−27	16	−9	−9	25	−2	−11	−15	−2	15	5	−3	−2
		2		4		6		8		10		12		14		16

DCT coefficients (16×16)

Quantization matrix

	10	10	10	10	50	50	50	50	100	100	100	100	100	100	100	100
2	10	10	10	10	50	50	50	50	100	100	100	100	100	100	100	100
	10	10	10	10	50	50	50	50	100	100	100	100	100	100	100	100
4	10	10	10	10	50	50	50	50	100	100	100	100	100	100	100	100
	50	50	50	50	50	50	50	50	100	100	100	100	100	100	100	100
6	50	50	50	50	50	50	50	50	100	100	100	100	100	100	100	100
	50	50	50	50	50	50	50	50	100	100	100	100	100	100	100	100
8	50	50	50	50	50	50	50	50	100	100	100	100	100	100	100	100
	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
12	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
14	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
16	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		2		4		6		8		10		12		14		16

Quantization matrix αQ_{jk}

JPEG: Step 4 — Quantization

of DCT coefficients

DCT2 coefficients

		2		4		6		8		10		12		14		16
	−275	663	231	77	−34	−54	−37	−16	14	2	2	−2	−6	7	5	4
2	882	−343	−301	−116	−5	55	51	22	7	−11	−2	1	−1	−1	−12	−11
	154	−294	70	101	77	13	−28	−32	−28	−8	−2	1	9	8	9	6
4	−90	16	160	−19	−77	−77	−21	33	38	39	5	−15	−13	−13	0	5
	−153	101	−1	−43	25	56	43	−2	−41	−36	−16	10	19	8	2	−4
6	4	137	−103	−39	−3	−4	−11	−6	18	14	16	8	−8	−6	−7	−3
	−1	−35	−64	77	39	2	−15	−6	13	−7	−8	−7	−6	9	7	9
8	29	−61	46	44	−34	−29	−10	1	0	2	11	10	7	−2	−11	−12
	34	−36	56	−50	−40	17	34	24	−8	−4	−9	−14	−5	1	9	4
10	−5	−13	21	−33	33	19	−20	−22	−24	7	16	12	9	−5	−3	0
	−16	28	−25	−8	28	−13	−18	15	20	9	−2	−10	−8	−9	−3	2
12	−22	32	−38	27	−5	−16	7	10	5	−21	−13	2	9	15	3	0
	−9	12	−15	29	−30	2	22	−10	0	−1	11	9	−6	−4	−15	−7
14	−1	−14	9	−1	−12	17	7	−13	−6	3	2	−9	−12	5	22	18
	14	−23	24	−20	18	−3	−21	5	−1	10	7	2	0	−12	−9	−8
16	20	−13	10	−27	16	−9	−9	25	−2	−11	−15	−2	15	5	−3	−2

DCT coefficients

Quantized DCT2 coefficients

		2		4		6		8		10		12		14		16
	−28	66	23	8	−1	−1	−1	0	0	0	0	0	0	0	0	0
2	88	−34	−30	−12	0	1	1	0	0	0	0	0	0	0	0	0
	15	−29	7	10	2	0	−1	−1	0	0	0	0	0	0	0	0
4	−9	2	16	−2	−2	−2	0	1	0	0	0	0	0	0	0	0
	−3	2	0	−1	1	1	1	0	0	0	0	0	0	0	0	0
6	0	3	−2	−1	0	0	0	0	0	0	0	0	0	0	0	0
	0	−1	−1	2	1	0	0	0	0	0	0	0	0	0	0	0
8	1	−1	1	1	−1	−1	0	0	0	0	0	0	0	0	0	0
	0	0	1	−1	0	0	0	0	0	0	0	0	0	0	0	0
10	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
12	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
14	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
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Quantized coefficients F'_{jk}

JPEG: Step 4 — Quantization

of DCT coefficients

DCT2 coefficients

		2		4		6		8		10		12		14		16
	−275	663	231	77	−34	−54	−37	−16	14	2	2	−2	−6	7	5	4
2	882	−343	−301	−116	−5	55	51	22	7	−11	−2	1	−1	−1	−12	−11
	154	−294	70	101	77	13	−28	−32	−28	−8	−2	1	9	8	9	6
4	−90	16	160	−19	−77	−77	−21	33	38	39	5	−15	−13	−13	0	5
	−153	101	−1	−43	25	56	43	−2	−41	−36	−16	10	19	8	2	−4
6	4	137	−103	−39	−3	−4	−11	−6	18	14	16	8	−8	−6	−7	−3
	−1	−35	−64	77	39	2	−15	−6	13	−7	−8	−7	−6	9	7	9
8	29	−61	46	44	−34	−29	−10	1	0	2	11	10	7	−2	−11	−12
	34	−36	56	−50	−40	17	34	24	−8	−4	−9	−14	−5	1	9	4
10	−5	−13	21	−33	33	19	−20	−22	−24	7	16	12	9	−5	−3	0
	−16	28	−25	−8	28	−13	−18	15	20	9	−2	−10	−8	−9	−3	2
12	−22	32	−38	27	−5	−16	7	10	5	−21	−13	2	9	15	3	0
	−9	12	−15	29	−30	2	22	−10	0	−1	11	9	−6	−4	−15	−7
14	−1	−14	9	−1	−12	17	7	−13	−6	3	2	−9	−12	5	22	18
	14	−23	24	−20	18	−3	−21	5	−1	10	7	2	0	−12	−9	−8
16	20	−13	10	−27	16	−9	−9	25	−2	−11	−15	−2	15	5	−3	−2

DCT coefficients

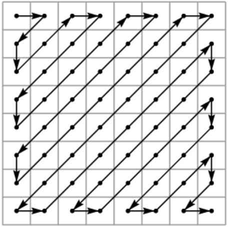
dct2 coefficients from the quantized data

		2		4		6		8		10		12		14		16
	−280	660	230	80	−50	−50	−50	0	0	0	0	0	0	0	0	0
2	880	−340	−300	−120	0	50	50	0	0	0	0	0	0	0	0	0
	150	−290	70	100	100	0	−50	−50	0	0	0	0	0	0	0	0
4	−90	20	160	−20	−100	−100	0	50	0	0	0	0	0	0	0	0
	−150	100	0	−50	50	50	50	0	0	0	0	0	0	0	0	0
6	0	150	−100	−50	0	0	0	0	0	0	0	0	0	0	0	0
	0	−50	−50	100	50	0	0	0	0	0	0	0	0	0	0	0
8	50	−50	50	50	−50	−50	0	0	0	0	0	0	0	0	0	0
	0	0	100	−100	0	0	0	0	0	0	0	0	0	0	0	0
10	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
12	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
14	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
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reconstructed coefficients $\alpha F'_{jk} Q_{jk}$

JPEG: Step 4 — Entropy coding

- Rearranging to coefficients (low to high frequencies)



- Predictive coding of F'_{00} (from previous block)
- Run length encoding (to eliminate runs of zeros)
- Huffman coding (or arithmetic coding)

JPEG example — photograph

Quality $q = 90$.

JPEG example — photograph

Quality $q = 70$.

JPEG example — photograph

Quality $q = 30$.

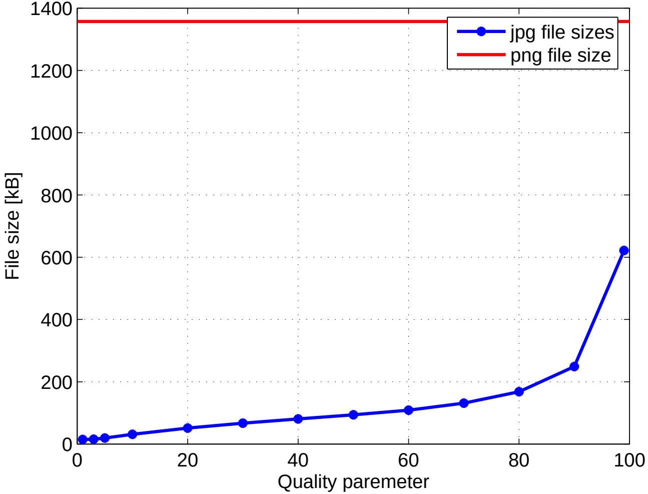
JPEG example — photograph

Quality $q = 10$.

JPEG example — photograph

Quality $q = 5$.

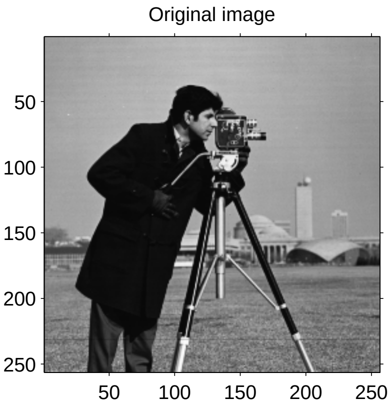
JPEG example — photograph



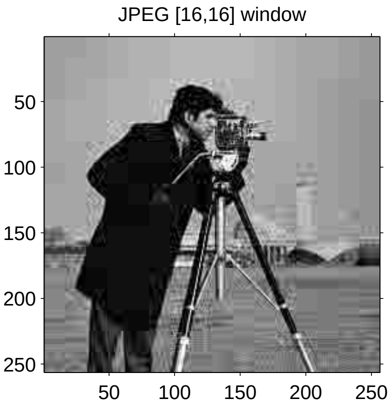
JPG versus PNG. File sizes.

JPEG disadvantages

- Blocking artifacts and ringing at low bitrates



original



reconstruction

JPEG disadvantages

- Blocking artifacts and ringing at low bitrates
- Loss of quality by each compression/decompression
- Low bit depth only (typically only 8 bits/channel)
- Unsuitable for line graphics, graphs, text, cartoons, ...

JPEG 2000

successor of JPEG, jp2, jpx

- Slightly better compression ($\sim 20\,\%$)
- Less visible artifacts (ringing) and (almost) no blocking
- Uses wavelet transform
- Multiple resolutions, progressive transmission
- Lossless and lossy modes
- Random access to region of interests
- Spatially varying compression
- Error resilience
- Flexible file format — metadata support, color space, interactivity, transparency and alpha channels...
- Licensed but free of charge

Which format to use

Buyer's guide

- **Lossy encoding** (JPEG,JPEG2000,PGF. . .)
  - Good compression, limited scalability
  - For photographs, natural images
- **Lossless encoding** (PNG,TIFF,. . .)
  - Medium compression, limited scalability
  - For computer generated images (graphs, logos, cartoon)
  - Scanned text and graphics, screen shots
  - Precious data (medical, astronomical)
  - Intermediary format for image editing
- **Vector formats** (EPS,PDF,SVG,AI,. . .)
  - For computer generated images with geometrical objects and text (graphs, logos, cartoons)
  - Unlimited scalability
  - Excellent compressibility

Conclusions

What to take home

- General compression methods
  - Run length encoding
  - Lempel-Ziv compression algorithms
  - Entropy coding — Huffman coding, arithmetic coding
  - Quantization
- Image specific methods
  - Predictive coding
  - Transform coding (KLT, DCT)
- Standard file formats (PNG,JPG)

Conclusions

What to take home

- General compression methods
  - Run length encoding
  - Lempel-Ziv compression algorithms
  - Entropy coding — Huffman coding, arithmetic coding
  - Quantization
- Image specific methods
  - Predictive coding
  - Transform coding (KLT, DCT)
- Standard file formats (PNG,JPG)
- There is more to be learned
  - Information theory, approximation theory, time-frequency analysis...
  - Video coding