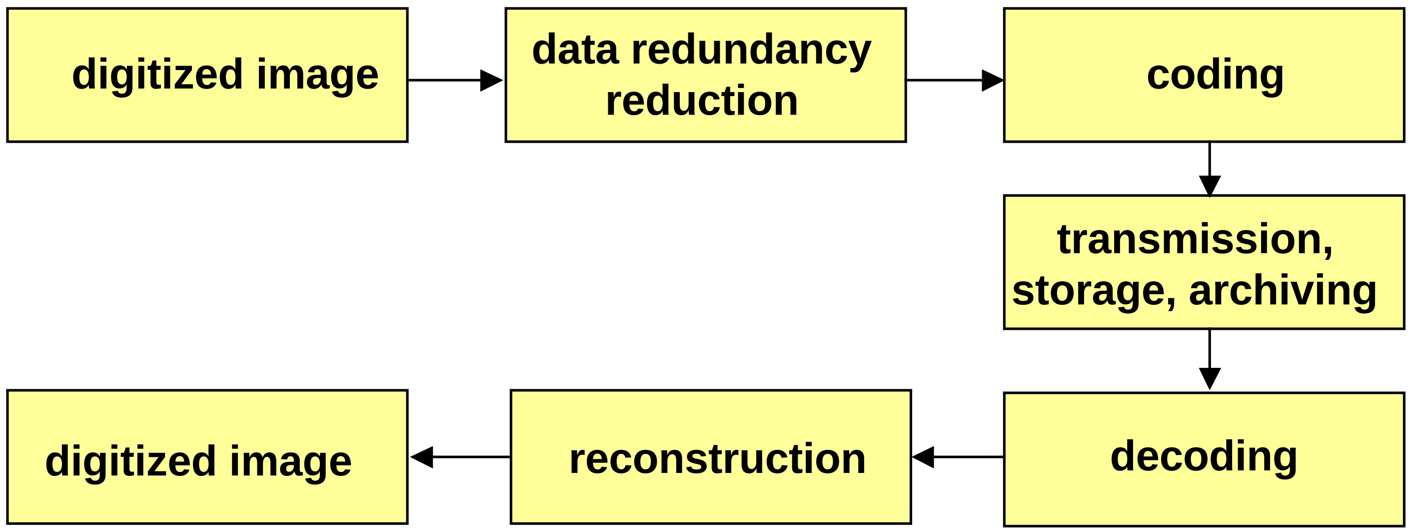
Image Data Compression

Image data compression is important for

- image archiving e.g. satellite data
- image transmission e.g. web data
- multimedia applications e.g. desk-top editing

Image data compression exploits redundancy for more efficient coding:



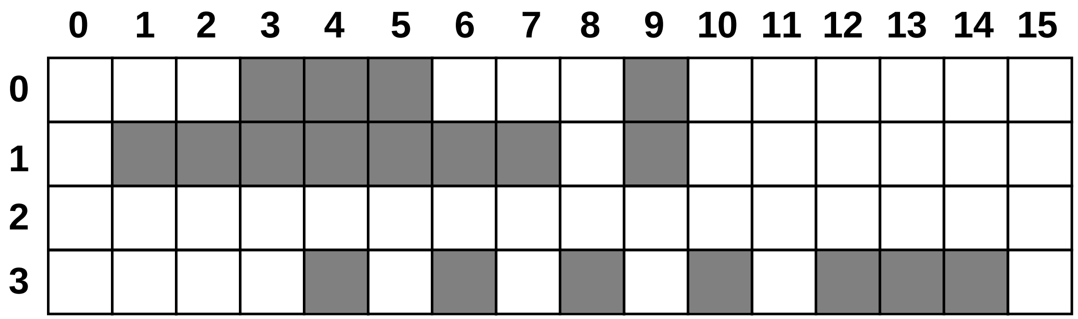
Run Length Coding

Images with repeating greyvalues along rows (or columns) can be compressed by storing "runs" of identical greyvalues in the format:

greyvalue1	repetition1	greyvalue2	repetition2	...

For B/W images (e.g. fax data) another run length code is used:

row #	column # run1 begin	column # run1 end	column # run2 begin	column # run2 end	...



run length coding:

(0 3 5 9 9)
(1 1 7 9 9)
(3 4 4 6 6 8 8 10 10 12 14)

Probabilistic Data Compression

A discrete image encodes information redundantly if

1. **the greyvalues of individual pixels are not equally probable**
2. **the greyvalues of neighbouring pixels are correlated**

Information Theory provides limits for minimal encoding of probabilistic information sources.

Redundancy of the encoding of individual pixels with G greylevels each:

$$r = b - H$$

b = number of bits used for each pixel $= \lceil \log_2 G \rceil$

$$H = \sum_{g=0}^{G-1} P(g) \log_2 \frac{1}{P(g)}$$

H = entropy of pixel source
= mean number of bits required to encode information of this source

The entropy of a pixel source with equally probable greyvalues is equal to the number of bits required for coding.

Huffman Coding

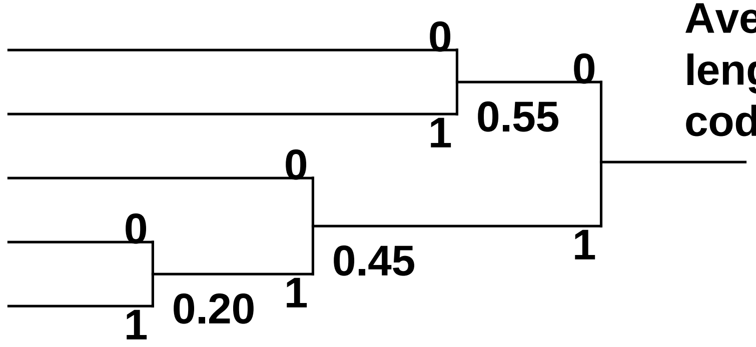
The Huffman coding scheme provides a <u>variable-length code</u> with minimal average code-word length, i.e. <u>least possible redundancy</u>, for a discrete message source. (Here messages are greyvalues)

1. **Sort messages along increasing probabilities such that $g^{(1)}$ and $g^{(2)}$ are the least probable messages**
2. **Assign 1 to code word of $g^{(1)}$ and 0 to codeword of $g^{(2)}$**
3. **Merge $g^{(1)}$ and $g^{(2)}$ by adding their probabilities**
4. **Repeat steps 1 - 4 until a single message is left.**

<u>Example</u>:

message	*probability*	*code word*
g1	0.3	00
g2	0.25	01
g3	0.25	10
g4	0.10	110
g5	0.10	111

coding tree



Entropy: H = 2.185

Average code word length of Huffman code: 2.2

Statistical Dependence

An image may be modelled as a set of <u>statistically dependent</u> random variables with a multivariate distribution $p(x_1, x_2, ..., x_N) = p(\underline{x})$.

Often the exact distribution is unknown and only <u>correlations</u> can be (approximately) determined.

<u>Correlation</u> of two variables:

$E[x_i x_j] = c_{ij}$

<u>Covariance</u> of two variables:

$E[(x_i - m_i)(x_j - m_j)] = v_{ij}$ with m_k = mean of x_k

<u>Correlation matrix</u>:

$$E[\underline{x}\,\underline{x}^T] = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \dots \\ c_{21} & c_{22} & c_{23} & \\ c_{31} & c_{32} & c_{33} & \\ \dots & & & \end{bmatrix}$$

<u>Covariance matrix</u>:

$$E[(\underline{x}-\underline{m})\,(\underline{x}-\underline{m})^T] = \begin{bmatrix} v_{11} & v_{12} & v_{13} & \dots \\ v_{21} & v_{22} & v_{23} & \\ v_{31} & v_{32} & v_{33} & \\ \dots & & & \end{bmatrix}$$

Uncorrelated variables need not be statistically independent:

$$E[x_i x_j] = 0 \quad \not\Rightarrow \quad p(x_i x_j) = p(x_i)\, p(x_j)$$

For Gaussian random variables, uncorrelatedness implies statistical independence.

Karhunen-Loève Transform

(also known as *Hotelling Transform* or *Principal Components Transform*)

Determine uncorrelated variables $\underline{y}$ from correlated variables $\underline{x}$ by a linear transformation.

$\underline{y} = A(\underline{x} - \underline{m})$

$E[\underline{y}\,\underline{y}^T] = A\,E[(\underline{x} - \underline{m})(\underline{x} - \underline{m})^T]\,A^T = A\,V\,A^T = D$ **D is a diagonal matrix**

- **An orthonormal matrix A which diagonalizes the real symmetric covariance matrix V always exists.**
- **A is the matrix of eigenvectors of V, D is the matrix of corresponding eigenvalues.**

$\underline{x} = A^T\underline{y} + \underline{m}$ **reconstruction of $\underline{x}$ from $\underline{y}$**

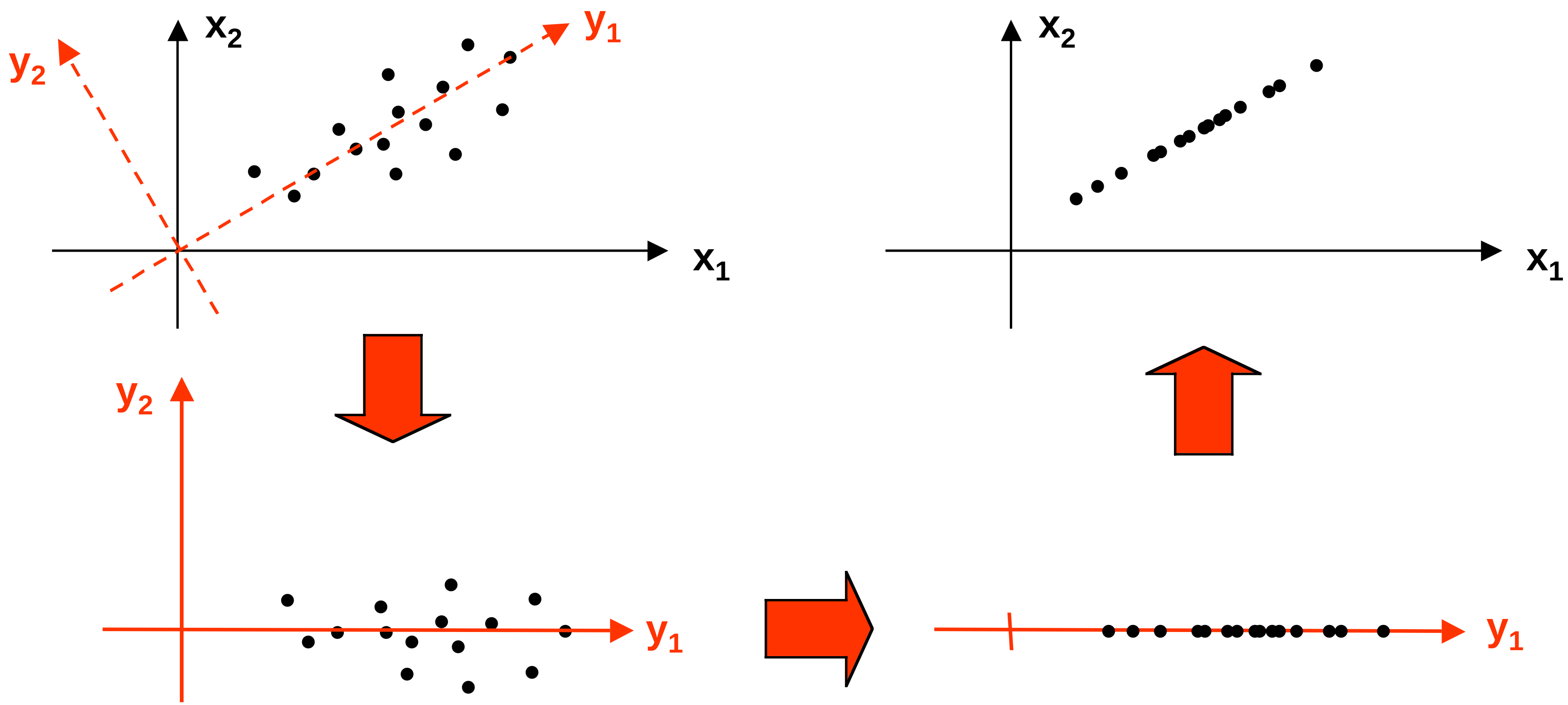
If $\underline{x}$ is viewed as a point in n-dimensional Euclidean space, then A defines a rotated coordinate system.

Illustration of Minimum-loss Dimension Reduction

Using the Karhunen-Loève transform data compression is achieved by

- **changing (rotating) the coordinate system**
- **omitting the least informative dimension(s) in the new coodinate system**

<u>Example</u>:



Compression and Reconstruction with the Karhunen-Loève Transform

Assume that the eigenvalues λ_n and the corresponding eigenvectors in A are sorted in decreasing order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \ldots \\ 0 & \lambda_2 & 0 & \\ 0 & 0 & \lambda_3 & \\ \ldots & & & \end{bmatrix}$$

Eigenvectors $\underline{a}$ and eigenvalues λ are defined by $V\,\underline{a} = \lambda\,\underline{a}$ and can be determined by solving $\det[V - \lambda I] = 0$.

There exist special procedures for determining eigenvalues of real symmetric matrices V.

Then $\underline{x}$ can be transformed into a K-dimensional vector $\underline{y}_K$, $K < N$, with a transformation matrix A_K containing only the first K eigenvectors of A corresponding to the largest K eigenvalues.

$$\underline{y}_K = A_K(\underline{x} - \underline{m})$$

The approximate reconstruction $\underline{x}'$ minimizing the MSE is

$$\underline{x}' = A_K^T\,\underline{y}_K + \underline{m}$$

Hence $\underline{y}_K$ can be used for data compression!

Example for Karhunen-Loève Compression

$N = 3$

$\underline{x}^T = [x_1\ x_2\ x_3]$

$$V = \begin{bmatrix} 2 & -0{,}866 & -0{,}5 \\ -0{,}866 & 2 & 0 \\ -0{,}5 & 0 & 2 \end{bmatrix} \qquad \underline{m} = \underline{0}$$

$\det(V - \lambda I) = 0 \Rightarrow \lambda_1 = 3 \quad \lambda_2 = 2 \quad \lambda_3 = 1$

$$A^T = \begin{bmatrix} 0{,}707 & 0 & 0{,}707 \\ -0{,}612 & 0{,}5 & 0{,}612 \\ -0{,}354 & -0{,}866 & 0{,}354 \end{bmatrix} \qquad D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compression into K=2 dimensions:

$$\underline{y}_2 = A_2\,\underline{x} = \begin{bmatrix} 0{,}707 & -0{,}612 & -0{,}354 \\ 0 & 0{,}5 & -0{,}866 \end{bmatrix} \underline{x}$$

Reconstruction from compressed values:

$$\underline{x}' = A_2^T\,\underline{y} = \begin{bmatrix} 0{,}707 & 0 \\ -0{,}612 & 0{,}5 \\ -0{,}354 & 0{,}354 \end{bmatrix} \underline{y}$$

Note the discrepancies between the original and the approximated values:

$x_1' = 0{,}5\ x_1 - 0{,}43\ x_2 - 0{,}25\ x_3$

$x_2' = -0{,}085\ x_1 - 0{,}625\ x_2 + 0{,}39\ x_3$

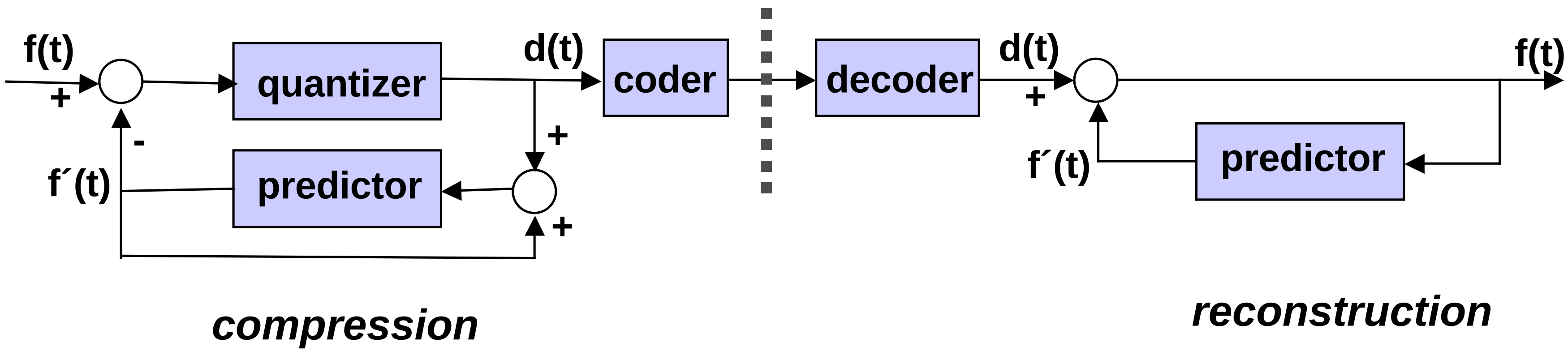
$x_3' = 0{,}273\ x_1 + 0{,}39\ x_2 + 0{,}25\ x_3$

Predictive Compression

Principle:

- estimate g_{mn}' from greyvalues in the neighbourhood of (mn)
- encode difference $d_{mn} = g_{mn} - g_{mn}'$
- transmit difference data + predictor

For a 1D signal this is known as Differential Pulse Code Modulation (DPCM):



Linear predictor for a neighbourhood of K pixels:

$$g_{mn}' = a_1 g_1 + a_2 g_2 + \ldots + a_K g_K$$

Computation of $a_1 \ldots a_K$ by minimizing the expected reconstruction error

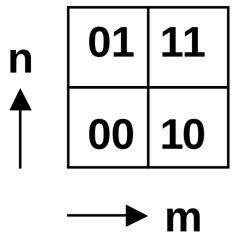
Example of Linear Predictor

For images, a linear predictor based on 3 pixels (3rd order) is often sufficient:

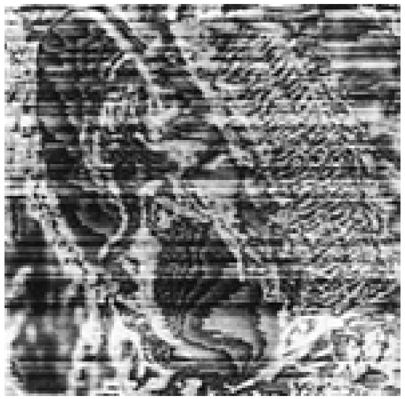
$$g_{mn}' = a_1\, g_{m,n-1} + a_2\, g_{m-1,n-1} + a_3\, g_{m-1,n}$$

If g_{mn} is a zero mean stationary random process with autocorrelation C, then minimizing the expected error gives

$$a_1 c_{00} + a_2 c_{01} + a_3 c_{11} = c_{10}$$
$$a_1 c_{01} + a_2 c_{00} + a_3 c_{10} = c_{11}$$
$$a_1 c_{11} + a_2 c_{10} + a_3 c_{00} = c_{01}$$



This can be solved for a_1, a_2, a_3 using Cramer´s Rule.



Example:

Predictive compression with 2nd order predictor and Huffman coding, ratio 6.2

Left: Reconstructed image

Right: Difference image (right) with maximal difference of 140 greylevels

Discrete Cosine Transform (DCT)

Discrete Cosine Transform is commonly used in image compression, e.g. in JPEG (Joint Photographic Expert Group) Baseline System standard.

Definition of DCT:

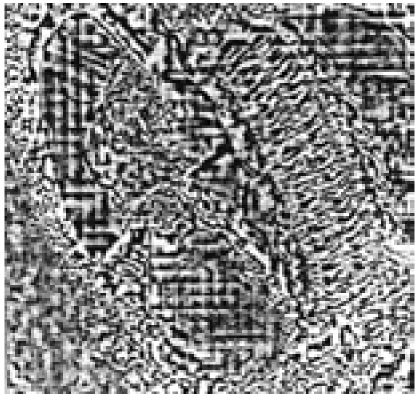
$$G_{00} = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_{mn}$$

$$G_{uv} = \frac{1}{2N^3} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_{mn} \cos[(2m+1)u\pi] \; \cos[(2n+1)v\pi]$$

Inverse DCT:

$$g_{mn} = \frac{1}{N} G_{00} + \frac{1}{2N^3} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} G_{uv} \cos[(2m+1)u\pi] \; \cos[(2n+1)v\pi]$$

Compression is accomplished by blockwise DCT + Huffman coding



Example:

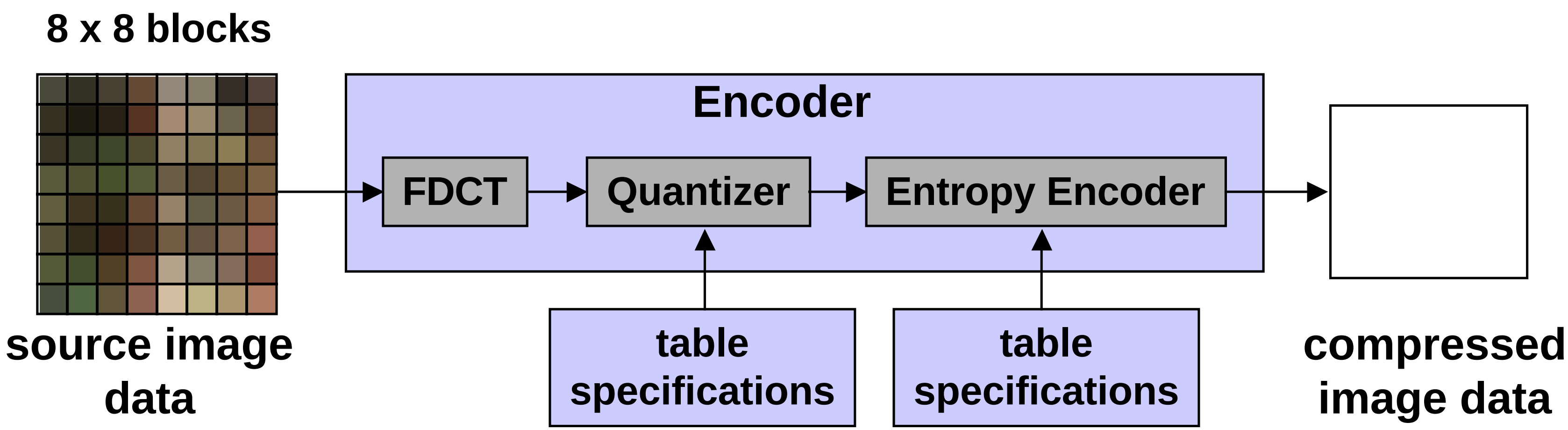
DCT compression with ratio 5.6

Left: Reconstructed image

Right: Difference image (right) with maximal difference of 125 greylevels

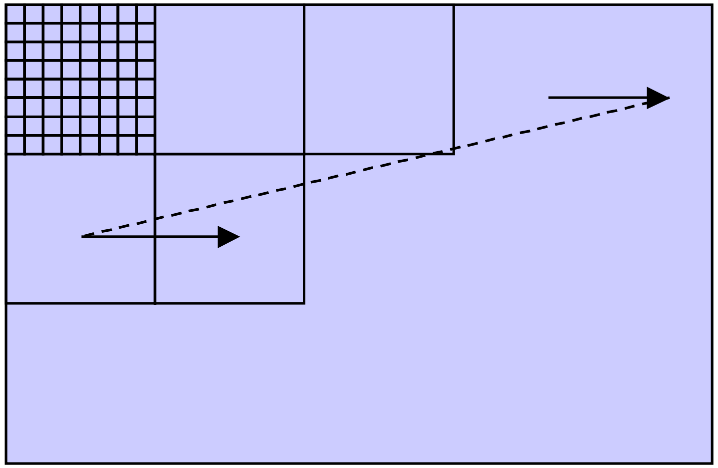
Principle of Baseline JPEG

(Source: Gibson et al., Digital Compression for Multimedia, Morgan Kaufmann 98)

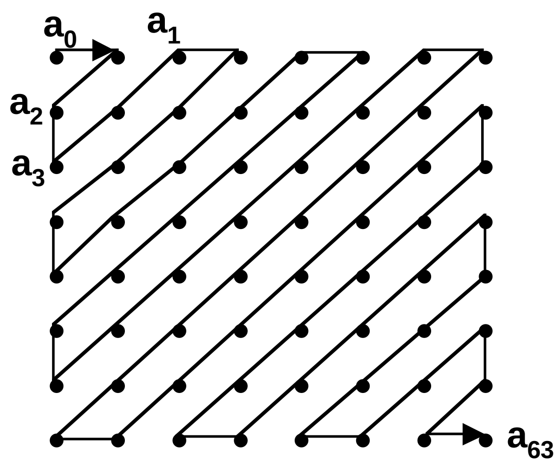


- partition image into 8 x 8 blocks, left-to-right, top-to-bottom
- compute Discrete Cosine Transform (DCT) of each block
- quantize coefficients according to psychovisual quantization tables
- order DCT coefficients in zigzag order
- perform runlength coding of bitstream of all coefficients of a block
- perform Huffman coding for symbols formed by bit patterns of a block

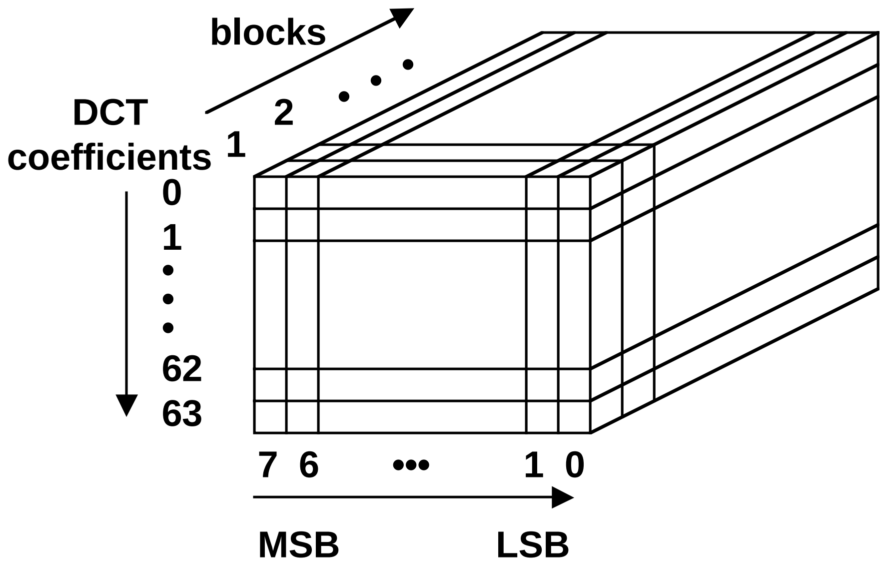
Illustrations for Baseline JPEG



partitioning the image into blocks



DCT coefficient ordering for
efficient runlength coding

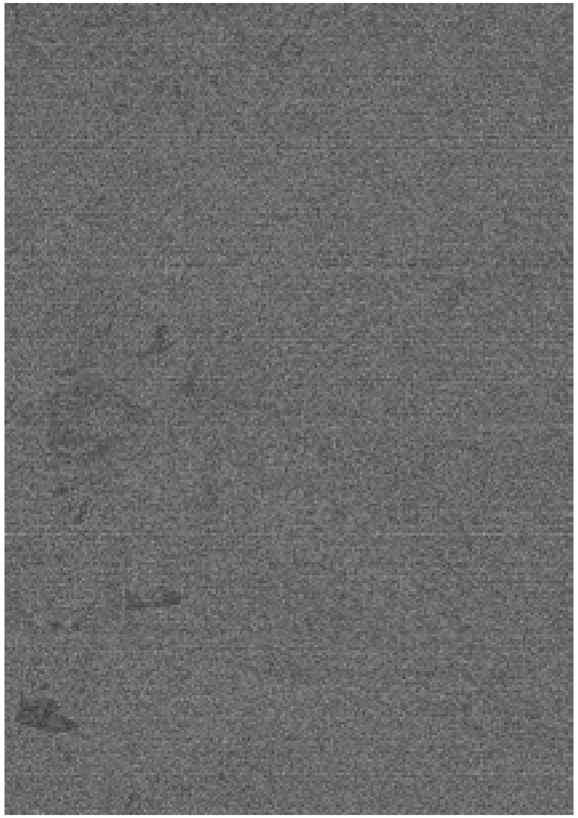


transmission sequence
for blocks of image

JPEG-compressed Image

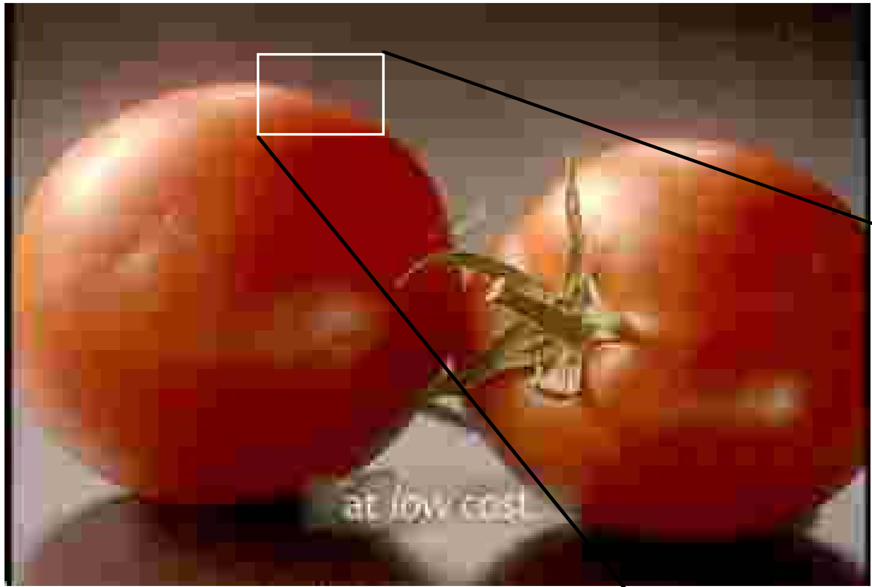
original
5.8 MB

JPEG-compressed
450 KB



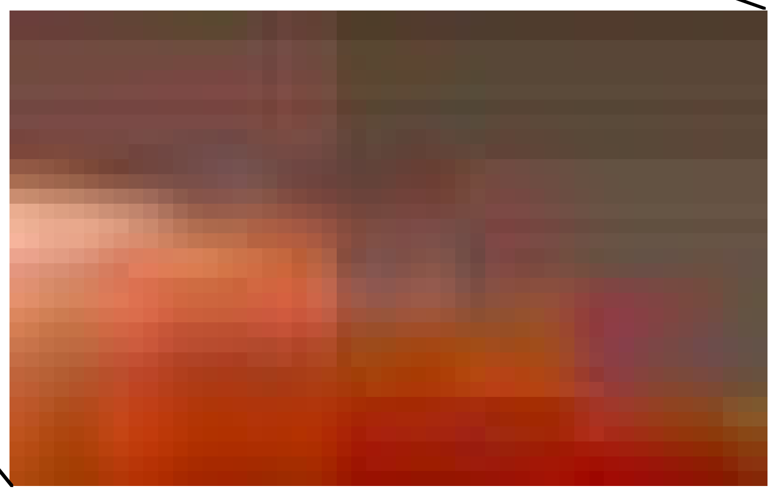
difference image
standard deviation of luminance differences: 1,44

Problems with Block Structure of JPEG



JPEC encoding with compression ratio 1:70

block boundaries are visible



Progressive Encoding

Progressive encoding allows to first transmit a coarse version of the image which is then progressively refined (convenient for browsing applications).

<u>Spectral selection</u>

1. transmission: DCT coefficients a_0 ... a_{k1}

2. transmission: DCT coefficients a_{k1} ... a_{k2}

⋮

low frequency coefficients first

<u>Successive approximation</u>

1. transmission: bits 7 ... n_1

2. transmission: bits n_1+1 ... n_2

⋮

most significant bits first

MPEG Compression

Original goal:

Compress a 120 Mbps video stream to be handled by a CD with 1 Mbps.

Basic procedure:

- **temporal prediction to exploit redundancy between image frames**
- **frequency domain decomposition using the DCT**
- **selective reduction of precision by quantization**
- **variable length coding to exploit statistical redundancy**
- **additional special techniques to maximize efficiency**

Motion compensation:

16 x 16 blocks luminance with 8 x 8 blocks chromaticity of the current image frame are transmitted in terms of

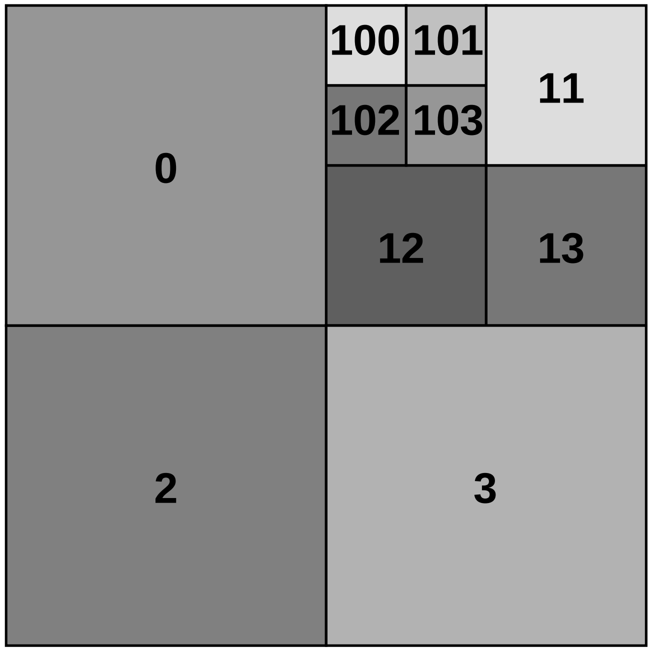
- an offset to the best-fitting block in a reference frame (motion vector)

- the compressed differences between the current and the reference block

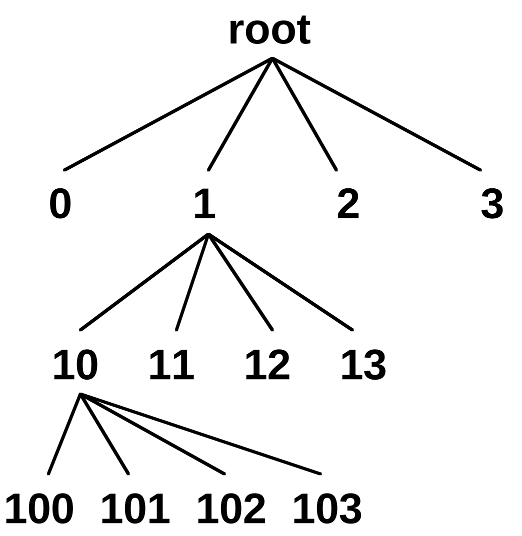
Quadtree Image Representation

Properties of quadtree:

- **every node represents a squared image area, e.g. by its mean greyvalue**
- **every node has 4 children except leaf nodes**
- **children of a node represent the 4 subsquares of the parent node**
- **nodes can be refined if necessary**



quadtree structure:



Quadtree Image Compression

A complete quadtree represents an image of N = 2^Kx 2^K pixels with 1 + 4 + 16 + ... + 2^{2K} nodes ≈ 1.33 N nodes.

An image may be compressed by

- **storing at every child node the <u>greyvalue difference</u> between child and parent node**
- **<u>omitting</u> subtrees with equal greyvalues**

Quadtree image compression supports <u>progressive image transmission</u>:

- **images are transmitted by increasing quadtree levels, i.e. images are progressively refined**
- **intermediate image representations provide useful information, e.g. for image retrieval**