Information theory and coding – Image, video and audio compression

Markus Kuhn

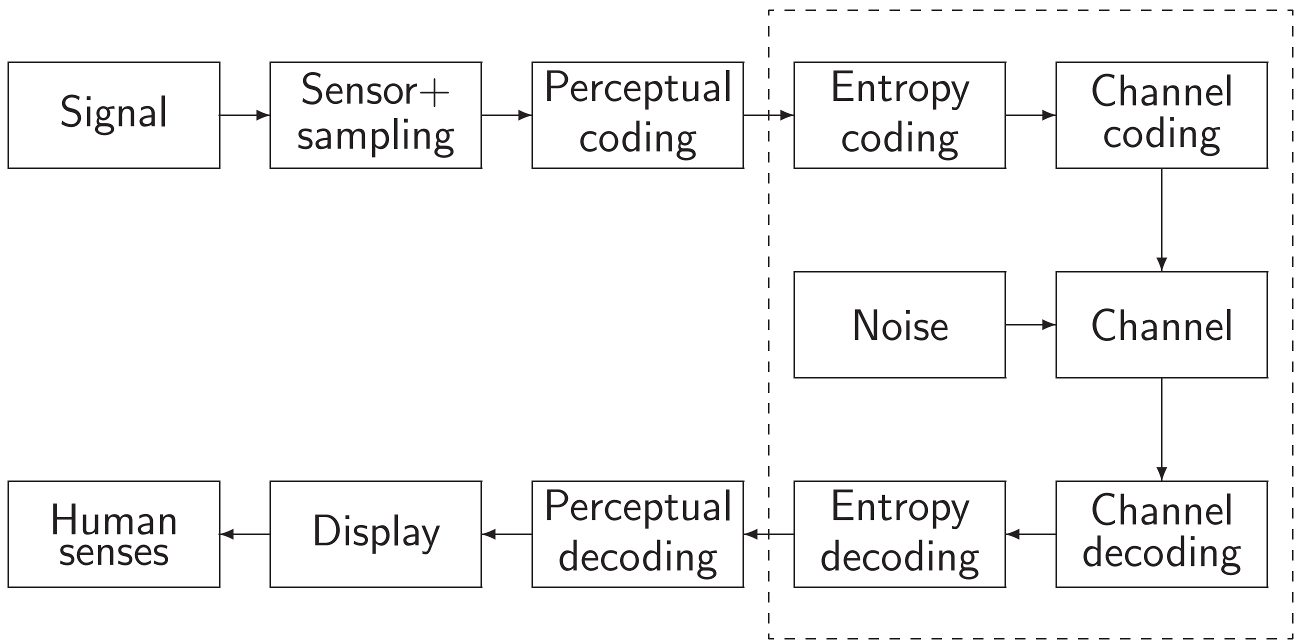
UNIVERSITY OF CAMBRIDGE

Computer Laboratory

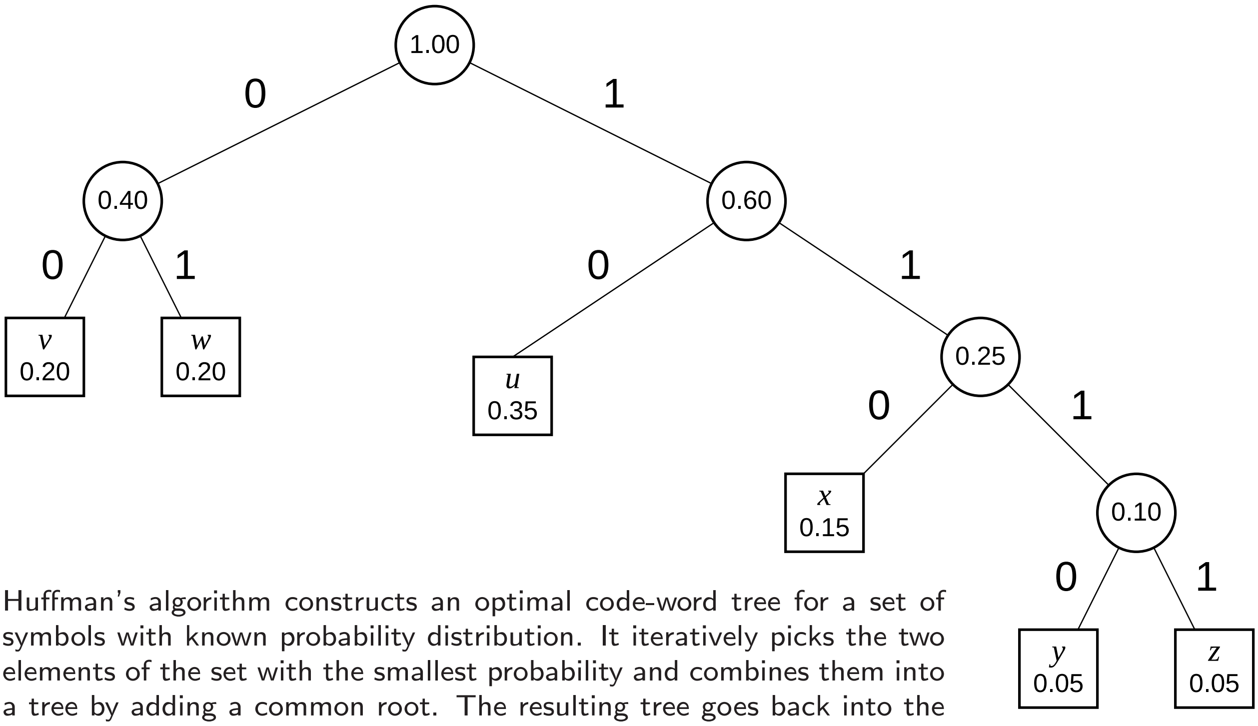
http://www.cl.cam.ac.uk/Teaching/2003/InfoTheory/mgk/

Michaelmas 2003 – Part II

Structure of modern audiovisual communication systems



Entropy coding review – Huffman



Huffman's algorithm constructs an optimal code-word tree for a set of symbols with known probability distribution. It iteratively picks the two elements of the set with the smallest probability and combines them into a tree by adding a common root. The resulting tree goes back into the set, labeled with the sum of the probabilities of the elements it combines. The algorithm terminates when less than two elements are left.

Other variable-length code tables

Huffman's algorithm generates an optimal code table.
Disadvantage: this code table (or the distribution from which is was generated) needs to be stored or transmitted.

Adaptive variants of Huffman's algorithm modify the coding tree in the encoder and decoder synchronously, based on the distribution of symbols encountered so far. This enables one-pass processing and avoids the need to transmit or store a code table, at the cost of starting with a less efficient encoding.

Unary code

Encode the natural number n as the bit string 1^n0. This code is optimal when the probability distribution is $p(n) = 2^{-(n+1)}$.

Example: $3, 2, 0 \rightarrow 1110, 110, 0$

Golomb code

Select an encoding parameter b. Let n be the natural number to be encoded, $q = \lfloor n/b \rfloor$ and $r = n - qb$. Encode n as the unary code word for q, followed by the $(\log_2 b)$-bit binary code word for r.

Where b is not a power of 2, encode the lower values of r in $\lfloor \log_2 b \rfloor$ bits, and the rest in $\lceil \log_2 b \rceil$ bits, such that the leading digits distinguish the two cases.

Examples:

$b = 1$: 0, 10, 110, 1110, 11110, 111110, ... (this is just the unary code)
$b = 2$: 00, 01, 100, 101, 1100, 1101, 11100, 11101, 111100, 111101, ...
$b = 3$: 00, 010, 011, 100, 1010, 1011, 1100, 11010, 11011, 11100, 111010, ...
$b = 4$: 000, 001, 010, 011, 1000, 1001, 1010, 1011, 11000, 11001, 11010, ...

Golomb codes are optimal for geometric distributions of the form $p(n) = u^n(u-1)$ (e.g., run lengths of Bernoulli experiments) if b is chosen suitably for a given u.

S.W. Golomb: Run-length encodings. IEEE Transactions on Information Theory, IT-12(3):399–401, July 1966.

Elias gamma code

Start the code word for the positive integer n with a unary-encoded length indicator $m = \lfloor \log_2 n \rfloor$. Then append from the binary notation of n the rightmost m digits (to cut off the leading 1).

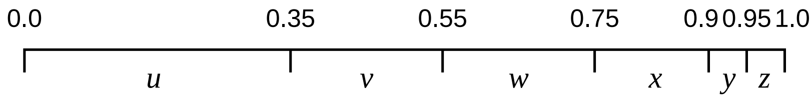
1	=	0	4	=	11000	7	=	11011	10	=	1110010
2	=	100	5	=	11001	8	=	1110000	11	=	1110011
3	=	101	6	=	11010	9	=	1110001		...	

P. Elias: Universal codeword sets and representations of the integers. IEEE Transactions on Information Theory, IT-21(2)194–203, March 1975.

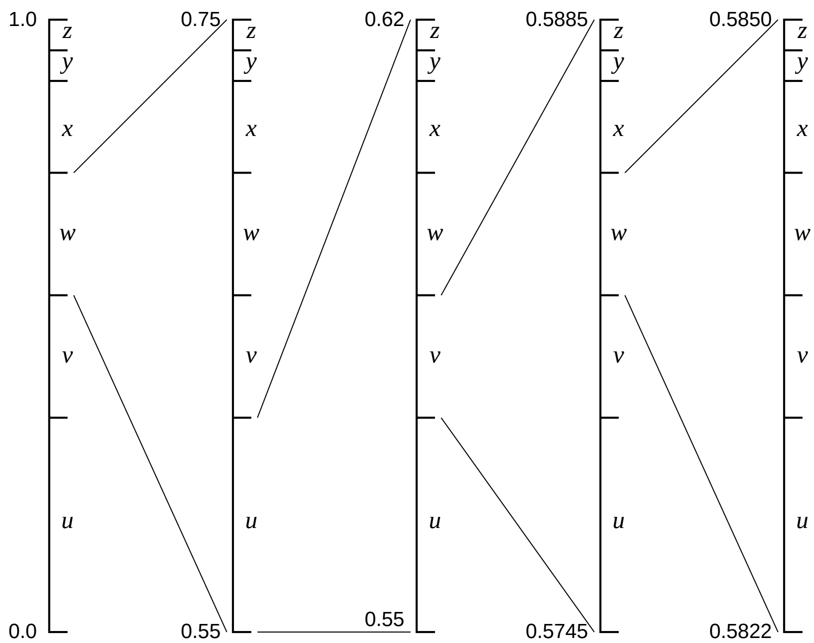
More such variable-length integer codes are described by Fenwick in IT-48(8)2412–2417, August 2002. (Available on `http://ieeexplore.ieee.org/`)

Entropy coding review – arithmetic coding

Partition [0,1] according to symbol probabilities:



Encode text $wuvw\ldots$ as numeric value $(0.58\ldots)$ in nested intervals:



Arithmetic coding

Several advantages:

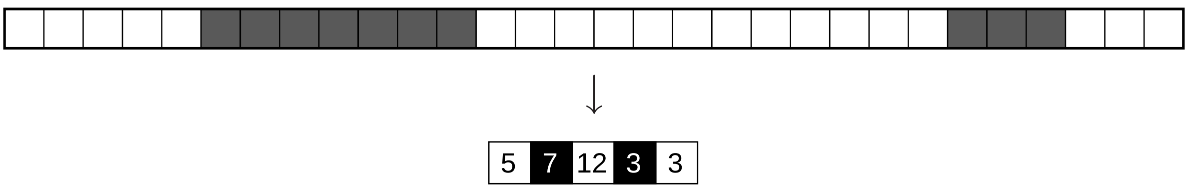
- → Length of output bitstring can approximate the theoretical information content of the input to within 1 bit.
- → Performs well with probabilities > 0.5, where the information per symbol is less than one bit.
- → Interval arithmetic makes it easy to change symbol probabilities (no need to modify code-word tree) $\Rightarrow$ convenient for adaptive coding

Can be implemented efficiently with fixed-length arithmetic by rounding probabilities and shifting out leading digits as soon as leading zeros appear in interval size. Usually combined with adaptive probability estimation.

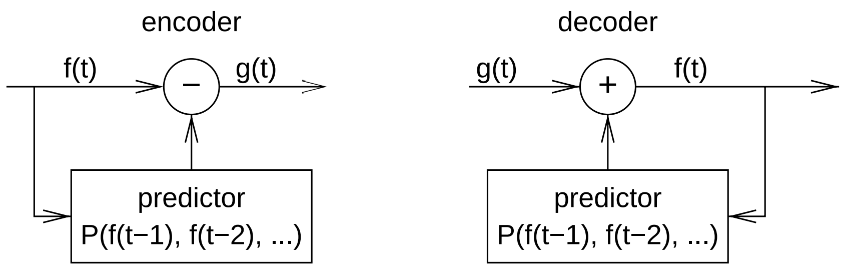
Huffman coding remains popular because of its simplicity and lack of patent licence issues.

Coding of sources with memory and correlated symbols

Run-length coding:



Predictive coding:



$$\text{Delta coding (DPCM):} \quad P(x) = x$$

$$\text{Linear predictive coding:} \quad P(x_1, \ldots, x_n) = \sum_{i=1}^{n} a_i x_i$$

Fax compression

International Telecommunication Union specifications:

→ Group 1 and 2: obsolete analog 1970s fax systems, required several minutes for uncompressed transmission of each page.

→ Group 3: fax protocol used on the analogue telephone network (9.6–14.4 kbit/s), with "modified Huffman" (MH) compression of run-length codes.
Modern G3 analog fax machines also support the better G4 and JBIG encodings.

→ Group 4: enhanced fax protocol for ISDN (64 kbit/s), introduced "modified modified relative element address designate (READ)" (MMR) coding.

ITU-T Recommendations, such as the ITU-T T.4 and T.6 documents that standardize the fax coding algorithms, are available on `http://www.itu.int/ITU-T/publications/recs.html`.

Group 3 MH fax code

- Run-length encoding plus modified Huffman code
- Fixed code table (from eight sample pages)
- separate codes for runs of white and black pixels
- *termination code* in the range 0–63 switches between black and white code
- *makeup code* can extend length of a run by a multiple of 64
- termination run length 0 needed where run length is a multiple of 64
- single white column added on left side before transmission
- makeup codes above 1728 equal for black and white
- 12-bit end-of-line marker: 000000000001 (can be prefixed by up to seven zero-bits to reach next byte boundary)

Example: line with 2 w, 4 b, 200 w, 3 b, EOL →
1000|011|010111|10011|10|000000000001

pixels	white code	black code
0	00110101	0000110111
1	000111	010
2	0111	11
3	1000	10
4	1011	011
5	1100	0011
6	1110	0010
7	1111	00011
8	10011	000101
9	10100	000100
10	00111	0000100
11	01000	0000101
12	001000	0000111
13	000011	00000100
14	110100	00000111
15	110101	000011000
16	101010	0000010111
...	...	...
63	00110100	000001100111
64	11011	0000001111
128	10010	000011001000
192	010111	000011001001
...	...	...
1728	010011011	0000001100101

Group 4 MMR fax code

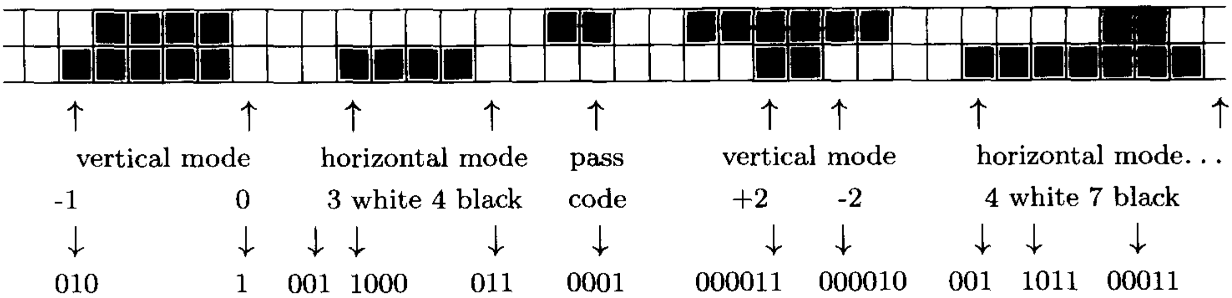
⟶ 2-dimensional code, references previous line

⟶ *Vertical mode* encodes transitions that have shifted up to ±3 pixels horizontally.

−1	→	010	0	→	1	1	→	011
−2	→	000010				2	→	000011
−3	→	0000010				3	→	0000011

⟶ *Pass mode* skip edges in previous line that have no equivalent in current line (0001)

⟶ *Horizontal mode* uses 1-dimensional run-lengths independent of previous line (001 plus two MH-encoded runs)



JBIG (Joint Bilevel Experts Group)

⟶ lossless algorithm for 1–6 bits per pixel

⟶ main applications: fax, scanned text documents

⟶ context-sensitive arithmetic coding

⟶ adaptive context template for better prediction efficiency with rastered photographs (e.g. in newspapers)

⟶ support for resolution reduction and progressive coding

⟶ "deterministic prediction" avoids redundancy of progr. coding

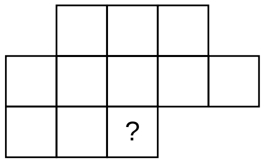
⟶ "typical prediction" codes common cases very efficiently

⟶ typical compression factor 20, 1.1–1.5× better than Group 4 fax, about 2× better than "`gzip -9`" and about ≈3–4× better than GIF (all on 300 dpi documents).

Information technology — Coded representation of picture and audio information — progressive bi-level image compression. International Standard ISO 11544:1993.
Example implementation: `http://www.cl.cam.ac.uk/~mgk25/jbigkit/`

JBIG encoding

Both encoder and decoder maintain statistics on how the black/white probability of each pixel depends on these 10 previously transmitted neighbours:



Based on the counted numbers n_{LPS} and n_{MPS} of how often the less and more probable symbol (e.g., black and white) have been encountered so far in each of the 1024 contexts, their probabilities are estimated as

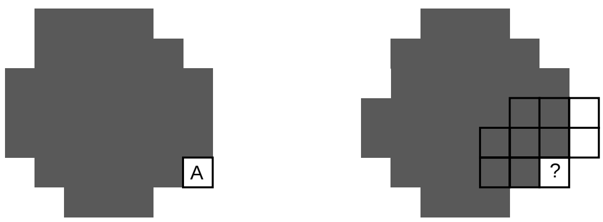
$$p_{\mathrm{LPS}} = \frac{n_{\mathrm{LPS}} + \delta}{n_{\mathrm{LPS}} + \delta + n_{\mathrm{MPS}} + \delta}$$

Parameter $\delta = 0.45$ is an empirically optimized start-up aid. To keep the estimation adaptable (for font changes, etc.) both counts are divided by a common factor before $n_{\mathrm{LPS}} > 11$.

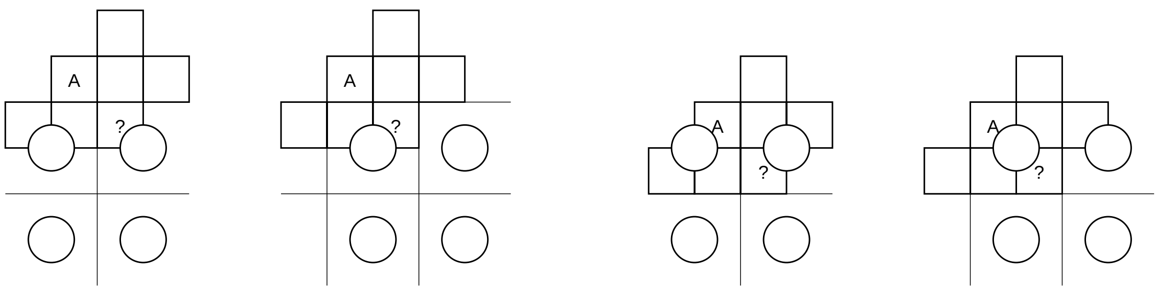
To simplify hardware implementation, the estimator is defined as an FSM with 113 states, each representing a point in the $(n_{\mathrm{LPS}}, n_{\mathrm{MPS}})$ plane. Is makes a transition only when the arithmetic-coding interval is renormalized to output another bit. The new state depends on whether renormalization was initiated by the less and more probable symbol.

Other JBIG features

One pixel of the context template can be moved, to exploit long-distance correlation in dither patterns:

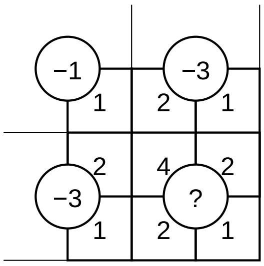


Differential encoding first transmits a lowres image, followed by additional differential layers (each doubling the resolution) that are encoded using context from the current (6 pixels) *and* previous (4 pixels) layer:

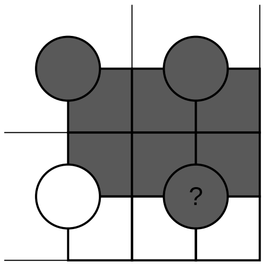


JBIG resolution reduction

Multiply neighbour pixels with filter coefficients and make the new reduced-resolution pixel black if the sum is 5 or more:



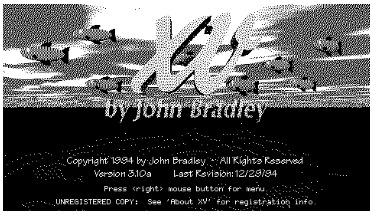
filter coefficients



example from exception list

This applies a low-pass filter to the high-resolution layer to avoid aliasing and a high-pass filter to the low-resolution layer to preserve gray-scale dithering. An exception list suppresses zigzag edges.

Example:



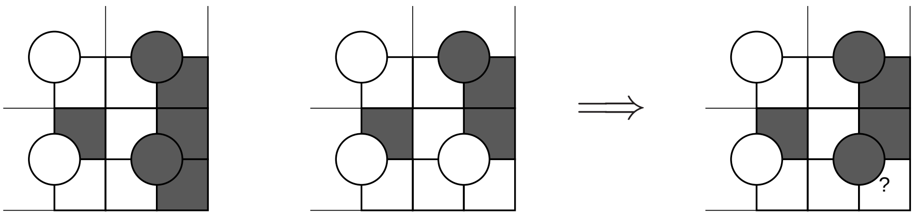
original

subsampled

JBIG reduced

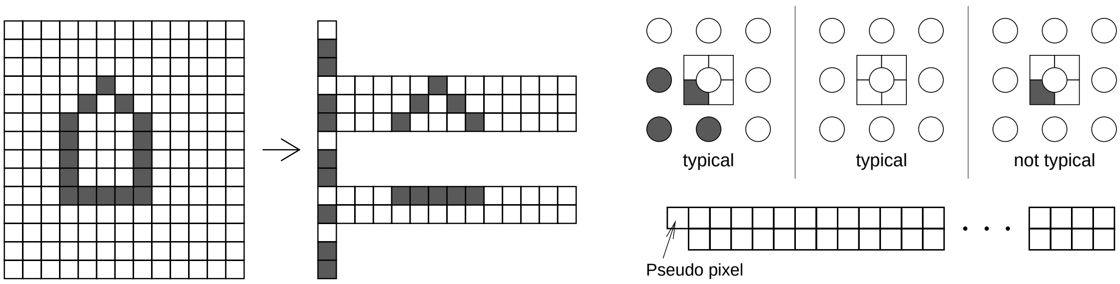
Two more JBIG optimization tricks

Deterministic prediction:



The next pixel can sometimes be predicted in differential layers from the already decoded lower-resolution layer and knowledge of the resolution-reduction table. The "?" pixel above must be black. If it were white, so would have to be its lowres equivalent. Such pixels are not coded.

Typical prediction:



A line identical to its predecessor is marked in an additional leftmost pseudo-pixel row as such and its pixels will not be coded. In differential mode, if in a highres line pair for each pixel four lowres neighbors with the same value imply that pixel's value ("typical pixel"), then the line pair will be marked and all its pixels with four identical low-res neighbours will not be coded.

Dependence and correlation

Random variables X, Y are *dependent* iff $\exists x, y$:

$$P(X = x \wedge Y = y) \neq P(X = x) \cdot P(Y = y).$$

If X, Y are dependent, then

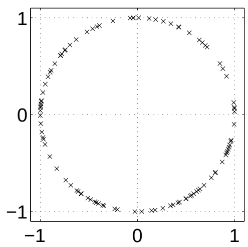
$$\begin{aligned} \Rightarrow \quad \exists x, y: \quad & P(X = x \,|\, Y = y) \neq P(X = x) \vee \\ & P(Y = y \,|\, X = x) \neq P(Y = y) \\ \Rightarrow \quad H(X|Y) < H(X) & \vee H(Y|X) < H(Y) \end{aligned}$$

Random variables are *correlated* iff

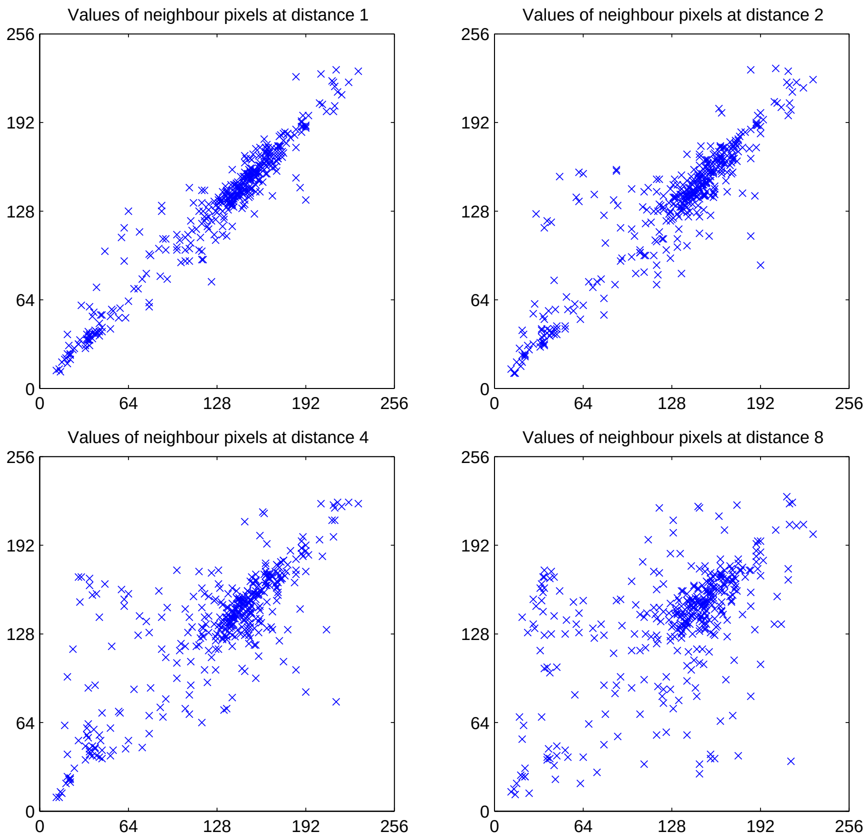
$$E((X - E(X)) \cdot (Y - E(Y))) \neq 0$$

Correlation implies dependence, but dependence does not always lead to correlation. However, most dependency in audiovisual data is a consequence of correlation, which is algorithmically much easier to exploit.

Dependent but not correlated:



Correlation of neighbour pixels



Covariance and correlation

We define the *covariance* of two random vectors X and Y as

$$\mathrm{Cov}(X,Y) = E\{[X-E(X)]\cdot[Y-E(Y)]\} = E(X\cdot Y)-E(X)\cdot E(Y)$$

and the *variance* as $\mathrm{Var}(X) = \mathrm{Cov}(X,X)$.

The correlation coefficient

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\cdot\mathrm{Var}(Y)}}$$

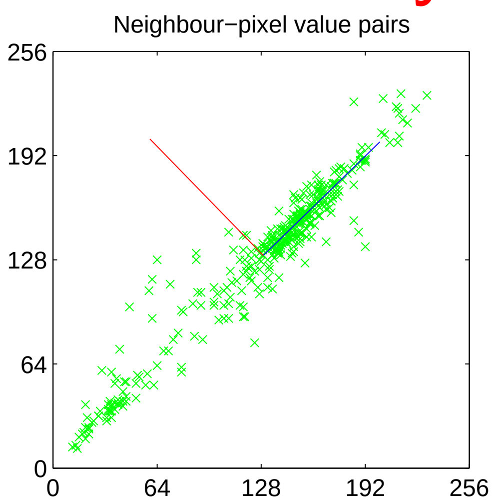
is a normalized form of the covariance in the range $[-1,1]$ with $\rho_{X,Y} = \pm 1 \Leftrightarrow \exists a,b : Y = aX+b$.

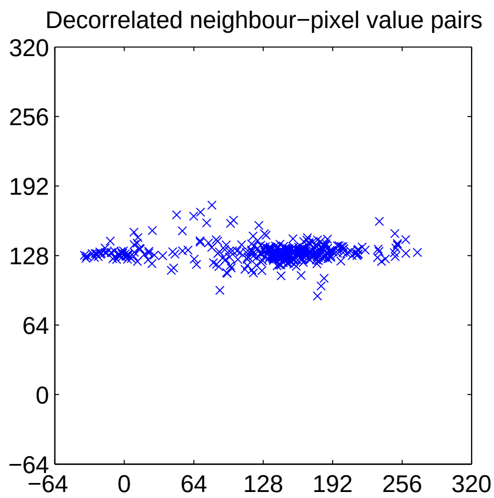
For a random vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ we define the covariance matrix

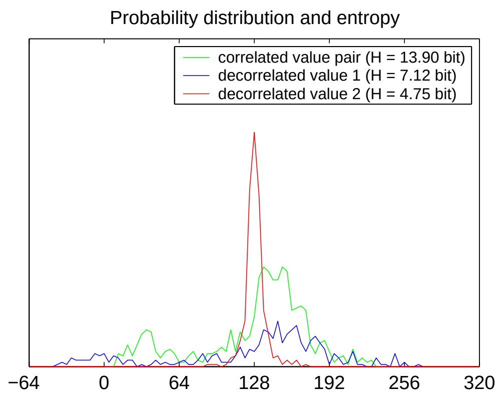
$$\mathrm{Cov}(\mathbf{X}) = (\mathrm{Cov}(X_i,X_j))_{i,j} = E\left((\mathbf{X}-E(\mathbf{X}))\cdot(\mathbf{X}-E(\mathbf{X}))^\mathsf{T}\right)$$

The elements of a random vector $\mathbf{X}$ are uncorrelated iff $\mathrm{Cov}(\mathbf{X})$ is a diagonal matrix.

Decorrelation by coordinate transform







Take the values of a group of pixels as a random vector. Find a coordinate transform (multiplication with an orthonormal matrix) that leads to a new random vector whose covariance matrix is diagonal. The vector components in this transformed coordinate system will no longer be correlated. This will hopefully reduce the entropy of some of these components.

Theorem: Let $\mathbf{X}$ and $\mathbf{Y}$ be random vectors with $\mathbf{Y} = A\mathbf{X} + b$, where A is a matrix and b is a vector of suitable size. Then

$$\begin{aligned} E(\mathbf{Y}) &= A \cdot E(\mathbf{X}) + b \\ \mathrm{Cov}(\mathbf{Y}) &= A \cdot \mathrm{Cov}(\mathbf{X}) \cdot A^\mathsf{T} \end{aligned}$$

Proof: The first equation follows from the linearity of the expected-value operator $E(\cdot)$, as does $E(A \cdot \mathbf{X} \cdot B) = A \cdot E(\mathbf{X}) \cdot B$ for matrices A, B. With that, we can transform

$$\begin{aligned} \mathrm{Cov}(\mathbf{Y}) &= E\left((\mathbf{Y} - E(\mathbf{Y})) \cdot (\mathbf{Y} - E(\mathbf{Y}))^\mathsf{T}\right) \\ &= E\left((A\mathbf{X} - AE(\mathbf{X})) \cdot (A\mathbf{X} - AE(\mathbf{X}))^\mathsf{T}\right) \\ &= E\left(A(\mathbf{X} - E(\mathbf{X})) \cdot (\mathbf{X} - E(\mathbf{X}))^\mathsf{T} A^\mathsf{T}\right) \\ &= A \cdot E\left((\mathbf{X} - E(\mathbf{X})) \cdot (\mathbf{X} - E(\mathbf{X}))^\mathsf{T}\right) \cdot A^\mathsf{T} \\ &= A \cdot \mathrm{Cov}(\mathbf{X}) \cdot A^\mathsf{T} \end{aligned}$$

Karhunen-Loève transform (KLT)

Take the n pixel values of an image (or in practice a small 8×8 pixel block) as an n-dimensional random vector $\mathbf{X}$.

How can we find a transform matrix A such that $\mathrm{Cov}(A\mathbf{X}) = A \cdot \mathrm{Cov}(\mathbf{X}) \cdot A^\mathsf{T}$ becomes a diagonal matrix? A would provide us the transformed representation $\mathbf{Y} = A\mathbf{X}$ of our image, in which all pixels are uncorrelated.

Note that $\mathrm{Cov}(\mathbf{X})$ is symmetric. It therefore has n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and a set of associated mutually orthogonal eigenvectors $b_1, b_2, \ldots, b_n$ of length 1 with

$$\mathrm{Cov}(\mathbf{X}) b_i = \lambda_i b_i.$$

We convert this set of equations into matrix notation using the matrix $B = (b_1, b_2, \ldots, b_n)$ that has these eigenvectors as columns and the diagonal matrix $D = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ that consists of the corresponding eigenvalues:

$$\mathrm{Cov}(\mathbf{X}) B = BD$$

B is orthonormal, that is $BB^{\mathsf{T}} = I$.

Multiplying the above from the right with B^{T} leads to the *spectral decomposition*

$$\mathrm{Cov}(\mathbf{X}) = BDB^{\mathsf{T}}$$

of the covariance matrix. Similarly multiplying instead from the left with B^{T} leads to

$$B^{\mathsf{T}}\,\mathrm{Cov}(\mathbf{X})B = D$$

and therefore shows with

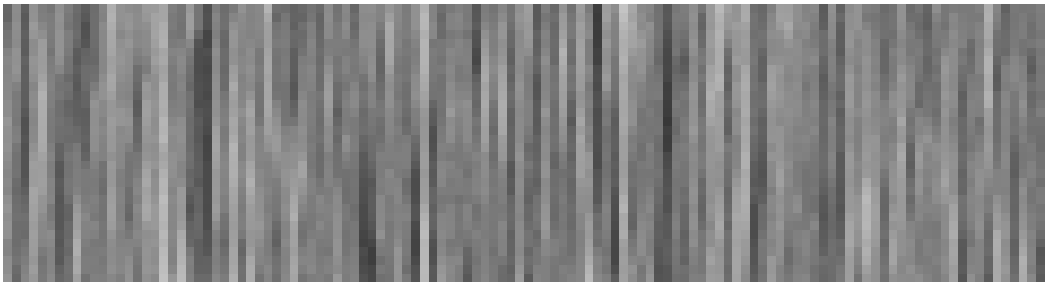
$$\mathrm{Cov}(B^{\mathsf{T}}\mathbf{X}) = D$$

that the eigenvector matrix B^{T} is the wanted transform.

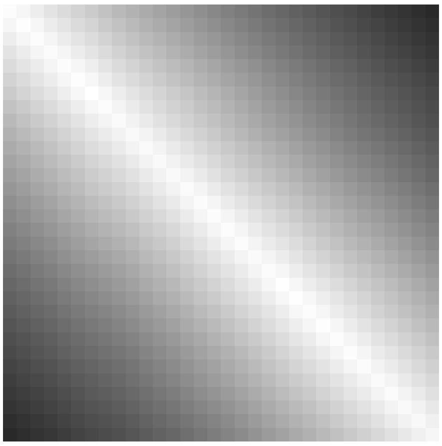
The *Karhunen-Loève transform* (also known as *Hotelling transform*) is the multiplication of a correlated random vector $\mathbf{X}$ with the orthonormal eigenvector matrix B^{T} from the spectral decomposition $\mathrm{Cov}(\mathbf{X}) = BDB^{\mathsf{T}}$ of its covariance matrix, leading to a decorrelated random vector $B^{\mathsf{T}}\mathbf{X}$ whose covariance matrix is diagonal.

Karhunen-Loève transform example

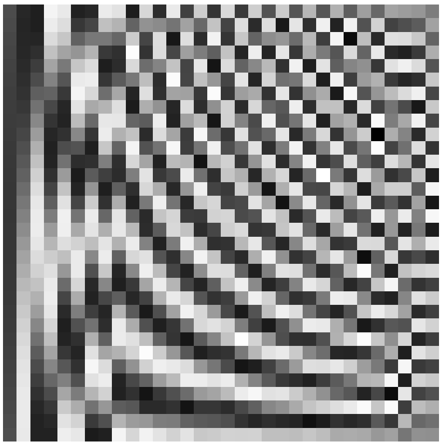
Matrix columns filled with samples of 1/f filtered noise



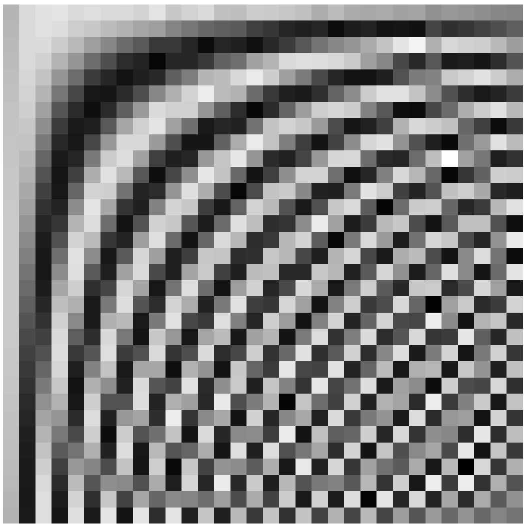
Covariance matrix



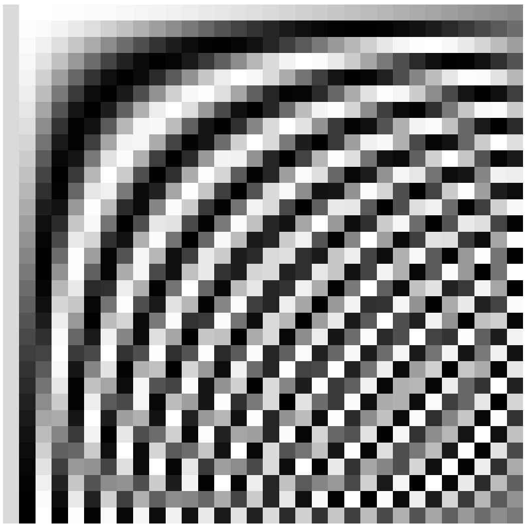
Matrix with eigenvector columns



Matrix with normalised KLT eigenvector columns



Matrix with Discrete Cosine Transform base vector columns



Breakthrough: Ahmed/Natarajan/Rao discovered the DCT as an excellent approximation of the KLT for typical photographic images, but far more efficient to calculate.

Ahmed, Natarajan, Rao: Discrete Cosine Transform. IEEE Transactions on Computers, Vol. 23, January 1974, pp. 90–93.

Discrete cosine transform (DCT)

The forward and inverse discrete cosine transform

$$\begin{aligned} S(u) &= \frac{C(u)}{\sqrt{N/2}} \sum_{x=0}^{N-1} s(x) \cos \frac{(2x+1)u\pi}{2N} \\ s(x) &= \sum_{u=0}^{N-1} \frac{C(u)}{\sqrt{N/2}} S(u) \cos \frac{(2x+1)u\pi}{2N} \end{aligned}$$

with

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & u = 0 \\ 1 & u > 0 \end{cases}$$

is an orthonormal transform:

$$\sum_{x=0}^{N-1} \frac{C(u)}{\sqrt{N/2}} \cos \frac{(2x+1)u\pi}{2N} \cdot \frac{C(u')}{\sqrt{N/2}} \cos \frac{(2x+1)u'\pi}{2N} = \begin{cases} 1 & u = u' \\ 0 & u \neq u' \end{cases}$$

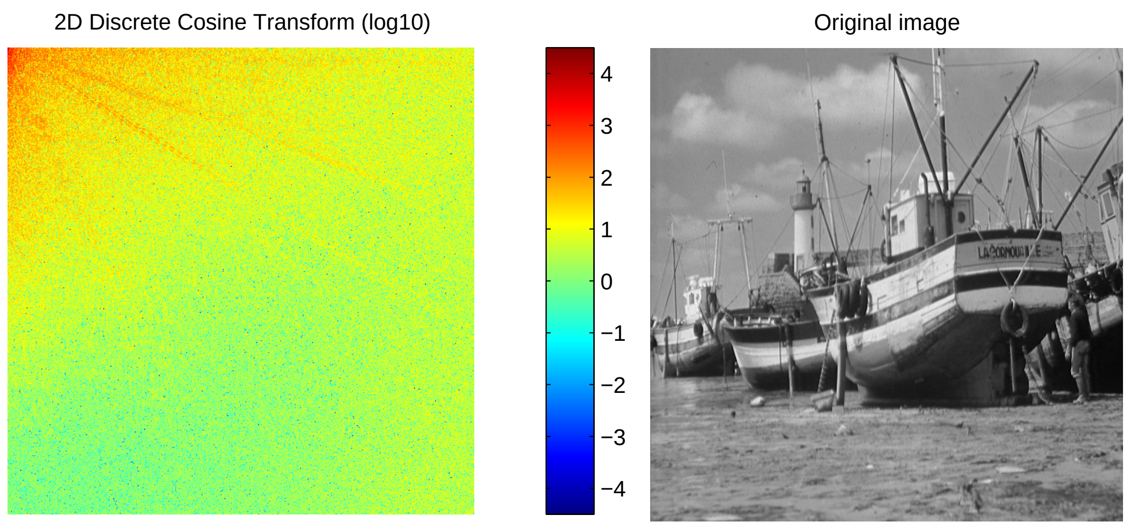
The 2-dimensional variant of the DCT applies the 1-D transform on both rows and columns of an image:

$$S(u,v) = \frac{C(u)}{\sqrt{N/2}} \frac{C(v)}{\sqrt{N/2}} \cdot \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x,y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$

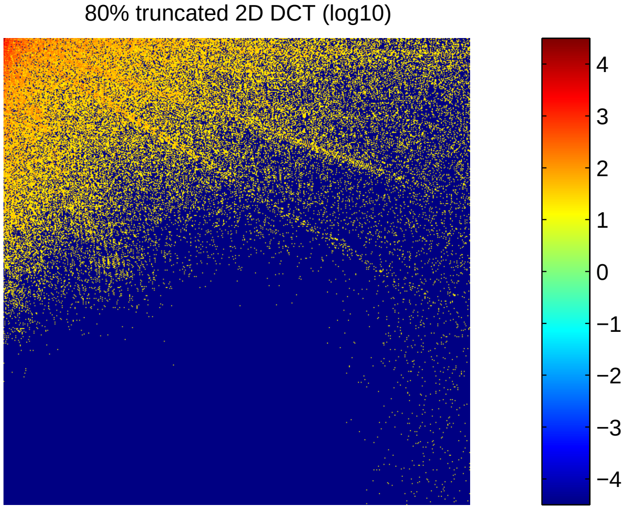
$$s(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \frac{C(u)}{\sqrt{N/2}} \frac{C(v)}{\sqrt{N/2}} \cdot S(u,v) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$

A range of fast algorithms have been found for calculating 1-D and 2-D DCTs (e.g., Ligtenberg/Vetterli).

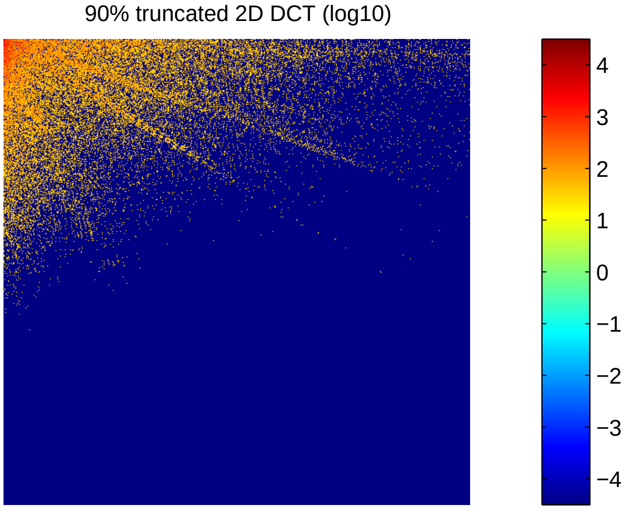
Whole-image DCT



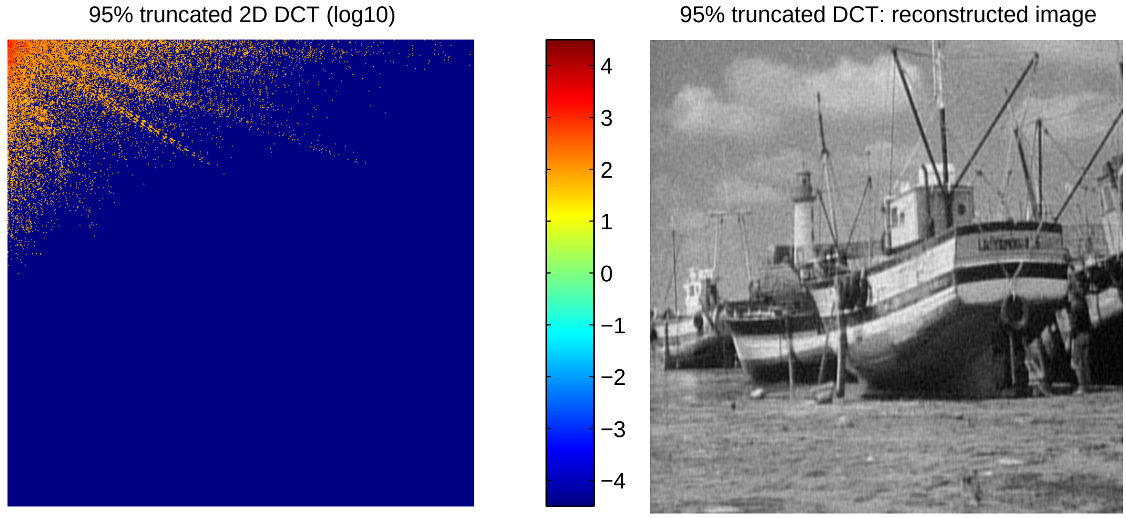
Whole-image DCT, 80% coefficient cutoff



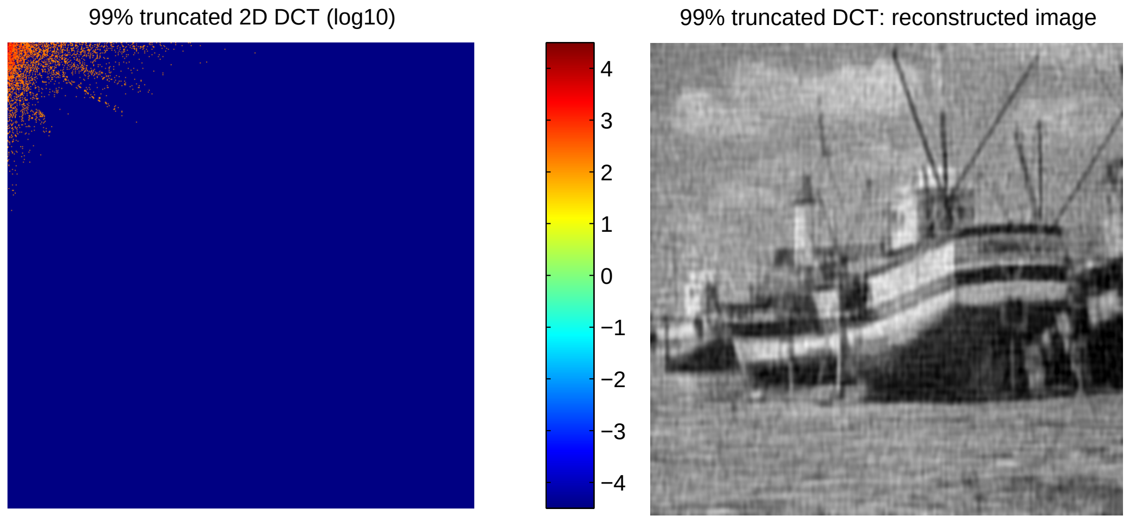
Whole-image DCT, 90% coefficient cutoff



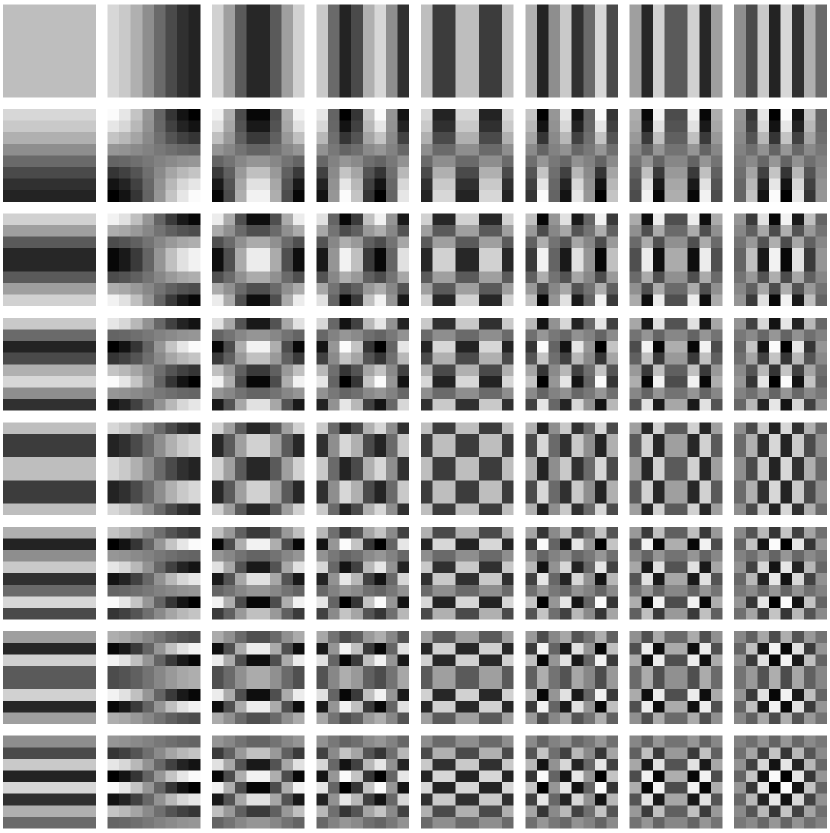
Whole-image DCT, 95% coefficient cutoff



Whole-image DCT, 99% coefficient cutoff



Base vectors of 8×8 DCT



Psychophysics of perception

Sensation limit (SL) = lowest intensity stimulus that can still be perceived

Difference limit (DL) = smallest perceivable stimulus difference at given intensity level

Weber's law

Difference limit $\Delta\phi$ is proportional to the intensity ϕ of the stimulus (except for a small correction constant a describe deviation of experimental results near SL):

$$\Delta\phi = c \cdot (\phi + a)$$

Fechner's scale

Define a perception intensity scale ψ using the sensation limit ϕ_0 as the origin and the respective difference limit $\Delta\phi = c \cdot \phi$ as a unit step. The result is a logarithmic relationship between stimulus intensity and scale value:

$$\psi = \log_c \frac{\phi}{\phi_0}$$

Fechner's scale matches older subjective intensity scales that follow differentiability of stimuli, e.g. the astronomical magnitude numbers for star brightness introduced by Hipparchos (≈150 BC).

Stevens' law

A sound that is 20 DL over SL is perceived as more than twice as loud as one that is 10 DL over SL, i.e. Fechner's scale does not describe well perceived intensity. A rational scale attempts to reflect subjective relations perceived between different values of stimulus intensity ϕ. Stevens observed that such rational scales ψ follow a power law:

$$\psi = k \cdot (\phi - \phi_0)^a$$

Example coefficients a: temperature 1.6, weight 1.45, loudness 0.6, brightness 0.33.

Decibel

Communications engineers often use logarithmic units:

- $\longrightarrow$ Quantities often vary over many orders of magnitude $\rightarrow$ difficult to agree on a common SI prefix
- $\longrightarrow$ Quotient of quantities (amplification/attenuation) usually more interesting than difference
- $\longrightarrow$ Signal strength usefully expressed as field quantity (voltage, current, pressure, etc.) or power, but quadratic relationship between these two ($P = U^2/R = I^2R$) rather inconvenient
- $\longrightarrow$ Weber/Fechner: perception is logarithmic

Plus: Using magic special-purpose units has its own odd attractions ($\rightarrow$ typographers, navigators)

Neper (Np) denotes the natural logarithm of the quotient of a field quantity F and a reference value F_0.

Bel (B) denotes the base-10 logarithm of the quotient of a power P and a reference power P_0. Common prefix: 10 decibel (dB) = 1 bel.

Where P is some power and P_0 a 0 dB reference power, or equally where F is a field quantity and F_0 the corresponding reference level:

$$10\text{ dB} \cdot \log_{10} \frac{P}{P_0} = 20\text{ dB} \cdot \log_{10} \frac{F}{F_0}$$

Common reference values are indicated with an additional letter after the "dB":

$$\begin{aligned}
0\text{ dBW} &= 1\text{ W} \\
0\text{ dBm} &= 1\text{ mW} = -30\text{ dBW} \\
0\text{ dB}\mu\text{V} &= 1\ \mu\text{V} \\
0\text{ dB}_{\mathrm{SPL}} &= 20\ \mu\text{Pa} \quad \text{(sound pressure level)} \\
0\text{ dB}_{\mathrm{SL}} &= \text{perception threshold (sensation limit)}
\end{aligned}$$

3 dB = double power, 6 dB = double pressure/voltage/etc.
10 dB = 10× power, 20 dB = 10× pressure/voltage/etc.

W.H. Martin: Decibel – the new name for the transmission unit. Bell System Technical Journal, January 1929.

RGB video colour coordinates

Hardware interface (VGA): red, green, blue signals with 0–0.7 V

Electron-beam current and photon count of cathode-ray displays are roughly proportional to $(v - v_0)^\gamma$, where v is the video-interface or control-grid voltage and γ is a device parameter that is typically in the range 1.5–3.0. In broadcast TV, this CRT non-linearity is compensated electronically in TV cameras. A welcome side effect is that it approximates Stevens' scale and therefore helps to reduce perceived noise.

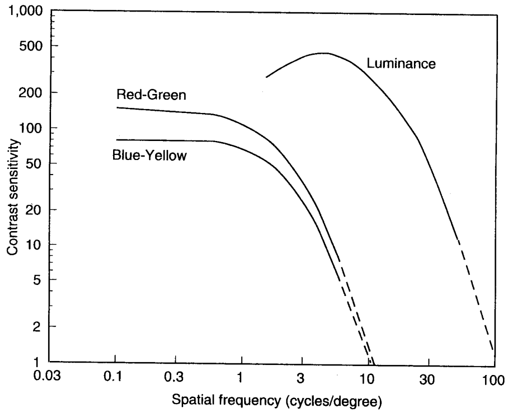
Software interfaces map RGB voltage linearly to $\{0, 1, \ldots, 255\}$ or 0–1

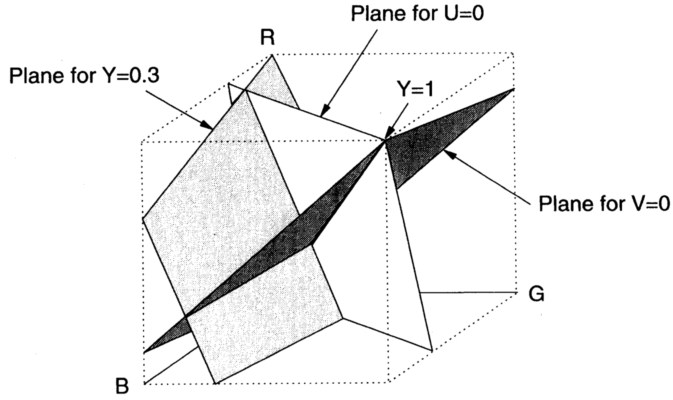
How numeric RGB values map to colour and luminosity depends at present still highly on the hardware and sometimes even on the operating system or device driver.

The new specification "sRGB" aims to standardize the meaning of an RGB value with the parameter $\gamma = 2.2$ and with standard colour coordinates of the three primary colours.

http://www.w3.org/Graphics/Color/sRGB, http://www.srgb.com/, IEC 61966

YUV video colour coordinates





The human eye processes colour and luminosity at different resolutions. To exploit this phenomenon, many image transmission systems use a colour space with a luminance coordinate

$$Y = 0.3R + 0.6G + 0.1B$$

and colour (“chrominance”) components

$$\begin{aligned} V &= R - Y = 0.7R - 0.6G - 0.1B \\ U &= B - Y = -0.3R - 0.6G + 0.9B \end{aligned}$$

YCrCb video colour coordinates

Since $-0.7 \leq V \leq 0.7$ and $-0.9 \leq U \leq 0.9$, a more convenient normalized encoding of chrominance is:

$$\begin{aligned} Cb &= \frac{U}{2.0} + 0.5 \\ Cr &= \frac{V}{1.6} + 0.5 \end{aligned}$$

Modern image compression techniques operate on Y, Cr, Cb channels separately, using half the resolution of Y for storing Cr, Cb.

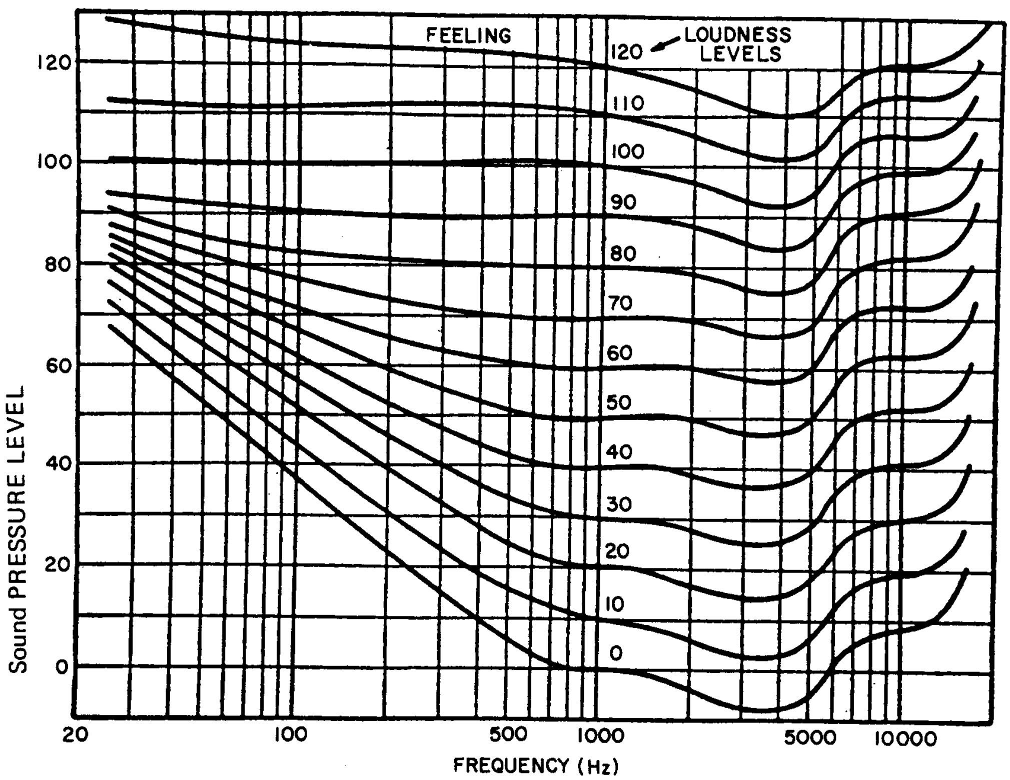
Some digital-television engineering terminology:

If each pixel is represented by its own Y, Cr and Cb byte, this is called a “4:4:4” format. In the compacter “4:2:2” format, a Cr and Cb value is transmitted only for every second pixel, reducing the horizontal chrominance resolution by a factor two. The “4:2:0” format transmits in alternating lines either Cr or Cb for every second pixel, thus halving the chrominance resolution both horizontally and vertically. The “4:1:1” format reduces the chrominance resolution horizontally by a quarter and “4:1:0” does so in both directions. [ITU-R BT.601]

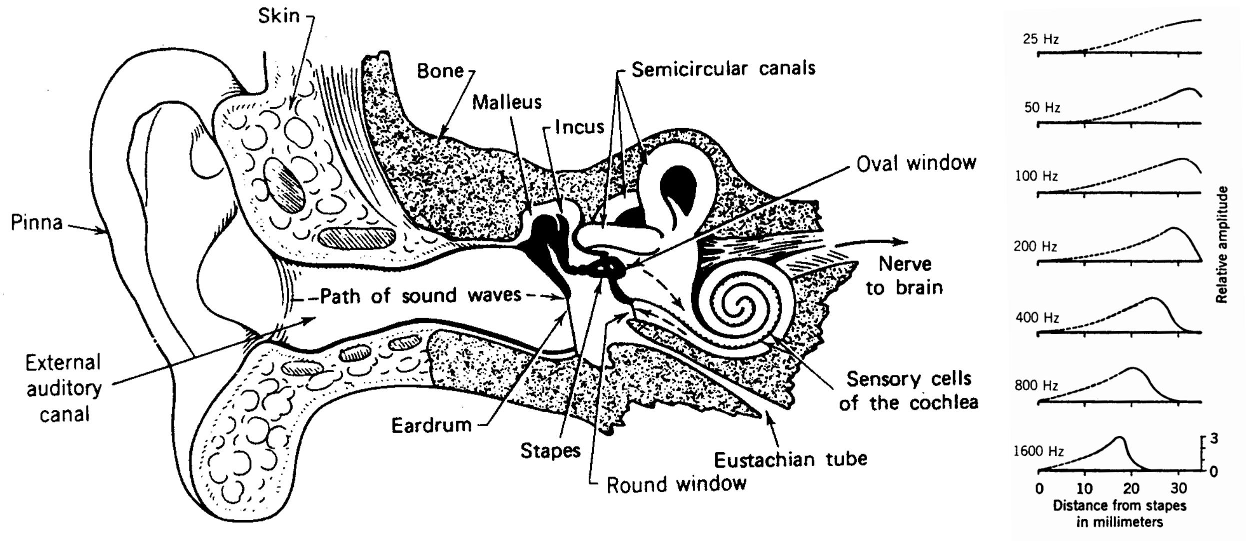
The human auditory system

- $\longrightarrow$ frequency range 20–16000 Hz (babies: 20 kHz)
- $\longrightarrow$ sound pressure range 0–140 dB_{SPL} (about 10^{-5}–10^{2} pascal)
- $\longrightarrow$ mechanical filter bank (cochlea) splits input into frequency components, physiological equivalent of Fourier transform
- $\longrightarrow$ most signal processing happens in the frequency domain where phase information is lost
- $\longrightarrow$ some time-domain processing below 500 Hz and for directional hearing
- $\longrightarrow$ sensitivity and difference limit are frequency dependent

Equiloudness curves and the unit “phon”



Each curve represents a loudness level in phon. At 1 kHz, the loudness unit phon is identical to dB_{SPL} and 0 phon is the sensation limit.



Sound waves cause vibration in the eardrum. The three smallest human bones in the middle ear (malleus, incus, stapes) provide an "impedance match" between air and liquid and conduct the sound via a second membrane, the oval window, to the cochlea. Its three chambers are rolled up into a spiral. The basilar membrane that separates the two main chambers decreases in stiffness along the spiral, such that the end near the stapes vibrates best at the highest frequencies, whereas for lower frequencies that amplitude peak moves to the far end.

Frequency discrimination and critical bands

A pair of pure tones (sine functions) cannot be distinguished as two separate frequencies if both are in the same frequency group ("critical band"). Their loudness adds up, and both are perceived with their average frequency.

The human ear has about 24 critical bands whose width grows non-linearly with the center frequency.

Each audible frequency can be expressed on the "Bark scale" with values in the range 0–24. A good closed-form approximation is

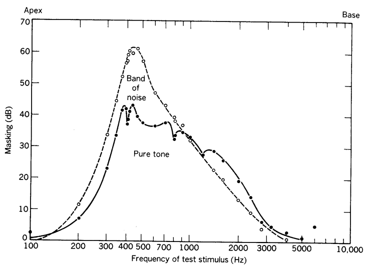
$$b \approx \frac{26.81}{1 + \frac{1960\ \mathrm{Hz}}{f}} - 0.53$$

where f is the frequency and b the corresponding point on the Bark scale.

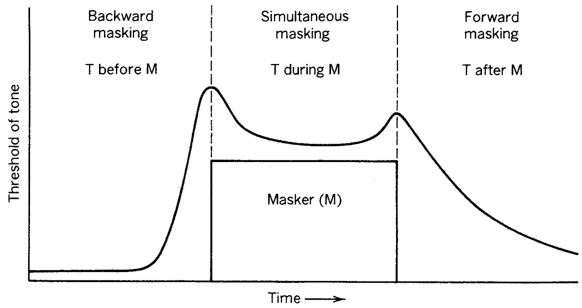
Two frequencies are in the same critical band if their distance is below 1 bark.

Masking

- → Louder tones increase the sensation limit for nearby frequencies and suppress the perception of quieter tones.
- → This increase is not symmetric. It extends about 3 barks to lower frequencies and 8 barks to higher ones.
- → The sensation limit is increased less for pure tones of nearby frequencies, as these can still be perceived via their beat frequency. For the study of masking effects, pure tones therefore need to be distinguished from narrowband noise.
- → Temporal masking: SL rises shortly before and after a masker.



Level of masking in decibels for test tones of various frequencies presented with a masker of either an 80-dB SPL 400 Hz tone or an 80-dB SPL narrow-band noise (90 Hz) with a center frequency of 410 Hz. From Egan and Hake (11).



Audio demo: loudness and masking

loudness.wav

Two sequences of tones with frequencies 40, 63, 100, 160, 250, 400, 630, 1000, 1600, 2500, 4000, 6300, 10000, and 16000 Hz.

- → Sequence 1: tones have equal amplitude
- → Sequence 2: tones have roughly equal perceived loudness
  Amplitude adjusted to IEC 60651 "A" weighting curve for soundlevel meters.

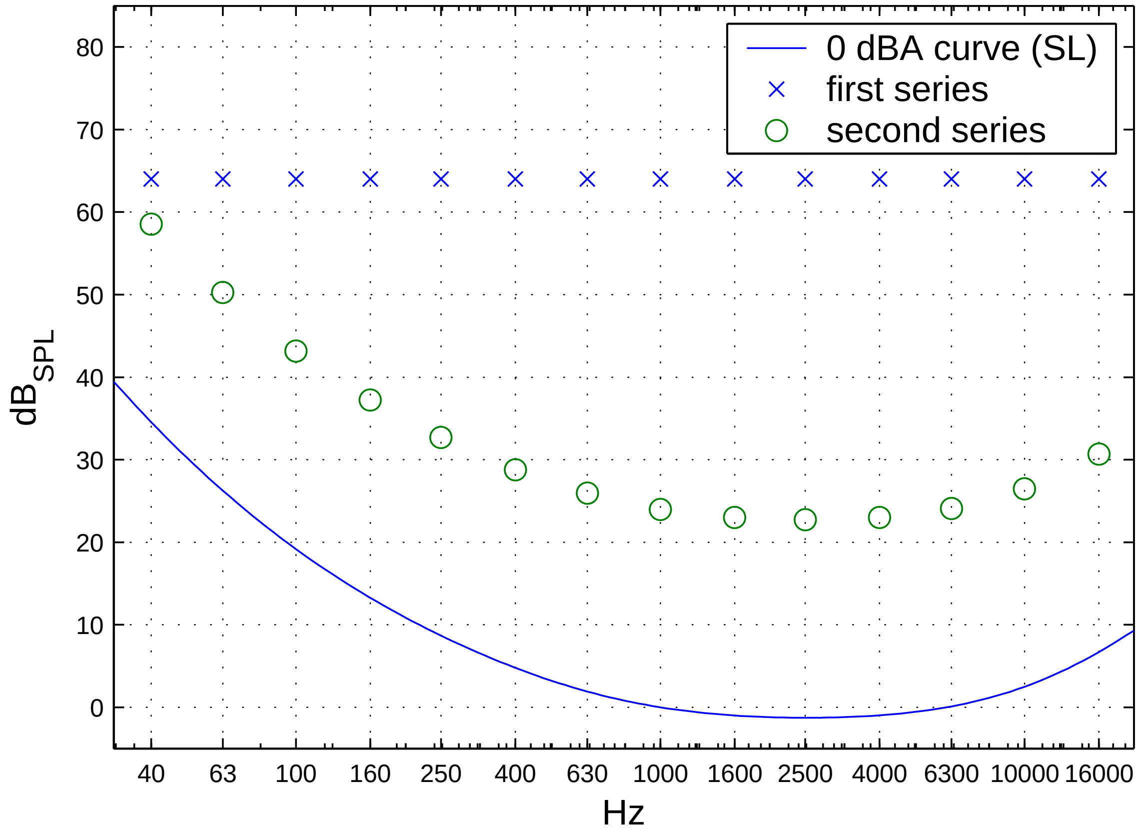
masking.wav

Twelve sequences, each with twelve probe-tone pulses and a 1200 Hz masking tone during pulses 5 to 8.

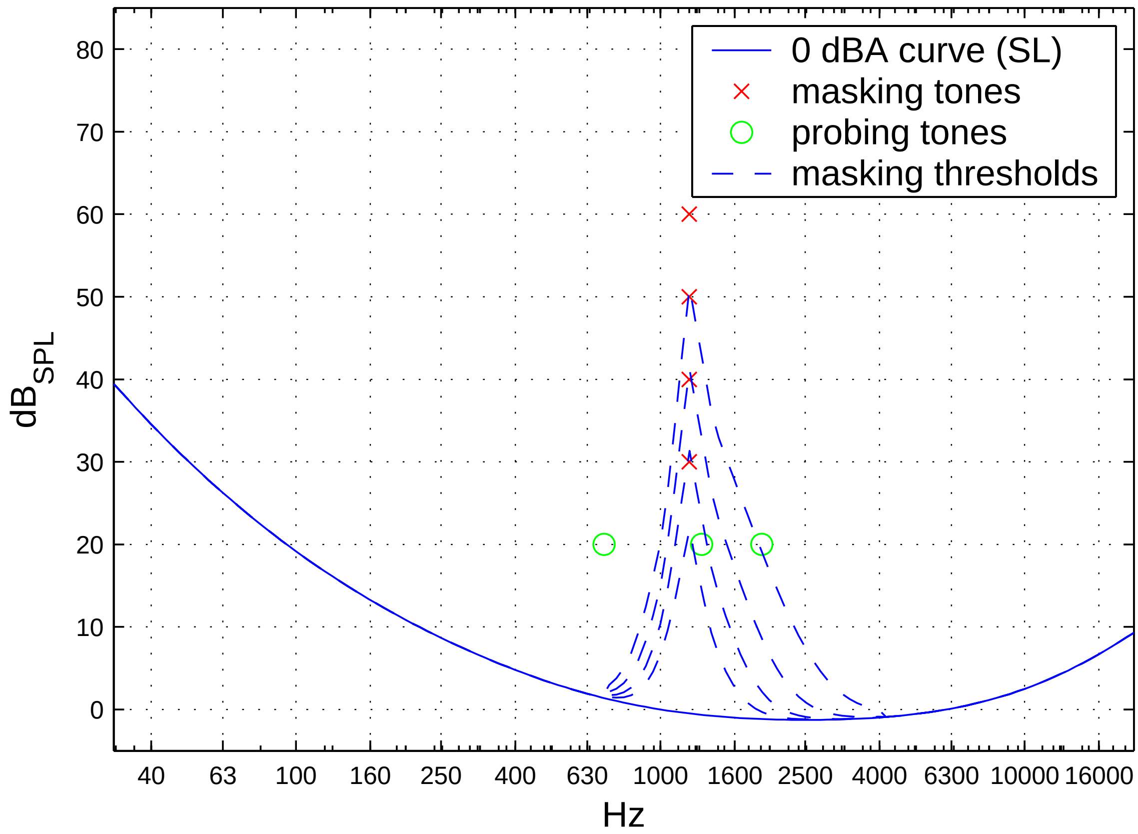
Probing tone frequency and relative masking tone amplitude:

	10 dB	20 dB	30 dB	40 dB
1300 Hz				
1900 Hz				
700 Hz				

Audio demo: loudness.wav

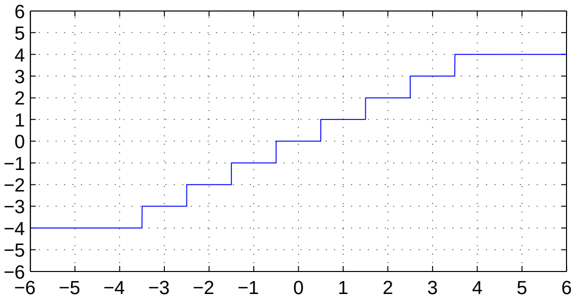


Audio demo: masking.wav

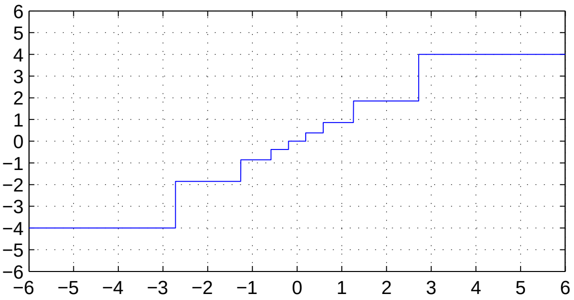


Quantization

Uniform/linear quantization:



Non-uniform quantization:



Quantization is the mapping from a continuous or large set of values (e.g., analog voltage, floating-point number) to a smaller set of (typically 2^8, 2^{12} or 2^{16}) values.

This introduces two types of error:

- ⟶ the amplitude of *quantization noise* reaches up to half the maximum difference between neighbouring quantization levels
- ⟶ *clipping* occurs where the input amplitude exceeds the value of the highest (or lowest) quantization level

Example of a linear quantizer (resolution R, peak value V):

$$y = \max\left\{-V, \min\left\{V, R\left\lfloor \frac{x}{R} + \frac{1}{2} \right\rfloor\right\}\right\}$$

Adding a noise signal that is uniformly distributed on $[0,1]$ instead of adding $\frac{1}{2}$ helps to spread the frequency spectrum of the quantization noise more evenly. This is known as *dithering*.

Variant with even number of output values (no zero):

$$y = \max\left\{-V, \min\left\{V, R\left(\left\lfloor \frac{x}{R} \right\rfloor + \frac{1}{2}\right)\right\}\right\}$$

Improving the resolution by a factor of two (i.e., adding 1 bit) reduces the quantization noise by 6 dB.

Linearly quantized signals are easiest to process, but analog input levels need to be adjusted carefully to achieve a good tradeoff between the signal-to-quantization-noise ratio and the risk of clipping. Non-uniform quantization can reduce quantization noise where input values are not uniformly distributed and can approximate human perception limits.

Logarithmic quantization

Rounding the logarithm of the signal amplitude makes the quantization error scale-invariant and is used where the signal level is not very predictable. Two alternative schemes are widely used to make the logarithm function odd and linearize it across zero before quantization:

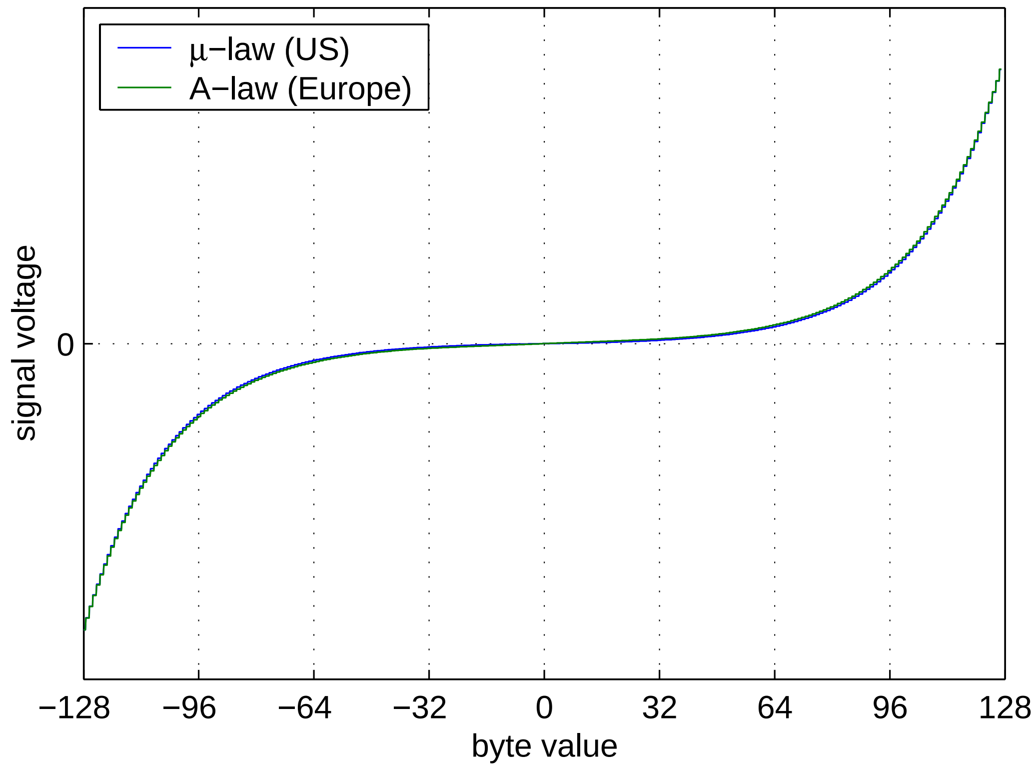
μ-law:

$$y = \frac{V \log(1 + \mu|x|/V)}{\log(1 + \mu)} \operatorname{sgn}(x) \quad \text{for } -V \le x \le V$$

A-law:

$$y = \begin{cases} \frac{A|x|}{1+\log A} \operatorname{sgn}(x) & \text{for } 0 \le |x| \le \frac{V}{A} \\ \frac{V\left(1+\log \frac{A|x|}{V}\right)}{1+\log A} \operatorname{sgn}(x) & \text{for } \frac{V}{A} \le |x| \le V \end{cases}$$

European digital telephone networks use A-law quantization ($A = 87.6$), North American ones use μ-law (μ=255), both with 8-bit resolution and 8 kHz sampling frequency (64 kbit/s). [ITU-T G.711]



Lloyd's algorithm: finds least-square-optimal non-uniform quantization function for a given probability distribution of sample values.

S.P. Lloyd: Least Squares Quantization in PCM. IEEE Trans. IT-28, March 1982, pp 129–137.

Joint Photographic Experts Group – JPEG

Working group "ISO/TC97/SC2/WG8 (Coded representation of picture and audio information)" was set up in 1982 by the International Organization for Standardization.

Goals:

- $\longrightarrow$ continuous tone gray-scale and colour images
- $\longrightarrow$ recognizable images at 0.083 bit/pixel
- $\longrightarrow$ useful images at 0.25 bit/pixel
- $\longrightarrow$ excellent image quality at 0.75 bit/pixel
- $\longrightarrow$ indistinguishable images at 2.25 bit/pixel
- $\longrightarrow$ feasibility of 64 kbit/s (ISDN fax) compression with late 1980s hardware (16 MHz Intel 80386).
- $\longrightarrow$ workload equal for compression and decompression

JPEG standard (ISO 10918) was finally published in 1994.

William B. Pennebaker, Joan L. Mitchell: JPEG still image compression standard. Van Nostrad Reinhold, New York, ISBN 0442012721, 1993.

Gregory K. Wallace: The JPEG Still Picture Compression Standard. Communications of the ACM 34(4)30–44, April 1991, `http://doi.acm.org/10.1145/103085.103089`

Summary of the baseline JPEG algorithm

The most widely used lossy method from the JPEG standard:

- $\longrightarrow$ Colour component transform: 8-bit RGB $\rightarrow$ 8-bit YCrCb
- $\longrightarrow$ Reduce resolution of Cr and Cb by a factor 2
- $\longrightarrow$ For the rest of the algorithm, process Y, Cr and Cb components independently (like separate gray-scale images)

  The above steps are obviously skipped where the input is a gray-scale image.
- $\longrightarrow$ Split each image component into 8×8 pixel blocks

  Partial blocks at the right/bottom margin may have to be padded by repeating the last column/row until a multiple of eight is reached. The decoder will remove these padding pixels.
- $\longrightarrow$ Apply the 8×8 forward DCT on each block

  On unsigned 8-bit input, the resulting DCT coefficients will be signed 11-bit integers.

→ Quantization: divide each DCT coefficient with the corresponding value from an 8×8 table, then round to the nearest integer:

The two standard quantization-matrix examples for luminance and chrominance are:

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

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

→ apply DPCM coding to quantized DC coefficients from DCT

→ read remaining quantized values from DCT in zigzag pattern

→ locate sequences of zero coefficients (run-length coding)

→ apply Huffman coding on zero run-lengths and magnitude of AC values

→ add standard header with compression parameters

http://www.jpeg.org/

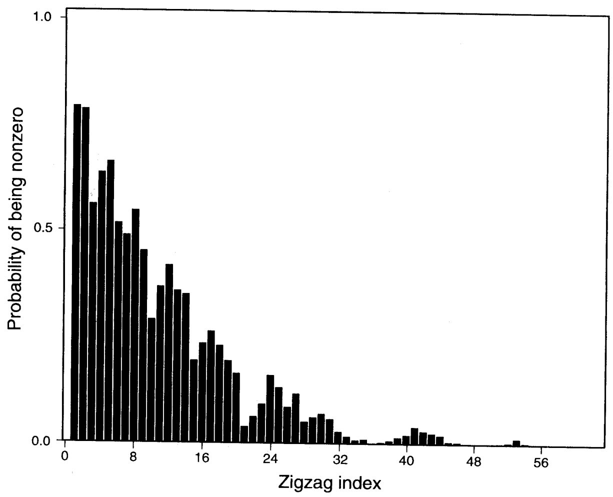
Example implementation: http://www.ijg.org/

Storing DCT coefficients in zigzag order

horizontal frequency →

vertical frequency ↓

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



After the 8×8 coefficients produced by the discrete cosine transform have been quantized, the values are processed in the above zigzag order by a run-length encoding step.

The idea is to group all higher-frequency coefficients together at the end of the sequence. As many image blocks contain little high-frequency information, the bottom-right corner of the quantized DCT matrix is often entirely zero. The zigzag scan helps the run-length coder to make best use of this observation.

Huffman coding in JPEG

s	value range
0	0
1	$-1, 1$
2	$-3, -2, 2, 3$
3	$-7 \ldots -4, 4 \ldots 7$
4	$-15 \ldots -8, 8 \ldots 15$
5	$-31 \ldots -16, 16 \ldots 31$
6	$-63 \ldots -32, 32 \ldots 63$
$\ldots$	$\ldots$
i	$-(2^i - 1) \ldots -2^{i-1}, 2^{i-1} \ldots 2^i - 1$

DCT coefficients have 11-bit resolution and would lead to huge Huffman tables (up to 2048 code words). JPEG therefore uses a Huffman table only to encode the magnitude category $s = \lceil \log_2(|v| + 1) \rceil$ of a DCT value v. A sign bit plus the $(s-1)$-bit binary value $|v| - 2^{s-1}$ are appended to each Huffman code word, to distinguish between the 2^s different values within magnitude category s.

When storing DCT coefficients in zigzag order, the symbols in the Huffman tree are actually tuples (r, s), where r is the number of zero coefficients preceding the coded value (run-length).

Arithmetic coding in JPEG

As an option, the Huffman coder in JPEG can be replaced with an arithmetic coder. The coder used is identical to the JBIG one (113-state adaptive estimator, etc.). It processes a sequence of binary decisions, therefore each integer value (DC coefficient difference, AC coefficient, lossless difference) to be coded is first transformed into a bit string using a variant of the Elias gamma code, which is then fed bit-by-bit into the arithmetic coder with a suitable context.

If the integer v to be coded is zero, only the bit 0 is fed into the arithmetic coder. Otherwise, the coder receives a 1 bit, followed by the sign bit of v, followed by the unary-coded value $\lceil \log_2 |v| \rceil$, followed by the $\lceil \log_2 |v| \rceil - 1$ bits after the leading 1 bit of the binary notation of $|v| - 1$.

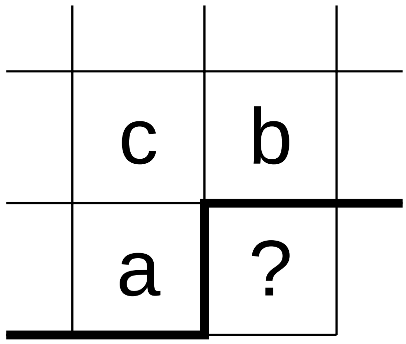
The coding context used depends on the bit position, in the case of the third bit ($|v| > 1$?) also on the second bit ($v < 0$?). In the case of the first three bits, the context also depends on the zigzag index number for AC coefficients, or on the first few bits of the previous DC coefficient difference for a DC coefficient.

In other words, integer values are first coded with a fixed Huffman code that outputs bits with roughly equal probability, and then the arithmetic coder adapts to exploit the remaining bit bias, as well as the dependence on a selected small set of previously coded bits.

Lossless JPEG algorithm

In addition to the DCT-based lossy compression, JPEG also defines a lossless mode. It offers a selection of seven linear prediction mechanisms based on three previously coded neighbour pixels:

$$
\begin{aligned}
1:&\quad x = a\\
2:&\quad x = b\\
3:&\quad x = c\\
4:&\quad x = a + b - c\\
5:&\quad x = a + (b - c)/2\\
6:&\quad x = b + (a - c)/2\\
7:&\quad x = (a + b)/2
\end{aligned}
$$



Predictor 1 is used for the top row, predictor 2 for the left-most row. The predictor used for the rest of the image is chosen in a header. The difference between the predicted and actual value is fed into either a Huffman or arithmetic coder.

Advanced JPEG features

Beyond the baseline and lossless modes already discussed, JPEG provides these additional features:

- $\longrightarrow$ 8 or 12 bits per pixel input resolution for DCT modes
- $\longrightarrow$ 2–16 bits per pixel for lossless mode
- $\longrightarrow$ progressive mode permits the transmission of more-significant DCT bits or lower-frequency DCT coefficients first, such that a low-quality version of the image can be displayed early during a transmission
- $\longrightarrow$ the transmission order of colour components, lines, as well as DCT coefficients and their bits can be interleaved in many ways
- $\longrightarrow$ the hierarchical mode first transmits a low-resolution image, followed by a sequence of differential layers that code the difference to the next higher resolution (like JBIG's progressive mode)

Moving Pictures Experts Group – MPEG

- MPEG-1: Coding of video and audio optimized for 1.5 Mbit/s (1× CD-ROM). ISO 11172 (1993).
- MPEG-2: Adds support for interlaced video scan, optimized for broadcast TV (2–8 Mbit/s) and HDTV, scalability options. Used by DVD and DVB. ISO 13818 (1995).
- MPEG-4: Adds algorithmic or segmented description of audio-visual objects for very-low bitrate applications. ISO 14496 (2001).
- System layer multiplexes several audio and video streams, time stamp synchronization, buffer control.
- Standard defines decoder semantics.
- Asymmetric workload: Encoder needs significantly more computational power than decoder (for bit-rate adjustment, motion estimation, perceptual modeling, etc.)

`http://mpeg.telecomitalialab.com/`

MPEG video coding

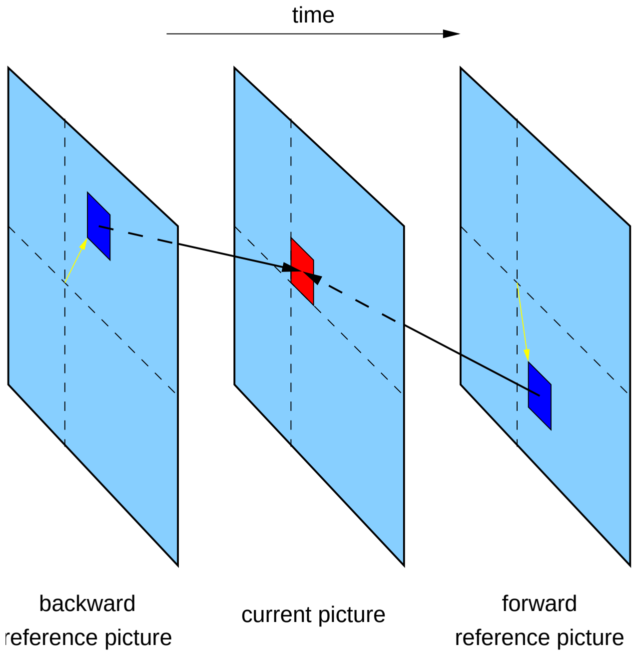
- Uses YCrCb colour transform, 8×8-pixel DCT, quantization, zigzag scan, run-length and Huffman encoding, similar to JPEG
- the zigzag scan pattern is adapted to handle interlaced fields
- Huffman coding with fixed code tables defined in the standard
  MPEG has no arithmetic coder option.
- adaptive quantization
- SNR and spatially scalable coding (enables separate transmission of a moderate-quality video signal and an enhancement signal to reduce noise or improve resolution)
- Predictive coding with motion compensation based on 16×16 macro blocks.

J. Mitchell, W. Pennebaker, Ch. Fogg, D. LeGall: MPEG video compression standard. ISBN 0412087715, 1997. (CL library: I.4.20)

B. Haskell et al.: Digital Video: Introduction to MPEG-2. Kluwer Academic, 1997. (CL library: I.4.27)

John Watkinson: The MPEG Handbook. Focal Press, 2001. (CL library: I.4.31)

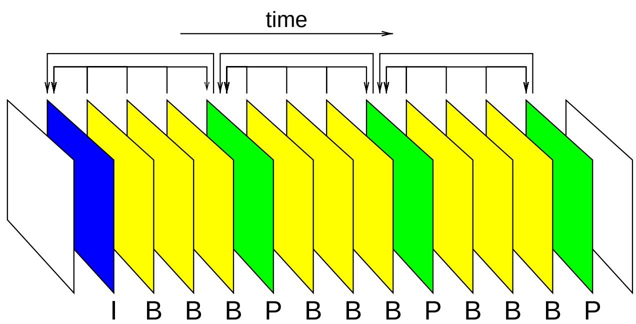
MPEG motion compensation



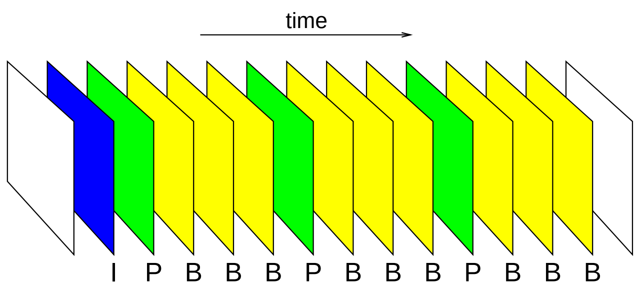
Each MPEG image is split into 16×16-pixel large *macroblocks*. The predictor forms a linear combination of the content of one or two other blocks of the same size in a preceding (and following) reference image. The relative positions of these reference blocks are encoded along with the differences.

MPEG reordering of reference images

Display order of frames:

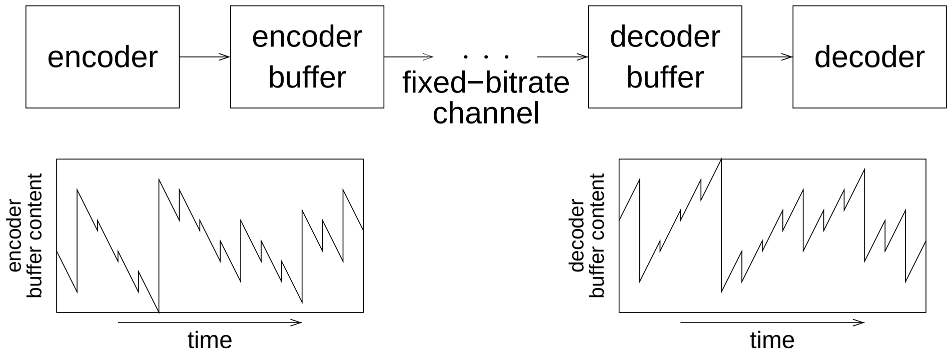


Coding order:



MPEG distinguishes between I-frames that encode an image independent of any others, P-frames that encode differences to a previous P- or I-frame, and B-frames that interpolate between the two neighboring B- and/or I-frames. A frame has to be transmitted before the first B-frame that makes a forward reference to it. This requires the coding order to differ from the display order.

MPEG system layer: buffer management



MPEG can be used both with variable-bitrate (e.g., file, DVD) and fixed-bitrate (e.g., ISDN) channels. The bitrate of the compressed data stream varies with the complexity of the input data and the current quantization values. Buffers match the short-term variability of the encoder bitrate with the channel bitrate. A control loop continuously adjusts the average bitrate via the quantization values to prevent under- or overflow of the buffer.

The MPEG system layer can interleave many audio and video streams in a single data stream. Buffers match the bitrate required by the codecs with the bitrate available in the multiplex and encoders can dynamically redistribute bitrate among different streams.

MPEG encoders implement a 27 MHz clock counter as a timing reference and add its value as a *system clock reference (SCR)* several times per second to the data stream. Decoders synchronize with a phase-locked loop their own 27 MHz clock with the incoming SCRs.

Each compressed frame is annotated with a *presentation time stamp (PTS)* that determines when its samples need to be output. *Decoding timestamps* specify when data needs to be available to the decoder.

MPEG audio coding

Three different algorithms are specified, each increasing the processing power required in the decoder.

Supported sampling frequencies: 32, 44.1 or 48 kHz.

Layer I

→ Waveforms are split into segments of 384 samples each (8 ms at 48 kHz).

→ Each segment is passed through an orthogonal filter bank that splits the signal into 32 subbands, each 750 Hz wide (for 48 kHz).
This approximates the critical bands of human hearing.

→ Each subband is then sampled at 1.5 kHz (for 48 kHz).
12 samples per window → again 384 samples for all 32 bands

→ This is followed by scaling, bit allocation and uniform quantization.
Each subband gets a 6-bit scale factor (2 dB resolution, 120 dB range, like floating-point coding). Layer I uses a fixed bitrate without buffering. A bit allocation step uses the psychoacoustic model to distribute all available resolution bits across the 32 bands (0–15 bits for each sample). With a sufficient bit rate, the quantization noise will remain below the sensation limit.

→ Encoded frame contains bit allocation, scale factors and sub-band samples.

Layer II

Uses better encoding of scale factors and bit allocation information.

Unless there is significant change, only one out of three scale factors is transmitted. Explicit zero code leads to odd numbers of quantization levels and wastes one codeword. Layer II combines several quantized values into a *granule* that is encoded via a lookup table (e.g., 3×5 levels: 125 values require 7 instead of 9 bits). Layer II is used in Digital Audio Broadcasting (DAB).

Layer III

⟶ Modified DCT step decomposes subbands further into 18 or 6 frequencies

⟶ dynamic switching between MDCT with 36-samples (28 ms, 576 freq.) and 12-samples (8 ms, 192 freq.)
enables control of pre-echos before sharp percussive sounds (Heisenberg)

⟶ non-uniform quantization

⟶ Huffman entropy coding

⟶ buffer with short-term variable bitrate

⟶ joint stereo processing

MPEG audio layer III is the widely used "MP3" music compression format.

Psychoacoustic models

MPEG audio encoders use a psychoacoustic model to estimate the spectral and temporal masking that the human ear will apply. The subband quantization levels are selected such that the quantization noise remains below the masking threshold in each subband.

The masking model is not standardized and each encoder developer can chose a different one. The steps typically involved are:

⟶ Fourier transform for spectral analysis

⟶ Group the resulting frequencies into "critical bands" within which masking effects will not vary significantly

⟶ Distinguish tonal and non-tonal (noise-like) components

⟶ Apply masking function

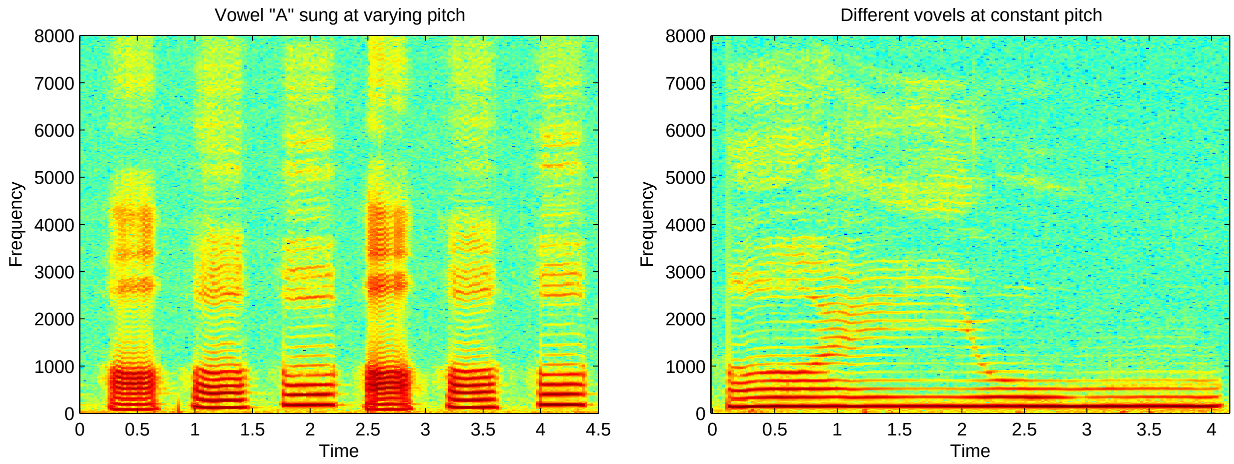
⟶ Calculate threshold per subband

⟶ Calculate signal-to-mask ratio (SMR) for each subband

Masking is not linear and can be estimated accurately only if the actual sound pressure levels reaching the ear are known. Encoder operators usually cannot know the sound pressure level selected by the decoder user. Therefore the model must use worst-case SMRs.

Voice encoding

The human vocal tract can be modeled as a variable-frequency impulse source (used for vowels) and a noise source (used for fricatives and plosives), to which a variable linear filter is applied which is shaped by mouth and tongue.



Vector quantization

A multi-dimensional signal space can be encoded by splitting it into a finite number of volumes. Each volume is then assigned a single codeword to represent all values in it.

Example: The colour-lookup-table file format GIF requires the compressor to map RGB pixel values using vector quantization to 8-bit code words, which are then entropy coded.

Literature

References used in the preparation of this part of the course in addition to those quoted previously:

- D. Salomon: A guide to data compression standards. ISBN 0387952608, 2002.
- A.M. Kondoz: Digital speech – Coding for low bit rate communications systems. ISBN 047195064.
- L. Gulick, G. Gescheider, R. Frisina: Hearing. ISBN 0195043073, 1989.
- H. Schiffman: Sensation and perception. ISBN 0471082082, 1982.
- British Standard BS EN 60651: Sound level meters. 1994.

Exercise 1 Compare the quantization techniques used in the digital telephone network and in audio compact disks. Which factors to you think led to the choice of different techniques and parameters here?

Exercise 2 Which steps of the JPEG (DCT baseline) algorithm cause a loss of information? Distinguish between accidental loss due to rounding errors and information that is removed for a purpose.

Exercise 3 How can you rotate by multiples of $\pm 90^\circ$ or mirror a DCT-JPEG compressed image without losing any further information. Why might the resulting JPEG file not have the exact same file length?

Exercise 4 Decompress this G3-fax encoded line:
1101011011110111101100110100000000000001

Exercise 5 You adjust the volume of your 16-bit linearly quantizing soundcard, such that you can just about hear a 1 kHz sine wave with a peak amplitude of 200. What peak amplitude do you expect will a 90 Hz sine wave need to have, to appear equally loud (assuming ideal headphones)?

Some final thoughts about redundancy . . .

```
Aoccdrnig to rsceearh at Cmabrigde Uinervtisy, it
deosn't mttaer in waht oredr the ltteers in a wrod are,
the olny iprmoetnt tihng is taht the frist and lsat
ltteer be at the rghit pclae. The rset can be a total
mses and you can sitll raed it wouthit porbelm. Tihs is
bcuseae the huamn mnid deos not raed ervey lteter by
istlef, but the wrod as a wlohe.
```

. . . and perception

Count how many Fs there are in this text:

```
FINISHED FILES ARE THE RE-
SULT OF YEARS OF SCIENTIF-
IC STUDY COMBINED WITH THE
EXPERIENCE OF YEARS
```