matfyz

Computer Graphics Charles University

Compression methods: the 1st generation

© 1998-2017 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
http://cgg.mff.cuni.cz/~pepca/

Basic terminology

Source alphabet: $S = \{ x_1, x_2, \ldots x_n \}$

Probability of occurence of the symbol x_i is $\underline{p_i}$

Code K of the symbol x_i has length $\underline{d_i}$

Message (source string) is sequence:

$$X = x_{i_1}, x_{i_2}, \ldots x_{i_k}$$

Entropy (information) of the symbol x_i:

$$E_i = -\log_2 p_i \quad \textbf{bits}$$

Basic terminology II

Average entropy (information) of a symbol:

$$AE = \sum_{i=1}^{n} p_i E_i = -\sum_{i=1}^{n} p_i \log_2 p_i \quad \text{bits}$$

Message entropy:

$$E(X) = -\sum_{j=1}^{k} p_{i_j} \log_2 p_{i_j} \quad \text{bits}$$

Length of the message X:

$$L(X) = \sum_{j=1}^{k} d_{i_j} \quad \text{bits}$$

Basic terminology III

Code redundancy **K** for message **X**:

$$R(K) = L(X) - E(X) = \sum_{j=1}^{k} \left(d_{i_j} + p_{i_j} \log_2 p_{i_j} \right) \quad \text{bits}$$

Average length of
a codeword for the code **K**:

$$AL(K) = \sum_{i=1}^{n} p_i d_i \quad \text{bits}$$

Average redundancy of the code **K**:

$$AR(K) = AL(K) - AE = \sum_{i=1}^{n} p_i \left(d_i + \log_2 p_i \right) \quad \text{bits}$$

Image compression terminology

Source sample (pixel value): $u_k, u_{j,k}$

Sample after **quantization (reconstructed)**: $\dot{u}_k, \dot{u}_{j,k}$

Sample prediction: $\overline{u}_k, \overline{u}_{j,k}$

Prediction error: $e_k, e_{j,k}$

Reconstruction quality

Mean squared error (MSE):

$$\mathbf{RMS}^2 = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(u_{i,j} - u^{\bullet}_{i,j}\right)^2$$

Signal to noise ratio:

$$\mathbf{SNR} = 10\log_{10}\frac{P^2}{RMS^2}\ \mathrm{dB}$$

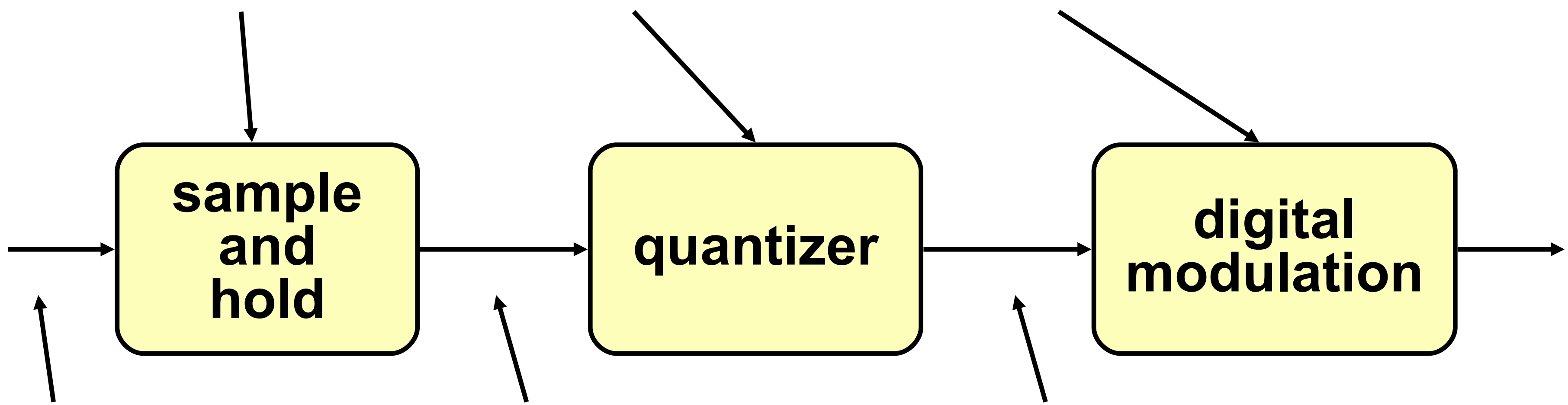
P is range of source values, e.g.:

$$\mathbf{SNR} = 10\log_{10}\frac{255^2}{RMS^2}\ \mathrm{dB}$$

Pulse-code modulation (PCM)

- **continuous analog signal – digital channel**

  time sampling, **quantization**, **coding** (modulation)



$u(t) \in \mathbf{R}$ — **everything is continuous**

$u_t \in \mathbf{R}$ — **real values in discrete time**

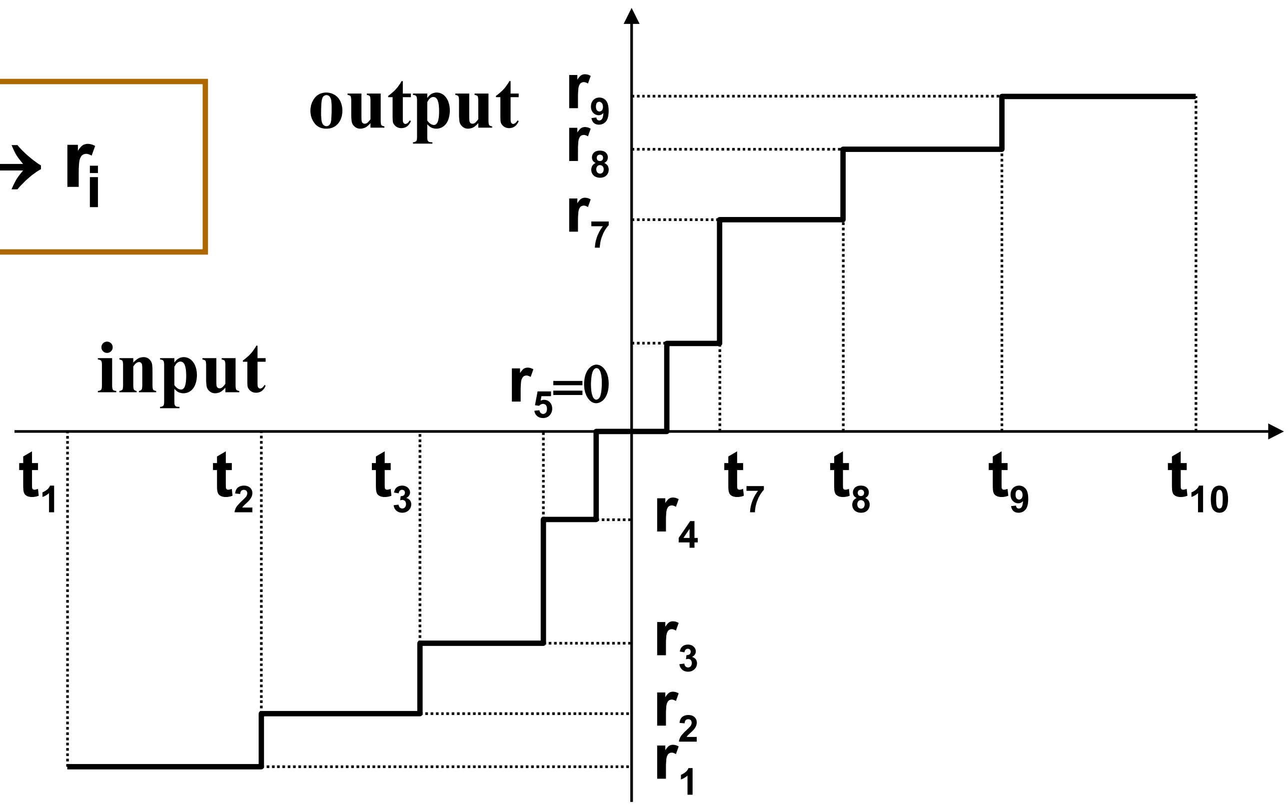
$r_t \in \mathbf{N}$ — **discrete values in discrete time**

Quantization

Decision values: $\{\ t_i,\ i = 1,2,\ldots L+1 \ \ t_i < t_{i+1}\ \}$

Reconstruction values: $\{\ r_i,\ i = 1,2,\ldots L\ \}$

$$q: \langle t_i, t_{i+1}] \rightarrow r_i$$



Examples:

A-law, µ-law

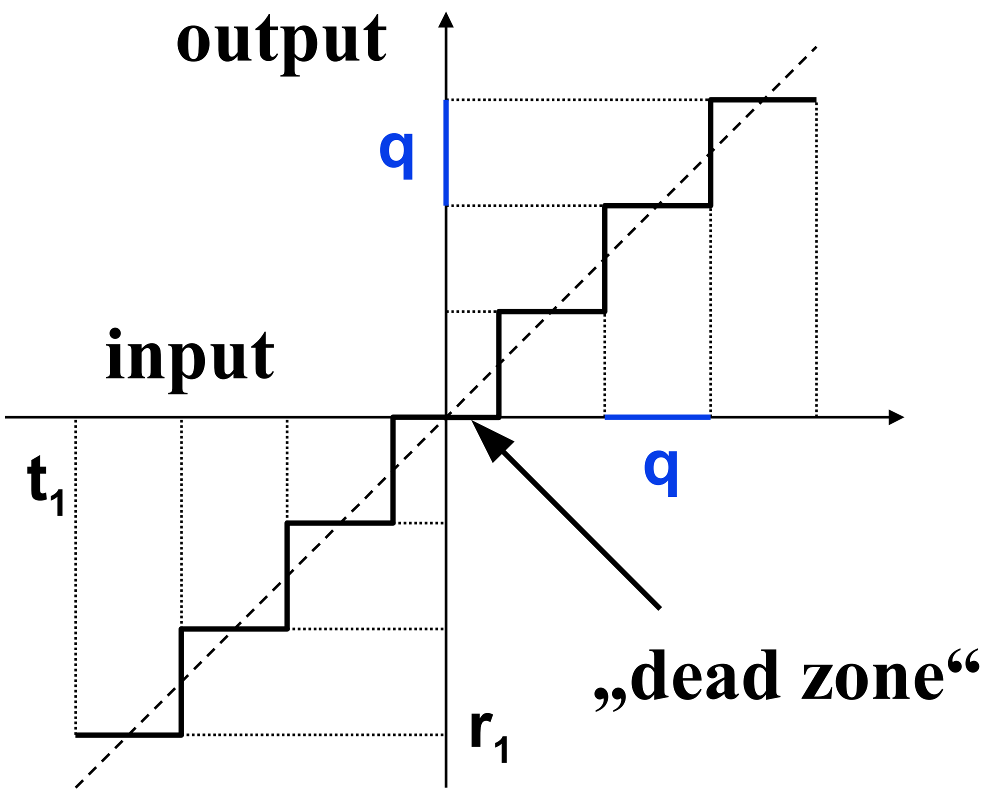
Linear quantizer

Best for **uniformly distributed source values**:

$$t_k = t_1 + (k-1)\,q$$

$$r_k = t_k + q/2$$

$$q = \frac{t_{L+1} - t_1}{L}$$



Lloyd-Max quantizer

Minimizes **mean squared error of the quantization**:

$$E = E\left[\left(u - u^{\bullet}\right)^2\right] = \int_{t_1}^{t_{T+1}} \left[x - u^{\bullet}(x)\right]^2 p(x)\ dx$$

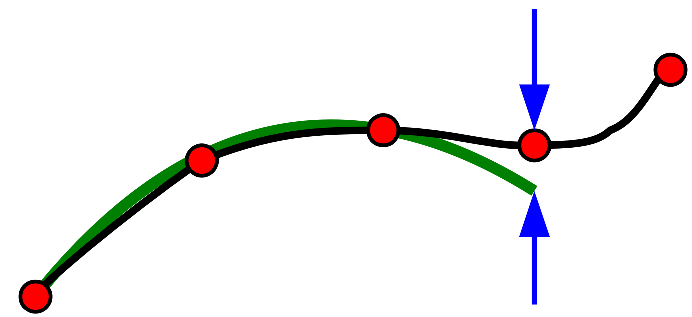
p(x) .. probability density function of **x** (discr. – histogram)

$$t_k = \tfrac{1}{2}\left(r_{k-1} + r_k\right)$$

$$r_k = \int_{t_k}^{t_{k+1}} x\,p(x)\ dx \Big/ \int_{t_k}^{t_{k+1}} p(x)\ dx$$

Predictive compression

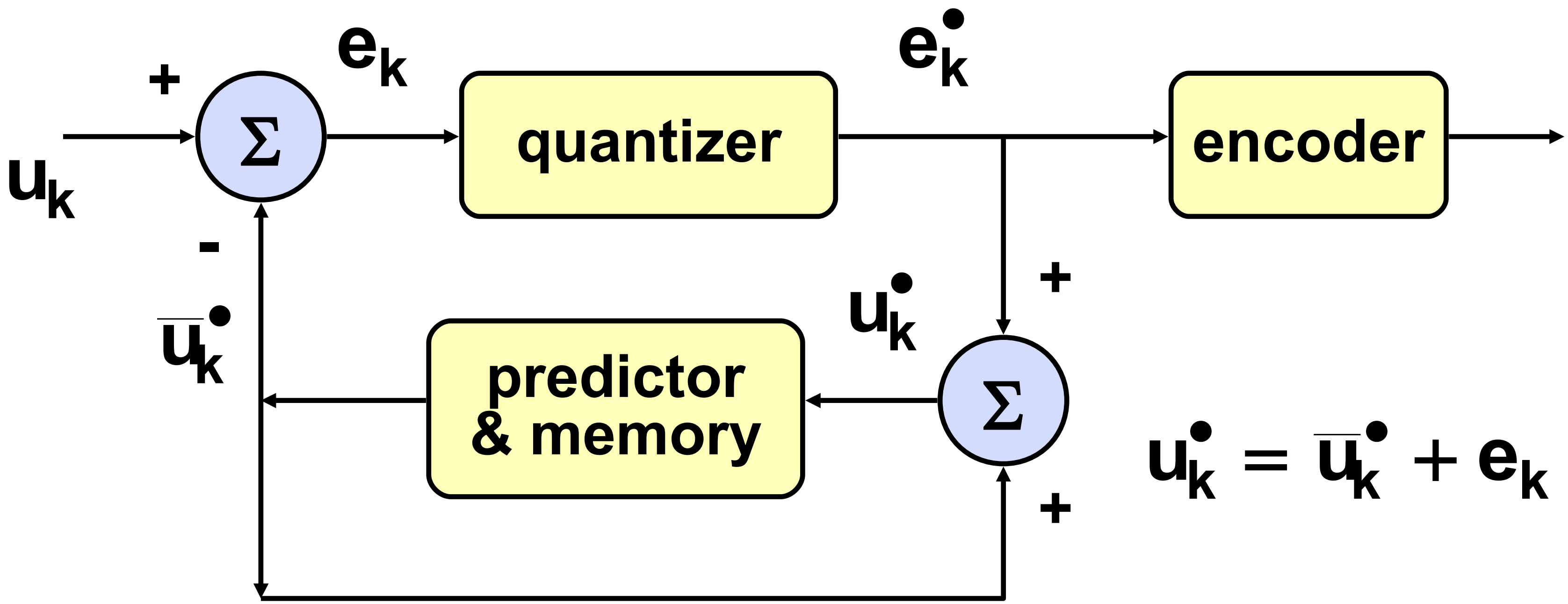
- **neighbour samples are dependent** (spatial correlation)
  – pixel values usually change slowly



- implementation using **predictor function**
  – previous (neighbour) samples are using to predict recent value
  – only residuals are coded (differences between predicted and actual values)
  – if successful, residuals have smaller amplitude

Predictive quantization

DPCM encoder (Differential Pulse-Code Modulation):



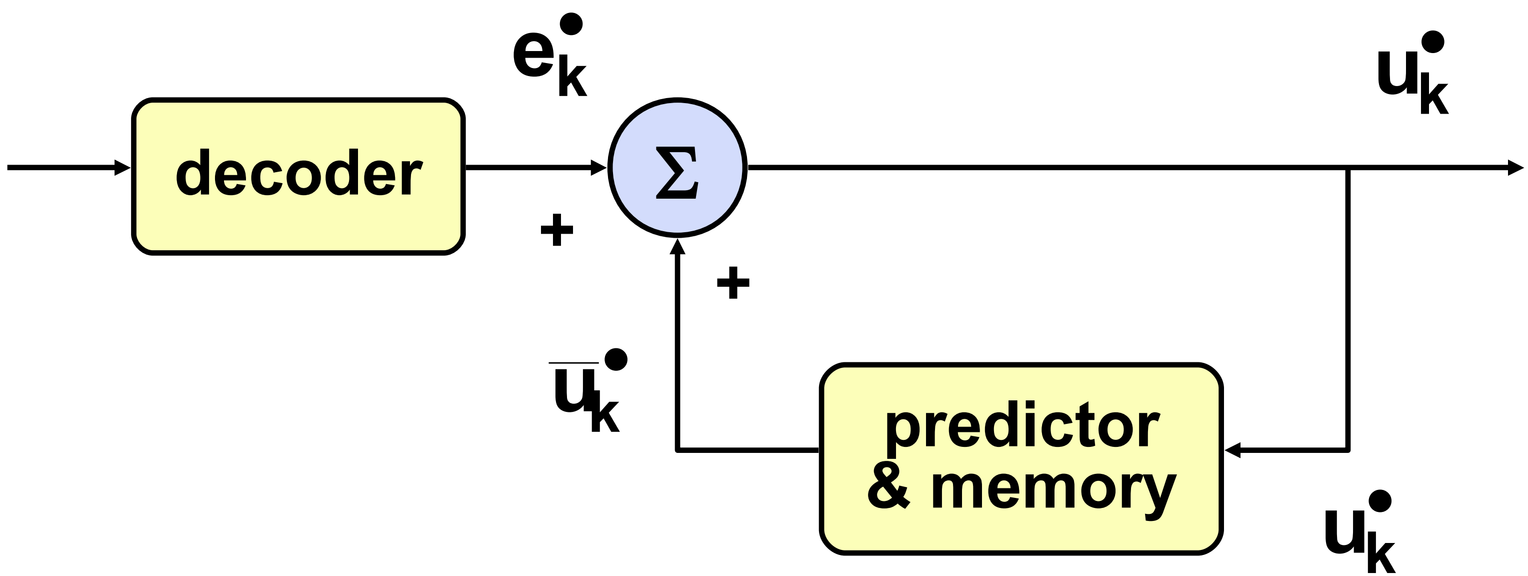
$$u_k^{\bullet} = \overline{u}_k^{\bullet} + e_k$$

$$e_k = u_k - \overline{u}_k^{\bullet}$$

prediction: $\overline{u}_k^{\bullet} = P\left(u_{k-1}^{\bullet}, u_{k-2}^{\bullet}, \ldots\right)$

Predictive quantization II

DPCM decoder:



Delta modulation

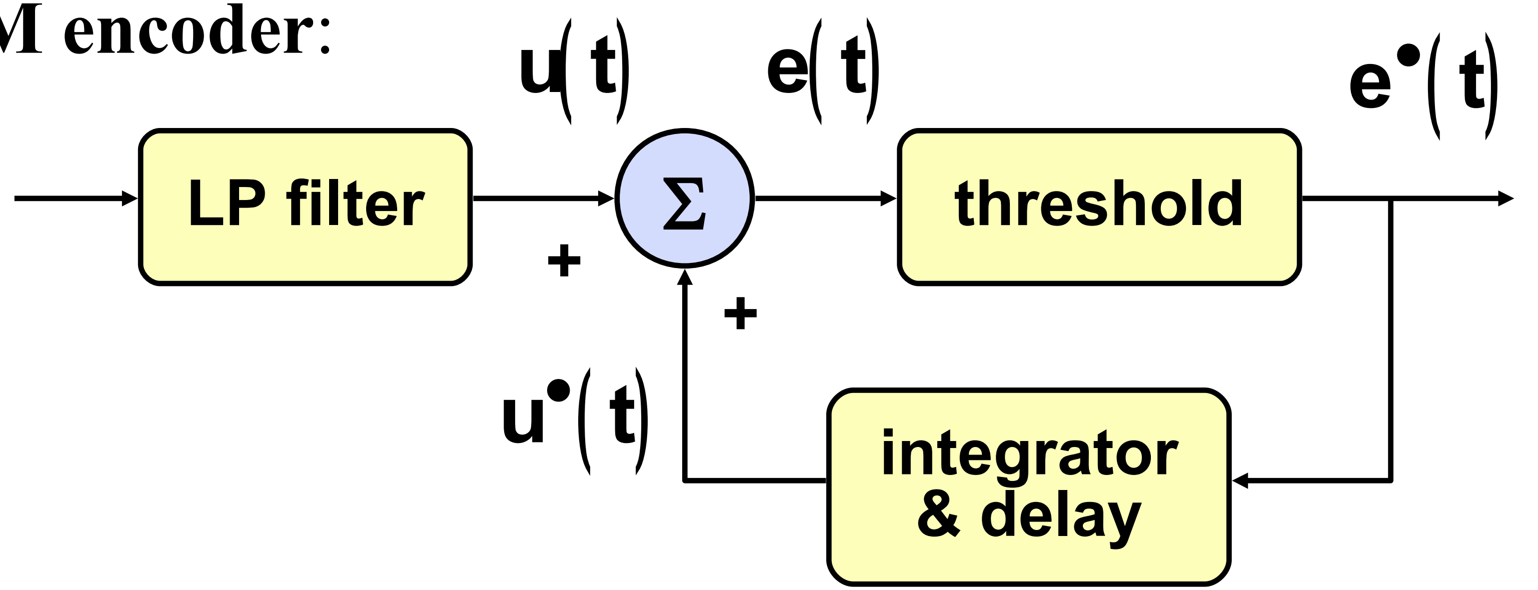
- quantizer has only **two output levels (+q,-q)**
  - basically a thresholding circuit
  - prediction is implemented by a <u>delay circuit</u>

$$\bar{u}_k^{\bullet} = u_{k-1}^{\bullet} \qquad e_k = u_k - u_{k-1}^{\bullet}$$

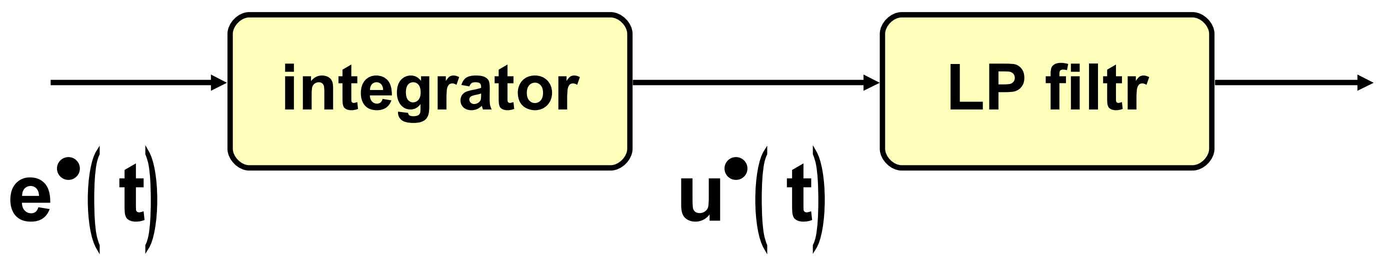
- output signal **needs not be quantized**
  - <u>integrator circuit</u> instead of predictor
- **analog signal coding** (audio)

Delta modulation II

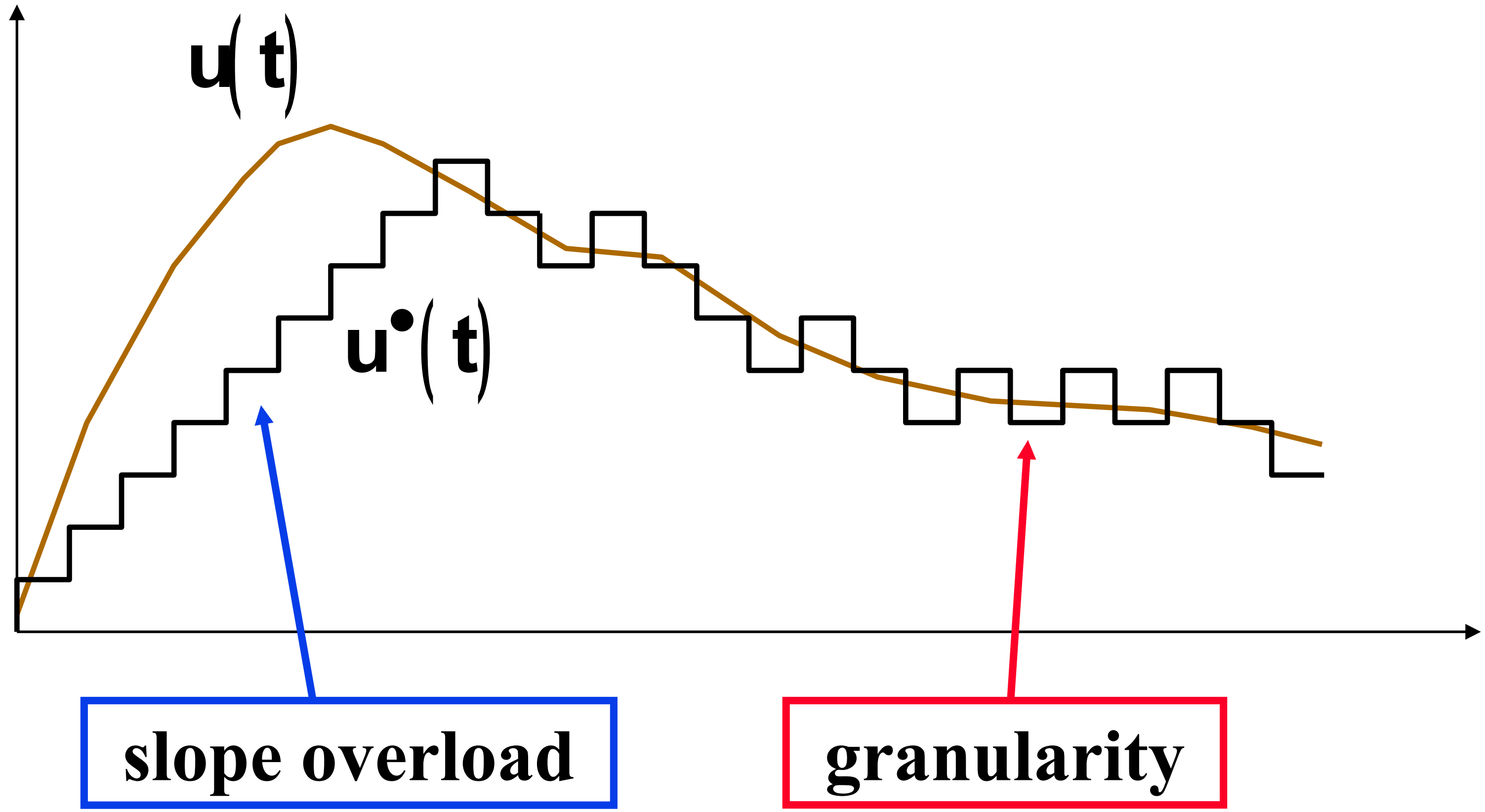
DM encoder:



DM decoder:



DM flaws



granularity reduction … **<u>three-state DM</u>** (**+q**,**0**,**-q**)

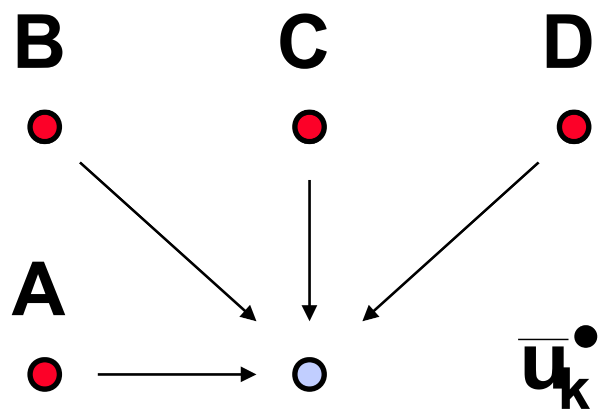
DPCM

- **Markov chain** (order of **p**)
  – mean value of the current sample is linearly dependent on **p** previous samples

$$\overline{u_k^{\bullet}} = \sum_{j=1}^{p} a_j \, u_{k-j}^{\bullet}$$

- **nonlinear** predictors
  – polynomials are more popular
- **2D** and **ND** predictors
  – match 2D image characteristics (3D in case of video)

2D DPCM



$$\overline{u}^{\bullet}_{k} = aA + bB + cC + dD$$

optimal case:

$$a = c = 0.95$$
$$b = -ac$$
$$d = 0$$

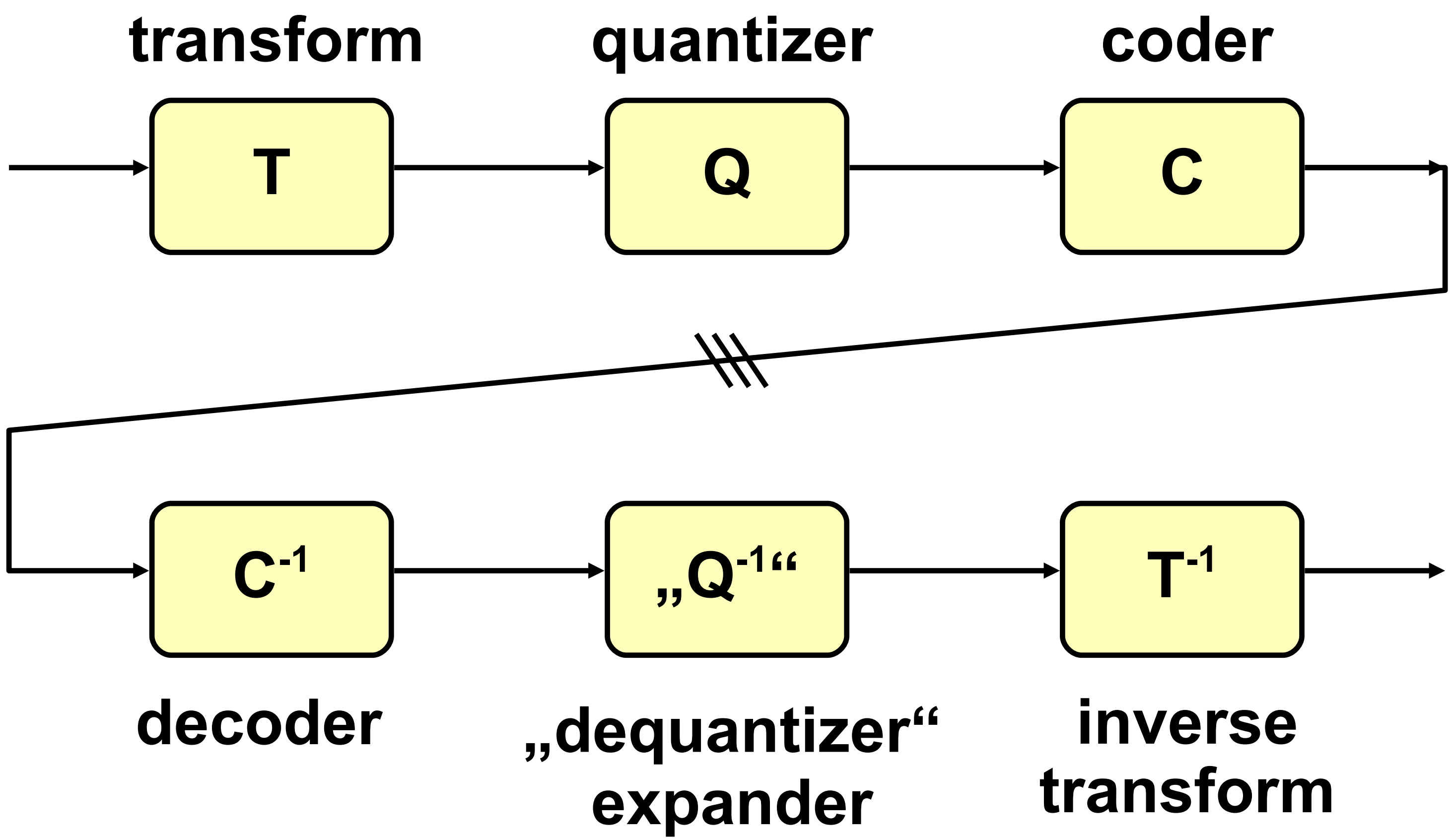
B/W image: **DPCM** is in theory by cca 20dB better than **PCM** (ratio 3-3.5 : 1)

Adaptive predictive techniques

- predictor sets for **different correlation directions** (horizontal, vertical, oblique, ..)
  - algorithm can adapt itselft – picks more effective predictor (max correlation)
- adaptive **gain control** (quantization error)
  - quality (distortion) of quantization is controlled acording to variance of predictor error
- **image region classification**
  - quick adaptation to local pixel neighbourhood
  - <u>block classification</u> – e.g. 16×16px blocks, ~4 patterns

Transform methods

General scheme:



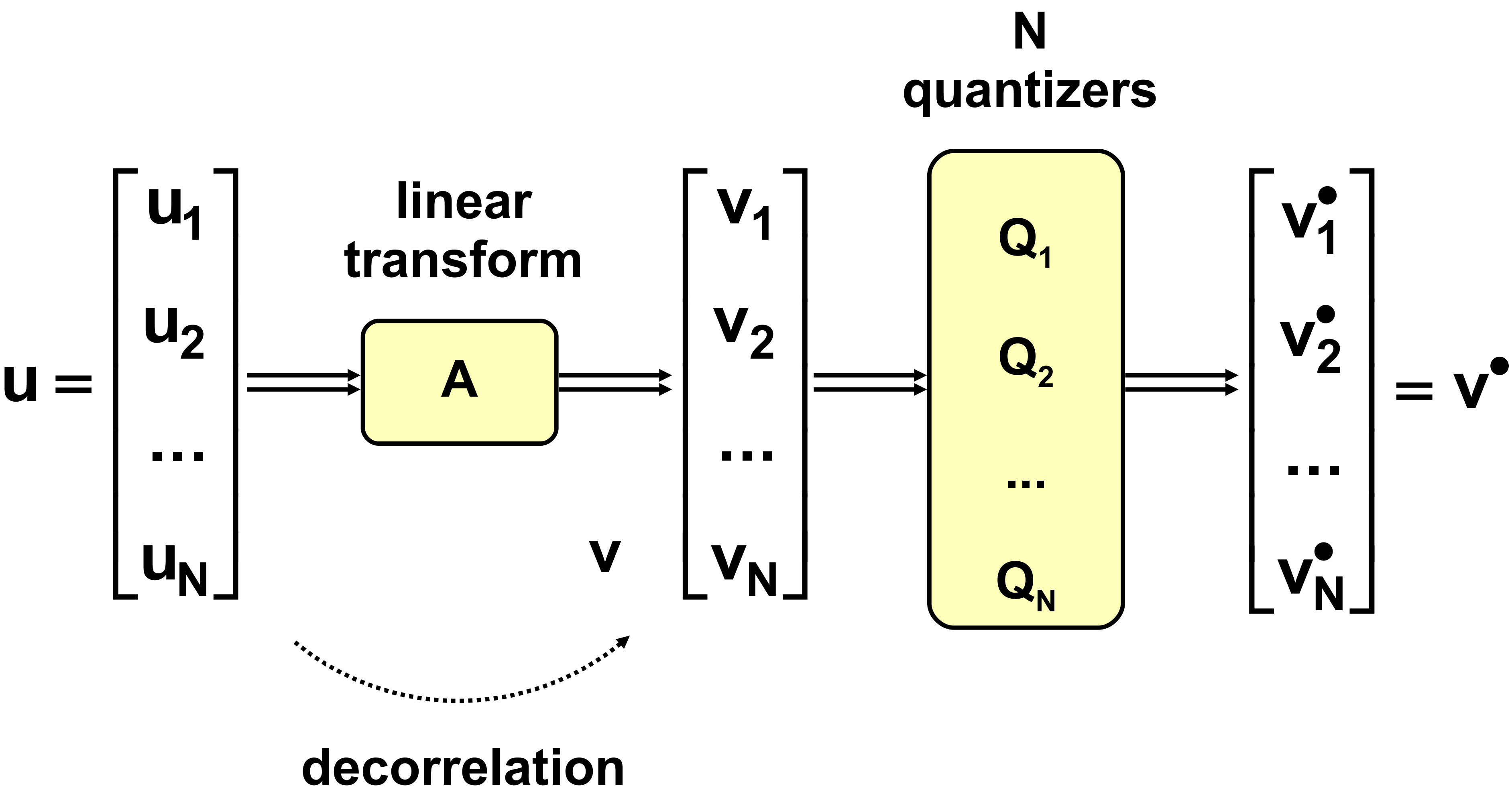
Transform methods II

- try to process the whole **image block (vector)** at once
  - – can be extended to „block quantization"
- **reducing** maximum amount of **redundancy** (among components of coded vector)
  - – actually used transform function is important
- **transform function**
  - – 1D (image scanline)
  - – 2D (the whole image, blocks **K×K**)

Transform function T

- the **most important information** should be concentrated in a **small number of coefficients**
  - throwing away of the rest of coefficients leads to a big compression ratio
- **quantization distortion** or **dropping coefficients** should have as little effect on image quality as possible
- output coefficients should be **little/not dependent**
- **computation efficiency:** **T** and $\mathbf{T}^{-1}$ (SW, HW)

1D linear transform methods



Karhunen-Loeve transform

- input values (vector components) have the **Gaussian distribution**
  – covariance matrix (dependency of the components) is known
- **goal:** to define transform coder with **minimum RMS error** and **maximum decorrelation**:
- optimal **Karhunen-Loeve** transform
  – coefficients ($\mathbf{v}_i$) are completely independent
  – big time complexity (covariance matrix → eigenvectors)

Practical transforms

- **suboptimal** transforms
  – a little bit worse than optimal KLT
  – better eficiency: fast algorihtms in **O(N log N)**, **O(N)**
- **goniometric** transforms (Fourier, sin, cos, Hartley)
  – complex & real arithmetics, good decorrelation
- **Walsh**-type transforms
  – partially constant functions, only additive operations are needed (Hadamard), very fast implementations
- **wavelets** (Haar, CDF(9,7), ...)
  – good representation of inhomogeneous data, hierarchical

Zonal coding of coefficients

Accuracy according to information value (variance) of coefficients. Example of the bit-allocation:

7	5	3	2	1	1	1	0
5	3	3	2	1	1	1	0
3	3	2	2	1	1	0	0
2	2	2	1	1	1	0	0
1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Zonal method (only a fixed region/zone is transferred)

Threshold coding

Only **K** coefficients with **biggest amplitudes** are transferred.

Example (actual coefficient values):

14	-5	4	8	7	-3	1	0
8	-3	5	4	-4	1	-1	0
-9	6	-8	5	3	3	0	0
7	3	6	1	1	4	0	2
4	1	-5	0	-2	1	0	0
-3	-1	4	2	0	-2	0	1
3	-2	-2	0	0	1	0	0
0	0	-1	0	1	0	0	0

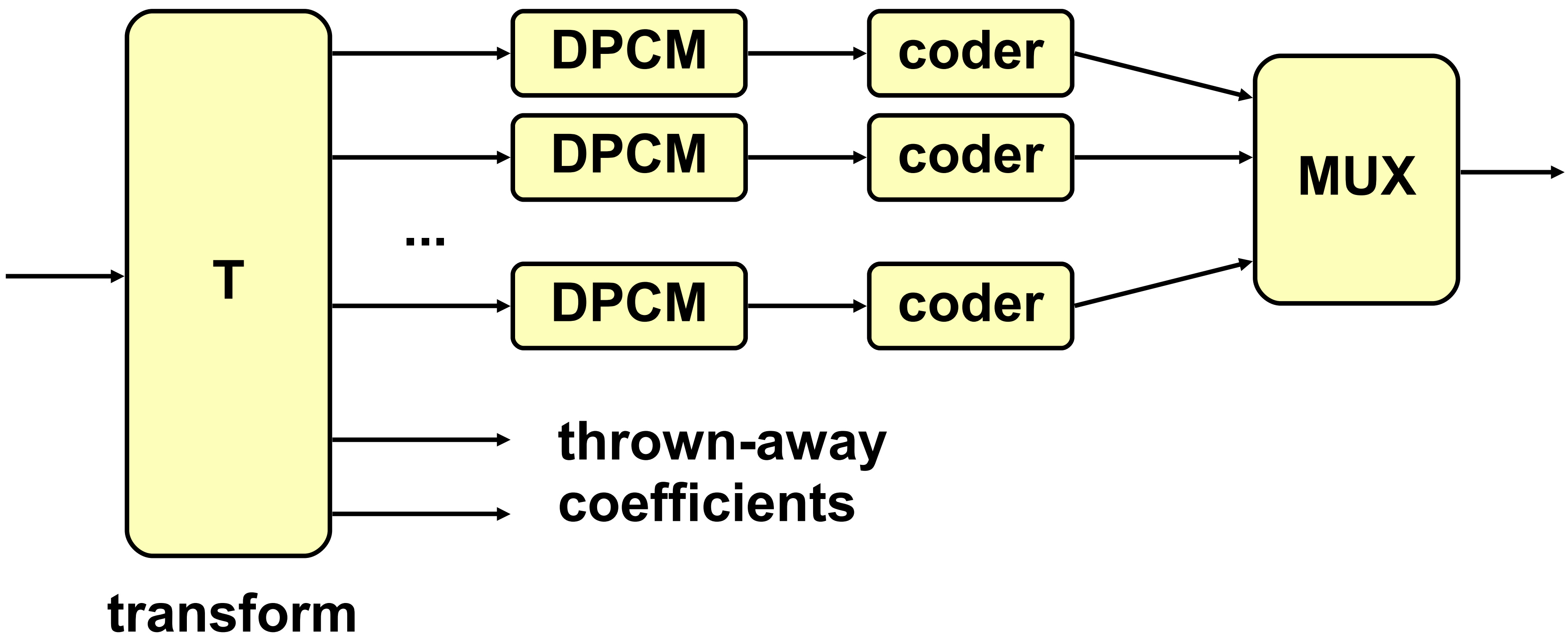
Transferred zone is different block to block (address coding)

Adaptive transform methods

- **(statistical) analysis** of image blocks
  – transform algorithm adaptates to actual data
- **transform function switching** (parameter-sets switching)
- **zone adaptation** (set of transferred coefficients)
- **quantizer adaptation**

Hybrid methods

Transform & **predictive** approach together

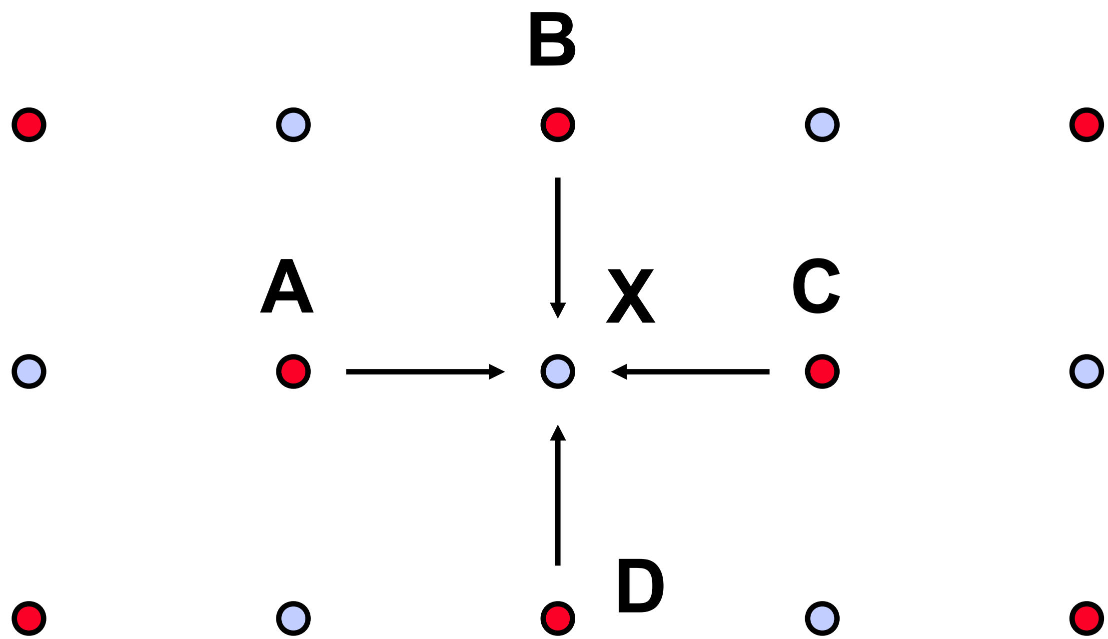


Interpolation methods

- only **some pixels/samples** are coded
  - omitted samples are reconstructed by interpolation
  - linear interpolation (higher-order polynomials are not much better)
- static methods
  - set of encoded samples is constant
- dynamic methods
  - adaptation to image characteristics (e.g. variance)

Alternating interpolation

Half of the pixels is encoded, adaptive **linear interpolation** is used for the rest:



$$X = \frac{A + C}{2} \quad \text{or} \quad \frac{B + D}{2}$$

(the one with less diference)

The End

More information:

- **A. Jain**: ***Image Data Compression: A Review***, Proceedings of the IEEE, vol.69, #3, 1981
- **A. Jain**: ***Fundamentals of Digital Image Processing***, Prentice-Hall, 1989
- **ed. by H.-M. Hang, J. Woods**: ***Handbook of Visual Communications***, Academic Press, San Diego, 1995