Predictive Quantization
Last Lectures: Transform Coding with Orthogonal Block Transform

- **Forward transform**: \( u = A \cdot s \)
- **Scalar quantization**: \( q_k = \alpha_k(u_k) \)
- **Entropy coding**: \( b = \gamma(\{q_k\}) \)

**Entropy decoding**: \( \{q_k\} = \gamma^{-1}(b) \)

**Dequantization**: \( u'_k = \beta_k(q_k) \)

**Inverse transform**: \( s' = A^{-1} \cdot u' \)
Last Lectures: Transform Coding with Orthogonal Block Transform

- **Forward transform:** $u = A \cdot s$
- **Scalar quantization:** $q_k = \alpha_k(u_k)$
- **Entropy coding:** $b = \gamma(\{q_k\})$
- **Entropy decoding:** $\{q_k\} = \gamma^{-1}(b)$
- **Dequantization:** $u'_k = \beta_k(q_k)$
- **Inverse transform:** $s' = A^{-1} \cdot u'$

**Orthogonal Block Transforms**

- Inverse matrix is equal to transposed matrix: $A^{-1} = A^T$
- Geometric Interpretation: Rotation (and possible reflection) in $N$-d signal space
  - Can be interpreted as lattice vector quantizer with orthogonal lattice
  - Preservation of MSE distortion: Independent quantization of individual transform coefficients

$A$ is an orthogonal matrix, and its inverse is equal to its transpose. This property is crucial for orthogonal block transforms as it ensures that the distortion introduced by quantization is minimized. The process consists of three main steps:

1. **Forward Transform:**
   - $u = A \cdot s$

2. **Scalar Quantization:**
   - $q_k = \alpha_k(u_k)$

3. **Entropy Coding:**
   - $b = \gamma(\{q_k\})$

4. **Entropy Decoding:**
   - $\{q_k\} = \gamma^{-1}(b)$

5. **Dequantization:**
   - $u'_k = \beta_k(q_k)$

6. **Inverse Transform:**
   - $s' = A^{-1} \cdot u'$

The diagram illustrates these steps visually, with arrows connecting the variables and processes. The encoder (left) and decoder (right) blocks show the flow of data through the transform, quantization, and reconstruction stages.
**Last Lectures: High-Rate Approximations**

**High-Rate Approximation of Operational Distortion-Rate Function**

- Optimal bit allocation at high rates: $D_k(R_k) = \text{const}$ (for URQs: $\Delta_k = \text{const}$)

- High-rate distortion-rate function for transform coding (with optimal bit allocation)

$$D_{TC}(R) = \bar{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \quad \text{with} \quad \bar{\varepsilon}^2 = \left( \prod_k \varepsilon_k^2 \right)^{\frac{1}{N}}, \quad \tilde{\sigma}^2 = \left( \prod_k \sigma_k^2 \right)^{\frac{1}{N}}$$
High-Rate Approximation of Operational Distortion-Rate Function

- Optimal bit allocation at high rates: \( D_k(R_k) = \text{const} \) (for URQs: \( \Delta_k = \text{const} \))
- High-rate distortion-rate function for transform coding (with optimal bit allocation)

\[
D_{TC}(R) = \bar{\varepsilon}^2 \cdot \bar{\sigma}^2 \cdot 2^{-2R}
\]

with
\[
\bar{\varepsilon}^2 = \left( \prod_k \varepsilon_k^2 \right)^\frac{1}{N}, \quad \bar{\sigma}^2 = \left( \prod_k \sigma_k^2 \right)^\frac{1}{N}
\]

Goal of Transform Selection

- Minimize distortion \( D_{TC}(R) \) for given rate \( R \) \quad \Rightarrow \quad Maximize transform coding gain \( G_T \)

\[
G_T = \frac{D_{SQ}(R)}{D_{TC}(R)} = \frac{\bar{\varepsilon}^2 \cdot \sigma_S^2}{\varepsilon^2 \cdot \bar{\sigma}^2}
\]

- Typically, ignore impact of pdf shapes \quad \Rightarrow \quad Maximize energy compaction \( G_{EC} \) of transform

\[
G_{EC} = \frac{\sigma_S^2}{\bar{\sigma}^2} = \frac{1}{N} \sum_{k=0}^{N-1} \frac{\sigma_k^2}{\sigma_k^2} \frac{\sqrt{\prod_{k=0}^{N-1} \sigma_k^2}}{\sqrt{\prod_{k=0}^{N-1} \sigma_k^2}}
\]
Last Lectures:  Transform Selection

Karhunen Loève Transform (KLT)

- Basis vectors (rows of $A$) are unit-norm eigenvectors of the $N$-th order auto-covariance matrix $C_N$
- KLT yields uncorrelated transform coefficients ($C_{UU}$ becomes a diagonal matrix)
- KLT achieves maximum possible energy compaction

$$G_{EC}^{(KLT)} = \left| \frac{1}{\sigma_S^2} C_N \right|^{-\frac{1}{N}}$$

(note: determinant $| \cdot | = \text{product of eigenvalues}$)
Karhunen Loève Transform (KLT)

- Basis vectors (rows of $A$) are unit-norm eigenvectors of the $N$-th order auto-covariance matrix $C_N$
- KLT yields uncorrelated transform coefficients ($C_{UU}$ becomes a diagonal matrix)
- KLT achieves maximum possible energy compaction

\[
G_{EC}^{(\text{KLT})} = \left| \frac{1}{\sigma_S^2} C_N \right|^{-\frac{1}{N}}
\]

(note: determinant $| \cdot | = $ product of eigenvalues)

Discrete Cosine Transform of Type II (DCT-II)

- Represents discrete Fourier transform of mirrored signal (with half-sample symmetry of both sides)
- Signal independent: Basis vectors (rows of $A$) are sampled cosines of different frequencies
- KLT approaches DCT-II for highly correlated sources ($\rho \to 1$)
- For typical stationary signals: Only small loss in coding efficiency relative to KLT

- Most widely used transform in image and video coding (fast implementations possible)
High-Rate Approximation for KLT and Gauss-Markov

- High-rate operational distortion-rate function and transform coding gain (for transform size $N$)

$$D_N(R) = \varepsilon^2 \cdot \sigma^2_S \cdot (1 - \varrho^2)^{N-1} \cdot 2^{-2R}$$

and

$$G_T^N = G_{EC}^N = (1 - \varrho^2)^{\frac{1-N}{N}}$$

- Transform gain increases with transform size $N$, but approaches a limit

$$\text{Comparison to Rate-Distortion Bound}$$

Example: Gauss-Markov, KLT, and optimal scalar quantizers (ECSQ)

$$D_{\text{KLT}}(R) = \pi \cdot e \cdot 6 \cdot \sigma^2_S \cdot (1 - \varrho^2)^{N-1} \cdot 2^{-2R}$$

$$\sigma^2_S \cdot (1 - \varrho^2)^{\frac{1-N}{N}} = \pi \cdot e \cdot 6 \cdot (1 - \varrho^2)^{\frac{1-N}{N}}$$

Large transform sizes ($N \to \infty$): Performance gap reduces to space-filling gain (1.53 dB)

Other sources: Transform coding cannot utilize all dependencies (non-linear dependencies)
Last Lectures: Coding Efficiency of Transform Coding

High-Rate Approximation for KLT and Gauss-Markov

- High-rate operational distortion-rate function and transform coding gain (for transform size $N$)

\[
D_N(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R}
\]

and

\[
G_T^N = G_{EC}^N = (1 - \varrho^2)^{\frac{1-N}{N}}
\]

- Transform gain increases with transform size $N$, but approaches a limit

Comparison to Rate-Distortion Bound

- Example: Gauss-Markov, KLT, and optimal scalar quantizers (ECSQ)

- Distortion increase relative to Shannon lower bound

\[
\frac{D_N^{(KLT)}(R)}{D_{SLB}(R)} = \frac{\pi \varepsilon}{6} \cdot \sigma_S^2 \cdot (1 - \varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R}
\]

\[
= \frac{\pi \varepsilon}{6} \cdot \left( \frac{1}{1 - \varrho^2} \right)^{\frac{1}{N}}
\]

- Large transform sizes ($N \to \infty$): Performance gap reduces to space-filling gain (1.53 dB)

- Other sources: Transform coding cannot utilize all dependencies (non-linear dependencies)
**Predictive Lossless Coding**

- Predict current sample $s_n$ using a function of preceding samples (referred to as observation set $b_n$)
  
  $$\hat{s}_n = f(s_{n-1}, s_{n-2}, \cdots) = f(b_n)$$

- Entropy coding of prediction error samples
  
  $$u_n = s_n - \hat{s}_n$$

- Decoder reconstructs the original samples according to
  
  $$s_n = \hat{s}_n + u_n$$

**Examples of observation sets**

1D signals:

2D signals:
Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set $b_n = \{b_1, \cdots, b_n\}$

$$\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = a^T \cdot b_n$$

with

$$a = (a_1, \cdots, a_N)^T$$
Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set \( b_n = \{b_1, \cdots, b_n\} \)

\[
\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = \mathbf{a}^T \cdot \mathbf{b}_n \quad \text{with} \quad \mathbf{a} = (a_1, \cdots, a_N)^T
\]

Prediction error variance \( \sigma^2_U \) is given by

\[
\sigma^2_U(\mathbf{a}) = \sigma^2_S - 2 \mathbf{a}^T \mathbf{c} + \mathbf{a}^T \mathbf{C}_B \mathbf{a}
\]

\[
\mathbf{C}_B = \mathbb{E}\left\{ (\mathbf{B}_n - \mathbb{E}\{\mathbf{B}_n\}) (\mathbf{B}_n - \mathbb{E}\{\mathbf{B}_n\})^T \right\}
\]

\[
\mathbf{c} = \mathbb{E}\left\{ (\mathbf{S}_n - \mathbb{E}\{\mathbf{S}_n\}) (\mathbf{B}_n - \mathbb{E}\{\mathbf{B}_n\}) \right\}
\]
Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set $b_n = \{b_1, \cdots, b_n\}$

$$\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = a^T \cdot b_n \quad \text{with} \quad a = (a_1, \cdots, a_N)^T$$

$\Rightarrow$ Prediction error variance $\sigma_U^2$ is given by

$$\sigma_U^2(a) = \sigma_S^2 - 2a^T c + a^T C_B a \quad \text{with}$$

$$C_B = \mathbb{E}\left\{ (B_n - \mathbb{E}\{B_n\}) (B_n - \mathbb{E}\{B_n\})^T \right\}$$

$$c = \mathbb{E}\left\{ (S_n - \mathbb{E}\{S_n\}) (B_n - \mathbb{E}\{B_n\}) \right\}$$

$\Rightarrow$ Prediction error variance is minimized by solution of the Yule-Walker equations

$$C_B \cdot a_{\text{opt}} = c \quad \text{(linear equation system)}$$
Review: Linear and Affine Prediction

Linear Predictor

- Current sample is predicted using weighted sum over observation set \( b_n = \{b_1, \cdots, b_n\} \)

\[
\hat{s}_n = \sum_{k=1}^{N} a_k \cdot b_k = a^T \cdot b_n \quad \text{with} \quad a = (a_1, \cdots, a_N)^T
\]

⇒ **Prediction error variance** \( \sigma_U^2 \) is given by

\[
\sigma_U^2(a) = \sigma_S^2 - 2a^Tc + a^T C_B a \quad \text{with} \quad C_B = E\left\{ (B_n - E\{B_n\})(B_n - E\{B_n\})^T \right\}
\]

\[
c = E\left\{ (S_n - E\{S_n\})(B_n - E\{B_n\}) \right\}
\]

⇒ **Prediction error variance** is minimized by solution of the **Yule-Walker equations**

\[
C_B \cdot a_{opt} = c \quad \text{(linear equation system)}
\]

⇒ **Prediction error variance** for optimal linear prediction

\[
\sigma_U^2 = \sigma_S^2 - a_{opt}^T c = \sigma_S^2 - c^T C_B^{-1} c
\]
Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

\[
\hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + \mathbf{a}^T \cdot \mathbf{b}_n
\]
Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

\[ \hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + a^T \cdot b_n \]

- Prediction error variance \( \sigma^2_U \) is not impacted by constant offset \( a_0 \)
Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

\[ \hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + \mathbf{a}^T \cdot \mathbf{b}_n \]

- Prediction error variance \( \sigma^2_U \) is not impacted by constant offset \( a_0 \)
- Mean of prediction error \( \mu_U \) can be forced to zero by choosing

\[ a_0 = \mu_S \left( 1 - \sum_{k=1}^{N} a_k \right) \]
Review: Linear and Affine Prediction

Affine Predictor

- Linear predictor with additional constant offset

\[ \hat{s}_n = a_0 + \sum_{k=1}^{N} a_k \cdot b_k = a_0 + a^T \cdot b_n \]

- Prediction error variance \( \sigma_U^2 \) is not impacted by constant offset \( a_0 \)

- Mean of prediction error \( \mu_U \) can be forced to zero by choosing

\[ a_0 = \mu_S \left( 1 - \sum_{k=1}^{N} a_k \right) \]

- Affine prediction reduces mean squared prediction error (compared to linear prediction)

\[ \varepsilon_U^2 = \mathbb{E}\left\{ \left( S_n - \hat{S}_n \right)^2 \right\} = \sigma_U^2 + \mu_U^2 \]
Lossy Coding with Prediction: (1) Prediction of Quantization Indexes

**Encoder**

- **Scalar quantization**
  
  \[ q_n = \alpha(s_n) \]

- **Prediction of quantization indexes**
  
  \[ \hat{q}_n = f_{\text{pred}}(q_{n-1}, q_{n-2}, \cdots) \]

- **Determining prediction residual**
  
  \[ u_n = q_n - \hat{q}_n \]

- **Entropy coding of prediction residual**
Lossy Coding with Prediction: (1) Prediction of Quantization Indexes

**Encoder**
- Scalar quantization
  \[ q_n = \alpha(s_n) \]
- Prediction of quantization indexes
  \[ \hat{q}_n = f_{\text{pred}}(q_{n-1}, q_{n-2}, \cdots) \]
- Determining prediction residual
  \[ u_n = q_n - \hat{q}_n \]
- Entropy coding of prediction residual

**Decoder**
- Entropy decoding of prediction residual
- Prediction of quantization indexes
  \[ \hat{q}_n = f_{\text{pred}}(q_{n-1}, q_{n-2}, \cdots) \]
- Reconstruct quantization indexes
  \[ q_n = u_n + \hat{q}_n \]
- Dequantization (typically: scaling)
  \[ s'_n = \beta(q_n) \]
Lossy Coding with Prediction: (1) Prediction of Quantization Indexes

Prediction after Quantization
- Sources with memory: Quantization indexes have statistical dependencies
Lossy Coding with Prediction: (1) Prediction of Quantization Indexes

Prediction after Quantization

- Sources with memory: Quantization indexes have statistical dependencies
- Prediction can improve lossless coding of quantization indexes
  - More accurate: With prediction, entropy coding can be simplified
  - Require: Predicted value $\hat{q}_n$ must be integer (since $u_n$ must be integer)
**Lossy Coding with Prediction: (1) Prediction of Quantization Indexes**

**Prediction after Quantization**
- Sources with memory: Quantization indexes have statistical dependencies
- Prediction can improve lossless coding of quantization indexes
  - More accurate: With prediction, entropy coding can be simplified
  - Require: Predicted value $\hat{q}_n$ must be integer (since $u_n$ must be integer)

**Extension: Prediction after Transform and Quantization**
- Prediction of some transform coefficients (e.g., DC coefficient)
- Examples: JPEG, MPEG-2 Video, H.263, MPEG-4 Visual
**Lossy Coding with Prediction: (2) Quantization of Prediction Error**

### Prediction Before Quantization

- **Idea:** Reduce statistical dependencies before quantization (similar to transform coding)
- **Have to use same prediction signal**: $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)
Lossy Coding with Prediction: (2) Quantization of Prediction Error

Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)
- Prediction value $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)$$
Lossy Coding with Prediction: (2) Quantization of Prediction Error

Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal \( \hat{s}_n \) at encoder and decoder (otherwise: error accumulation)
- Prediction value \( \hat{s}_n \) has to be derived based on reconstructed samples

\[
\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)
\]
Lossy Coding with Prediction: (2) Quantization of Prediction Error

**Prediction Before Quantization**

- **Idea:** Reduce statistical dependencies before quantization (similar to transform coding)
- **Have to use same prediction signal** $\hat{s}_n$ at encoder and decoder (otherwise: error accumulation)
- **Prediction value** $\hat{s}_n$ has to be derived based on reconstructed samples

$$\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \ldots)$$

- **Encoder includes decoder** (except for entropy decoding)
Lossy Coding with Prediction: (2) Quantization of Prediction Error

Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal \( \hat{s}_n \) at encoder and decoder (otherwise: error accumulation)
- Prediction value \( \hat{s}_n \) has to be derived based on reconstructed samples

\[
\hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)
\]

- Encoder includes decoder (except for entropy decoding)
Lossy Coding with Prediction: (2) Quantization of Prediction Error

Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal \( \hat{s}_n \) at encoder and decoder (otherwise: error accumulation)
  - Prediction value \( \hat{s}_n \) has to be derived based on reconstructed samples
    \[
    \hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)
    \]
- Encoder includes decoder (except for entropy decoding)
Lossy Coding with Prediction: (2) Quantization of Prediction Error

Prediction Before Quantization

- Idea: Reduce statistical dependencies before quantization (similar to transform coding)
- Have to use same prediction signal \( \hat{s}_n \) at encoder and decoder (otherwise: error accumulation)
- Prediction value \( \hat{s}_n \) has to be derived based on reconstructed samples
  \[
  \hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots)
  \]
- Encoder includes decoder (except for entropy decoding)
- Concept also referred to as differential pulse code modulation (DPCM)
**Lossy Coding with Prediction: (2) Quantization of Prediction Error**

\[ s_n + u_n \rightarrow \text{quant.} \rightarrow q_n \rightarrow \text{entropy encoder} \rightarrow \text{bitstream} \rightarrow \text{entropy decoder} \rightarrow \text{dequant.} \rightarrow u'_n + \hat{s}_n \rightarrow s'_n \]

**DPCM Encoder**

- **Prediction:** \( \hat{s}_n = f_{\text{pred}}(s'_{n-1}, s'_{n-2}, \cdots) \)
- **Quantization:** \( q_n = \alpha(s_n - \hat{s}_n) \)
- **Reconstruction:** \( s'_n = \hat{s}_n + \beta(q_n) \)
- **Entropy coding of quantization indexes** \( q_n \)
Lossy Coding with Prediction: (2) Quantization of Prediction Error

DPCM Encoder
- Prediction: \( \hat{s}_n = f_{\text{pred}}(s'_n, s'_{n-1}, \ldots) \)
- Quantization: \( q_n = \alpha(s_n - \hat{s}_n) \)
- Reconstruction: \( s'_n = \hat{s}_n + \beta(q_n) \)
- Entropy coding of quantization indexes \( q_n \)

DPCM Decoder
- Entropy decoding of quantization indexes \( q_n \)
- Prediction: \( \hat{s}_n = f_{\text{pred}}(s'_n, s'_{n-1}, \ldots) \)
- Reconstruction: \( s'_n = \hat{s}_n + \beta(q_n) \)
DPCM Encoder and Decoder

- Redrawing yields typical DPCM structure
- Note: Encoder contains decoder except entropy decoding
Design of Optimal Predictor and Quantizer

Interdependencies between Predictor and Quantizer

- Optimal quantizer depends on statistical properties of prediction error signal
- Optimal predictor depends on statistical properties of quantization error
Design of Optimal Predictor and Quantizer

Interdependencies between Predictor and Quantizer

- Optimal quantizer depends on statistical properties of prediction error signal
- Optimal predictor depends on statistical properties of quantization error

Joint Design of Predictor and Quantizer

- Choose sufficiently large training set and Lagrange multiplier
- Start: Design predictor for original training data
- Iteratively refine quantizer and predictor
  - Design quantizer for predictor error signal (using current predictor)
  - Run DPCM encoding with current predictor and quantizer
  - Update predictor using obtained reconstructed signal
Design of Optimal Predictor and Quantizer

Interdependencies between Predictor and Quantizer
- Optimal quantizer depends on statistical properties of prediction error signal
- Optimal predictor depends on statistical properties of quantization error

Joint Design of Predictor and Quantizer
- Choose sufficiently large training set and Lagrange multiplier
- Start: Design predictor for original training data
- Iteratively refine quantizer and predictor
  - Design quantizer for predictor error signal (using current predictor)
  - Run DPCM encoding with current predictor and quantizer
  - Update predictor using obtained reconstructed signal

Simplified Design
- Design predictor for original signal (i.e., neglect quantization in predictor design)
- Typically: Works reasonably well
Example: DPCM for Gauss-Markov

Gauss-Markov Model

- Zero-mean AR(1): Ignore mean, since it can be removed as a first step
  \[ S_n = \varrho \cdot S_{n-1} + Z_n \]  
  (dependencies are specified by correlation coefficient \( \varrho \))
- Innovation process \( Z_n \) is zero-mean Gaussian iid
Example: DPCM for Gauss-Markov

Gauss-Markov Model

- Zero-mean AR(1): Ignore mean, since it can be removed as a first step
  \[ S_n = \rho \cdot S_{n-1} + Z_n \] (dependencies are specified by correlation coefficient \( \rho \))
- Innovation process \( Z_n \) is zero-mean Gaussian iid

DPCM Coding of Gauss-Markov Source

- Consider optimal linear predictor for original source (i.e., one-tap filter \( a = \rho \))
  \[ \hat{s}_n = \rho \cdot s'_{n-1} = \rho \cdot (s_{n-1} - x_{n-1}) \]
  with quantization error
  \[ x_n = u_n - u'_n = s_n - s'_n \]
Example: DPCM for Gauss-Markov

Gauss-Markov Model

- Zero-mean AR(1): Ignore mean, since it can be removed as a first step
  \[ S_n = \varrho \cdot S_{n-1} + Z_n \] (dependencies are specified by correlation coefficient \( \varrho \))
- Innovation process \( Z_n \) is zero-mean Gaussian iid

DPCM Coding of Gauss-Markov Source

- Consider optimal linear predictor for original source (i.e., one-tap filter \( a = \varrho \))
  \[ \hat{s}_n = \varrho \cdot s'_{n-1} = \varrho \cdot (s_{n-1} - x_{n-1}) \] with quantization error \( x_n = u_n - u'_n = s_n - s'_n \)
- Resulting prediction error
  \[ u_n = s_n - \hat{s}_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \]
DPCM for Gauss-Markov: Prediction Error Variance

- **Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma_U^2 = E \left\{ U_n^2 \right\}
\]
DPCM for Gauss-Markov: Prediction Error Variance

**Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma_U^2 = \mathbb{E}\{ U_n^2 \} \\
= \mathbb{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} 
\]
**DPCM for Gauss-Markov: Prediction Error Variance**

**Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma_u^2 = E\{ U_n^2 \} = E\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} = E\{ S_n^2 \}
\]
DPCM for Gauss-Markov: Prediction Error Variance

Variance of prediction error \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s_n' \)

\[ \sigma_U^2 = \mathbb{E}\{ U_n^2 \} \]

\[ = \mathbb{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \]

\[ = \mathbb{E}\{ S_n^2 \} + \varrho^2 \cdot \mathbb{E}\{ S_{n-1}^2 \} \]
DPCM for Gauss-Markov: Prediction Error Variance

→ Variance of prediction error  \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s_n' \)

\[
\sigma_U^2 = E\{ U_n^2 \} \\
= E\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \\
= E\{ S_n^2 \} + \varrho^2 \cdot E\{ S_{n-1}^2 \} - 2\varrho \cdot E\{ S_n S_{n-1} \}
\]
DPCM for Gauss-Markov: Prediction Error Variance

- **Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma^2_U = \mathbb{E}\{U_n^2\} = \mathbb{E}\{(S_n - \varrho S_{n-1} + \varrho X_{n-1})^2\} = \mathbb{E}\{S_n^2\} + \varrho^2 \mathbb{E}\{S_{n-1}^2\} - 2\varrho \mathbb{E}\{S_n S_{n-1}\} + \varrho^2 \mathbb{E}\{X_{n-1}^2\}
\]
DPCM for Gauss-Markov: Prediction Error Variance

- Variance of prediction error
  \[ u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \quad \text{with} \quad x_n = s_n - s'_n \]

  \[
  \sigma^2_U = \mathbb{E}\{ U_n^2 \} \\
  = \mathbb{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \\
  = \mathbb{E}\{ S_n^2 \} + \varrho^2 \cdot \mathbb{E}\{ S_{n-1}^2 \} - 2\varrho \cdot \mathbb{E}\{ S_n S_{n-1} \} + \varrho^2 \cdot \mathbb{E}\{ X_{n-1}^2 \} \\
  + 2\varrho \cdot \mathbb{E}\{ S_n X_{n-1} \} 
  \]
DPCM for Gauss-Markov: Prediction Error Variance

Variance of prediction error

\[ u_n = s_n - \rho \cdot s_{n-1} + \rho \cdot x_{n-1} \quad \text{with} \quad x_n = s_n - s'_n \]

\[
\sigma_U^2 = \mathbb{E}\{ u_n^2 \} = \mathbb{E}\{ (S_n - \rho S_{n-1} + \rho X_{n-1})^2 \} = \mathbb{E}\{ S_n^2 \} + \rho^2 \cdot \mathbb{E}\{ S_{n-1}^2 \} - 2\rho \cdot \mathbb{E}\{ S_n S_{n-1} \} + \rho^2 \cdot \mathbb{E}\{ X_{n-1}^2 \} + 2\rho \cdot \mathbb{E}\{ S_n X_{n-1} \} - 2\rho^2 \cdot \mathbb{E}\{ S_{n-1} X_{n-1} \}
\]
DPCM for Gauss-Markov: Prediction Error Variance

- **Variance of prediction error**
  \[ u_n = s_n - \rho \cdot s_{n-1} + \rho \cdot x_{n-1} \quad \text{with} \quad x_n = s_n - s'_n \]

\[
\sigma_U^2 = E\{ U_n^2 \} \\
= E\{ (S_n - \rho S_{n-1} + \rho X_{n-1})^2 \} \\
= E\{ S_n^2 \} + \rho^2 \cdot E\{ S_{n-1}^2 \} - 2\rho \cdot E\{ S_n S_{n-1} \} + \rho^2 \cdot E\{ X_{n-1}^2 \} \\
+ 2\rho \cdot E\{ S_n X_{n-1} \} - 2\rho^2 \cdot E\{ S_{n-1} X_{n-1} \} \\
= \sigma_S^2
\]
DPCM for Gauss-Markov: Prediction Error Variance

**Variance of prediction error**

\[ u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \quad \text{with} \quad x_n = s_n - s'_n \]

\[ \sigma_U^2 = \mathbb{E} \{ U_n^2 \} \]

\[ = \mathbb{E} \{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \]

\[ = \mathbb{E} \{ S_n^2 \} + \varrho^2 \cdot \mathbb{E} \{ S_{n-1}^2 \} - 2\varrho \cdot \mathbb{E} \{ S_n S_{n-1} \} + \varrho^2 \cdot \mathbb{E} \{ X_{n-1}^2 \} \]

\[ + 2\varrho \cdot \mathbb{E} \{ S_n X_{n-1} \} - 2\varrho^2 \cdot \mathbb{E} \{ S_{n-1} X_{n-1} \} \]

\[ = \sigma_S^2 + \varrho^2 \sigma_S^2 \]
DPCM for Gauss-Markov: Prediction Error Variance

**Variance of prediction error** $u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1}$ with $x_n = s_n - s'_n$

\[
\sigma_U^2 = E\{ U_n^2 \} = E\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \}
\]

\[
= E\{ S_n^2 \} + \varrho^2 E\{ S_{n-1}^2 \} - 2\varrho E\{ S_n S_{n-1} \} + \varrho^2 E\{ X_{n-1}^2 \} + 2\varrho E\{ S_n X_{n-1} \} - 2\varrho^2 E\{ S_{n-1} X_{n-1} \}
\]

\[
= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2
\]
**DPCM for Gauss-Markov: Prediction Error Variance**

**Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s_n' \)

\[
\sigma_U^2 = \mathbb{E}\{ U_n^2 \} \\
= \mathbb{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \\
= \mathbb{E}\{ S_n^2 \} + \varrho^2 \cdot \mathbb{E}\{ S_{n-1}^2 \} - 2\varrho \cdot \mathbb{E}\{ S_n S_{n-1} \} + \varrho^2 \cdot \mathbb{E}\{ X_{n-1}^2 \} \\
+ 2\varrho \cdot \mathbb{E}\{ S_n X_{n-1} \} - 2\varrho^2 \cdot \mathbb{E}\{ S_{n-1} X_{n-1} \} \\
= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \\
\text{(note: distortion } D = \mathbb{E}\{ (S_n - S_n')^2 \} \text{)}
DPCM for Gauss-Markov: Prediction Error Variance

**Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s_n' \)

\[
\sigma_U^2 = \mathbb{E}\{U_n^2\} = \mathbb{E}\{(S_n - \varrho S_{n-1} + \varrho X_{n-1})^2\} = \mathbb{E}\{S_n^2\} + \varrho^2 \cdot \mathbb{E}\{S_{n-1}^2\} - 2\varrho \cdot \mathbb{E}\{S_nS_{n-1}\} + \varrho^2 \cdot \mathbb{E}\{X_{n-1}^2\} + 2\varrho \cdot \mathbb{E}\{S_nX_{n-1}\} - 2\varrho^2 \cdot \mathbb{E}\{S_{n-1}X_{n-1}\}
\]

\[
= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \quad \text{(note: distortion } D = \mathbb{E}\{(S_n - S_n')^2\}) + 2\varrho \mathbb{E}\{(\varrho S_{n-1} + Z_n)E_{n-1}\}
\]
DPCM for Gauss-Markov: Prediction Error Variance

- **Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma_U^2 = \mathbb{E}\{ U_n^2 \} = \mathbb{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} = \mathbb{E}\{ S_n^2 \} + \varrho^2 \cdot \mathbb{E}\{ S_{n-1}^2 \} - 2\varrho \cdot \mathbb{E}\{ S_n S_{n-1} \} + \varrho^2 \cdot \mathbb{E}\{ X_{n-1}^2 \} + 2\varrho \cdot \mathbb{E}\{ S_n X_{n-1} \} - 2\varrho^2 \cdot \mathbb{E}\{ S_{n-1} X_{n-1} \}
\]

\[
= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \quad \text{ (note: distortion } D = \mathbb{E}\{ (S_n - S'_n)^2 \}) + 2\varrho \mathbb{E}\{ (\varrho S_{n-1} + Z_n)E_{n-1} \} - 2\varrho^2 \mathbb{E}\{ S_{n-1} E_{n-1} \}
\]
**DPCM for Gauss-Markov: Prediction Error Variance**

**Variance of prediction error**

\[ u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \quad \text{with} \quad x_n = s_n - s'_n \]

\[
\sigma^2_U = \mathbb{E}\{ U_n^2 \} \\
= \mathbb{E}\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \\
= \mathbb{E}\{ S_n^2 \} + \varrho^2 \cdot \mathbb{E}\{ S_{n-1}^2 \} - 2\varrho \cdot \mathbb{E}\{ S_n S_{n-1} \} + \varrho^2 \cdot \mathbb{E}\{ X_{n-1}^2 \} \\
+ 2\varrho \cdot \mathbb{E}\{ S_n X_{n-1} \} - 2\varrho^2 \cdot \mathbb{E}\{ S_{n-1} X_{n-1} \} \\
= \sigma^2_S + \varrho^2 \sigma^2_S - 2\varrho^2 \sigma^2_S + \varrho^2 D \\
\quad \text{(note: distortion} \quad D = \mathbb{E}\{ (S_n - S'_n)^2 \}) \\
+ 2\varrho \mathbb{E}\{ (\varrho S_{n-1} + Z_n)E_{n-1} \} - 2\varrho^2 \mathbb{E}\{ S_{n-1}E_{n-1} \} \\
= \sigma^2_S (1 - \varrho^2) + \varrho^2 D
DPCM for Gauss-Markov: Prediction Error Variance

**Variance of prediction error** \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma^2_U = E\{ U_n^2 \} = E\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} = E\{ S_n^2 \} + \varrho^2 \cdot E\{ S_{n-1}^2 \} - 2\varrho \cdot E\{ S_n S_{n-1} \} + \varrho^2 \cdot E\{ X_{n-1}^2 \} + 2\varrho \cdot E\{ S_n X_{n-1} \} - 2\varrho^2 \cdot E\{ S_{n-1} X_{n-1} \}
\]

\[
= \sigma^2_S + \varrho^2 \sigma^2_S - 2\varrho^2 \sigma^2_S + \varrho^2 D = \sigma^2_S (1 - \varrho^2) + \varrho^2 D + 2\varrho^2 E\{ S_{n-1} E_{n-1} \} - 2\varrho^2 E\{ S_{n-1} E_{n-1} \} + 2\varrho E\{ Z_n E_{n-1} \}
\]

(note: distortion \( D = E\{ (S_n - S'_n)^2 \} \))
DPCM for Gauss-Markov: Prediction Error Variance

⇒ Variance of prediction error \( u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \) with \( x_n = s_n - s'_n \)

\[
\sigma_U^2 = E\{ U_n^2 \}
= E\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \}
= E\{ S_n^2 \} + \varrho^2 \cdot E\{ S_{n-1}^2 \} - 2\varrho \cdot E\{ S_n S_{n-1} \} + \varrho^2 \cdot E\{ X_{n-1}^2 \}
+ 2\varrho \cdot E\{ S_n X_{n-1} \} - 2\varrho^2 \cdot E\{ S_{n-1} X_{n-1} \}
= \sigma_S^2 + \varrho^2 \sigma_S^2 - 2\varrho^2 \sigma_S^2 + \varrho^2 D \quad \text{(note: distortion } D = E\{ (S_n - S'_n)^2 \})
+ 2\varrho E\{ (\varrho S_{n-1} + Z_n)E_{n-1} \} - 2\varrho^2 E\{ S_{n-1} E_{n-1} \}
= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho^2 E\{ S_{n-1} E_{n-1} \} - 2\varrho^2 E\{ S_{n-1} E_{n-1} \} + 2\varrho E\{ Z_n E_{n-1} \}
= \sigma_S^2 (1 - \varrho^2) + \varrho^2 D + 2\varrho E\{ Z_n E_{n-1} \} \]
DPCM for Gauss-Markov: Prediction Error Variance

\[ u_n = s_n - \varrho \cdot s_{n-1} + \varrho \cdot x_{n-1} \quad \text{with} \quad x_n = s_n - s'_n \]

\[ \sigma^2_U = E\{ U_n^2 \} \]

\[ = E\{ (S_n - \varrho S_{n-1} + \varrho X_{n-1})^2 \} \]

\[ = E\{ S_n^2 \} + \varrho^2 \cdot E\{ S_{n-1}^2 \} - 2\varrho \cdot E\{ S_n S_{n-1} \} + \varrho^2 \cdot E\{ X_{n-1}^2 \} \]

\[ + 2\varrho \cdot E\{ S_n X_{n-1} \} - 2\varrho^2 \cdot E\{ S_{n-1} X_{n-1} \} \]

\[ = \sigma^2_S + \varrho^2 \sigma^2_S - 2\varrho^2 \sigma^2_S + \varrho^2 D \quad \text{(note: distortion} \ D = E\{ (S_n - S'_n)^2 \} \])

\[ + 2\varrho E\{ (\varrho S_{n-1} + Z_n)E_{n-1} \} - 2\varrho^2 E\{ S_{n-1}E_{n-1} \} \]

\[ = \sigma^2_S (1 - \varrho^2) + \varrho^2 D + 2\varrho^2 E\{ S_{n-1}E_{n-1} \} - 2\varrho^2 E\{ S_{n-1}E_{n-1} \} + 2\varrho E\{ Z_n E_{n-1} \} \]

\[ = \sigma^2_S (1 - \varrho^2) + \varrho^2 D + 2\varrho E\{ Z_n E_{n-1} \} \]

\[ = \sigma^2_S (1 - \varrho^2) + \varrho^2 D \]
DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor \( \hat{s}_n = \varrho \cdot s'_n \)

- Prediction error variance \( \sigma^2_U \) depends on distortion \( D \)

\[
\sigma^2_U = \sigma^2_S (1 - \varrho^2) + \varrho^2 D
\]
DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor $\hat{s}_n = \varrho \cdot s'_n$

- Prediction error variance $\sigma^2_U$ depends on distortion $D$

$$\sigma^2_U = \sigma^2_S (1 - \varrho^2) + \varrho^2 D$$

- Distortion $D$ is caused by scalar quantization of prediction error $u_n$

$$D(R) = \sigma^2_U \cdot g(R)$$

where $g(R)$ is the distortion-rate function for a unit-variance Gaussian
DPCM for Gauss-Markov: Prediction Error Variance

Prediction Error Variance for Predictor $\hat{s}_n = \varrho \cdot s'_n$

- Prediction error variance $\sigma^2_U$ depends on distortion $D$
  \[ \sigma^2_U = \sigma^2_S (1 - \varrho^2) + \varrho^2 D \]

- Distortion $D$ is caused by scalar quantization of prediction error $u_n$
  \[ D(R) = \sigma^2_U \cdot g(R) \]
  where $g(R)$ is the distortion-rate function for a unit-variance Gaussian

- Rate-dependent prediction error variance
  \[ \sigma^2_U = \sigma^2_S (1 - \varrho^2) + \varrho^2 \cdot \sigma^2_U \cdot g(R) \]
DPCM for Gauss-Markov: Prediction Error Variance

**Prediction Error Variance for Predictor** \( \hat{s}_n = \varrho \cdot s'_n \)

- Prediction error variance \( \sigma^2_U \) depends on distortion \( D \)
  \[
  \sigma^2_U = \sigma^2_S (1 - \varrho^2) + \varrho^2 D
  \]
- Distortion \( D \) is caused by scalar quantization of prediction error \( u_n \)
  \[
  D(R) = \sigma^2_U \cdot g(R)
  \]
  where \( g(R) \) is the distortion-rate function for a unit-variance Gaussian

- **Rate-dependent prediction error variance**
  \[
  \sigma^2_U = \sigma^2_S (1 - \varrho^2) + \varrho^2 \cdot \sigma^2_U \cdot g(R)
  \]
  \[
  \sigma^2_U = \sigma^2_S \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)}
  \]
DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

\[ D(R) = \sigma_U^2 \cdot g(R) \]
DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

\[ D(R) = \sigma_U^2 \cdot g(R) = \sigma_S^2 \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)} \cdot g(R) \]
DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

\[ D(R) = \sigma_u^2 \cdot g(R) = \sigma_s^2 \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)} \cdot g(R) \]

High-Rate Approximation

- High rates \( R \): \( g(R) = \varepsilon^2 \cdot 2^{-2R} \) and \( g(R) \ll 1 \)
DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

\[ D(R) = \sigma^2_U \cdot g(R) = \sigma^2_S \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)} \cdot g(R) \]

High-Rate Approximation

- High rates \( R \): \( g(R) = \varepsilon^2 \cdot 2^{-2R} \) and \( g(R) \ll 1 \)

→ Operational distortion-rate function at high rates

\[ D(R) = \varepsilon^2 \cdot \sigma^2_S \cdot (1 - \varrho^2) \cdot 2^{-2R} \]
DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

\[ D(R) = \sigma_U^2 \cdot g(R) = \sigma_S^2 \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)} \cdot g(R) \]

High-Rate Approximation

- High rates \( R \): \( g(R) = \varepsilon^2 \cdot 2^{-2R} \) and \( g(R) \ll 1 \)

- Operational distortion-rate function at high rates

\[ D(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2) \cdot 2^{-2R} \]

- Same formula as for transform coding with large KLT (\( N \to \infty \))
DPCM for Gauss-Markov: Distortion-Rate Function

Distortion-Rate Function

- Approximation of operational distortion-rate function

\[ D(R) = \sigma_U^2 \cdot g(R) = \sigma_S^2 \cdot \frac{1 - \varrho^2}{1 - \varrho^2 \cdot g(R)} \cdot g(R) \]

High-Rate Approximation

- High rates \( R \): \( g(R) = \varepsilon^2 \cdot 2^{-2R} \) and \( g(R) \ll 1 \)

\( \rightarrow \) Operational distortion-rate function at high rates

\[ D(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2) \cdot 2^{-2R} \]

\( \rightarrow \) Same formula as for transform coding with large KLT \( (N \to \infty) \)

\( \rightarrow \) Gap to rate-distortion bound represents space-filling gain of vector quantization
DPCM for Gauss-Markov: Experimental Results for $\rho = 0.9$

- Prediction error variance $\sigma_U^2$ depends on bit rate

$$\frac{\sigma_U^2(R)}{\sigma_S^2}$$

$$\sigma_U^2(R) = \sigma_S^2 \cdot \frac{1 - \rho^2}{1 - \rho^2 g(R)}$$

Measurement

$$\sigma_U^2(\infty) = \sigma_S^2 \cdot (1 - \rho^2)$$
DPCM for Gauss-Markov: Experimental Results for $\varrho = 0.9$

\[ D(R) = \sigma_s^2 g(R) \]

Space-filling gain = 1.53 dB

Distortion-rate function $D(R)$

EC-Lloyd (without prediction)

EC-Lloyd and DPCM

$G_\infty = 7.21 \text{ dB}$

High rates: Shape and memory gain are achievable

Low rates: Worse than transform coding

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization
DPCM for Gauss-Markov: Experimental Results for $\rho = 0.9$

$D(R) = \sigma^2_u(R) g(R)$

$D(R) = \sigma^2_s g(R)$

$G_F^\infty = 7.21$ dB

High rates: Shape and memory gain are achievable
DPCM for Gauss-Markov: Experimental Results for $\rho = 0.9$

- High rates: Shape and memory gain are achievable
- Low rates: Worse than transform coding
Adaptive DPCM

Audio signals, images, videos

- Instationary signals
- Single predictor not suitable for entire signal
- Adapt predictor (incl. observation set) during encoding/decoding
Adaptive DPCM

Audio signals, images, videos
- Instationary signals
- Single predictor not suitable for entire signal
- Adapt predictor (incl. observation set) during encoding/decoding

Backward adaptation
- Simultaneously estimate predictor at encoder and decoder side
- Estimation has to be based on reconstructed samples
- No additional rate, but accuracy, complexity, error resilience issues
Adaptive DPCM

Audio signals, images, videos
- Instationary signals
- Single predictor not suitable for entire signal
- Adapt predictor (incl. observation set) during encoding/decoding

Backward adaptation
- Simultaneously estimate predictor at encoder and decoder side
- Estimation has to be based on reconstructed samples
- No additional rate, but accuracy, complexity, error resilience issues

Forward adaptation
- Analyze signal at encoder side
- Send predictor as part of the bitstream (requires additional bit rate)
- Simple method: Switched predictors
Backward-Adaptive DPCM

Encoder:
- \( s_n \)
- \( u_n \)
- Quantization
- Prediction
- Estimation

Decoder:
- \( s_n \)
- Prediction
- Estimation
Forward-Adaptive DPCM

\[ s_n \xrightarrow{u_n} \text{quant.} \xrightarrow{q_n} \text{entropy encoder} \xrightarrow{\text{bitstream}} \text{entropy decoder} \xrightarrow{q_n} \text{dequant.} \xrightarrow{u'_n} \text{prediction} \xrightarrow{s'_n} \text{estimation} \xrightarrow{s_n} \hat{s}_n + \]

\[ s'_n \]
Lossy Source Coding: DPCM or Transform Coding?

Differential Pulse Coding Modulation (DPCM)
- Worse coding efficiency than transform coding at low rates (interesting operation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization
Lossy Source Coding: DPCM or Transform Coding?

**Differential Pulse Coding Modulation (DPCM)**
- Worse coding efficiency than transform coding at low rates (interesting operation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization

**Transform Coding**
- Better coding efficiency than DPCM at low rates (interesting operation points)
- Better subjective quality than DPCM: Can better control perceived quality (frequency spectrum)
- Cannot utilize statistical dependencies between blocks (only inside blocks)
Lossy Source Coding: DPCM or Transform Coding?

Differential Pulse Coding Modulation (DPCM)
- Worse coding efficiency than transform coding at low rates (interesting operation points)
- Audio, images, video: Perceptual artifacts when using rather strong quantization

Transform Coding
- Better coding efficiency than DPCM at low rates (interesting operation points)
- Better subjective quality than DPCM: Can better control perceived quality (frequency spectrum)
- Cannot utilize statistical dependencies between blocks (only inside blocks)

Combination of Transform Coding and Prediction
- Improve coding efficiency of transform coding by prediction between transform blocks
- Two concepts:  
  1. Prediction after quantization: Only for certain transform coefficients
  2. Prediction before transform: DPCM with transform coding as quantizer
Image Coding Example: JPEG

Block-based Transform Coding

1. **Transform**: Separable DCT-II for $8 \times 8$ blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
**Image Coding Example: JPEG**

![Diagram of JPEG compression process]

Block-based Transform Coding

1. **Transform**: Separable DCT-II for 8×8 blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
   - Prediction of quantization index for DC coefficient (mean of block): \( \Delta q_{(0,0)}^n = q_{(0,0)}^n - q_{(0,0)}^{n-1} \)
**Image Coding Example: JPEG**

**Block-based Transform Coding**

1. **Transform:** Separable DCT-II for $8 \times 8$ blocks of samples
2. **Quantization:** Uniform reconstruction quantizer
3. **Entropy Coding:** DC Prediction and Run-Level Coding with Huffman tables
   - Prediction of quantization index for DC coefficient (mean of block): $\Delta q_{(0,0)}^n = q_{(0,0)}^n - q_{(0,0)}^{n-1}$
   - Huffman table for prediction error $\Delta q_{(0,0)}^n$ for DC coefficient
Image Coding Example: JPEG

**Image and Video Coding / Image Coding**

**Input video picture**
- Partitioned into blocks

**Original samples**

**2d block transform**

**Transform coefficients**

**Scalar quantization**

**Quantity indexes**

**Entropy coding**

**Bitstream**

---

**Block-based Transform Coding**

1. **Transform**: Separable DCT-II for $8 \times 8$ blocks of samples
2. **Quantization**: Uniform reconstruction quantizer
3. **Entropy Coding**: DC Prediction and Run-Level Coding with Huffman tables
   - **Prediction of quantization index for DC coefficient (mean of block):** \( \Delta q_{(0,0)}^n = q_{(0,0)}^n - q_{(0,0)}^{n-1} \)
   - **Huffman table for prediction error \( \Delta q_{(0,0)}^n \) for DC coefficient**
   - **Run-level coding of remaining quantization indexes (without prediction)**
Modern Image Coding: Forward-Adaptive DPCM Structure

\[ s_n[x, y] \quad u_n[x, y] \quad t_n[x, y] \quad scalar \quad quantization \quad q_n[x, y] \quad entropy \quad coding \quad bitstream \]

\[ s'_n[x, y] \quad u'_n[x, y] \quad t'_n[x, y] \quad inverse \quad quantization \]

\[ s'^*_n[x, y] \quad mode_n \]

\[ s'^{\prime}_{pic}[x, y] \quad s'^*_{pic}[x, y] \]

Predictor selection

*Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization*
Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal
Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding
- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

Intra-Picture Prediction
- Using surrounding reconstructed samples for predicting signal of current block
- Multiple modes with directional prediction
Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

Intra-Picture Prediction

- Using surrounding reconstructed samples for predicting signal of current block
- Multiple modes with directional prediction

Examples

- H.264 | AVC: 9 prediction modes
Block-Based Intra-Picture Prediction

Block-Based Prediction and Transform Coding

- Predict signal of current block using reconstructed neighboring blocks
- Transform coding of prediction error signal

Intra-Picture Prediction

- Using surrounding reconstructed samples for predicting signal of current block
- Multiple modes with directional prediction

Examples

- H.264 | AVC: 9 prediction modes
- H.265 | HEVC: 35 prediction modes
Successive Pictures in Video Sequences

successive picture are typically very similar
(exception: scene cuts)
Successive Pictures in Video Sequences

Important: Utilize large amount of dependencies between video picture

⇒ Basic Idea: Predict current picture from already coded previous picture

**successive picture are typically very similar**
(exception: scene cuts)
Simple Variant: Frame Difference Coding

**current original picture** $s_n$

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
- Get prediction error: $u_{n}[x,y] = s_{n}[x,y] - s'_{n-1}[x,y]$
- Transform coding: $u_{n}[x,y] \mapsto u'_{n}[x,y]$
- Transmit in bitstream: Quantization indexes $\{q_k\}$
- Reconstruction: $s'_{n-1}[x,y] = s'_{n-1}[x,y] + u'_{n}[x,y]$

Problem: Ineffective for moving regions
Simple Variant: Frame Difference Coding

- Partition current picture \( s_n \) into rectangular blocks
Simple Variant: Frame Difference Coding

- Partition current picture \( s_n \) into rectangular blocks
- Block-wise coding of current picture \( s_n \)
Simple Variant: Frame Difference Coding

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
Simple Variant: Frame Difference Coding

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s_{n-1}[x, y]$
Simple Variant: Frame Difference Coding

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x, y]$
  2. Transform coding: $u_n[x, y] \rightarrow u'_n[x, y]$
Simple Variant: Frame Difference Coding

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s_{n-1}'[x, y]$
  2. Transform coding: $u_n[x, y] \mapsto u_n'[x, y]$
  3. Transmit in bitstream: Quantization indexes $\{q_k\}$
Simple Variant: Frame Difference Coding

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x, y]$
  2. Transform coding: $u_n[x, y] \rightarrow u'_n[x, y]$
  3. Transmit in bitstream: Quantization indexes $\{q_k\}$
  4. Reconstruction: $s'_n[x, y] = s'_{n-1}[x, y] + u'_n[x, y]$

Problem: Ineffective for moving regions
Simple Variant: Frame Difference Coding

- Partition current picture $s_n$ into rectangular blocks
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x, y]$  
  2. Transform coding: $u_n[x, y] \mapsto u'_n[x, y]$  
  3. Transmit in bitstream: Quantization indexes $\{q_k\}$  
  4. Reconstruction: $s'_n[x, y] = s'_{n-1}[x, y] + u'_n[x, y]$  

→ Problem: Ineffective for moving regions
Improvement: Motion-Compensated Prediction

current original picture \( s_n \)
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
**Improvement: Motion-Compensated Prediction**

- **rec. previous picture** $s'_{n-1}$
- **current original picture** $s_n$

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
- Block-wise coding of current picture $s_n$
**Improvement: Motion-Compensated Prediction**

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x + m_x, y + m_y]$
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x + m_x, y + m_y]$
  2. Transform coding: $u_n[x, y] \mapsto u'_n[x, y]$
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s_{n-1}'$
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s_{n-1}'[x + m_x, y + m_y]$
  2. Transform coding: $u_n[x, y] \mapsto u'_n[x, y]$
  3. Transmit in bitstream: Motion vector $(m_x, m_y)$ and quantization indexes $\{q_k\}$
Improvement: Motion-Compensated Prediction

- Partition current picture $s_n$ into rectangular blocks
- Estimate motion vectors $(m_x, m_y)$ of blocks in current picture relative to previous picture $s'_{n-1}$
- Block-wise coding of current picture $s_n$
  1. Get prediction error: $u_n[x, y] = s_n[x, y] - s'_{n-1}[x + m_x, y + m_y]$
  2. Transform coding: $u_n[x, y] \mapsto u'_n[x, y]$
  3. Transmit in bitstream: Motion vector $(m_x, m_y)$ and quantization indexes $\{q_k\}$
  4. Reconstruction: $s'_n[x, y] = s'_{n-1}[x + m_x, y + m_y] + u'_n[x, y]$
Hybrid Video Coding: Encoder

first input picture $s_0$
Hybrid Video Coding: Encoder

first input picture $s_0$

(partitioned into blocks)

prediction signal $\hat{s}_n[x,y] = s'_{n-1}[x+mx, y+my]$
Hybrid Video Coding: Encoder

first input picture $s_0$

(partitioned into blocks)

$s_0[x, y]$

prediction signal

$\hat{s}_n[x, y] = s_{n-1}[x + m_x, y + m_y]$

prediction error
Hybrid Video Coding: Encoder

first input picture $s_0$

(partitioned into blocks)

$s_0[x, y] \xrightarrow{\text{transform & quantization}} \text{quantization indexes } \{k\}$

prediction signal

$\hat{s}_n[x, y] = s'_{n-1}[x+mx, y+my]$
Hybrid Video Coding: Encoder

**First input picture** \(s_0\)

\(s_0[x, y]\)  

**Transform & quantization**

**Quantization indexes** \(\{k\}\)

**Entropy coding**

**Bitstream**
**Hybrid Video Coding: Encoder**

- First input picture \( s_0 \)
- \( s_0[x, y] \) transformed and quantized
- Quantization indexes \( \{ k \} \)
- Entropy coding
- Scaling & inverse transform
- Prediction signal \( \hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \)
- Prediction error

(first input picture \( s_0 \) partitioned into blocks)
Hybrid Video Coding: Encoder

First input picture $s_0$

$(partitioned into blocks)$

$s_0[x, y]$ transform & quantization quantization indexes $\{k\}$ entropy coding

scaling & inv. transform

$s'_0[x, y]$

buffer for current picture

first input picture $s_0$

$(partitioned into blocks)$

$s_0[x, y]$

transform & quantization quantization indexes $\{k\}$ entropy coding

scaling & inv. transform

$s'_0[x, y]$

buffer for current picture
Hybrid Video Coding: Encoder

First input picture $s_0$

(partitioned into blocks)

$s_0[x, y]$ -> transform & quantization -> quantization indexes $\{k\}$ -> entropy coding -> bitstream

Scaling & inv. transform

$s'_0[x, y]$ -> buffer for current picture

Prediction signal

$\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$
Hybrid Video Coding: Encoder

- **First input picture** $s_0$
- **Current input picture** $s_n$ (partitioned into blocks)
- **Frame buffer** ($s'_0$)
- **Buffer for current picture**
- **Prediction signal** $\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$
- **Prediction error**
- **Entropy coding**

($m_x$ and $m_y$ are the motion vectors.)
Hybrid Video Coding: Encoder

first input picture $s_0$

(partitioned into blocks)

$s_0[x, y] \xrightarrow{\text{transform & quantization}} \text{quantization indexes } \{k\} \xrightarrow{\text{entropy coding}} \text{bitstream}$

scaling & inv. transform

$s'_0[x, y] \xrightarrow{\text{buffer for current picture}} \text{after completion}$

frame buffer ($s'_0$)
Hybrid Video Coding: Encoder

Current input picture $s_n$

Frame buffer ($s_{n-1}$)

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization
Hybrid Video Coding: Encoder

current input picture $s_n$

(partitioned into blocks)

frame buffer ($s_{n-1}$)
Hybrid Video Coding: Encoder

- **current input picture** $s_n$
- **(partitioned into blocks)**

$\hat{s}_n[x, y] = s_{n-1}[x + m_x, y + m_y]$

- **prediction signal**
- **prediction error**

Frame buffer ($s_{n-1}$)
Hybrid Video Coding: Encoder

- Current input picture \( s_n \)
- Prediction signal \( \hat{s}_n[x, y] = s_{n-1}[x + m_x, y + m_y] \)
- Prediction error \( s_n[x, y] - s_{n-1}[x, y] \)
- Motion estimation
- Frame buffer \( (s_{n-1}) \)

(partitioned into blocks)
Hybrid Video Coding: Encoder

current input picture $s_n$

$\hat{s}_n[x, y]$

$\hat{s}_n[x, y]$

$(m_x, m_y)$

$\hat{s}_n[1]$

prediction error

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$

$\hat{s}_n = s_n - \hat{s}_n$
Hybrid Video Coding: Encoder

\[ s_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \]

- **Predictor:**
  \[
  \hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]
  \]

- **Prediction Signal:**

- **Input Picture:**
  \[
  s_n \]

- **Output Signal:**
  \[
  s'_{n-1}
  \]

- **Buffer:**
  \[
  s_{n-1}
  \]

- **Motion Estimation:**
  \[
  (m_x, m_y)
  \]

- **Motion Compensation:**
  \[
  (m_x, m_y)
  \]

- **Effective Picture:**
  \[
  s_{n-1}
  \]

- **Output Picture:**
  \[
  s'_{n-1}
  \]
Hybrid Video Coding: Encoder

current input picture \( s_n \)

(partitioned into blocks)

prediction error

prediction signal

\( s_n[x, y] \)

\( u_n[x, y] \)

\( s_n[x, y] = s_{n-1}'[x + m_x, y + m_y] \)

motion vector \((m_x, m_y)\)

motion estimation

motion compensation

frame buffer \( s_{n-1} \)

entropy coding

bitstream

first input picture \( s_0 \)

current input picture \( s_n \)

(\( \hat{s}_n[x, y] = s_{n-1}'[x + m_x, y + m_y] \))
Hybrid Video Coding: Encoder

Current input picture $s_n$

(partitioned into blocks)

Prediction error

Prediction signal

$\hat{s}_n[x, y] = s_{n-1}[x + m_x, y + m_y]$
Hybrid Video Coding: Encoder

Current input picture $s_n$

Prediction error

Prediction signal

$u_n[x, y]$

$s_n[x, y]$

$\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$

Entropy coding

Entropy coding

Motion estimation

Motion compensation

Frame buffer ($s'_{n-1}$)

Motion vector $(m_x, m_y)$

Quantization indexes $\{k\}$

Scaling & inv. transform

Bitstream

First input picture

Current input picture
Hybrid Video Coding: Encoder

current input picture $s_n$

(partitioned into blocks)

prediction error

prediction signal

$s_n[x, y]$ $u_n[x, y]$ transform & quantization quantization indexes $\{k\}$ entropy coding

scaling & inv. transform

$\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$ $u'_n[x, y]$ $s'_n[x, y]$

motion compensation

$s'_{n-1}[x, y]$ frame buffer ($s_{n-1}$)

motion estimation

$(m_x, m_y)$ motion vector $(m_x, m_y)$

bitstream
Hybrid Video Coding: Encoder

- Current input picture $s_n$
- Prediction signal
- Prediction error
- Motion estimation
- Motion compensation
- Frame buffer
- Buffer for current picture
- Entropy coding
- Bitstream

Let $s_n[x, y]$ be the current input picture, partitioned into blocks.

- Prediction signal $u_n[x, y]$: $u_n[x, y] = s_n[x + m_x, y + m_y]$

- Prediction error $s'_n[x, y] = s_{n-1}[x, y] - u'_n[x, y]$

- Motion vector $(m_x, m_y)$

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization
Hybrid Video Coding: Encoder

- Current input picture $s_n$
  - partitioned into blocks

- Prediction error

- Prediction signal

- Transform & quantization
  - $s_n[x, y]$
  - $u_n[x, y]$

- Quantization indexes $\{k\}$

- Entropy coding

- Scaling & inverse transform

- Motion estimation
  - $(m_x, m_y)$

- Motion compensation
  - $s'_n[x, y]$

- Frame buffer ($s_{n-1}$)

- Prediction signal
  - $\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$

- Prediction error
  - $u'_n[x, y]$

- After completion

- Bitstream
Hybrid Video Coding: Decoder

decoding of first picture

bitstream
Hybrid Video Coding: Decoder

decoding of first picture

bitstream → entropy decoding → quantization indexes \( \{k\} \) → reconstructed pictures

\[ \hat{s}_n[x,y] = s'_n - 1[x + m_x, y + m_y] \]

after completion

buffer for current picture

frame buffer (\( s'_0 \))

frame buffer (\( s'_{n-1} \))
Hybrid Video Coding: Decoder

decoding of first picture

bitstream → entropy decoding → quantization indexes \( \{ k \} \) → scaling & inv. transform → reconstructed residual \( s'_0[x, y] \) → output picture

buffer for current picture
Hybrid Video Coding: Decoder

decoding of first picture

bitstream → entropy decoding → quantization indexes \( \{ k \} \) → scaling & inv. transform → reconstructed residual

buffer for current picture

\[ s'_0[x, y] = s'_1[x, y] - \text{motion compensation} \]

\[ \hat{s}_n[x, y] = s'_n[x, y] - 1 \]

\[ \hat{s}_n[x, y] = s'_n[x, y] - 1 \]

\[ \hat{s}_n[x, y] = s'_n[x, y] - 1 \]
Hybrid Video Coding: Decoder

decoding of first picture

bitstream → entropy decoding → quantization indexes \( \{k\} \) → scaling & inv. transform

\( s_0'[x,y] \)

buffer for current picture

\( \hat{s}_n[x,y] = s_0'[x,y] - s_{n-1}[x+mx,y+my] \)
Hybrid Video Coding: Decoder

Decoding of first picture

1. Bitstream
2. Entropy decoding
3. Quantization indexes \(\{k\}\)
4. Scaling & inverse transform
5. Buffer for current picture
6. After completion

\[ \hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \]
Hybrid Video Coding: Decoder

Decoding of first picture

bitstream

entropy decoding

quantization indexes \( \{ k \} \)

scaling & inv. transform

\( s'_0[x, y] \)

buffer for current picture

after completion

reconstructed pictures

frame buffer (\( s'_0 \))

output

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization
Hybrid Video Coding: Decoder

decoding of \( n \)-th picture

\[
\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]
\]

after completion

frame buffer (\( s'_{n-1} \))
Hybrid Video Coding: Decoder

decoding of \( n \)-th picture

\[
\hat{s}_n[x,y] = s_{n-1}[x + m_x, y + m_y]
\]

bitstream

entropy decoding

motion vector \((m_x, m_y)\)

frame buffer \((s'_{n-1})\)
Hybrid Video Coding: Decoder

decoding of n-th picture

bitstream

entropy decoding

entropy decoding

entropy decoding

entropy decoding

entropy decoding

entropy decoding

$\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$

motion vector $(m_x, m_y)$

motion compensation

frame buffer ($s'_{n-1}$)

reconstructed pictures

buffer for current picture

output

reconstructed residual

$\hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y]$
Hybrid Video Coding: Decoder

decoding of $n$-th picture

bitstream → entropy decoding → quantization indexes \{k\}

\[ \hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \]

motion compensation

motion vector \((m_x, m_y)\)

frame buffer \((s'_{n-1})\)
Hybrid Video Coding: Decoder

decoding of $n$-th picture

bitstream

entropy decoding

quantization indexes $\{k\}$

scaling & inv. transform

reconstructed residual

$s_n[x,y] = s'_{n-1}[x + m_x, y + m_y]$

motion vector $(m_x, m_y)$

motion compensation

frame buffer ($s'_{n-1}$)
Hybrid Video Coding: Decoder

decoding of n-th picture

bitstream → entropy decoding → quantization indexes \( \{ k \} \) → scaling & inv. transform

\[ s_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \]

\[ u'_n[x, y] \]

\[ s'_n[x, y] \]

motion vector \((m_x, m_y)\)

motion compensation

frame buffer \((s'_{n-1})\)

reconstructed residual

output

reconstructed pictures

buffer for current picture

frame buffer \((s'_{0})\)
Hybrid Video Coding: Decoder

decoding of \( n \)-th picture

bitstream → 
entropy decoding → 
quantization indexes \( \{k\} \) → 
scaling & inv. transform → 

\[ s_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \]

buffer for current picture → 
motion compensation → 

\[ s'_{n-1}[x, y] \]

frame buffer \( (s'_{n-1}) \) → 
reconstructed residual

\[ u'_n[x, y] \]
Hybrid Video Coding: Decoder

decoding of n-th picture

\[ \hat{s}_n[x, y] = s'_{n-1}[x + m_x, y + m_y] \]

bitstream

entropy decoding

quantization indexes \( \{k\} \)

scaling & inv. transform

reconstructed residual

buffer for current picture

motion vector \((m_x, m_y)\)

motion compensation

after completion

frame buffer \((s'_{n-1})\)

output reconstructed pictures

bitstream

buffer for current picture

reconstructed residual
Hybrid Video Coding: Decoder

Decoding of $n$-th picture

- **Bitstream**
  - Entropy decoding
  - **Quantization indexes** $\{k\}$
  - Scaling & inv. transform

- **Motion vector** $(m_x, m_y)$
  - Motion compensation

- **Reconstructed residual** $u'_n[x, y]$

- **Buffer for current picture**
  - $s'_n[x, y]$

- **Reconstructed residual** $s'_n[x, y]$

- **After completion**
  - $\hat{s}_n[x, y] = s'_n - 1[x + m_x, y + m_y]$

- **Output**
  - Reconstructed pictures

- Frame buffer ($s'_{n-1}$)
Efficiency of Hybrid Video Coding

![Graph showing PSNR (dB) vs. bit rate [Mbit/s] with a curve labeled "JPEG" for Kimono, 1920×1080, 24Hz]
Efficiency of Hybrid Video Coding

![Graph showing PSNR vs. bit rate for frame difference coding and JPEG.]

- *Kimono, 1920 × 1080, 24Hz*

- PSNR [dB]
  - Frame difference coding
  - JPEG

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization

33 / 36
Efficiency of Hybrid Video Coding

- PSNR [dB]
- bit rate [Mbit/s]

- Hybrid video coding
- Frame difference coding

Kimono, 1920 x 1080, 24Hz
Efficiency of Hybrid Video Coding

- PSNR [dB] vs. bit rate [Mbit/s] for hybrid video coding, frame difference coding, and JPEG.
- Kimono, 1920 × 1080, 24Hz

Heiko Schwarz (Freie Universität Berlin) — Data Compression: Predictive Quantization
Efficiency of Hybrid Video Coding

Hybrid video coding

Frame difference coding

≈ 80%
Summary of Lecture

Prediction after Quantization

- Prediction of quantization indexes can improve lossless coding
- Combination with transform coding: Only useful for certain transform coefficients

Prediction before Quantization: Differential Pulse Code Modulation (DPCM)

- Reduce variance (or statistical dependencies) before quantization
- Have to use reconstructed samples for prediction in order to avoid error accumulation
- Quantization impacts quality of prediction (worse at low bit rates)
- Combination with transform coding: Prediction of complete blocks of samples

Image and Video Coding

- Combination of block-based prediction and transform coding
- Prediction of blocks of samples: Intra-picture prediction or motion-compensated prediction
- Transform coding of prediction error blocks
Exercise 1: Compare Your Lossy Codec to JPEG

Evaluate the Coding Efficiency of JPEG

- Choose a PPM image from our data base
- Encoding: Convert image to JPEG using different quality parameters (e.g., using ImageMagick)
- Decoding: Convert the JPEG file back to PPM format (e.g., using ImageMagick)
- Measure the RGB-PSNR between original and reconstructed image (tool available in KVV)
- Measure the bit rate (in bits per sample) based on size of the JPEG file

Evaluate the Coding Efficiency of your Codec

- Encode and decode the PPM image (same as for JPEG) with varying quantization step sizes
- Measure the bit rate of the compressed file and the RGB-PSNR of the reconstructed image

Compare Coding Efficiency of your Codec with that of JPEG

- Plot the RGB-PSNR over the bit rate for both your codec and JPEG (for multiple operation points)
- Compare your codec and JPEG by plotting the PSNR-rate curves into one diagram
Exercise 2: Lossy Image Compression Challenge

**Improve your codec for lossy coding of PPM images**
- Use any implementation of last weeks exercise as basis (see KVV)
- Try different simple techniques discussed in lectures and exercises

**The following might be worth trying**
- Use YCoCg format for actual coding (see implementations for lossless coding)
- Add prediction between transform blocks:
  - Prediction of quantization index for DC coefficient (as in JPEG); or
  - Subtract mean of surrounding reconstructed samples before transform (should be better), add that mean after reconstruction (dequantization + inverse transform) of prediction error
- Improve entropy coding of quantization indexes:
  - Adaptive arithmetic coding (see implementations for lossless coding)
  - Adaptive binary arithmetic coding (see implementations for lossless coding)