

Compression methods of the 1st generation

© 1998-2015 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

<http://cgg.mff.cuni.cz/~pepca/>



Basic terminology

Source alphabet: $\mathbf{S} = \{ \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \}$

Probability of occurrence of the symbol \mathbf{x}_i is \mathbf{p}_i

Code \mathbf{K} of the symbol \mathbf{x}_i has length \mathbf{d}_i

Message (source string) is sequence:

$$\mathbf{X} = \mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_k}$$

Entropy (information) of the symbol \mathbf{x}_i :

$$\mathbf{E}_i = -\log_2 \mathbf{p}_i \quad \text{bits}$$



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:

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

Length of the message \mathbf{X} :

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



Basic terminology III

Code redundancy \mathbf{K} for message \mathbf{X} :

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

Average length of a codeword for the code \mathbf{K} :

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

Average redundancy of the code \mathbf{K} :

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

Image compression terminology

Source sample (pixel value):

$u_k, u_{j,k}$

Sample after quantization (reconstructed):

$u_k^\circ, u_{j,k}^\circ$

Sample prediction:

$\bar{u}_k, \bar{u}_{j,k}$

Prediction error:

$e_k, e_{j,k}$



Reconstruction quality

Mean squared error (MSE):

$$\text{RMS}^2 = \frac{1}{MN} \sum_{i=1}^N \sum_{j=1}^M (u_{i,j} - \hat{u}_{i,j})^2$$

Signal to noise ratio:

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

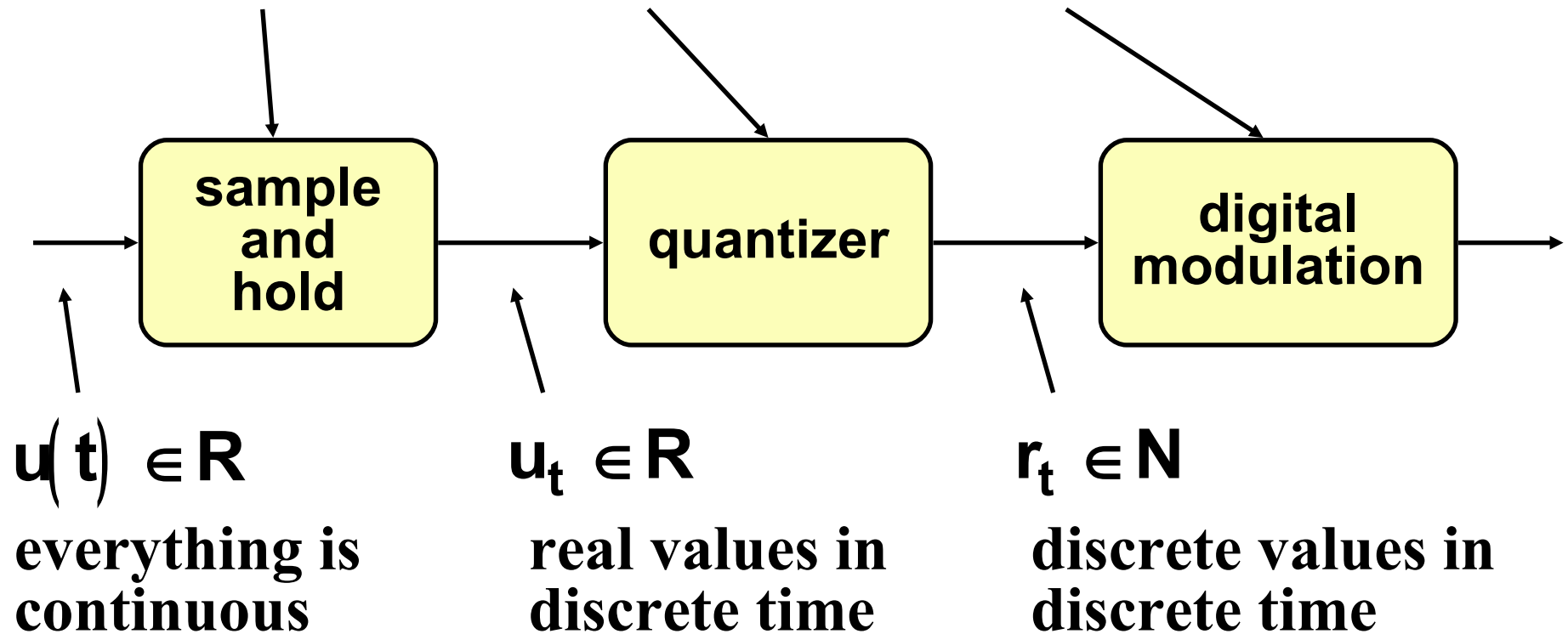
P is range of source values, e.g.:

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



Pulse-code modulation (PCM)

- continuous analog signal – digital channel
time sampling, quantization, coding (modulation)



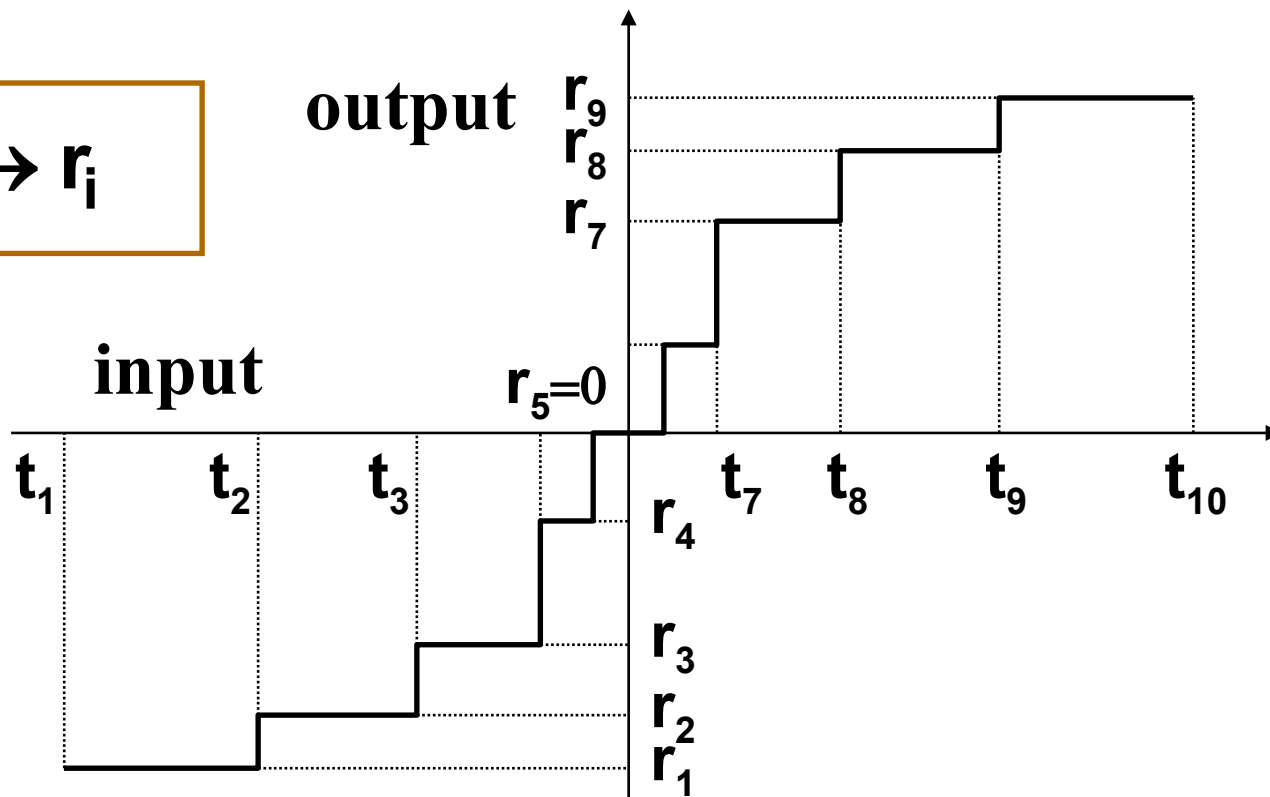


Quantization

Decision values: $\{ \underline{t}_i, i = 1, 2, \dots, L + 1 \mid t_i < t_{i+1} \}$

Reconstruction values: $\{ \underline{r}_i, i = 1, 2, \dots, L \}$

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



Examples:
A-law, μ -law



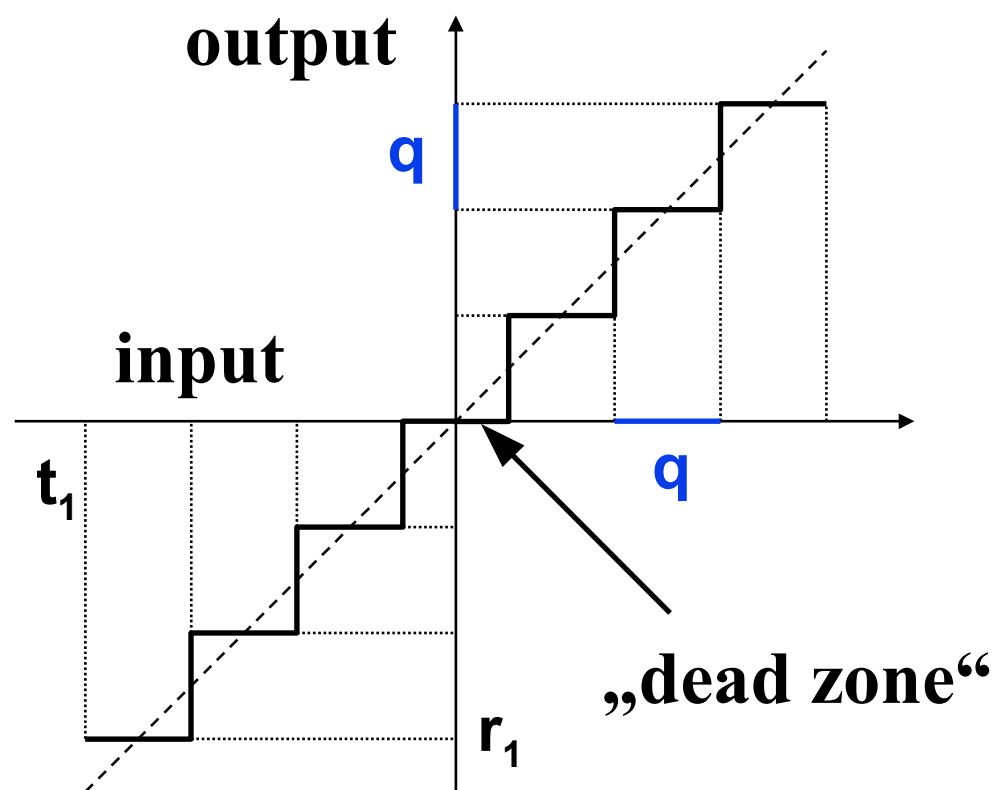
Linear quantizer

Best for **uniformly distributed source values**:

$$\underline{t}_k = t_1 + (k - 1) q$$

$$\underline{r}_k = t_k + q/2$$

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





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)

$$t_k = \frac{1}{2} (r_{k-1} + r_k)$$

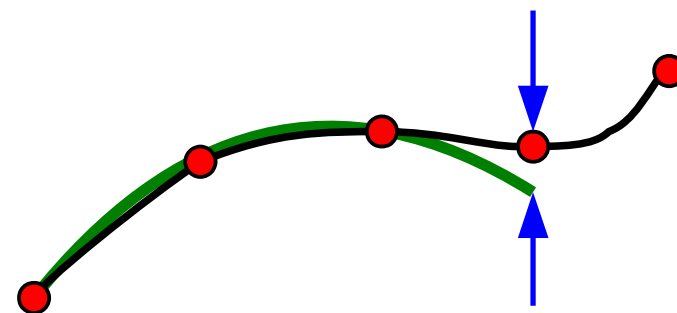
$$r_k = \frac{\int_{t_k}^{t_{k+1}} x p(x) dx}{\int_{t_k}^{t_{k+1}} p(x) dx}$$



Predictive compression

- ◆ **neighbour samples are dependent** (spatial correlation)

- pixel values usually change slowly



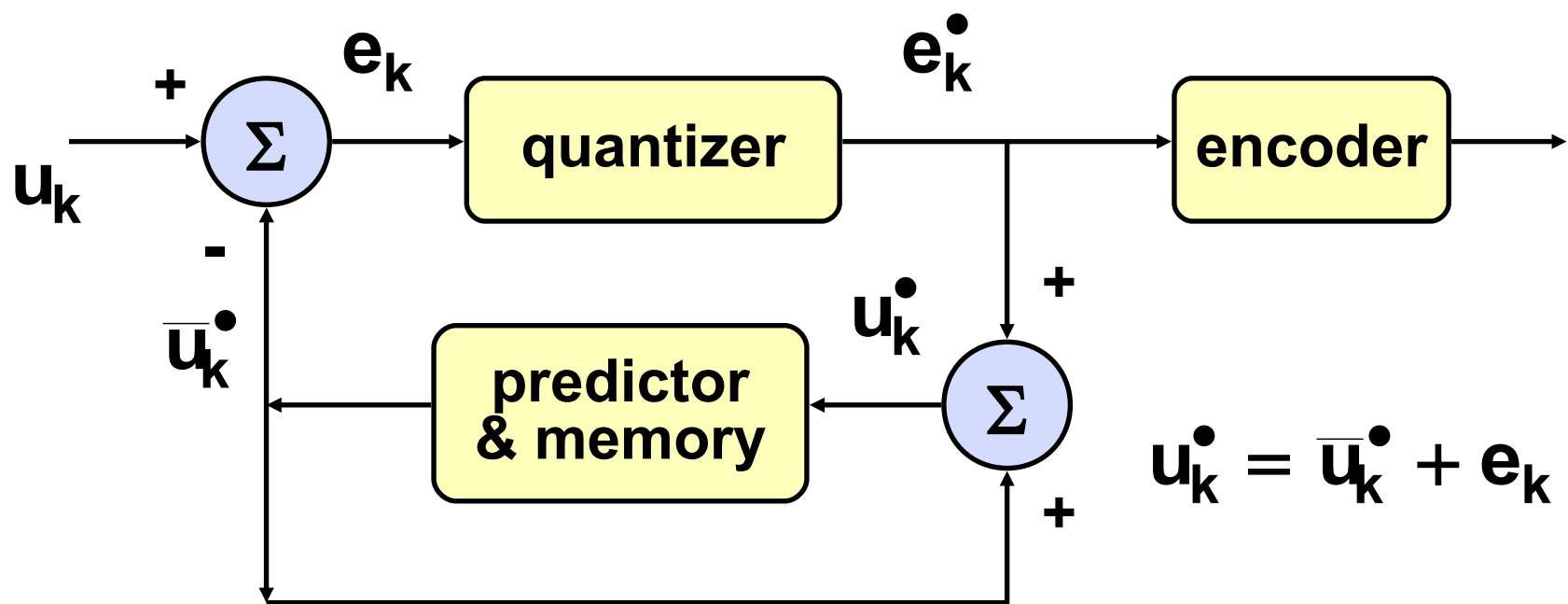
- ◆ implementation using **predictor function**

- previous (neighbour) samples are using to predict recent value
 - only residuals are coded (diferences between predicted and actual values)
 - if successful, residuals have smaller amplitude



Predictive quantization

DPCM encoder (Differential Pulse-Code Modulation):



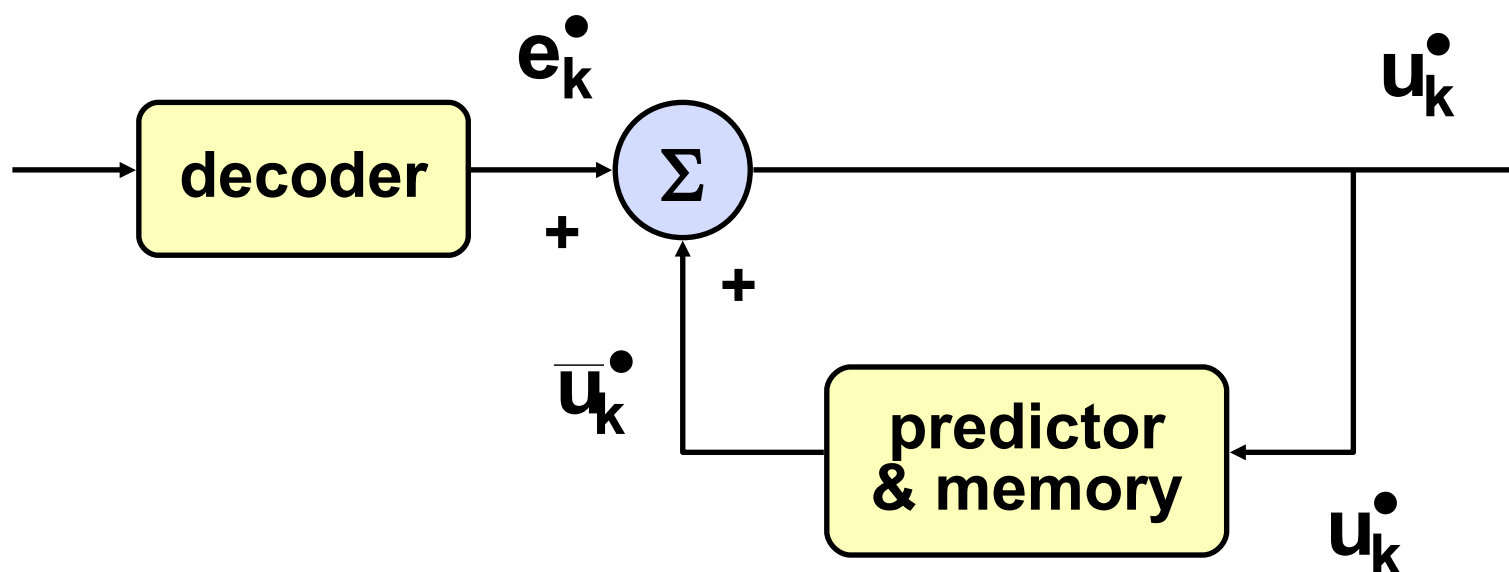
$$e_k = u_k - \bar{u}_k^{\bullet}$$

$$\text{prediction: } \bar{u}_k^{\bullet} = P(u_{k-1}^{\bullet}, u_{k-2}^{\bullet}, \dots)$$



Predictive quantization II

DPCM decoder:





Delta modulation

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

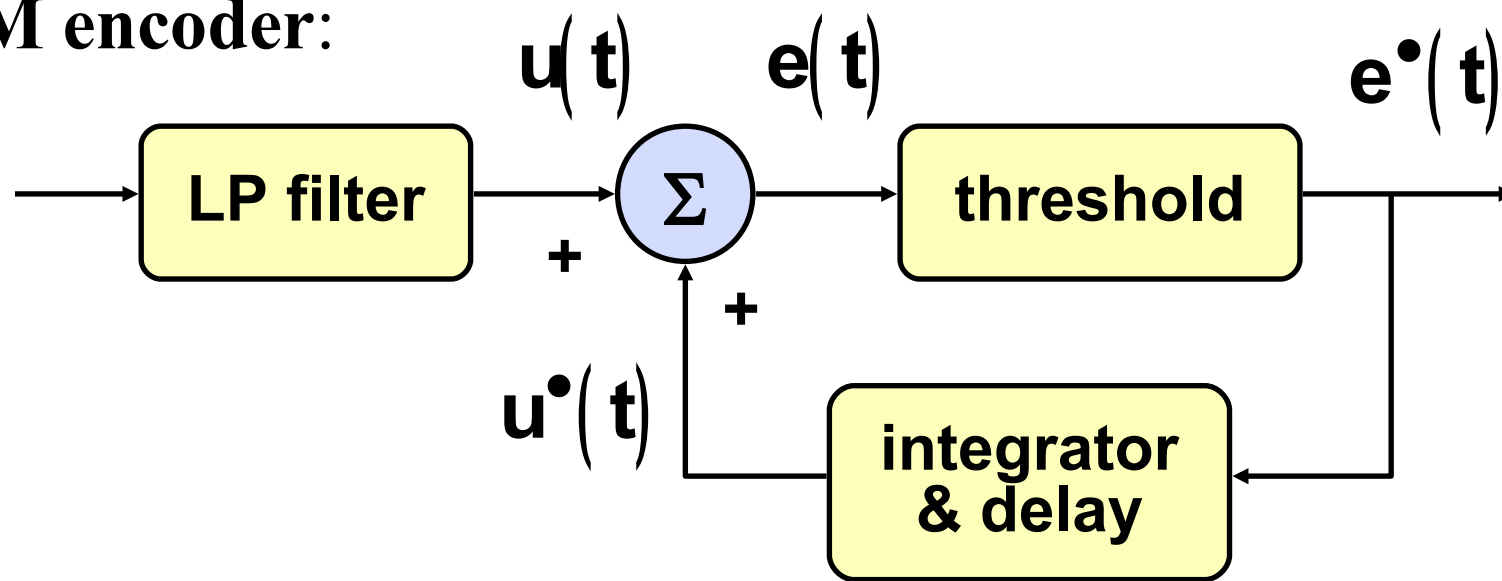
$$\bar{u}_k = u_{k-1} \quad e_k = u_k - u_{k-1}$$

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

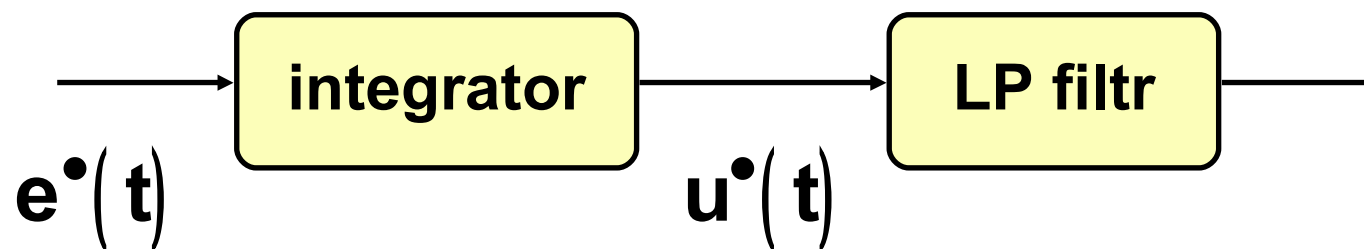


Delta modulation II

DM encoder:

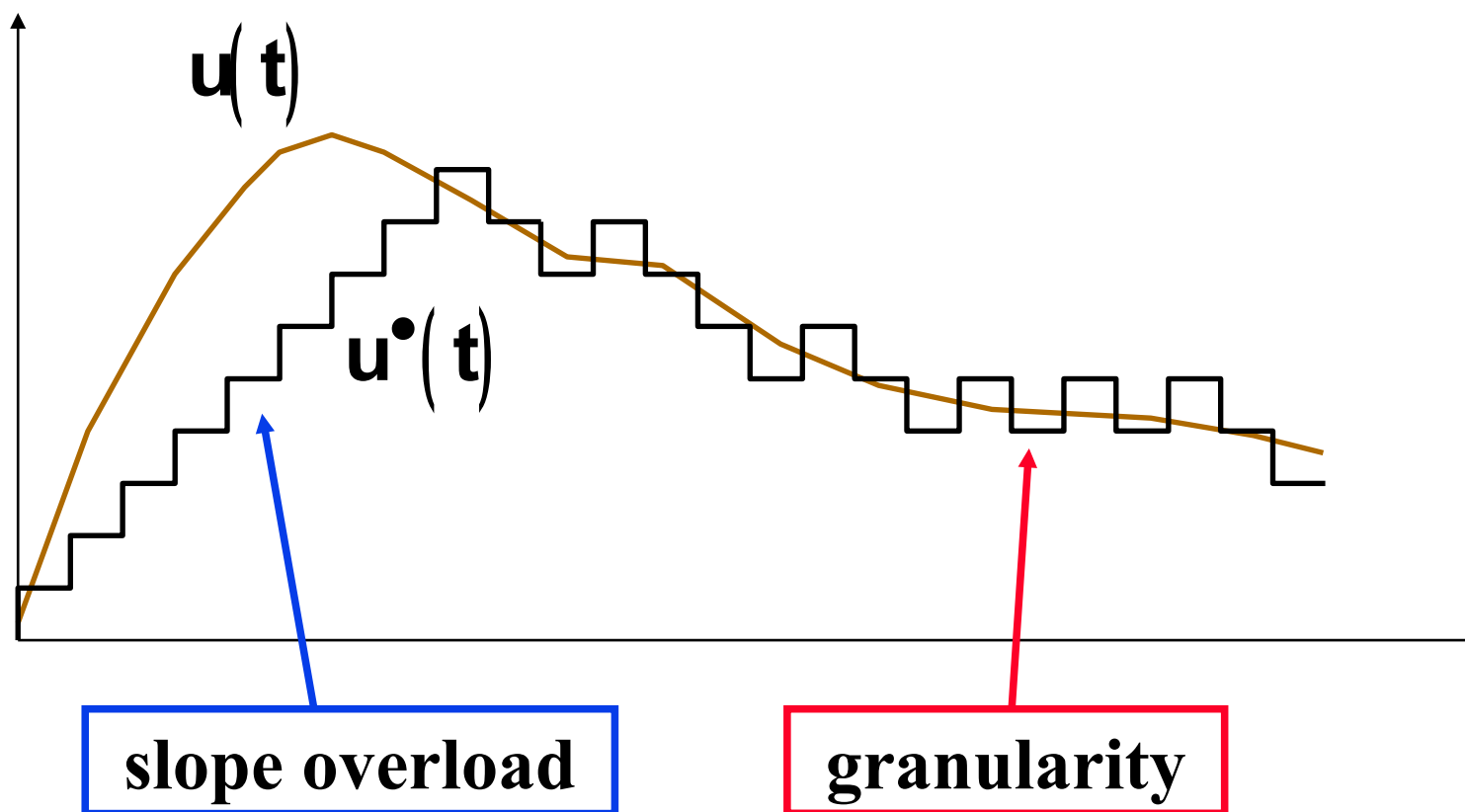


DM decoder:





DM flaws



granularity reduction ... three-state DM (+q,0,-q)



DPCM

- ◆ **Markov chain** (order of \mathbf{p})

- mean value of the current sample is linearly dependent on \mathbf{p} previous samples

$$\bar{u}_k = \sum_{j=1}^{\mathbf{p}} a_j u_{k-j}$$

- ◆ **nonlinear** predictors

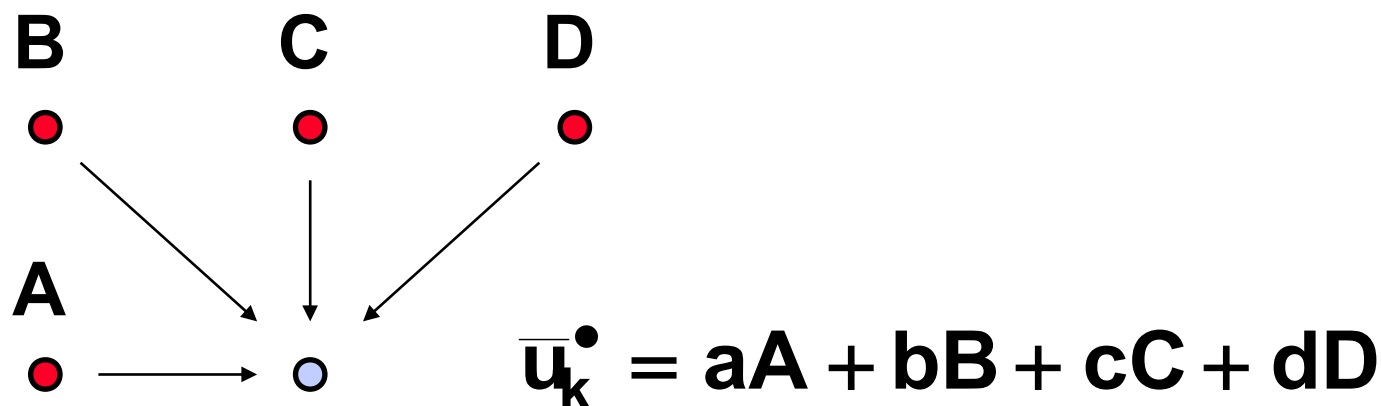
- polynomials are more popular

- ◆ **2D** and **ND** predictors

- match 2D image characteristics (3D in case of video)



2D DPCM



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)



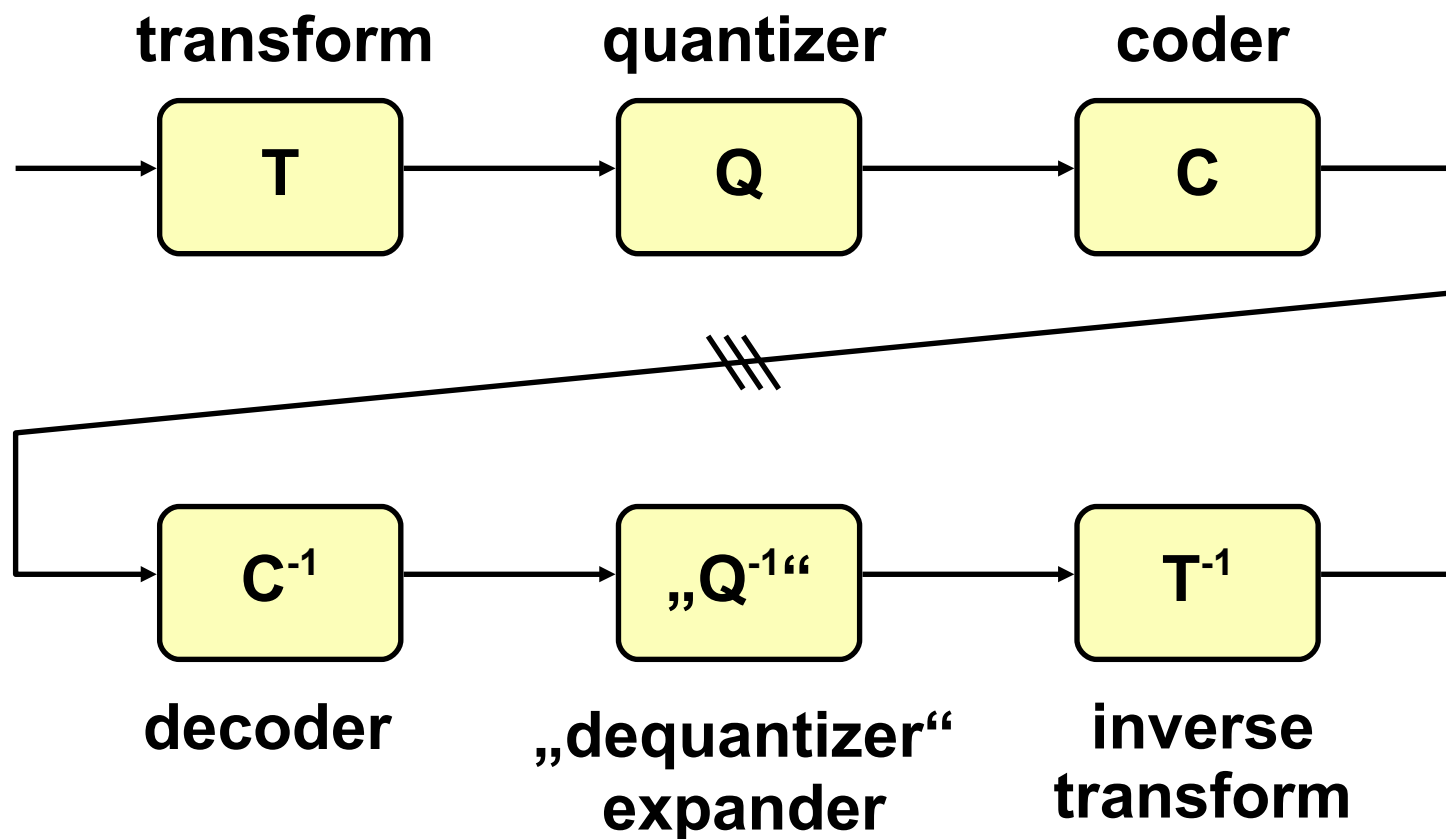
Adaptive predictive techniques

- ◆ predictor sets for **different correlation directions** (horizontal, vertical, oblique, ..)
 - algorithm can adapt itself – picks more effective predictor (max correlation)
- ◆ adaptive **gain control** (quantization error)
 - quality (distortion) of quantization is controlled according to variance of predictor error
- ◆ **image region classification**
 - quick adaptation to local pixel neighbourhood
 - block classification – e.g. 16×16 px 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 **$K \times K$**)

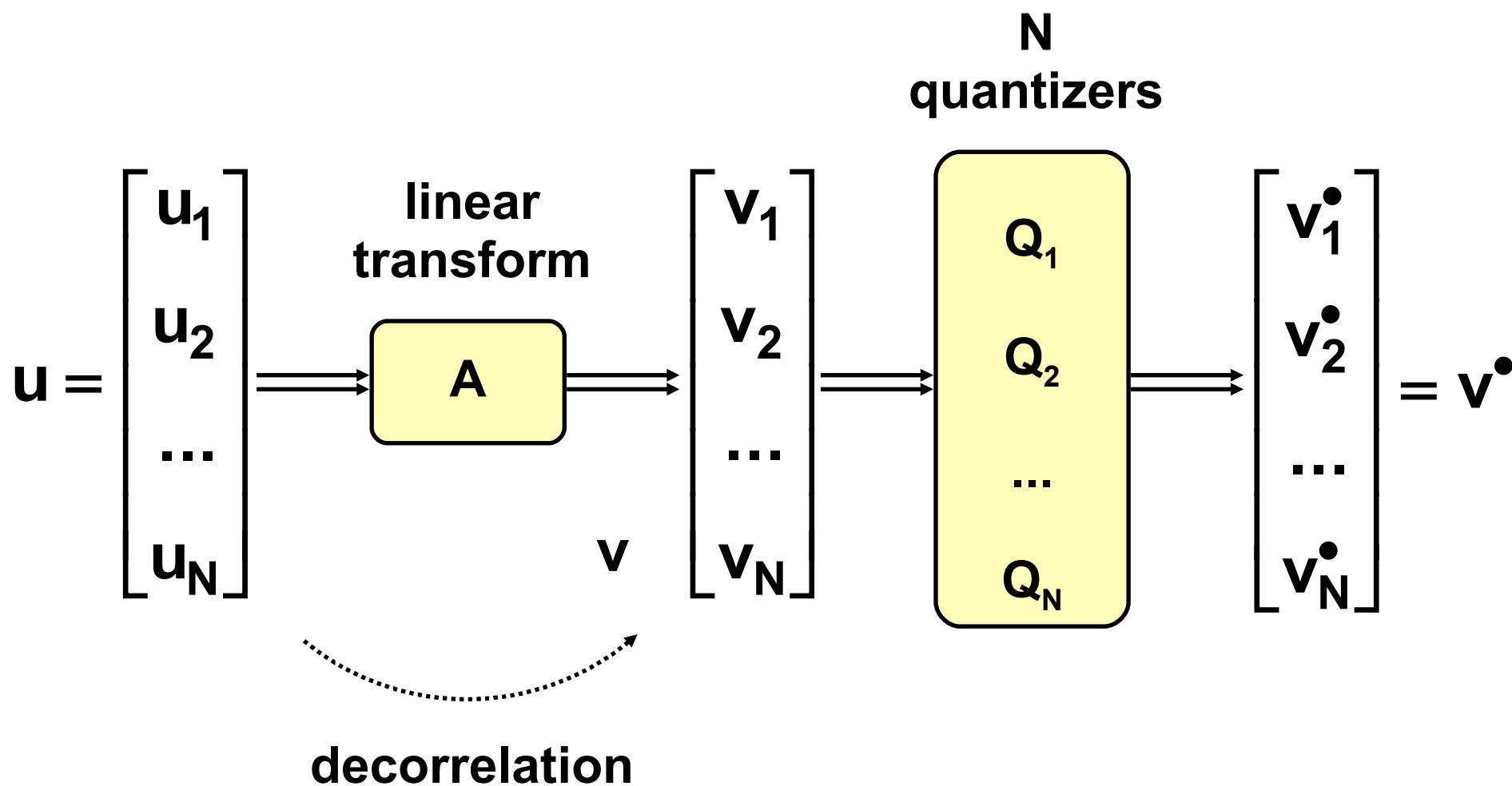


Transform function T

- 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 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 \rightarrow eigenvectors)



Practical transforms

- **suboptimal** transforms
 - a little bit worse than optimal KLT
 - better efficiency: fast algorithms 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)



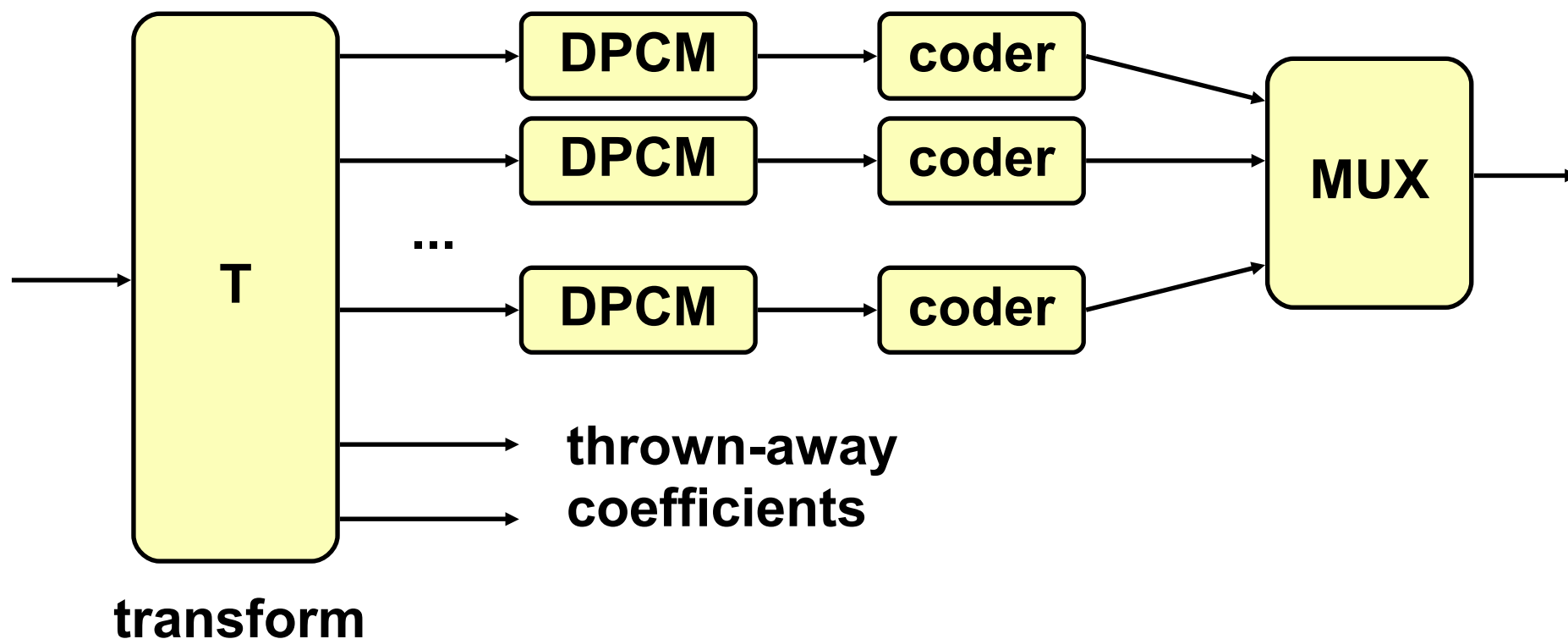
Adaptive transform methods

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



Hybrid methods

Transform & predictive approach together





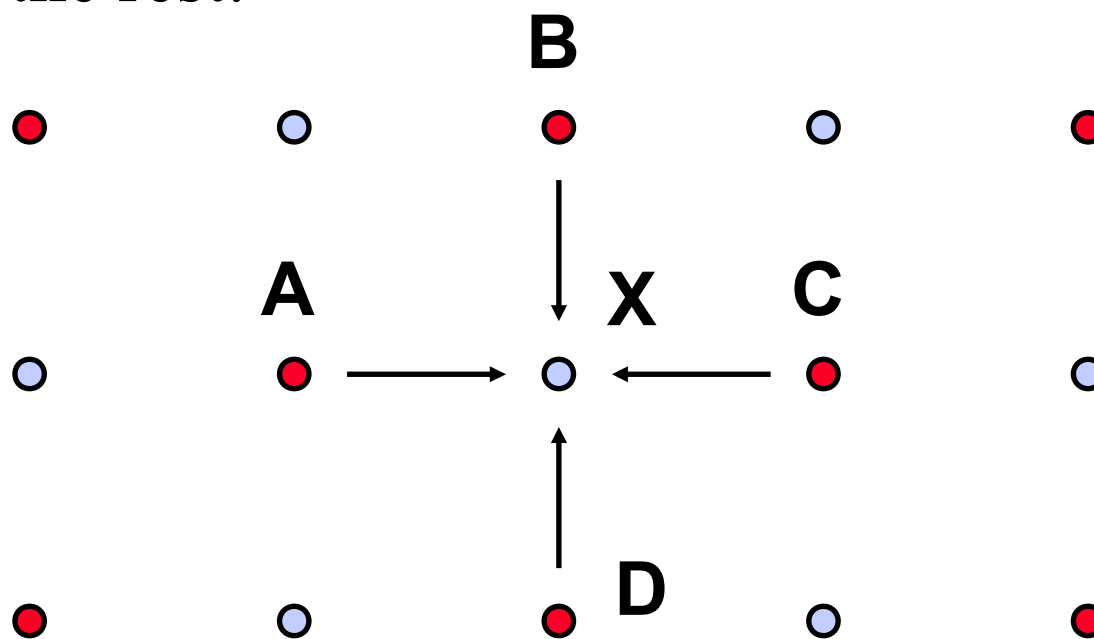
Interpolation methods

- 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:



$$X = \frac{A + C}{2} \quad \text{or} \quad \frac{B + D}{2} \quad (\text{the one with less difference})$$



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**