Innovations in Biomedical Waveform Coding: An Emerging Standard and Its Applications

Jonathan Pfaff
Fraunhofer HHI and University of Potsdam

Outline of the talk

1. Motivation for the emerging codec T.261
2. High-level structure and architectural overview
3. Prediction methods
4. Transform coding of prediction residuals
5. Experimental evaluation of T.261: Comparison to state-of-the-art audio codec
6. Experimental evaluation of T.261: Evaluation of individual coding tools
7. Illustration of the T.261 high-level functionality in a clinical use-case
8. Behavior of T.261 for AI based classifiers and within low power devices
9. Summary

Motivation for the emerging codec T.261

Motivation for biomedical waveform coding

Overview

Background

Personalized medicine in the age of AI

- Exponential growth in volume of biomedical waveform data (ECG, EEG, EMG, PPG)
- Data acquired with heterogenous recording devices
  - professional devices in hospitals
  - wearable medical devices or medical implants
  - smart watches and similar devices
- Acquired data should be processable worldwide by physicians as well as AI-based diagnostic tools

Unmet need: interoperable exchange and compression format for biomedical waveform data

Development of ITU-T Recommendation T.261

Overview

Standardization of T.261 in VCEG

Initiated by DICOM WG32

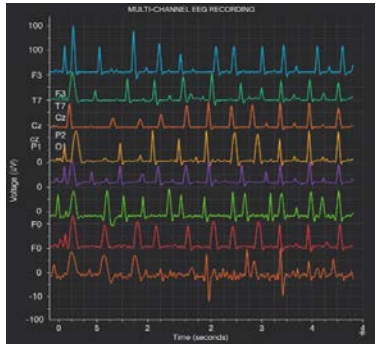
- Liaison statement from DICOM WG32 to ITU-T SG 21 Q6
- Points to "*absence of well accepted codec for compression of biomedical waveform data*"
- Asks for **development of a new codec**

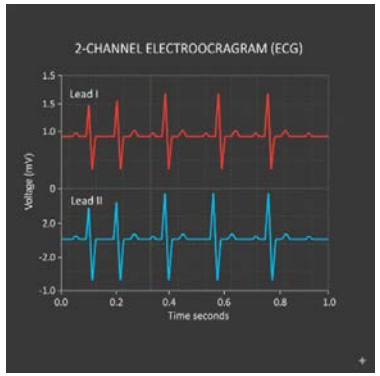
Process in ITU-T SG 21 Q 6 (VCEG):

- Call for Proposals issued for October 2024
- 4 responses: Fraunhofer HHI, Dolby, Philips, ETRI
- Development of standard in cooperation with DICOM WG32
- Developed jointly with ISO/IEC JTC1 SC29 WG6 (MPEG audio)
- Technical completion expected by July 2026
- Initial project name: BWC.

Use cases for ITU-T Recommendation T.261

Example



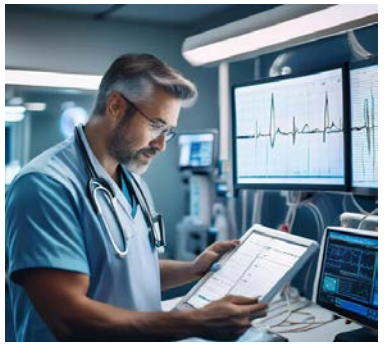


T.261 Encoder

110010…

storage or transmission

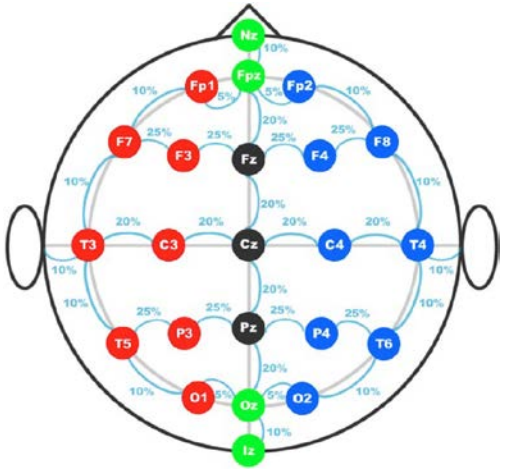
T.261 Decoder



Fraunhofer HHI

High-level structure and architectural overview

Channel groups



Basic functionality

- Describe partitioning of waveform signal into **channel groups**
- Channel group: Any group of channels

Syntax describes

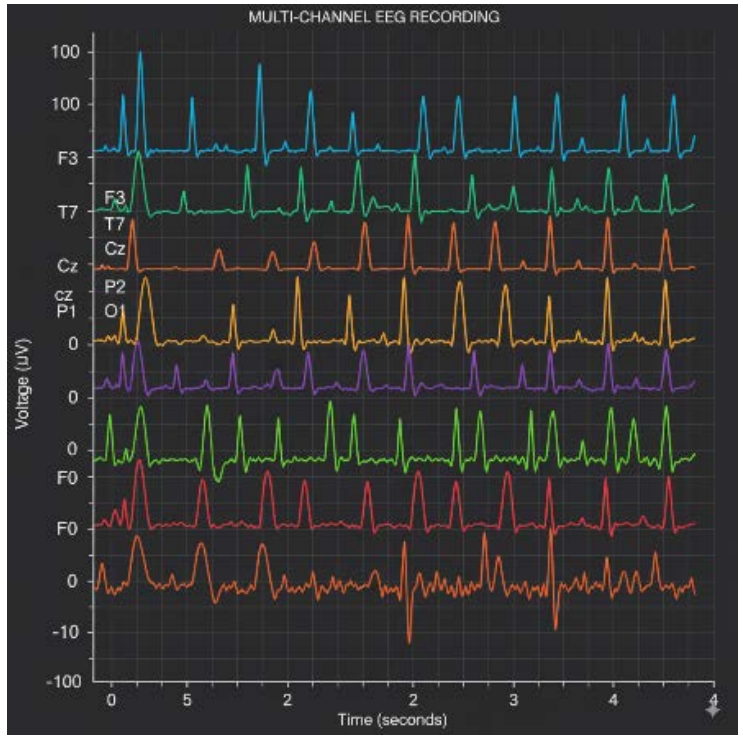
- Total number of channels
- Number of channels in each channel group
- A permutation on the set of all channels (signaled as sequence of swaps)

Syntax can represent

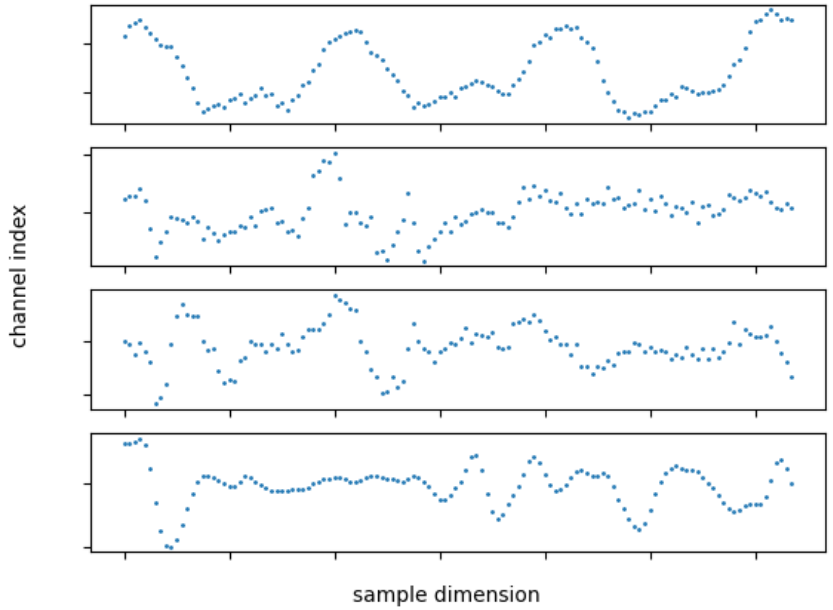
- Any re-ordering of channels for coding
- Any partitioning of re-ordered set of channels into channel groups
- Decoder restores original channel ordering from bitstream information

Channel group parameter set

- Flexible enabling of coding tools



Frame structure



Frames carry actual sample data

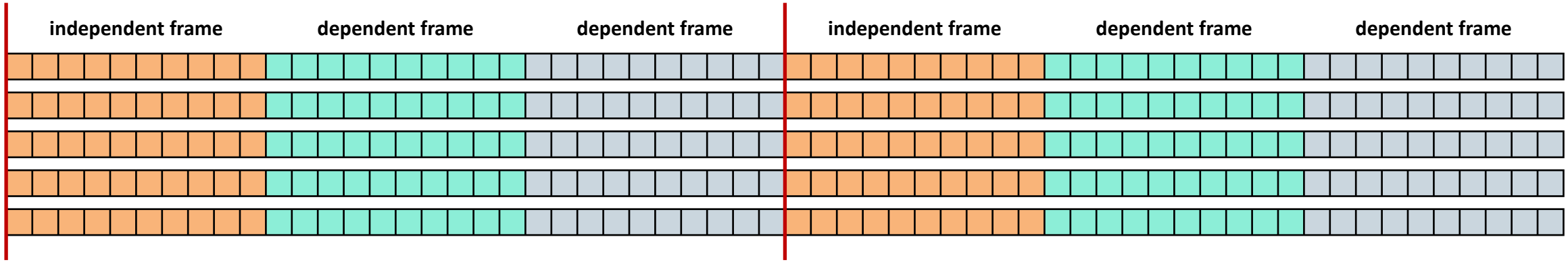
- After parsing and decoding process: Block

$$\{\hat{x}_c[i], 0 \leq c < M, 0 \leq i < N\}$$

of M channels and N samples is obtained.

Coding of Channel Groups

Independent and dependent frames



Independent Frame

- Represents random access point for channel group
- Does not use data of preceding frames for prediction
- Resets all context model states

Dependent Frame

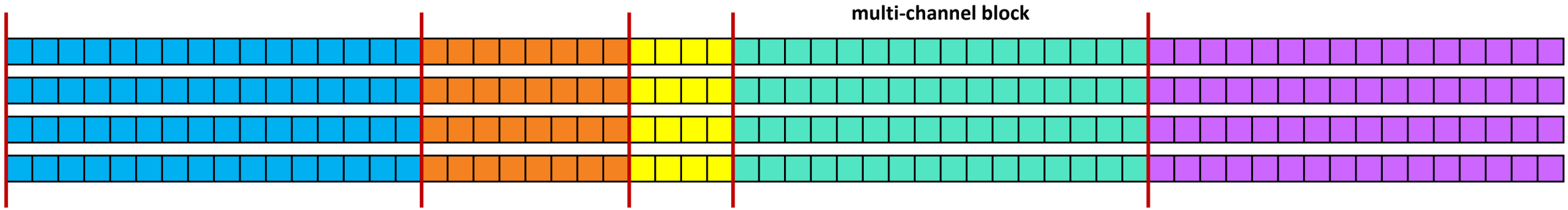
- Represents a single arithmetic codeword (limits delay)
- Can use data of preceding frames for prediction
- Context model states of preceding frames are reused

Independent frame sequence

- Collection of an independent frame and zero or more dependent frames
- Can be independently decoded

Partitioning of Frames

Variable block sizes



Partitioning of a frame into multi-channel blocks

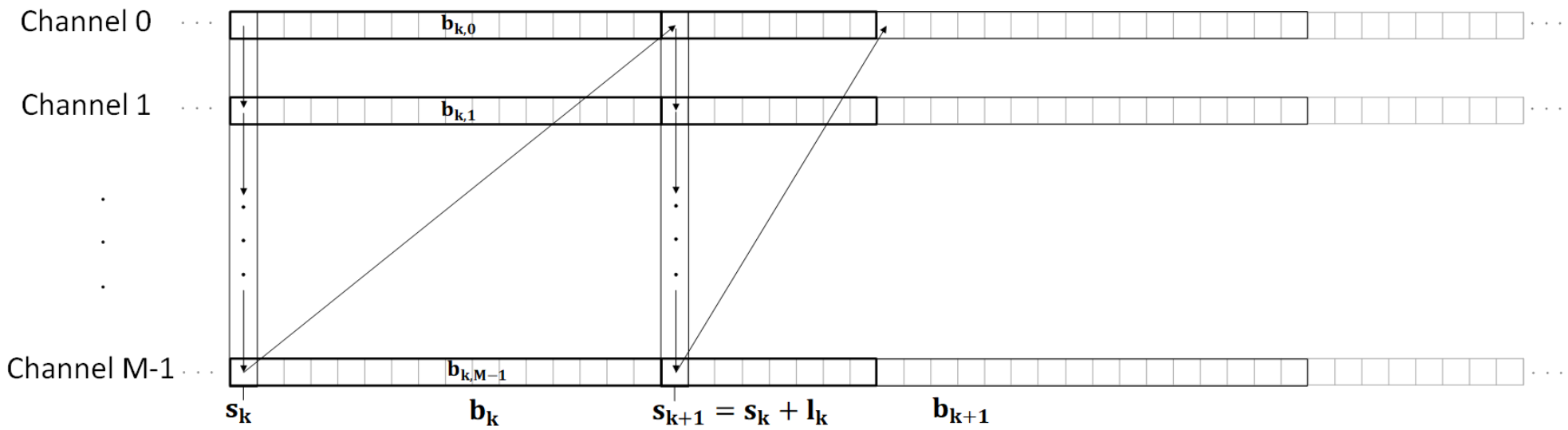
- Partitioning is the same for all channels inside a frame
- Length of a block is always an integer power of two (16, 32, 64, 128, 256, …, 8192, 16384)
- Length of a block is coded at the beginning of a block (using truncated unary code)
- Supports split-and-merge approach at the encoder side

Coding of individual sample blocks inside a multi-channel block

- 1d sample blocks are coded in channel order
- Reconstructed blocks of same channel or previously coded channel can be used for prediction

Partitioning of Frames

Coding order



Sequential coding

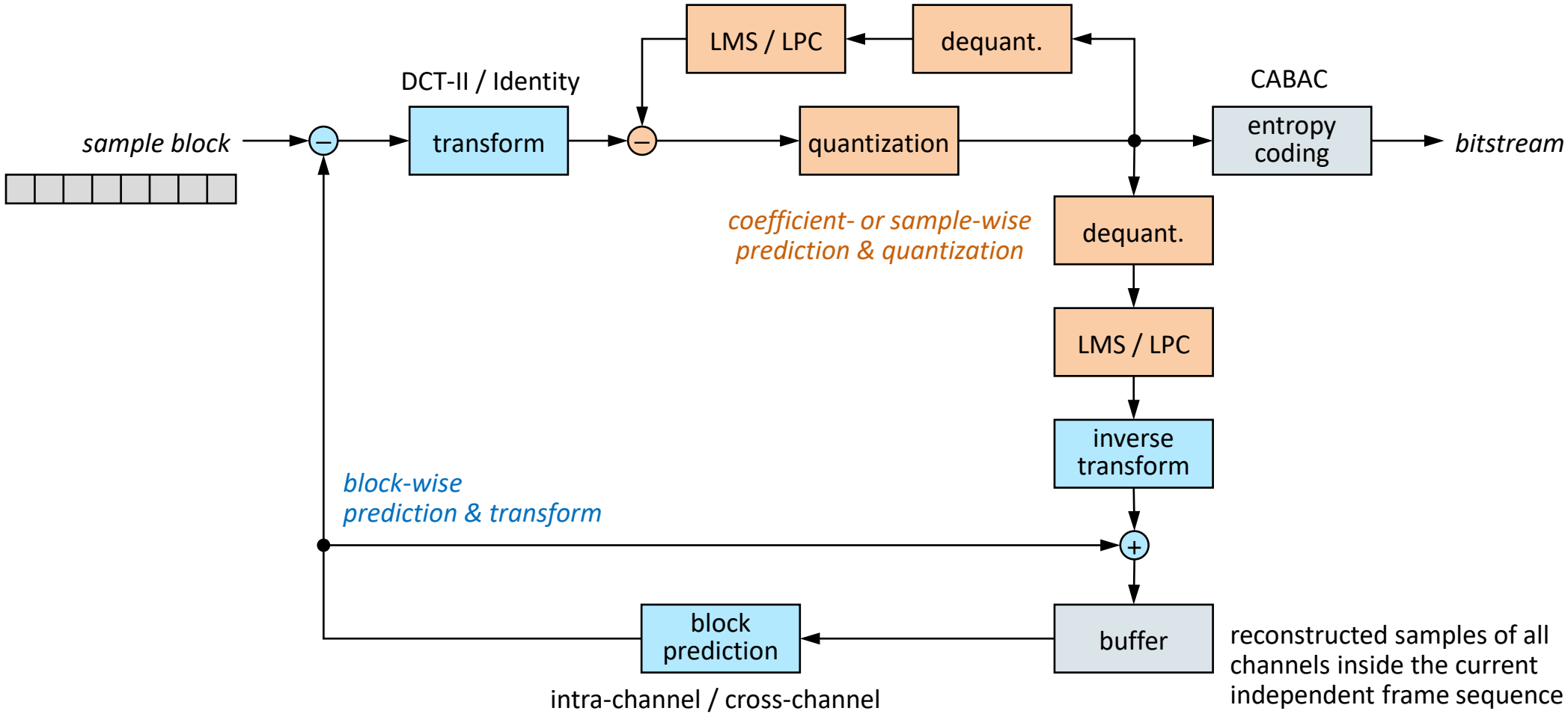
- Current starting position s_k, block length l_k
- Code block b_k consisting of all samples $\mathrm{x}_c[i], 0 \leq c < M, s_k \leq i < s_{k+1}$
- Proceed to next starting position $s_{k+1} = s_k + l_k$

Coding of block b_k

- Consecutive coding of per-channel blocks $\mathrm{b}_{\mathrm{k,c}} = \{\mathrm{x}_c[i] : s_k \leq i < s_{k+1}\}$
- Code in order $b_{k,0} \rightarrow b_{k,1} \rightarrow \cdots \rightarrow b_{k,M-1}$

Coding of One-Dimensional Sample Blocks

Overview

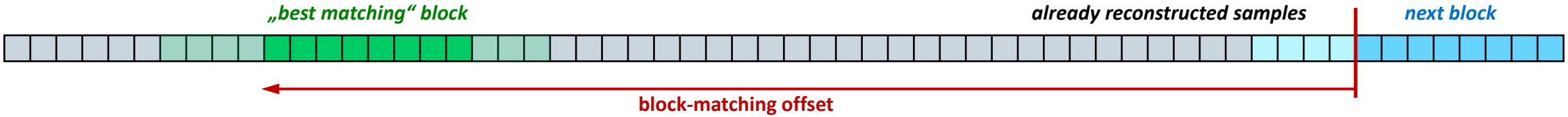


Fraunhofer HHI

Prediction methods

Block-Matching Prediction

Predict repeated pattern inside a channel



Block-matching prediction mode

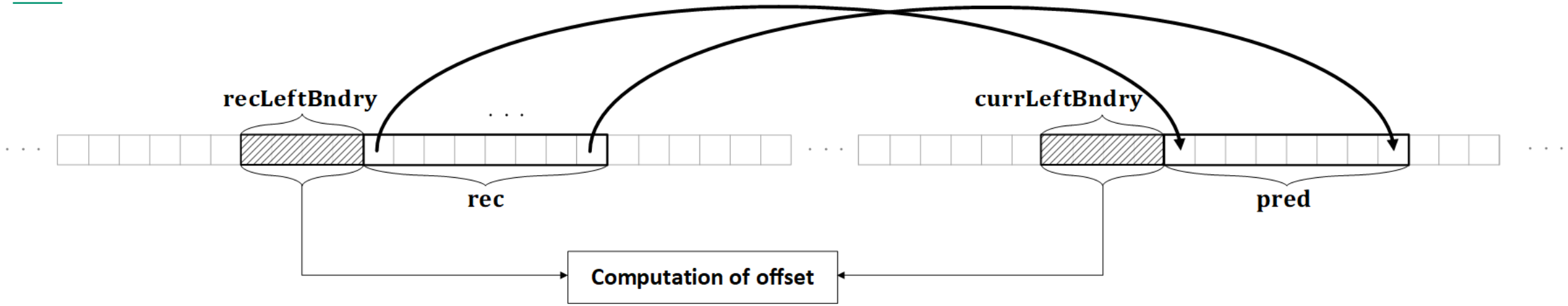
- Offset to reference block is transmitted (full-sample accuracy)
  - Can be predictively coded using offset of preceding channel or last offset of current channel as prediction
- In addition, filter identifier is transmitted
  1. No filter
  2. Smoothing filter: [-3, 0, 19, 32, 19, 0, -3] / 64
  3. Half-sample filter: [-1, -4, 8, 29, 29, 8, -4, -1] / 64

Supported extensions

- Supports multi-hypothesis prediction (one or two hypotheses)
- Supports prediction with adaptive offset

Block-Matching Extensions

Prediction with adaptive offset



Goal: Try to correct mean value deviation between prediction and target

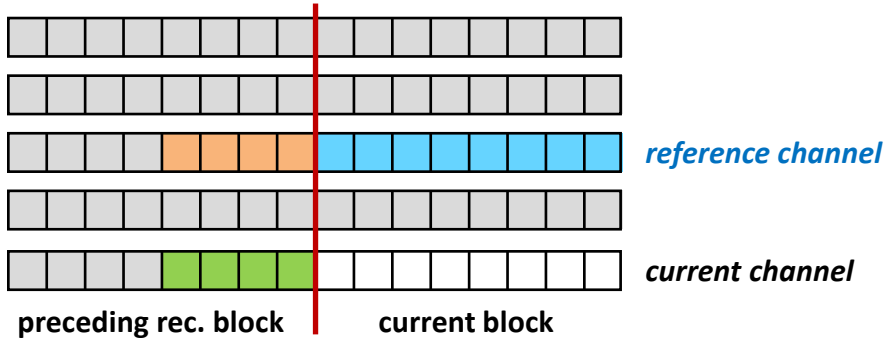
- Compute mean value $\mathbf{meanLeft}$ of reconstructed left boundary **recLeftBndry** adjacent to prediction input
- Compute mean value $\mathbf{meanCurr}$ of reconstructed left boundary **currLeftBndry** adjacent to current block
- Compute the offset

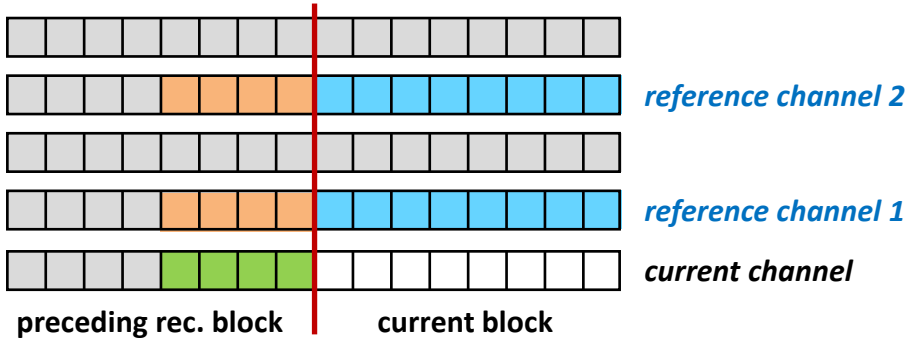
$$\mathbf{offset} = meanCurr - meanLeft$$

- Add offset to the prediction samples

Cross-Channel Prediction

Exploit linear dependencies across channels





Single-Hypothesis Cross-Channel Prediction

- Current block is predicted by affine model using an already coded and reconstructed reference channel

$$\hat{x}[k] = \alpha \cdot x'_{\mathrm{ref}}[k] + \gamma$$

- Reference channel index is transmitted
- The model parameter α and γ are determined by linear regression using the 4 preceding reconstructed samples of the current and the reference channel

$$\min \sum_{k=-4}^{-1} (x'[k] - \alpha \cdot x'_{\mathrm{ref}}[k] - \gamma)^2$$

Multi-Hypothesis Cross-Channel Prediction

- Current block is predicted by affine model using two already coded and reconstructed reference channels

$$\hat{x}[k] = \alpha \cdot x'_{\mathrm{ref1}}[k] + \beta \cdot x'_{\mathrm{ref2}}[k] + \gamma$$

- Two reference channel indices are transmitted
- The model parameter α, β and γ are determined by linear regression using the 4 preceding reconstructed samples of the current and the reference channel

$$\min \sum_{k=-4}^{-1} (x'[k] - \alpha \cdot x'_{\mathrm{ref1}}[k] - \beta \cdot x'_{\mathrm{ref2}}[k] - \gamma)^2$$

Cross-Channel Prediction Extensions

Offset only modes and filtering

Offset-Only Modes

- Weighting factors are fixed, only offset γ is calculated by linear regression
  - Single-hypothesis mode:

$$\hat{x}[k] = x'_{\mathrm{ref}}[k] + \gamma$$

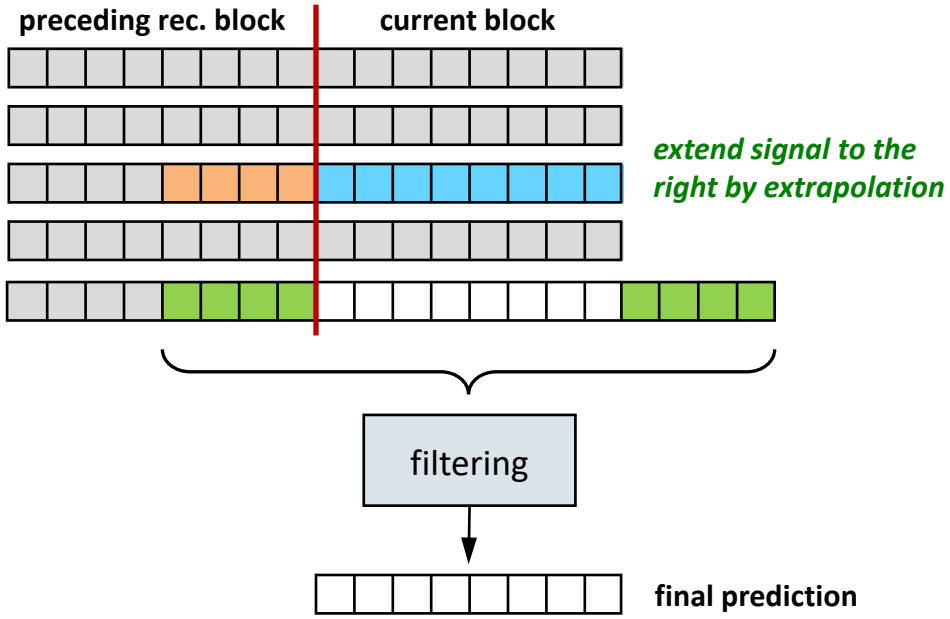
  - Multi-hypothesis mode:

$$\hat{x}[k] = 0.5 \cdot x'_{\mathrm{ref1}}[k] + 0.5 \cdot x'_{\mathrm{ref2}}[k] + \gamma$$

- Indicated by an additional flag

Cross-Channel Prediction with Filtering

- Cross-channel prediction signal $\hat{x}[k]$ is extended:
  - Left: Already coded samples
  - Right: Extrapolation by line-fitting predictor
- Final prediction signal is obtained by filtering the extended signal by one of two supported filters
- Filtering flag and filter index are transmitted

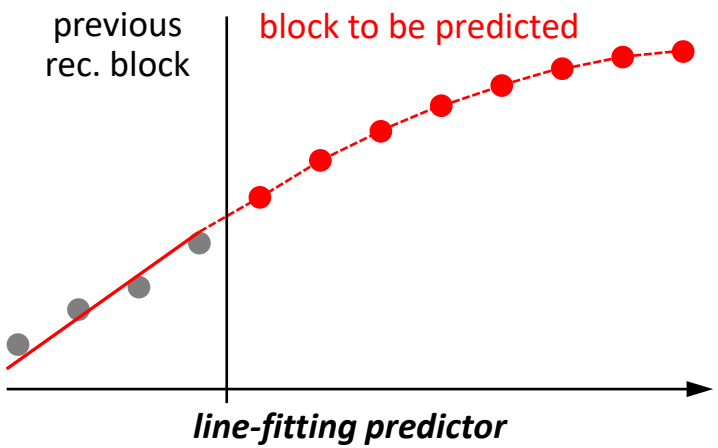


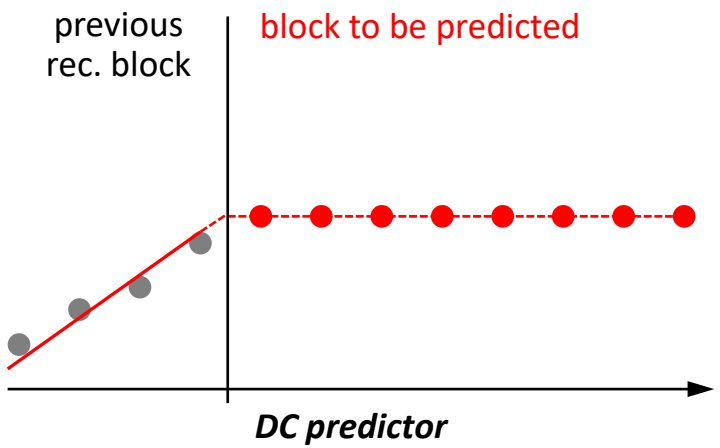
filtering modes (same as for block matching)**:**

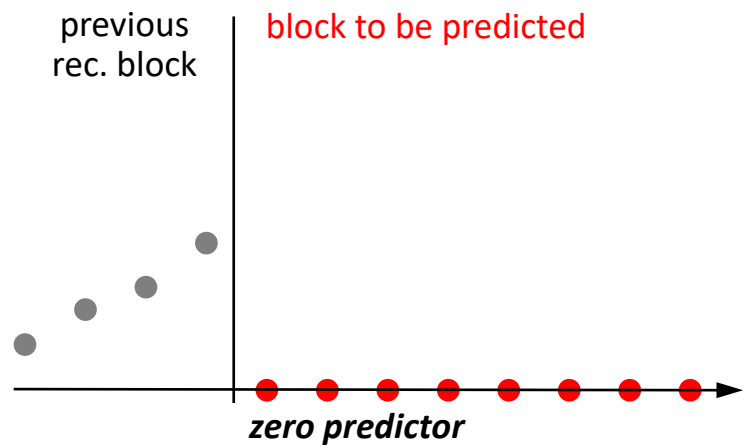
1. No filter
2. Smoothing filter: [-3, 0, 19, 32, 19, 0, -3] / 64
3. Half-sample filter: [-1, -4, 8, 29, 29, 8, -4, -1] / 64

Block Prediction by Extrapolation

Predict continuation of signal. Fall back modes, easy to implement at encoder/decoder.







Line-Fitting Predictor

- Last 4 samples of preceding reconstructed block are fitted by a straight line (affine function)
- Samples of current block are predicted by extrapolation, where absolute value of slope is decreased with the distance to the border (depending on block size)
- Results in extrapolation with quadratic function

DC Predictor

- Constant value for prediction of entire block
- DC value corresponds to weighting the last 4 samples with the filter [-43, 7, 57, 107] / 128

Zero Predictor

- Set all samples equal to zero

Transform coding of prediction residuals

Transformation and Quantization

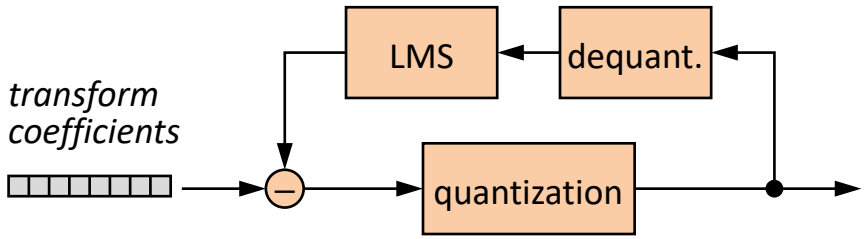
Possible interleaving of coefficient/sample prediction and quantization

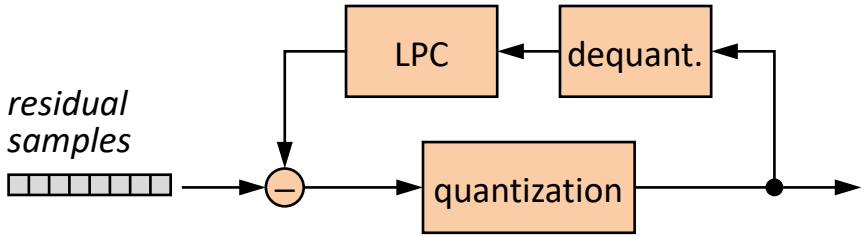
Transformation

1. Invertible integer DCT-II (lifting implementation, also applicable in lossless coding)
2. Transform skip (identity transform)

Quantization

- Three quantization options:
  1. Uniform reconstruction quantizer
  2. Dependent quantization (TCQ) with 4 states
  3. Dependent quantization (TCQ) with 8 states
- Optional interleaving (encoder) with prediction:
  - Backward-adaptive coefficient prediction (LMS), if coded in transform mode (DCT-II)
  - Forward-adaptive sample prediction (LPC), if coded in transform skip mode





Backward-Adaptive Transform Coefficient Prediction

Least mean square (LMS) predictor

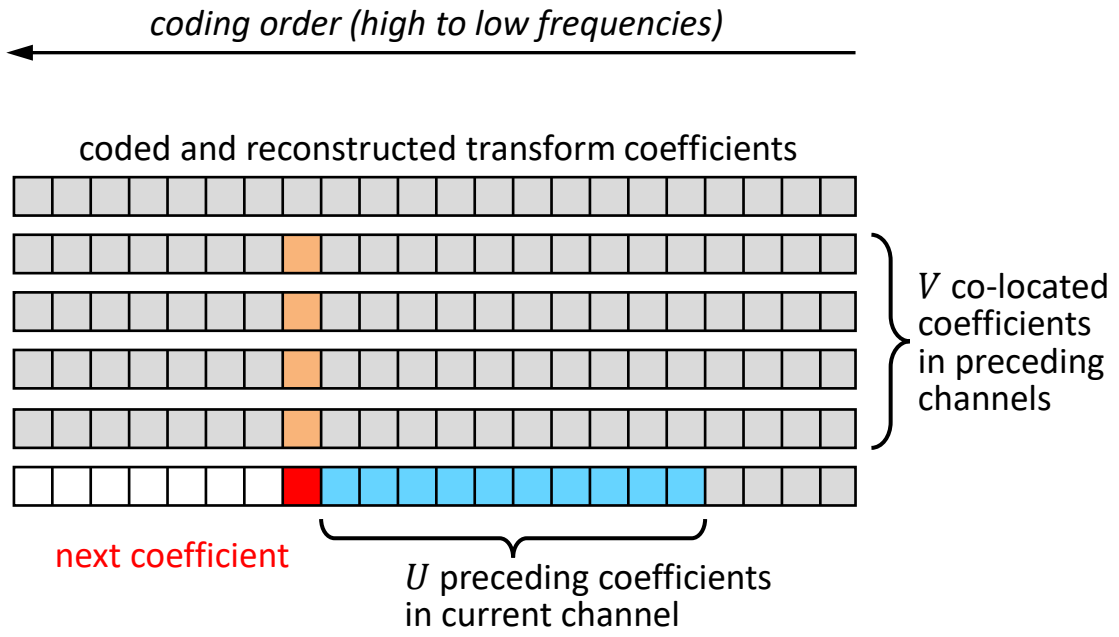
Goal

Exploit dependencies between current transform coefficient and

- Previously coded transform coefficients of same channel
- Collocated transform coefficients of previous channels

Linear prediction with signal adaptive weights

- Initialize prediction weights with zero
- Before predicting next coefficient: Gradient update on weights to reduce previous prediction error



$U, V \leq 64$ and $U + V \leq 72$

LMS prediction: Algorithmic description

Least mean square (LMS) predictor

Backward-Adaptive Transform Coefficient Prediction

1. Linear prediction of next transform coefficient

$$\hat{x}_c[k] = \sum_{u=1}^{U} a_k[u] \cdot x'_c[k+u] + \sum_{v=1}^{V} b_k[v] \cdot x'_{c-v}[k]$$

2. Quantization and reconstruction

$$r'_c[k] = \mathrm{DQ}(\mathrm{Q}(x_c[k] - \hat{x}_c[k]))$$
$$x'_c[k] = \hat{x}_c[k] + r'_c[k]$$

3. Update prediction coefficients for all u and v

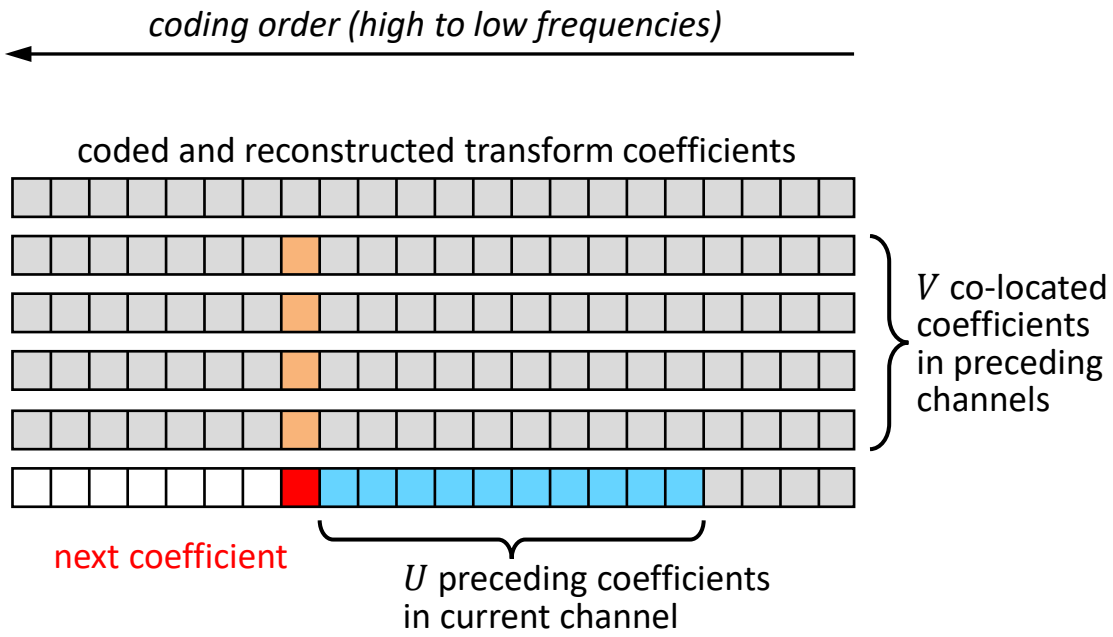
$$a_{k-1}[u] = a_k[u] + g_k \cdot x'_c[k+u]$$
$$b_{k-1}[v] = b_k[v] + g_k \cdot x'_{c+v}[k]$$

with gain

$$g_k = \frac{r'_c[k]}{4 \cdot n[k] \cdot \max(n[k], |r'_c[k]|)}$$

and norm

$$n[k] = \sqrt{\sum_{u=1}^{U} x'_c[k+u]^2 + \sum_{v=1}^{V} x'_{c+v}[k]^2}$$



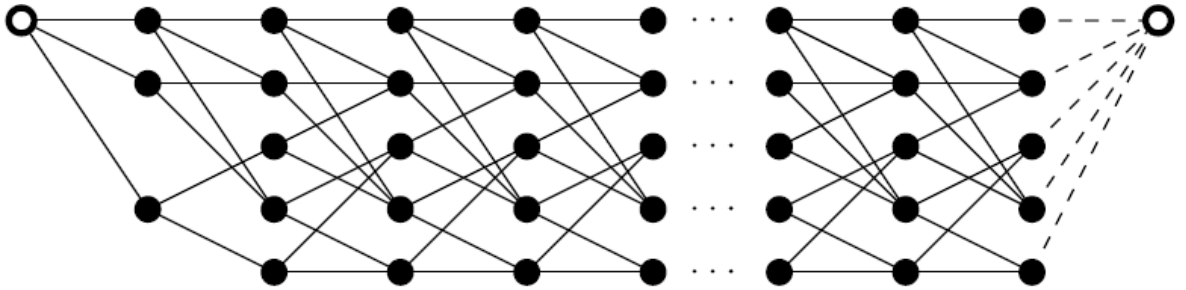
$U, V \leq 64$ and $U + V \leq 72$

LMS prediction and trellis coded quantization at encoder

LMS prediction coupled with the Viterbi Algorithm

TCQ without LMS prediction

- Need to quantize original transform coefficient $x[k]$
- **Same value** $x[k]$ for all states s_i
- State dependent quantizer



Trellis associated to 4-state TCQ. Top row: start states (all previous coefficients set to zero)

TCQ with LMS prediction

- For each state s_i , need to quantize LMS residual $r[k]$
- **Different value** of $r[k]$ for each state s_i
- ***Reason:*** **LMS prediction and LMS weights depend on state s_i .**

Computation of LMS prediction and LMS update per state:

- Obtain prediction input by backward-traversing trellis along optimal path according to Viterbi algorithm
- Obtain prediction weights from previous best state according to Viterbi algorithm
- Compute prediction
- Update state dependent prediction weights after state dependent residual reconstruction
- Instead of backward-traversing trellis: Can use FIFO buffer structure per state

Linear Predictive Coding (LPC) in Time Domain

Forward-adaptive prediction

Step 1: Prediction from one previous channel (optional)

- Modify (residual) block according to

$$x_c^*[k] = x_c[k] - \beta \cdot x'_{c-p}[k]$$

Step 2: Sample-wise linear predictive coding in time domain

1. Linear prediction of next (residual) sample

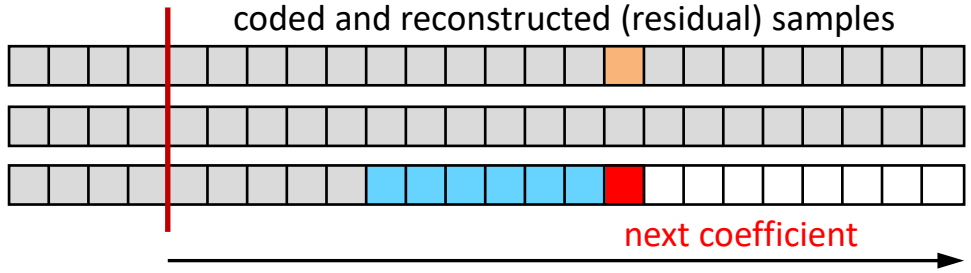
$$\hat{x}_c[k] = \sum_{i=1}^{K} \alpha_i \cdot y'_c[k-i]$$

2. Quantization and reconstruction

$$r'_c[k] = \mathrm{DQ}(\mathrm{Q}(x_c^*[k] - \hat{x}_c[k]))$$
$$y'_c[k] = \hat{x}_c[k] + r'_c[k]$$

3. Obtain final reconstructed transform coefficient as

$$x'_c[k] = y'_c[k] + \beta \cdot x'_{c-p}[k]$$



Supported prediction modes

1. Prediction modes with predetermine weights

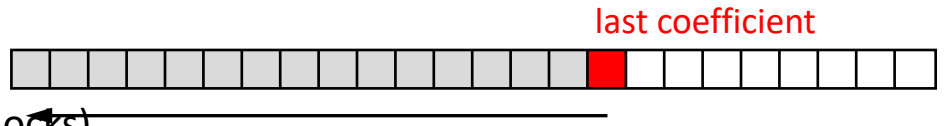
2. Flexible general sample-wise prediction mode
   - Previous channel offset p and weight β are transmitted
   - Temporal prediction order K (up to 16) is transmitted
   - Weights α_i are transmitted as reflection coefficients
3. When applicable, reconstructed samples of preceding block are used for prediction; otherwise, filter is shortened

Entropy Coding

Context-based adaptive binary arithmetic coding (CABAC)

Context-based Adaptive Binary Arithmetic Coding (CABAC)

- Coding engine and probability estimator are similar to those used in MPEG's NNC standard
- Separate context models per channel



Coding of transform blocks (from high to low frequencies)

- Location of last non-zero coefficient (predicted based only already coded blocks)
- Context-coded significance flag (context depends on state, position, template sum)
- Context-coded parity flag (only if template sum > 3)
- Context-coded truncated unary part (up to 20 flags)
- Remainder (EG0 with context-coded prefix part) and sign flag

Coding of transform-skip blocks

- Context-coded significance flag
- Context-coded truncated unary part (up to 4 flags)
- Rice code with adaptive Rice parameter and context-coded prefix part (up to 32 flags)
- Rice parameter derived on estimating coefficient magnitude from coded coefficients of same block
- Remainder (EG0 with context-coded prefix part) and sign flag

Other Aspects

Deblocking, normative encoder operation, additional data

Optional deblocking filter

- For blocks coded with DCT-II

Normative Encoder

- Goal:

  Ensure minimum reconstruction quality while providing flexible implementations

- Encoder can freely choose coding mode and associated parameters
- Quantization error must not exceed 1.5 times the quantization step size
- Requires specification of forward transform
- Requires normative prediction also at encoder side

Additional data packets

- Supports transmission of meta data
- Supports authentication (trusted data in healthcare)

Experimental evaluation of T.261: Comparison to state of the art audio codec

Setup for experimental evaluation

Reference: Extended HE-AAC

- State of the art audio codec
- Tuned for PSNR instead of perceptual audio quality
- Settings of anchor designed jointly with MPEG Audio

Reference for lossless: SLS

Reporting of results

- Bjontegaard Delta metric
- Distortion measure: PSNR

Datasets

- ECG data (2 channels and 12 channels)
- EEG data (between 24 and 64 channels)
- EMG data (4 channels)
- PPG data (2 channels and 6 channels)
- Provided by medical experts from DICOM WG 32

Coding Efficiency of T.261 over Extended HE AAC audio anchor (lossy)

Joint channel coding

Dataset	BD-PSNR	EncT
MIT (ECG)	-44,59%	511%
INCART (ECG)	-77,81%	1572%
CHBMIT (EEG)	-36,86%	977%
NMR55 (EEG)	-49,69%	638%
NMR57 (EEG)	-52,71%	2133%
Ozdemir (EMG)	-40,91%	591%
PTT (PPG)	-39,99%	2262%
WristPPG (PPG)	-40,14%	1198%
Overall	-47,84%	1235%

Coding Efficiency of T.261 over Extended HE AAC audio anchor (lossy)

Independent channel coding

Dataset	BD-PSNR	EncT
MIT (ECG)	-44,59%	758%
INCART (ECG)	-63,31%	1595%
CHBMIT (EEG)	-18,92%	477%
NMR55 (EEG)	-43,42%	426%
NMR57 (EEG)	-28,63%	550%
Ozdemir (EMG)	-40,33%	439%
PTT (PPG)	-45,14%	1774%
WristPPG (PPG)	-37,43%	945%
Overall	**-40,22%**	**870%**

Coding Efficiency of T.261 over SLS audio anchor (lossless)

Joint channel coding

Dataset	Bitrate saving	EncT
MIT (ECG)	-20,58%	604%
INCART (ECG)	-47,85%	697%
CHBMIT (EEG)	-17,57%	519%
NMR55 (EEG)	-28,97%	391%
NMR57 (EEG)	-28,72%	1079%
Tilt Illusion (EEG)	-11,03%	857%
Ozdemir (EMG)	-6,29%	1310%
PTT (PPG)	-32,23%	889%
WristPPG (PPG)	-13,75%	1013%
Overall	**-23,00%**	**818%**

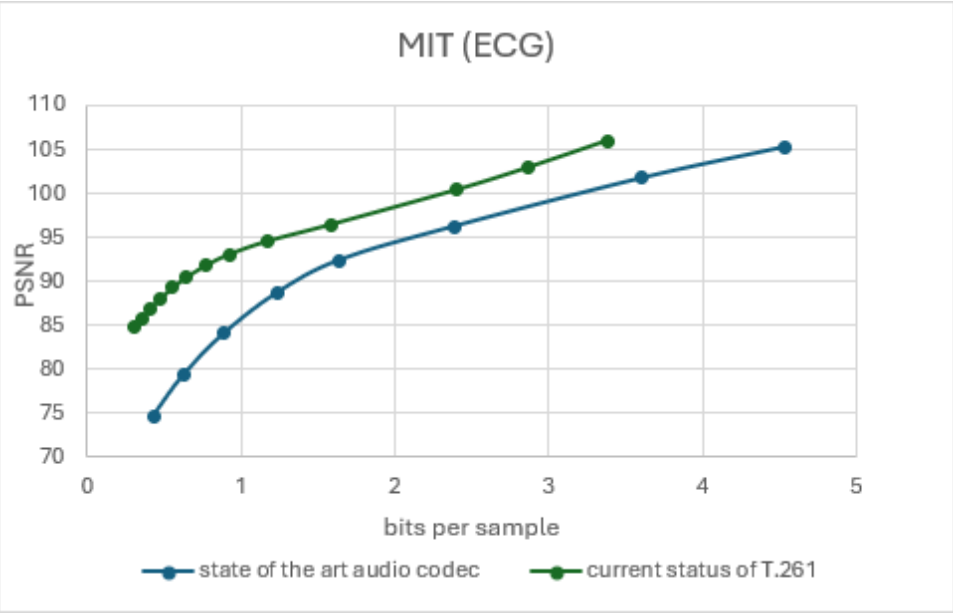
Coding Efficiency of T.261 over SLS audio anchor (lossless)

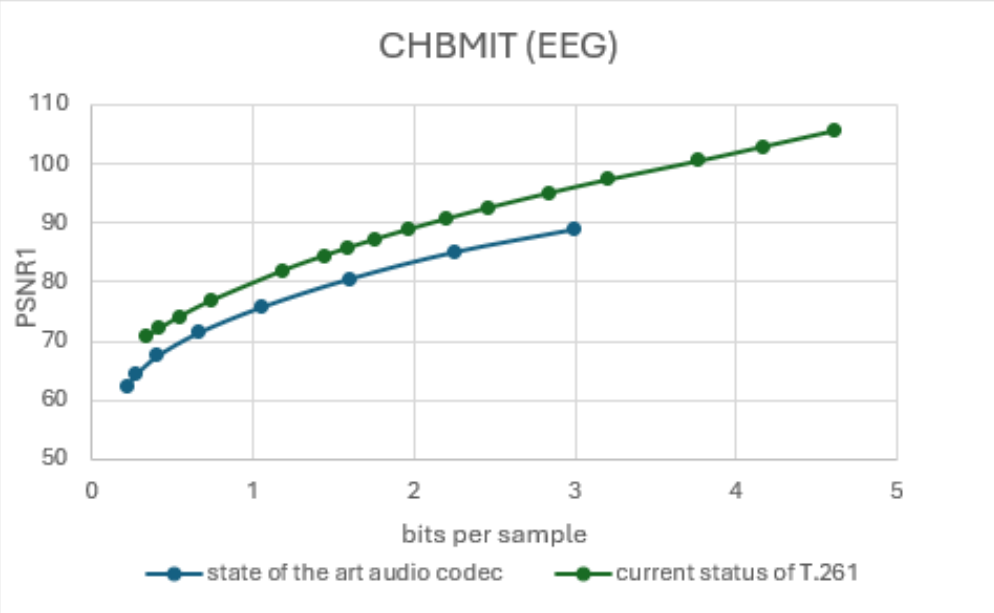
Independent channel coding

Dataset	Bitrate saving	EncT
MIT (ECG)	-20,64%	810%
INCART (ECG)	-25,32%	765%
CHBMIT (EEG)	-6,43%	88%
NMR55 (EEG)	-25,55%	88%
NMR57 (EEG)	-19,35%	87%
Tilt Illusion (EEG)	-2,35%	108%
Ozdemir (EMG)	-5,45%	897%
PTT (PPG)	-31,97%	767%
WristPPG (PPG)	-12,85%	885%
Overall	**-16,66%**	**500%**

Illustration of coding efficiency of T.261

Average rate-distortion curves for two datasets





Experimental evaluation of T.261: Evaluation of individual coding tools

Setup for evaluation of individual coding tools

Common test conditions

- Reference software BWC, publicly available
- All T.261 coding tools supported in the decoder
- Encoder:
  - Specific configuration of tools and specific search algorithms
  - Focus mainly on compression efficiency
  - Use joint channel coding
- Major coding tools can be disabled by configuration changes

Datatype-dependent partitioning

- Pre-configured block-sizes in the common test conditions depending on type of data
- Type of data: ECG, EMG, EEG, PPG,…
- Reasoning: Type of data is known by encoding device

Method for coding tool evaluation

- Use reference software in common test conditions as anchor
- Disable coding tool by configuration change
- Measure impact on BD rate, encoder runtime, decoder runtime

Disable block matching prediction

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	30,90%	28%	116%
INCART (ECG)	21,58%	47%	108%
CHBMIT (EEG)	0,27%	61%	101%
NMR55 (EEG)	2,20%	52%	103%
NMR57 (EEG)	1,08%	78%	102%
Tilt Illusion (EEG)	0,47%	72%	101%
Ozdemir (EMG)	0,98%	45%	106%
PTT (PPG)	6,50%	46%	102%
WristPPG (PPG)	1,55%	45%	99%
Overall	**7,28%**	**53%**	**104%**

Conclusion

- Block matching very effective for ECG and PPG signals
- No significant benefit for EEG and EMG signals

Disable cross channel prediction

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	0,26%	89%	98%
INCART (ECG)	23,01%	41%	116%
CHBMIT (EEG)	8,08%	51%	105%
NMR55 (EEG)	1,14%	59%	99%
NMR57 (EEG)	7,53%	25%	95%
Tilt Illusion (EEG)	4,92%	29%	101%
Ozdemir (EMG)	0,11%	70%	98%
PTT (PPG)	1,05%	62%	95%
WristPPG (PPG)	1,26%	64%	99%
Overall	**5,26%**	**54%**	**101%**

Conclusion

- Cross channel prediction very effective for 12-channel ECG (INCART) and for EEG signals
- No significant benefit for 2-channel ECG (MIT), EMG and PPG

Disable LMS prediction

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	2,79%	68%	82%
INCART (ECG)	13,45%	62%	74%
CHBMIT (EEG)	7,00%	58%	62%
NMR55 (EEG)	2,89%	63%	84%
NMR57 (EEG)	4,39%	65%	48%
Tilt Illusion (EEG)	1,30%	60%	55%
Ozdemir (EMG)	0,31%	63%	92%
PTT (PPG)	5,70%	65%	83%
WristPPG (PPG)	0,98%	60%	91%
Overall	**4,31%**	**63%**	**74%**

Conclusion

- LMS prediction very effective for 12-channel ECG (INCART), EEG signals and one PPG set (PTT)
- Less benefit for other datasets

Disable entropy constraint vector quantization

Use scalar quantization with simple rounding instead of 4-state trellis coded quantization

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	7,21%	67%	99%
INCART (ECG)	7,91%	57%	100%
CHBMIT (EEG)	6,47%	56%	100%
NMR55 (EEG)	5,05%	59%	100%
NMR57 (EEG)	6,70%	66%	100%
Tilt Illusion (EEG)	6,81%	60%	100%
Ozdemir (EMG)	4,35%	56%	101%
PTT (PPG)	5,56%	60%	102%
WristPPG (PPG)	2,52%	57%	100%
Overall	**5,84%**	**60%**	**100%**

Conclusion:

- Compression benefit rather balanced over all datasets
- General data-compression tool that exploits space filling advantage of vector quantization
- Runtime impact rather high. Reason: Combination of TCQ with LMS (see above)

Entropy constraint scalar instead of entropy constraint vector quantization

Use scalar quantization with entropy constraint instead of 4 state trellis coded quantization

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	3,92%	75%	98%
INCART (ECG)	4,23%	66%	99%
CHBMIT (EEG)	4,21%	64%	99%
NMR55 (EEG)	3,65%	67%	100%
NMR57 (EEG)	4,44%	72%	100%
Tilt Illusion (EEG)	3,67%	67%	99%
Ozdemir (EMG)	3,42%	66%	101%
PTT (PPG)	3,56%	68%	101%
WristPPG (PPG)	2,07%	68%	100%
Overall	**3,68%**	**68%**	**100%**

Conclusion :

- Compression benefit rather balanced over all datasets as for 4-state TCQ
- Encoder impact of entropy constraint scalar quantization much smaller than of 4-state TCQ
  Main reason: In scalar case, encoder needs to compute only one LMS-prediction per coefficient

Partitioning: Fixed block size 1024

CTC: Block size 1024 for EEG and EMG, block sizes 256 and 128 (split depth 1) for ECG and PPG

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	19,52%	57%	119%
INCART (ECG)	19,73%	62%	114%
CHBMIT (EEG)	0,00%	100%	99%
NMR55 (EEG)	0,00%	100%	99%
NMR57 (EEG)	0,00%	100%	100%
Tilt Illusion (EEG)	0,00%	100%	100%
Ozdemir (EMG)	0,00%	100%	99%
PTT (PPG)	8,58%	52%	105%
WristPPG (PPG)	0,36%	51%	101%
Overall	**5,35%**	**80%**	**104%**

Conclusion :

- Smaller block size very beneficial for ECG and for one PPG dataset (PTT)
- Smaller block size increase encoder runtime (due to more search)

Partitioning: Block sizes 256 and 128 (split depth 1)

CTC: Block size 1024 for EEG and EMG, block sizes 256 and 128 (split depth 1) for ECG and PPG

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	0,00%	100%	98%
INCART (ECG)	0,00%	100%	99%
CHBMIT (EEG)	4,98%	214%	92%
NMR55 (EEG)	1,09%	210%	99%
NMR57 (EEG)	4,15%	292%	94%
Tilt Illusion (EEG)	-0,16%	258%	95%
Ozdemir (EMG)	1,11%	213%	106%
PTT (PPG)	0,00%	100%	99%
WristPPG (PPG)	0,00%	100%	99%
Overall	**1,24%**	**176%**	**98%**

Conclusion :

- Larger block sizes beneficial for EEG and EMG data
- Larger block sizes beneficial in terms of encoder runtime

Partitioning: More exhaustive search

Instead of pre-configured partitioning, support all block sizes 2048, 1024, 512, 256, 128 (split depth 4)

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	-0,56%	311%	101%
INCART (ECG)	0,59%	286%	99%
CHBMIT (EEG)	-1,84%	595%	99%
NMR55 (EEG)	-1,27%	606%	96%
NMR57 (EEG)	-2,64%	630%	99%
Tilt Illusion (EEG)	-0,76%	603%	94%
Ozdemir (EMG)	-0,12%	614%	98%
PTT (PPG)	1,91%	278%	99%
WristPPG (PPG)	0,30%	267%	99%
Overall	**-0,49%**	**465%**	**98%**

Conclusion :

- Rather marginal compression benefit
- Tradeoff between encoder runtime and coding gain not attractive
- Pre-configured block sizes depending on dataset (as in CTC) seem to give better tradeoff

Disable all major coding tools

No block matching prediction, no cross channel prediction, no LMS, simple rounding in quantization

Dataset	BD-PSNR	EncT	DecT
MIT (ECG)	63,17%	6%	91%
INCART (ECG)	165,76%	5%	113%
CHBMIT (EEG)	34,78%	3%	67%
NMR55 (EEG)	17,86%	4%	87%
NMR57 (EEG)	46,76%	2%	59%
Tilt Illusion (EEG)	34,50%	2%	62%
Ozdemir (EMG)	5,55%	4%	92%
PTT (PPG)	26,91%	4%	85%
WristPPG (PPG)	8,31%	5%	95%
Overall	**44,84%**	**4%**	**84%**

Conclusion :

- Significant compression loss when disabling all major coding tools
- Significant reduction of encoder runtime
- Much less significant reduction of decoder runtime

Disable all major coding tools: Anchor Extended HE-AAC

Same ‘baseline’ version as before, but with Extended HE-AAC as reference

Dataset	BD-PSNR	EncT
MIT (ECG)	-11,53%	33%
INCART (ECG)	-25,98%	73%
CHBMIT (EEG)	-10,95%	29%
NMR55 (EEG)	-37,61%	26%
NMR57 (EEG)	-13,37%	32%
Ozdemir (EMG)	-38,51%	25%
PTT (PPG)	-16,73%	72%
WristPPG (PPG)	-32,08%	59%
Overall	-23,35%	44%

Conclusion :

- Even very low-complexity ‘baseline’ has significant compression benefit over audio anchor
- Main tools of baseline:
  - DCT-transform coding with dataset-dependent transform size
  - CABAC entropy coding of transform coefficient levels
  - Fallback prediction modes

Main conclusions from coding tool evaluations

1. Major coding tools significantly contribute to overall compression efficiency

- Results are reported when all other tools are enabled in anchor
- Thus: Gain that the measured tools provide is not exploited by other tools

2. Impact of coding tool on compression efficiency strongly depends on data type

- Performance of tools strongly varies with data-type
- Example 1: Block-matching prediction very effective for ECG, much less effective for EEG
- Example 2: Cross channel and LMS prediction very effective on EEG, much less effective for two-channel ECG *Reason:* Both (mainly) exploit inter-channel correlations.
- Exception: Trellis coded quantization / entropy constraint scalar quantization (general data compression tools)

3. Variable block sizes helpful, but block size can be configured per data type

- Fixed block size can be set per data type
- Different block sizes optimal for different data types
- Increasing partitioning options per sequence: Minor compression benefit at high encoder runtime increase
- Very different to behavior of modern video-codecs:
  Flexible partitioning per picture highly contributes to coding gain of e.g. H.265/HEVC or H.266/VVC

Illustration of the T.261 high-level functionality in a clinical use-case

Multi channel EEG waveform signals

Data acquisition

International 10-20 system

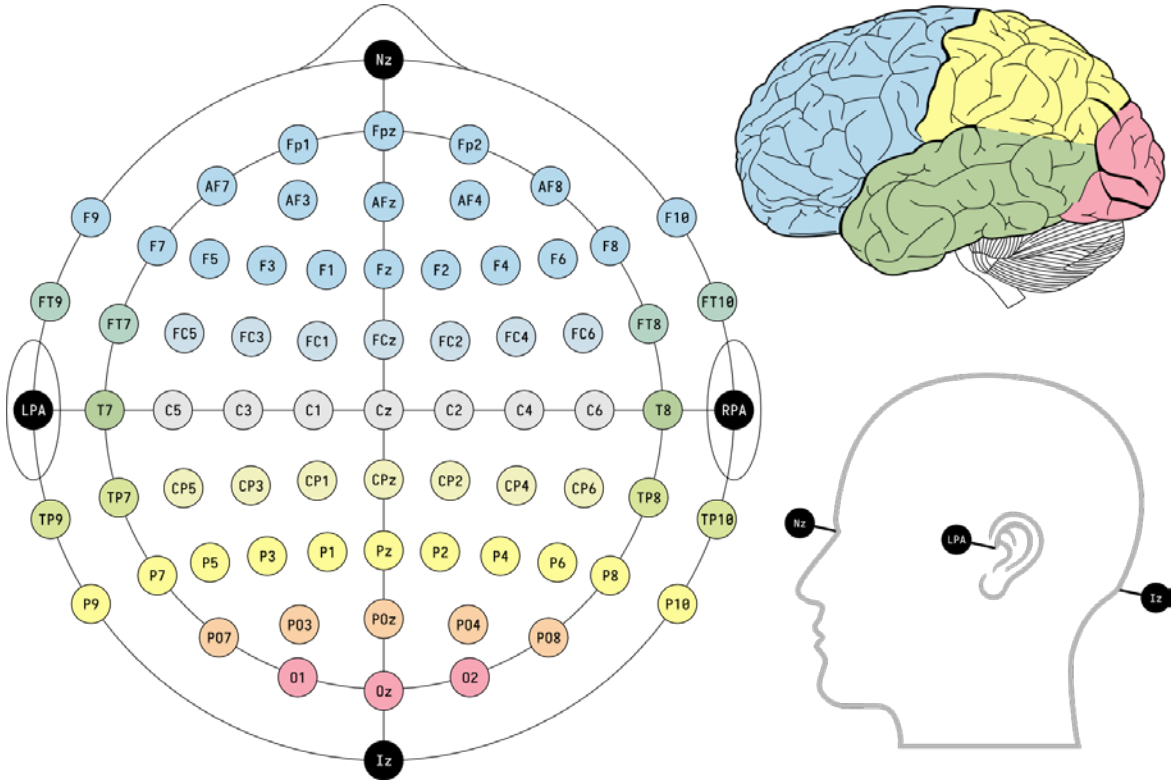
- Standard method for positioning EEG electrodes on scalp
- Electrodes positioned at sites of 10% / 20% of total distance between specific anatomical landmarks

Montage

- Definition of a set of predefine pairs of electrodes
- Example: Pairs of adjacent electrodes, pairs of fixed reference electrode and varying electrodes…

Multi-channel biomedical signal

- Channels correspond to the time-varying voltages between electrode pairs defined by the montage



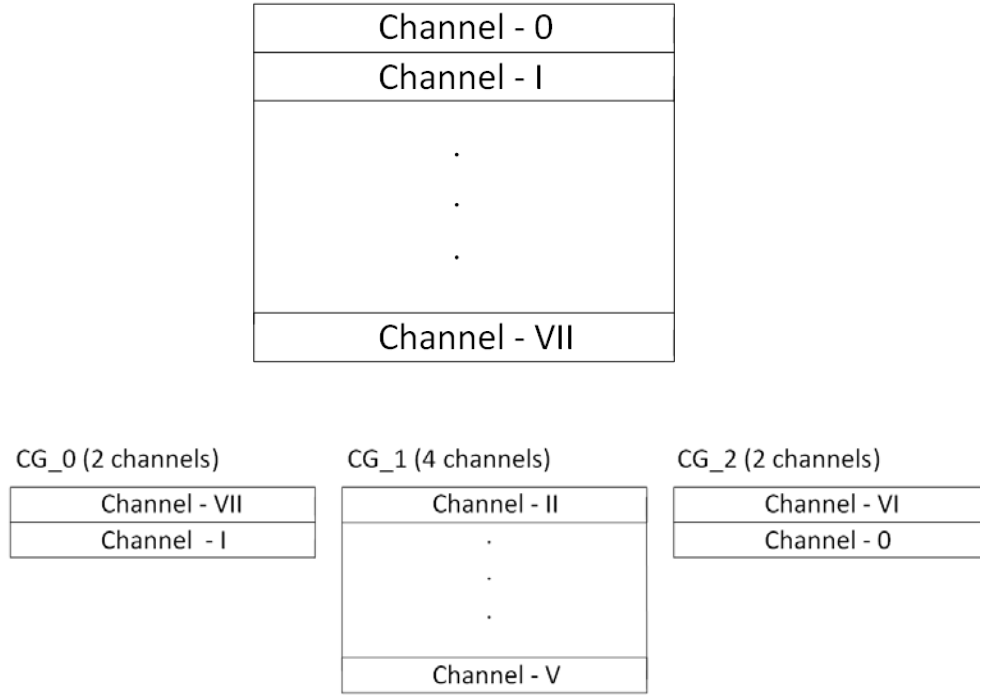
Channel grouping for EEG waveform signals

Setup

Problem statement

- Typical EEG waveform signals consist of many channels
- Coding of all channels together may cause:
  - Computational burden at encoder, e.g. due to large search range for cross channel prediction
  - Large memory requirement, both for encoder and decoder
  - Delay in accessing individual channels at decoder due to large channel-interdependencies
- **Problem**: Coding each channel individually might harm compression performance

Goal: Use the T.261 channel grouping mechanism for channel partitioning with smaller coding loss



Example of channel grouping for 8 channels. Code grouping and channel permutation in waveform parameter set

Example of a T.261 bitstream structure with channel grouping

CHB-MIT Scalp EEG Database

Original files

- Channels arbitrarily ordered
- Locations of electrode pairs in 10-20 system part of EDF medatada

Reordered files

- Group channels according to locations of electrode pairs within brain segments
- Obtain 6 channel groups

T.261 encoding

- T.261 compliant bitstream containing information on 6 channel groups
- Decoder can restore original ordering from bitstream

Original label	Reordered index	Original index	Group
FP1-F7	0	0	0
FP1-F3	1	4	
FP2-F4	2	8	
FP2-F8	3	12	
FT9-FT10	4	20	1
F7-T7	5	1	
F3-C3	6	5	2
FZ-CZ	7	16	
F4-C4	8	9	
F8-T8	9	13	
T7-FT9	10	19	3
T7-P7	11	2	
C3-P3	12	6	4
CZ-PZ	13	17	
C4-P4	14	10	
FT10-T8	15	21	5
T8-P8	16	14	
T8-P8	17	22	
P7-T7	18	18	6
P7-O1	19	3	
P3-O1	20	7	
P4-O2	21	11	
P8-O2	22	15	

Experimental assessment of channel-ordering functionality

Setup of assessment of compression behavior

- Experiment A: Encode with trivial splitting, i.e. simply split file into 6 files of consecutive channels and code each file with T.261
- Experiment B: Encode with channel grouping from previous slide, form a T.261 bitstream with 6 channel groups
- Compare both to joint T.261 – coding of all channels (always best *in terms of compression performance*)

Results

- Experiment A: 20% bitrate overhead, at 65% encoder runtime
- Experiment B: 10% bitrate overhead, at 60 % encoder runtime

Conclusion: Channel grouping very useful in terms of compression performance when high-level data-partitioning is required.

Behavior of T.261 for AI based classifiers and within low power devices

Suitability of T.261 for AI-based Processing of Biomedical Data

Experimental validation

Experimental setup (12-bit ECG):

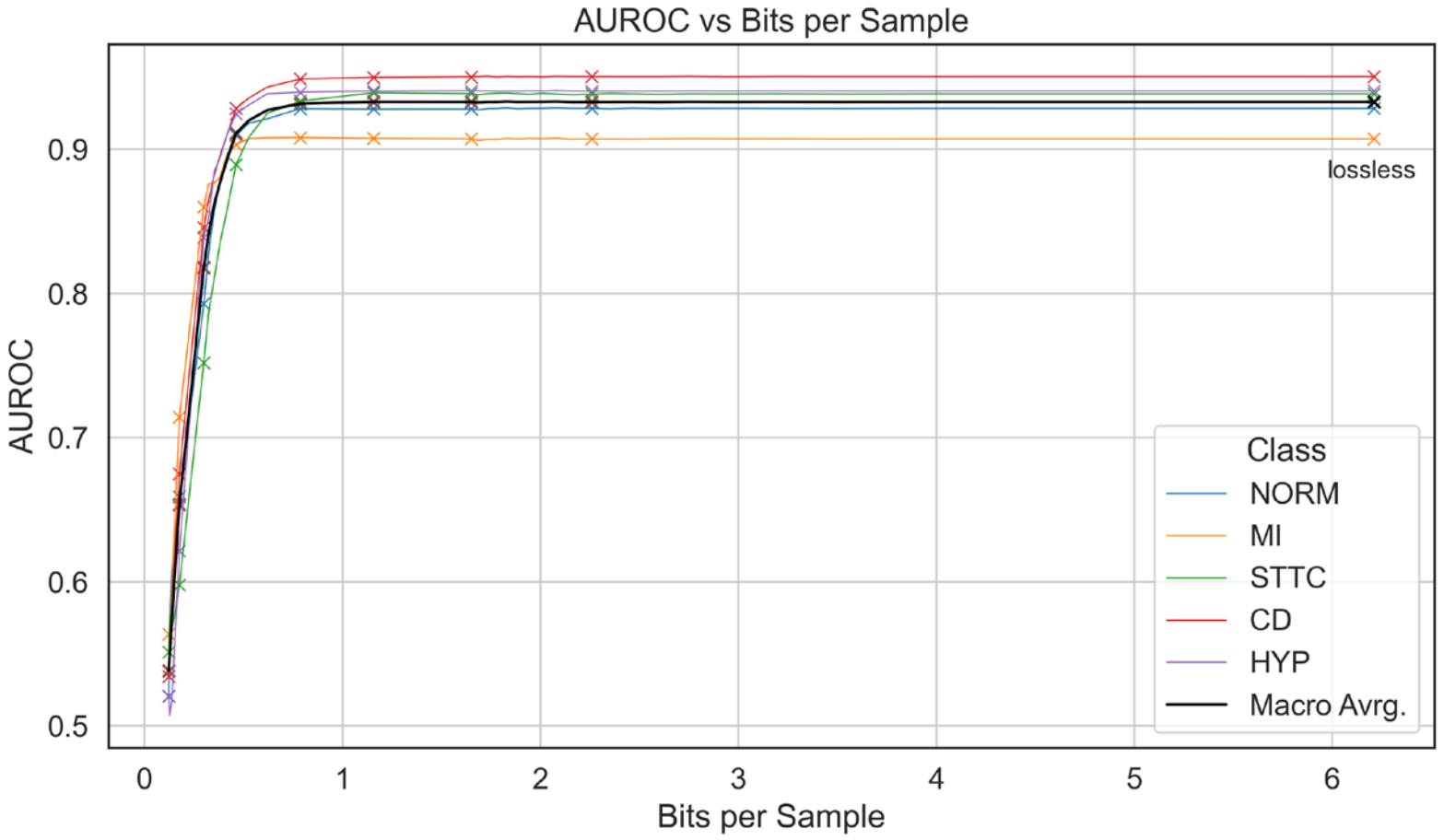
- AI-based classifier for diagnostic labels on ECG data
- Compress ECG data with T.261
- Compare classifier performance on compressed and uncompressed data

✓ No degradation in performance up to a compression factor of about 15

✓ Can save 80% of the bit rate relative to lossless compression

Note:

- Classifier trained on uncompressed data
- T.261 encoder optimizes for MSE

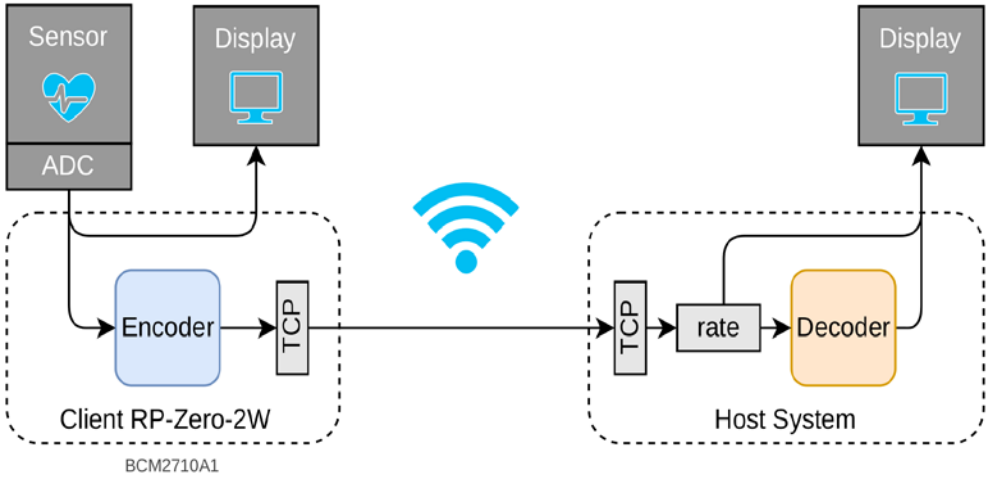


Suitability of T.261 for Implementation on Low-Power Devices

Setup of experimental prototype

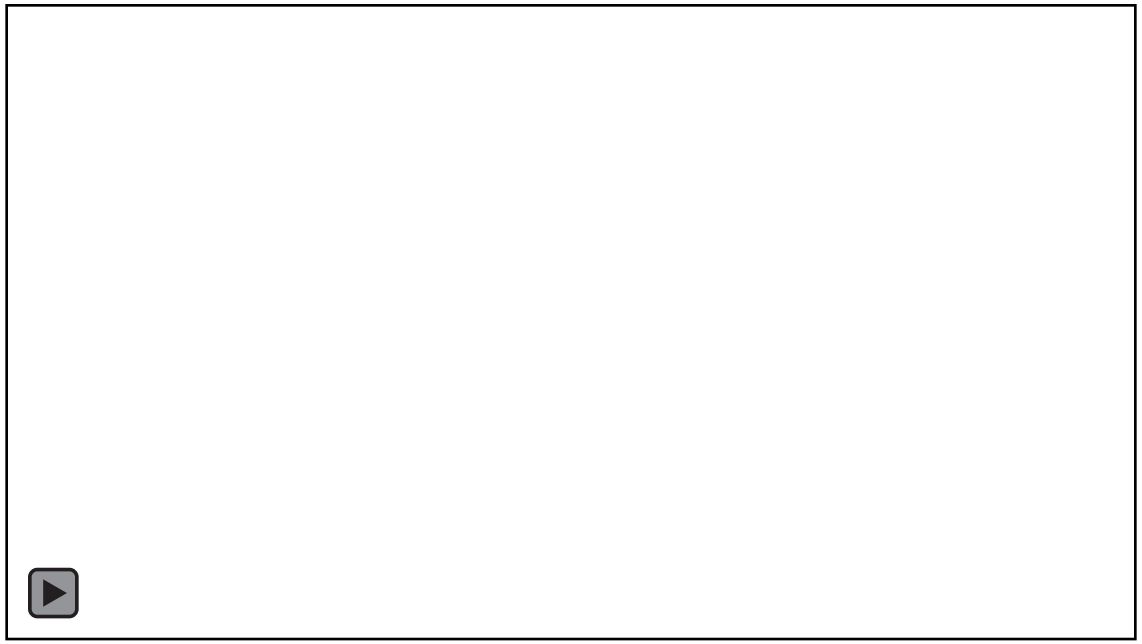
Experimental setup:

- Acquisition of single-channel data with optical pulse sensor (Iduino SE050)
- Analog-to-digital conversion with 16-bit TI ADS1115 (sampling rates 250, 475, and 860 Hz)
- Encoder on Raspberry Pi Zero 2W
  - Quad-core @ 1 GHz ARM Cortex®-A53, 512 MB RAM
  - Similar to Samsung Galaxy Watch 3 (2020)
  - 5-15% single core utilization, less than 6 Mbyte memory
- Transmission via WiFi
- Decoding and displaying on a computer



Suitability of T.261 for Implementation on Low-Power Devices

Experimental validation



Experimental validation:

- ✓ Significant data size reduction in real time
- ✓ Minimal overall latency from acquisition to display

Summary

Upcoming ITU-T Recommendation T.261

Summary

Key properties

- Interoperable exchange format for biomedical waveform data
- Developed jointly by standardization organizations from medicine and communications
- Guarantees efficient storage and transmission of biomedical waveform data
- Covers entire quality range up to lossless compression
- Significant compression benefit over state of the art

Key benefits

- Facilitates personalized health care
- Enables people to share their medical data acquired by low-end devices with clinicians
- Enables development and improvements of AI-based diagnostic algorithms
- Compatibility across devices worldwide

Thank you for your attention!