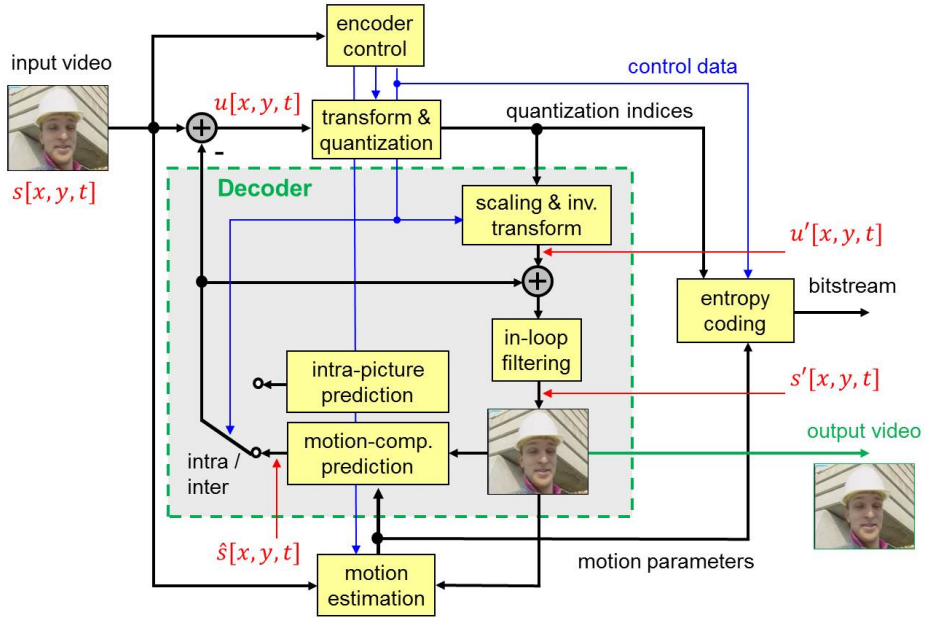
Source Coding and Compression

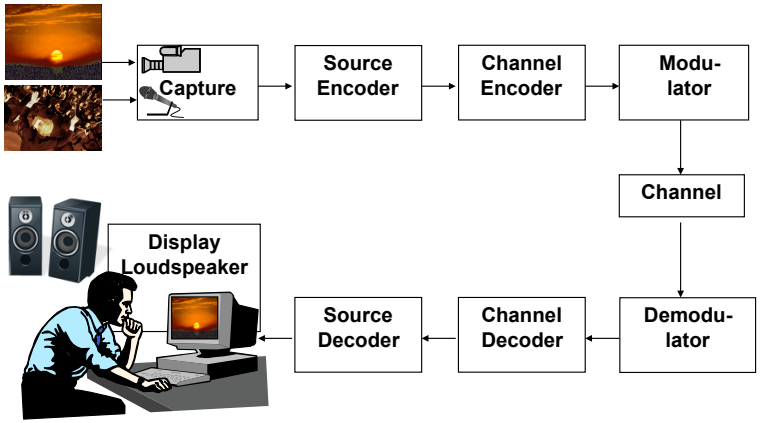
Heiko Schwarz



Contact:
Dr.-Ing. Heiko Schwarz
heiko.schwarz@hhi.fraunhofer.de

Outline

Introduction



Motivation for Source Coding

- **Source coding** or **compression** is required for efficient **transmission** or **storage**, leading to one or both of the following benefits:
  - Transmit more data given throughput (channel capacity or storage space)
  - Use less throughput given data
- Typically, source coding or compression are considered enabling technologies, i.e., technologies that enable an application
- Examples for source coding applications:
  - gzip, compress, winzip, ...
  - Mobile voice, audio, and video transmission
  - Internet voice, audio, and video transmission
  - Digital television
  - MP3 and portable video players (iPod, ...)
  - Digital Versatile Discs (DVDs) and Blu-Ray Discs

Practical Source Coding Problems

- **File compression** (text file, office document, program code, ...)
  - Example: Example 80 MByte down to 20 MByte (20%)
- **Audio compression**
  - Stereo with sampling frequency of 44.1 kHz
  - Each sample being represented with 16 bits

  $\Longrightarrow$ Raw data rate: $44.1 \times 16 \times 2 = 1.41$ Mbit/s

  $\Longrightarrow$ Typical data rate after compression: 64 kbit/s (4.5%)
- **Image compression**
  - Original picture size: 3000×2000 samples (6 MegaPixel)
  - 3 color components (red, green, blue) and 1 byte (8 bit) per sample

  $\Longrightarrow$ Raw file size: $3000 \times 2000 \times 3 = 18$ MByte

  $\Longrightarrow$ Typical compressed file size: 1 MByte (5.6%)
- **Video compression**
  - Picture size of 1920×1080 pixels and frame rate of 50 Hz
  - Each sample being digitized with 8 bit
  - 3 color components (red, green, blue)

  $\Longrightarrow$ Raw data rate: $1920 \times 1080 \times 8 \times 50 \times 3 = 2.49$ Gbit/s

  $\Longrightarrow$ Typical compressed data rate: 12 Mbit/s (0.5%)

Source Coding in Practice

- Source coding often **enables applications**:
  - Digital television (DVB-T)
  - Internet video streaming (YouTube)
- Source coding **makes applications economically feasible**
  - Distribution of digital images
  - High definition television (HDTV) over IPTV
- Many applications use source coding techniques
  - Software is often distributed in compressed form
  - Audio data are typically compressed (MP3, AAC)
    - Mobile audio players (IPod,...) and mobile phones
    - Audio download (ITunes) and streaming services (Internet radio)
  - Digital images are typically compressed (JPEG)
    - Compression is often done in camera
    - Picture found on web sites are compressed
  - Digital video data are typically compressed (MPEG-2, H.264/AVC)
    - Output of video cameras, optical discs
    - Video streaming (Youtube, Internet TV)
  - About 70% of the bits in the Internet are compressed video data

Pulse-Code Modulation

Analog-to-Digital Conversion: Pulse-Code Modulation

- Pulse-code modulation (PCM) is based on following principles
  - Sampling (obeying SHANNON-NYQUIST sampling theorem)
  - Quantizing sample values

- Sampling theorem asserts that a time-continuous signal $s(t)$ that contains only frequencies less than Ω Hz, can be recovered from a sequence of its sample values using

$$s(t) = \sum_{n=-\infty}^{\infty} s(t_n)\, \psi(t - t_n) \tag{1}$$

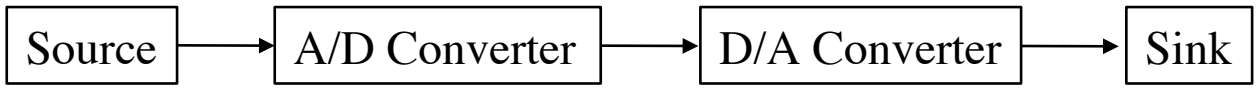
  where $s(t_n)$ is value of nth sampling instant $t_n = \frac{n}{2\Omega}$ and $\psi(\cdot)$ is given as

$$\psi(t) = \frac{\sin(2\pi\Omega t)}{2\pi\Omega t} \tag{2}$$

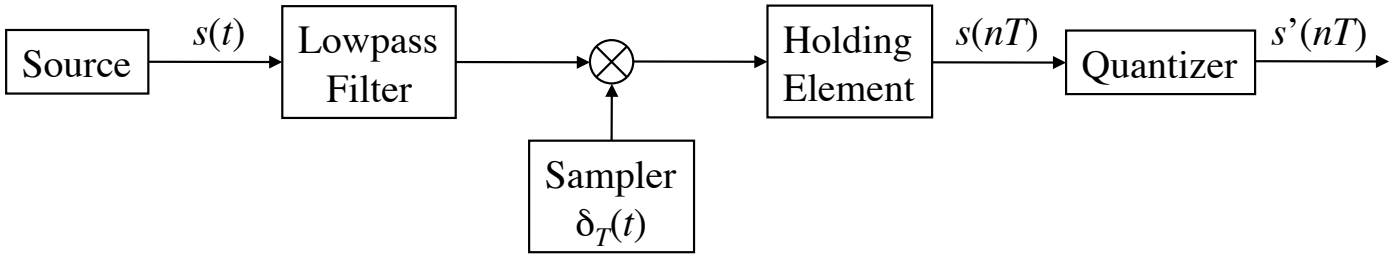
- The signal values $s(t_n)$ can be quantized allowing only an approximate reconstruction of $s(t)$

Analog-to-Digital Conversion: Overview

- Analog-to-digital and digital-to-analog conversion

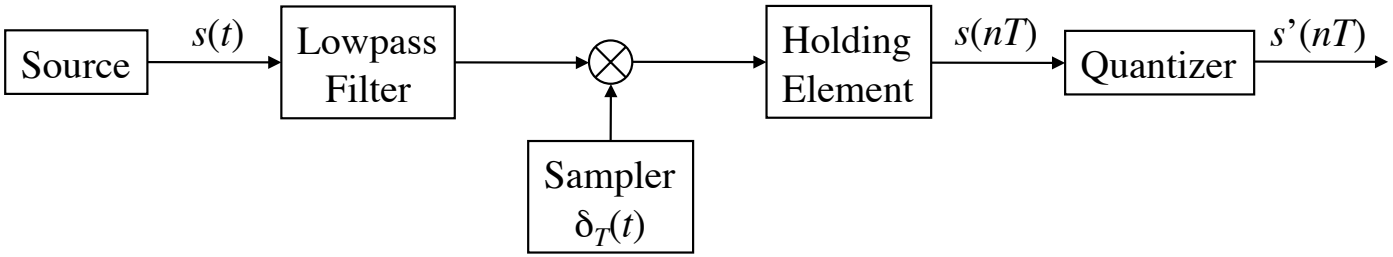


- Source and analog-to-digital converter



- Analog-to-digital converter turns analog signal into a discrete signal
  - **Analog signal**: Continuous-time and continuous-amplitude signal
  - **Discrete signal**: Discrete-time and discrete-amplitude signal

Analog-to-Digital Conversion



- Sample and hold operator turns continuous-time into discrete-time signal
- Low-pass filter ensures that signal is band-limited
- Quantizer turns continuous-amplitude signal into discrete-amplitude signal
  - A simple method is to quantize signal $s(nT)$ by mapping it to $K = 2^k$ possible amplitude values
  - A simple quantization rule is

$$s'(nT) = \lfloor s(nT) \times 2^k + 0.5 \rfloor / 2^k \tag{3}$$

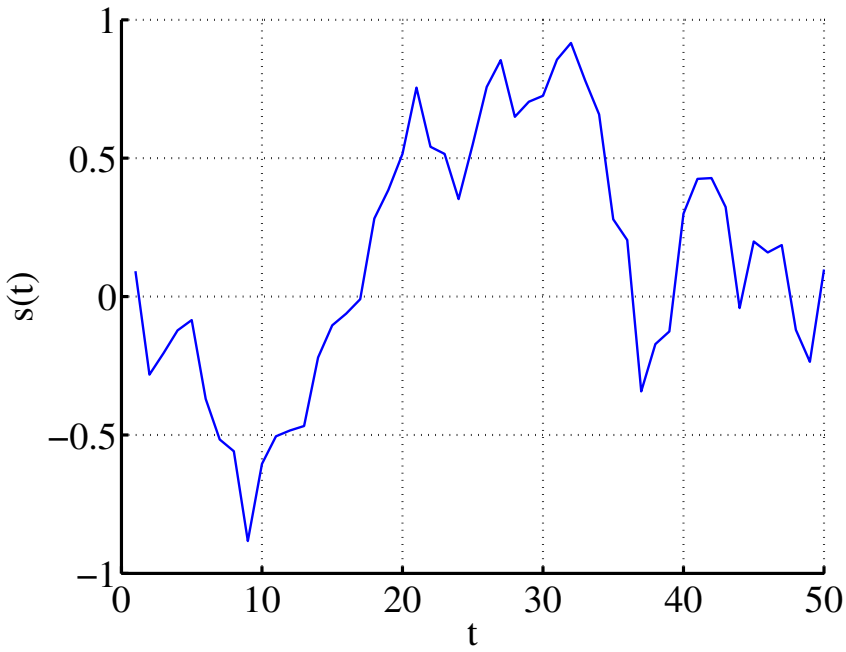
- We use the notation for the discrete signal $s[n]$ as an abbreviation for $s'(nT)$ with T being the sampling interval
- Digital values $s[n]$ are in practice numbers that are stored in a computer

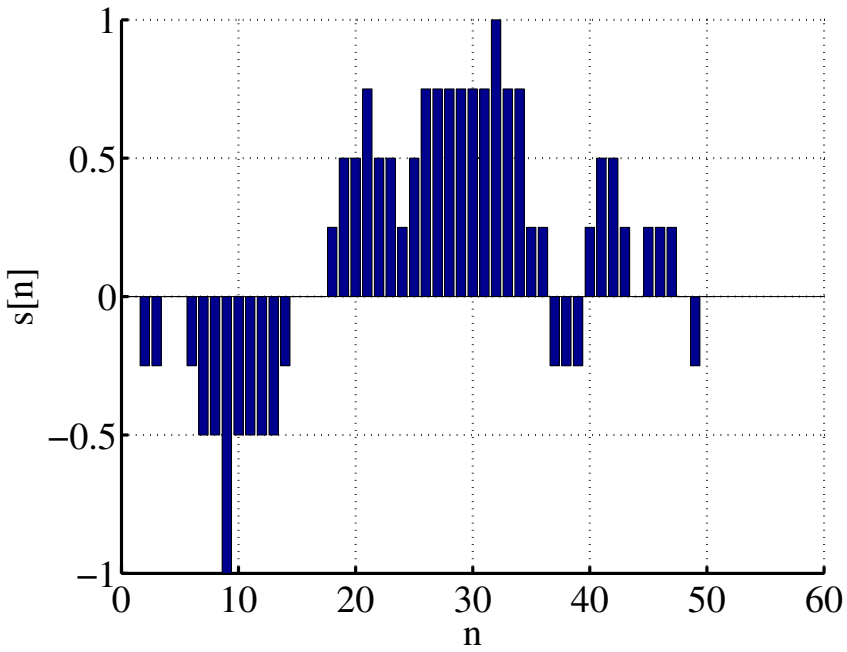
Why Analog-to-Digital Conversion?

- Required for processing data with a computer
- All compression methods discussed here are computer programs:
  - Encoder: Mapping of $s[n]$ into a bit stream b
  - Decoder: Mapping of the bit stream b into the discrete decoded signal $s'[n]$
- Although we will also discuss compression of analog signals in theory, in practice all algorithms will assume discrete versions of these analog signals that are very close approximation of these analog signals

One-Dimensional Signal Example

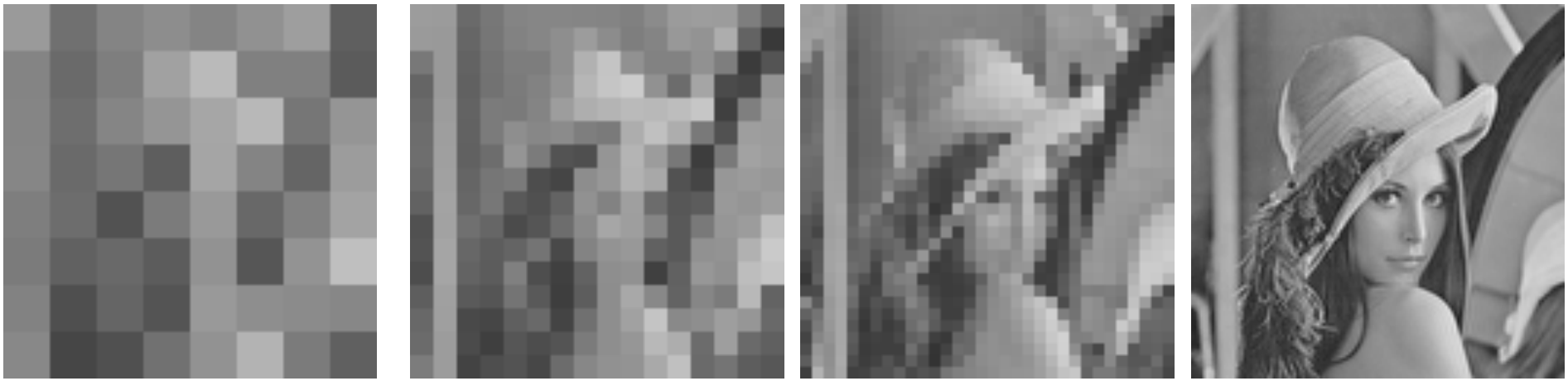
- Speech and audio signals are typically one-dimensional temporal signals
- Discrete signal below is temporally sampled and its amplitude is represented using $k = 3$ bits, i.e., $K = 8$ different values
- Note: Reconstruction value of -0.75 is not present in example, allowing us to represent this signal with $K = 8$ instead of $K = 9$ reconstruction values





Two-Dimensional Signal Example

- Pictures are two-dimensional spatial signals
- Videos are three-dimensional spatio-temporal signals
- Below sampling of picture **Lena** with different spatial sampling rates
  - 8×8, 16×16, 32×32, and 128×128 samples (from left to right)
  - Each sample is represented with $n = 8$ bits
  - Each square represents average of luminance values it covers



Two-Dimensional Signal Example

- Below quantization of picture **Lena** with different bits/sample
  - $k = 1, 2, 4,$ and 8 bits/sample (from left to right)
  - The spatial sampling rate is fixed to 128×128

Three-Dimensional Signal Examples

- Below, format, sampling rate and sampling method for different video signals yield corresponding PCM data rates

Picture format	Luma signal	Chroma signal	Sampling	Frames/s	Data rate
Common Intermediate Format (CIF)	352×288 (352×240)	$2 \times 176 \times 144$ $(2 \times 176 \times 120)$	progressive 8 bit	25 (30)	
ITU-R BT.601 Format ("Standard Television")	720×576 (720×480)	$2 \times 360 \times 576$ $(2 \times 360 \times 480)$	interlaced 8 bit	25 (30)	
ITU-R BT.709: 720p ("High Definition TV")	1280×720	$2 \times 640 \times 720$	progressive 8 bit	50 (60)	
ITU-R BT.709: 1080i ("Full HDTV")	1920×1080	$2 \times 960 \times 1080$	interlaced 8 bit	25 (30)	
ITU-R BT.2020: UHD-1 ("Ultra HDTV 4k")	3840×2160	$2 \times 1920 \times 1080$	progressive 10 bit	50 (60)	
ITU-R BT.2020: UHD-2 ("Ultra HDTV 8k")	7680×4320	$2 \times 3840 \times 2160$	progressive 12 bit	50 (60)	

Basic Communication Problem

- The **basic communication problem** may be posed as

  > **Conveying source data with highest fidelity possible within an available bit rate**

  or, equivalently, as

  > **Conveying source data using lowest bit rate possible while maintaining a specified reproduction fidelity**

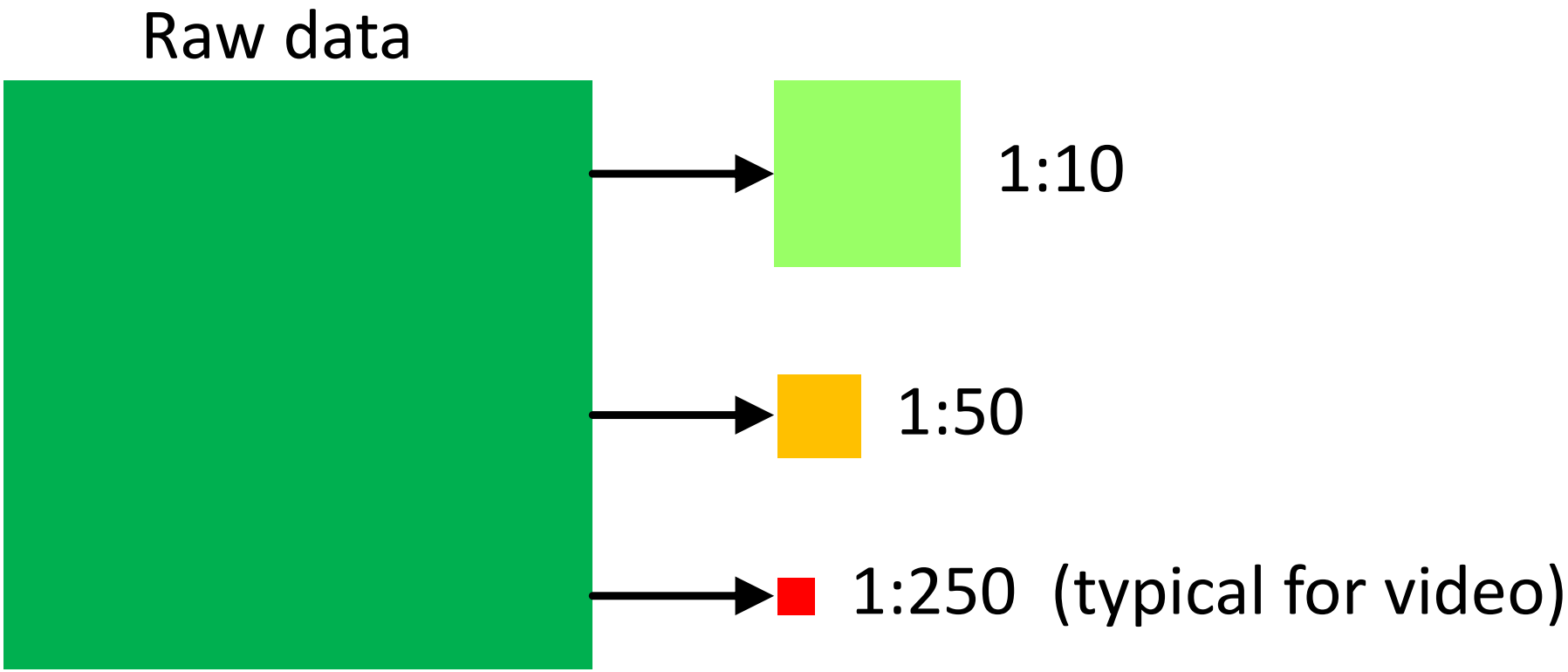
- In either case, a fundamental trade-off is made between bit rate and fidelity
- The ability of a source coding system to make this trade-off well is called its **coding efficiency** or **rate-distortion performance**, and the coding system itself is referred to as a **source codec**
- **Source codec**: a system comprising a **source co**der and a source **dec**oder

Example: JPEG (1:10 Compression)

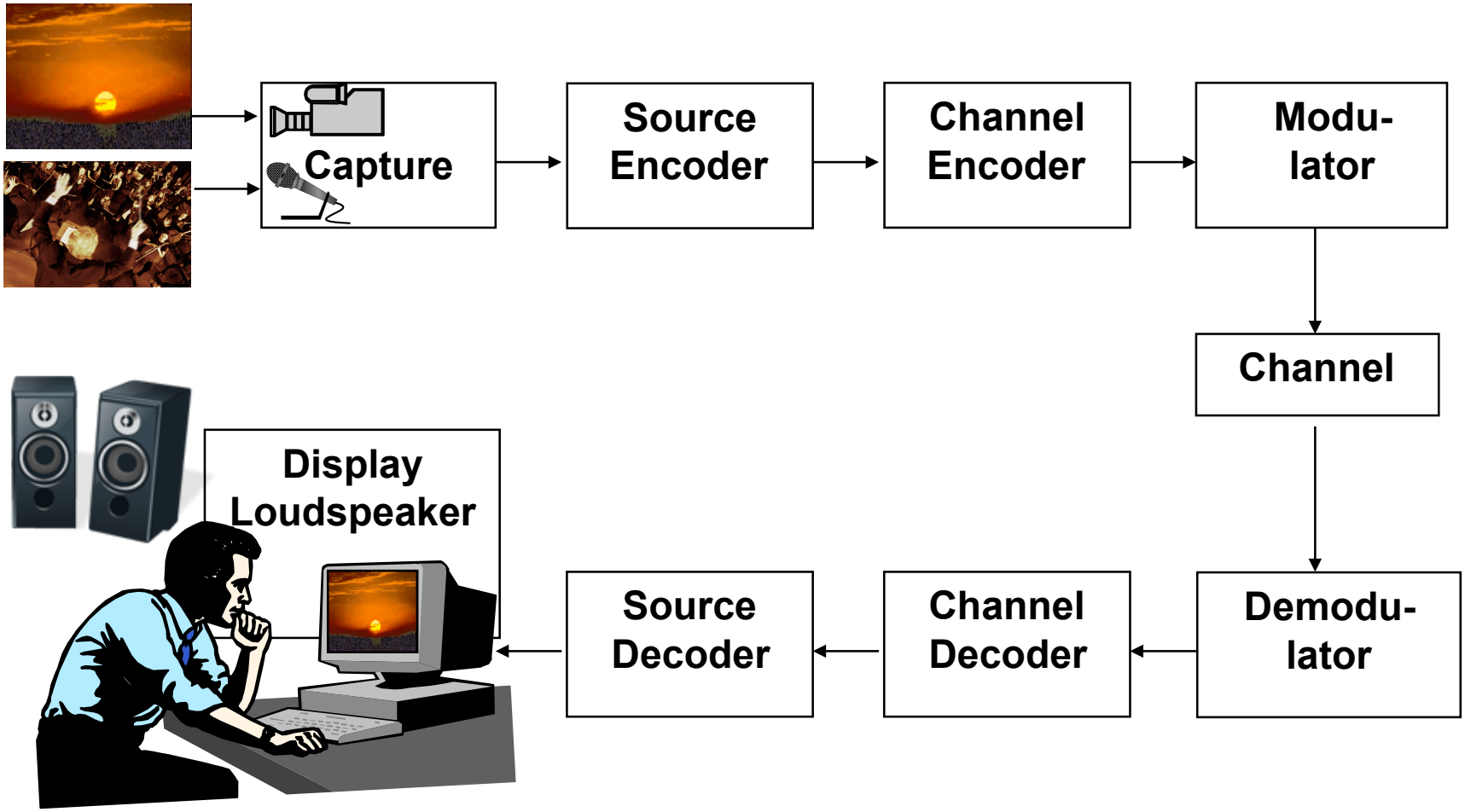
Example: JPEG (1:50 Compression)

Example: H.265/HEVC (1:50 Compression)

Geometrical Interpretation



Transmission System



Practical Communication Problem

- Source codecs are primarily characterized in terms of:
  - **Throughput of the channel**, a characteristic influenced by
    - transmission channel bit rate and
    - amount of protocol and error-correction coding overhead incurred by transmission system
  - **Distortion of the decoded signal**, which is primarily induced by
    - source encoder and
    - by channel errors introduced in path to source decoder

- The following additional constraints must also be considered
  - **Delay (start-up latency and end-to-end delay)** include
    - processing delay, buffering,
    - structural delays of source and channel codecs, and
    - speed at which data are conveyed through transmission channel
  - **Complexity (computation, memory capacity, memory access)** of
    - source codec,
    - protocol stacks, and network

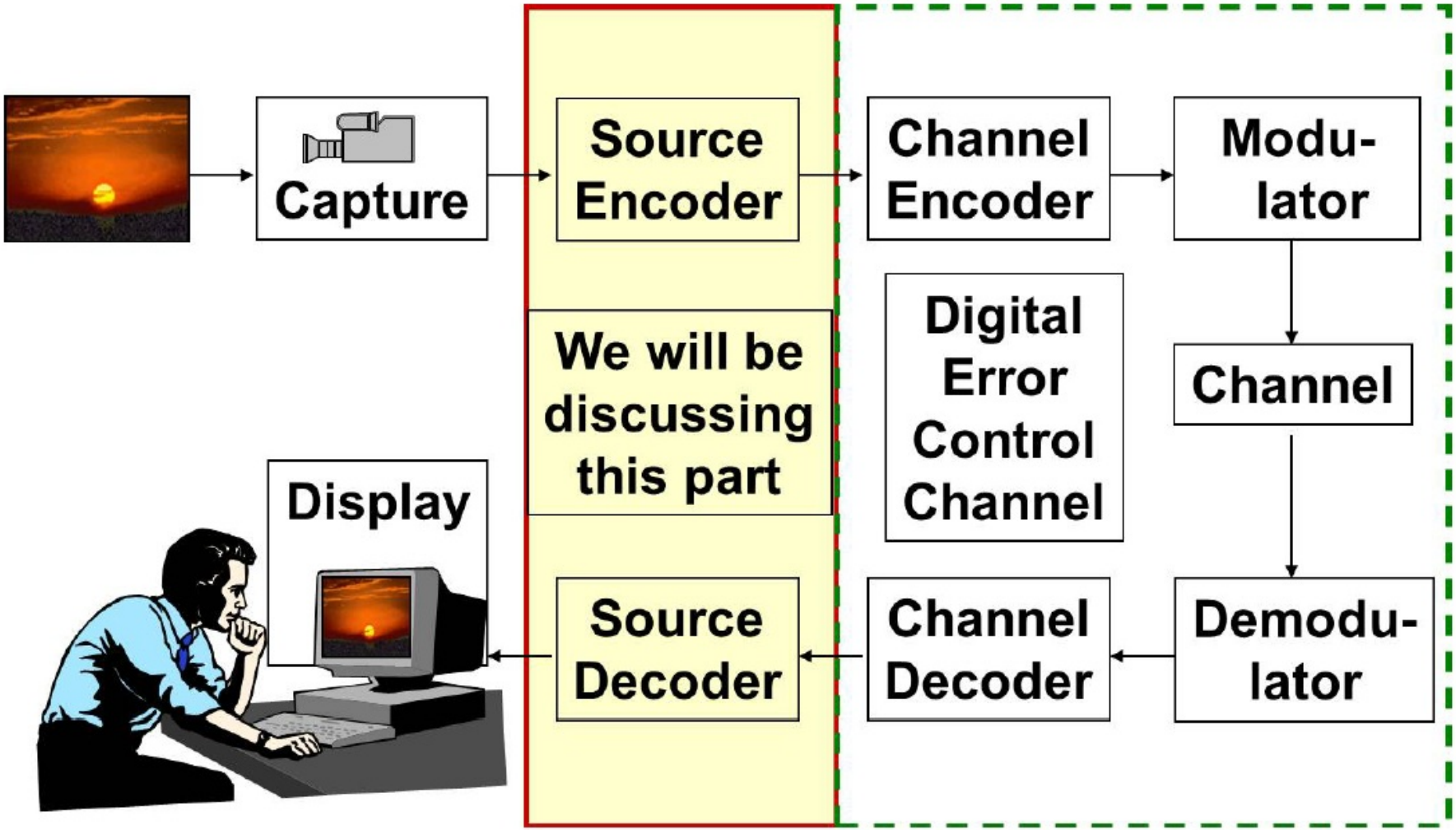
Formulation of the Practical Communication Problem

- The practical source coding design problem is posed as follows:

  Given a maximum allowed delay and a maximum allowed complexity, achieve an optimal trade-off between bit rate and distortion for the transmission problem in the targeted applications.

- Here, we will concentrate on source codec only
  - Delay is only evaluated for source codec
  - Complexity is also only assessed for the algorithm used in source codec

Scope of This Course



Transmission Channels and Optical Storage Media

- Fixed transmission lines:
  - ISDN line: 64 kbit/s
  - ADSL: 6 Mbit/s
  - VDSL: 25 Mbit/s or 50 Mbit/s
- Mobile networks:
  - GSM: 15 kbit/s
  - EDGE: 474 kbit/s (max)
  - HSDPA: 7.2 Mbit/s (peak)
  - LTE: 300 Mbit/s (peak)
- Broadcast channels
  - DVB-T: 13 Mbit/s (16QAM)
  - DVB-S: 38 Mbit/s (QPSK)
  - DVB-C: 38 Mbit/s (64QAM)
- Optical storage media
  - Compact Disc (CD): 650 MByte with 1.41 Mbit/s (12 cm)
  - Digital Versatile Dics (DVD): 4.7 GByte with 10.5 Mbit/s (DVD-5-SS-SL)
  - Blu-Ray Disc (BRD): 50 GByte with 36 Mbit/s (12 cm, DS-DL)

Types of Compression

- **Lossless coding**:
  - Uses redundancy reduction as the only principle and is therefore reversible
  - Also referred to as noiseless or invertible coding or data compaction
  - Well known use for this type of compression for data is Lempel-Ziv coding (gzip) and for picture and video signals JPEG-LS is well known
- **Lossy coding**:
  - Uses redundancy reduction and irrelevancy reduction and is therefore not reversible
  - It is the primary coding type in compression for speech, audio, picture, and video signals
  - The practically relevant bit rate reduction that is achievable through lossy compression is typically more than an order of magnitude larger than with lossless compression
  - Well known examples are for audio coding are the MPEG-1 Layer 3 (mp3), for still picture coding JPEG, and for video coding H.264/AVC

Distortion Measures

- The use of lossy compression requires the ability to measure distortion
- Often, the distortion that a human perceives in coded content is a very difficult quantity to measure, as the characteristics of human perception are complex
- Perceptual models are far more advanced for speech and audio codecs than for picture or video codecs
- In speech and audio coding,
  - Perceptual models are heavily used to guide encoding decisions
  - Listening tests are used to determine subjective quality of coding results
- In picture and video coding,
  - Perceptual models have limited use to guide encoding decisions (mainly focusing on properties of the human visual system)
  - Viewing tests are used to determine subjective quality of coding results
- This lecture: Use of objective distortion measures such as MSE and SNR

Mean Squared Error (MSE)

- **Speech and audio:** (N: duration in samples)

$$u[n] = s'[n] - s[n] \tag{4}$$

$$\mathrm{MSE} = \frac{1}{N} \sum_{n=0}^{N-1} u^2[n] \tag{5}$$

- **Pictures:** (X: picture height, Y: picture width):

$$u[x,y] = s'[x,y] - s[x,y] \tag{6}$$

$$\mathrm{MSE} = \frac{1}{X \cdot Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} u^2[x,y] \tag{7}$$

- **Videos:** (N: number of pictures, MSE_n: MSE of picture n):

$$\mathrm{MSE} = \frac{1}{N} \sum_{n=0}^{N-1} \mathrm{MSE}_n \tag{8}$$

Signal-to-Noise Ratio

- **Speech:**

$$\text{SNR} = 10 \cdot \log_{10}\left(\frac{\sigma^2}{\sigma_u^2}\right) \tag{9}$$

$$\text{with} \quad \sigma^2 = \frac{1}{N}\sum_{n=0}^{N-1}(s[n]-\mu_s)^2 \;\text{ and }\; \mu_s = \frac{1}{N}\sum_{n=0}^{N-1} s[n] \tag{10}$$

$$\sigma_u^2 = \frac{1}{N}\sum_{n=0}^{N-1}(u[n]-\mu_u)^2 \;\text{ and }\; \mu_u = \frac{1}{N}\sum_{n=0}^{N-1} u[n] \tag{11}$$

- **Pictures:** (k: bit depth of samples)

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{(2^k-1)^2}{\text{MSE}}\right) \tag{12}$$

- **Videos:** (N: number of pictures, PSNR_n: PSNR of picture n)

$$\text{PSNR} = \frac{1}{N}\sum_{n=0}^{N-1}\text{PSNR}_n \tag{13}$$

Recommended Literature

Source Coding

- T. M. Cover and J. A. Thomas, "Elements of Information Theory," John Wiley & Sons, New York, 1991.
- Gersho, A. and Gray, R. M. (1992), "Vector Quantization and Signal Compression," Kluwer Academic Publishers, Boston, Dordrecht, London.
- Jayant, N. S. and Noll, P. (1994), "Digital Coding of Waveforms," Prentice-Hall, Englewood Cliffs, NJ, USA.
- Wiegand, T. and Schwarz, H. (2010). Source Coding: Part I of Fundamentals of Source and Video Coding, Foundations and Trends in Signal Processing, vol. 4, no. 1-2. (http://iphome.hhi.de/wiegand/assets/pdfs/VBpart1.pdf)

Image and Video Coding

- W. Pennebaker and J. Mitchell, "JPEG Still Image Data Compression Standard," Van Nostrand Reinhold, New York, 1993.
- D. S. Taubman and M. W. Marcellin, "JPEG 2000 – Image Compression Fundamentals, Standards, and Practice," Kluwer Academic Publishers, 2002.
- Y. Wang, J. Ostermann, Y.-Q. Zhang, "Video Processing and Communications," Prentice-Hall, 2002.
- J.-R. Ohm, "Multimedia Communication Technology. Representation, Transmission and Identification of Multimedia Signals," Springer, Heidelberg/Berlin, 2004.

Organization

Lecture: Tuesday 10:30-12:00 & 12:15-13:45
Room 1.16

Lecturer: Dr.-Ing. Heiko Schwarz

Head, Image & Video Coding Group
Image Processing Department
Fraunhofer Heinrich Hertz Institute

heiko.schwarz@hhi.fraumhofer.de
http://iphome.hhi.de/schwarz

Course weights: Quizzes: 20%
Project: 20%
Midterm exam: 25%
Final exam: 35%

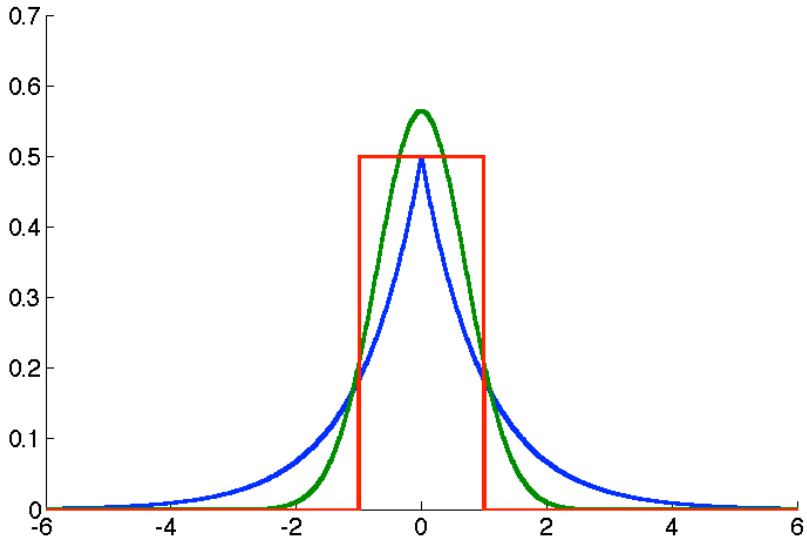
Copies of slides and solutions of exercises can be downloaded at:

http://iphome.hhi.de/schwarz/GUC-SourceCoding.htm

Part I:

Source Coding Fundamentals

Probability, Random Variables and Random Processes



Outline

Part I: Source Coding Fundamentals

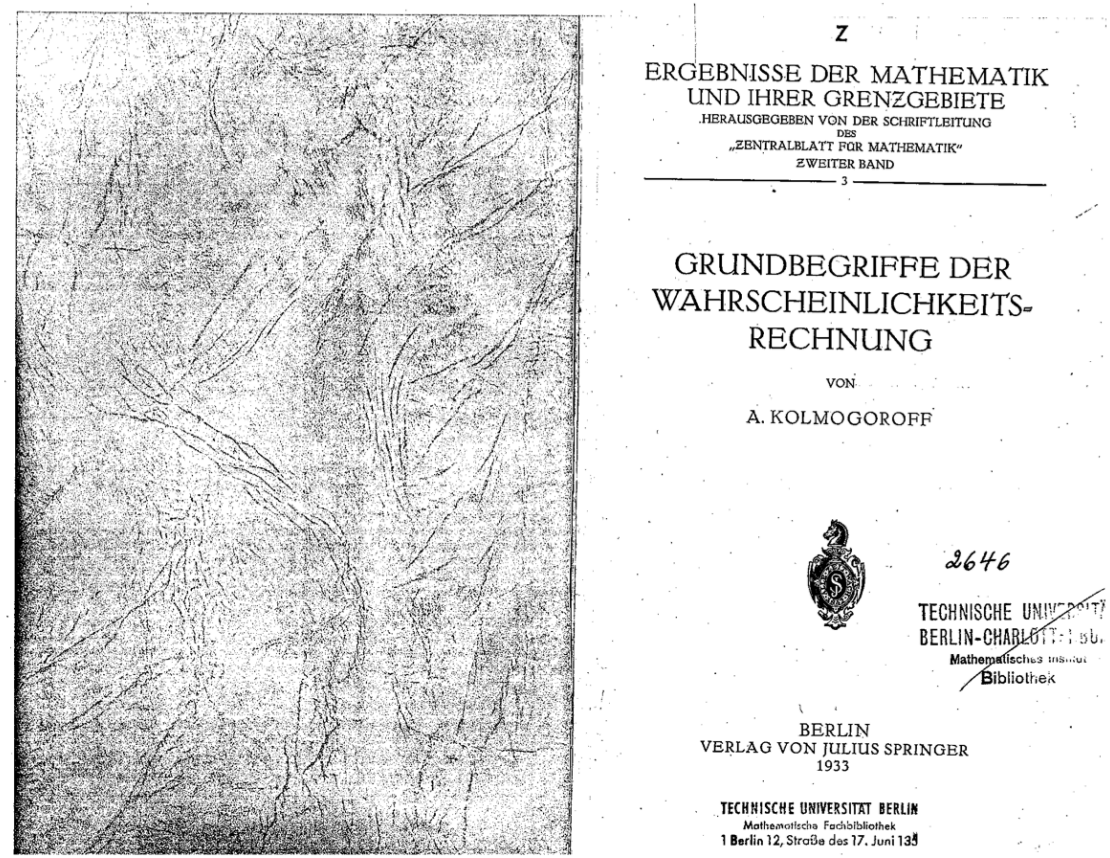
- **Review: Probability, Random Variables and Random Processes**
  - Probability
  - Random Variables
  - Random Processes
- Lossless Source Coding
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

Probability

- Probability theory:
  Branch of mathematics for description and modelling of random events
- Modern probability theory – the axiomatic definition of probability – introduced by KOLMOGOROV



Z

ERGEBNISSE DER MATHEMATIK
UND IHRER GRENZGEBIETE
HERAUSGEGEBEN VON DER SCHRIFTLEITUNG
DES
„ZENTRALBLATT FÜR MATHEMATIK"
ZWEITER BAND
3

GRUNDBEGRIFFE DER
WAHRSCHEINLICHKEITS-
RECHNUNG

VON

A. KOLMOGOROFF

BERLIN
VERLAG VON JULIUS SPRINGER
1933

Definition of Probability

- Experiment with an uncertain outcome: **Random experiment**
- Union of all possible **outcomes** ζ of the random experiment: **Certain event** or **sample space** $\mathcal{O}$ of the random experiment
- **Event**: Subset $\mathcal{A} \subseteq \mathcal{O}$
- **Probability**: Measure $P(\mathcal{A})$ assigned to $\mathcal{A}$ satisfying the following three axioms

1. Probabilities are non-negative real numbers: $P(\mathcal{A}) \geq 0, \quad \forall \mathcal{A} \subseteq \mathcal{O}$
2. Probability of the certain event: $P(\mathcal{O}) = 1$
3. If $\{\mathcal{A}_i : i = 0, 1, \cdots\}$ is a countable set of events such that $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for $i \neq j$, then

$$P\left(\bigcup_i \mathcal{A}_i\right) = \sum_i P(\mathcal{A}_i) \tag{14}$$

Independence and Conditional Probability

- Two events $\mathcal{A}_i$ and $\mathcal{A}_j$ are **independent** if

$$\boxed{P(\mathcal{A}_i \cap \mathcal{A}_j) = P(\mathcal{A}_i)\, P(\mathcal{A}_j)} \tag{15}$$

- The **conditional probability** of an event $\mathcal{A}_i$ given another event $\mathcal{A}_j$, with $P(\mathcal{A}_j) > 0$ is

$$\boxed{P(\mathcal{A}_i|\mathcal{A}_j) = \frac{P(\mathcal{A}_i \cap \mathcal{A}_j)}{P(\mathcal{A}_j)}} \tag{16}$$

- Direct consequence: BAYES' theorem

$$P(\mathcal{A}_i|\mathcal{A}_j) = P(\mathcal{A}_j|\mathcal{A}_i)\, \frac{P(\mathcal{A}_i)}{P(\mathcal{A}_j)} \quad \text{with} \quad P(\mathcal{A}_i),\, P(\mathcal{A}_j) > 0 \tag{17}$$

- Definitions (15) and (16) also imply that, if $\mathcal{A}_i$ and $\mathcal{A}_j$ are independent and $P(\mathcal{A}_j) > 0$, then

$$P(\mathcal{A}_i \,|\, \mathcal{A}_j) = P(\mathcal{A}_i) \tag{18}$$

Random Variables

- **Random variable** S:

  Function of the sample space $\mathcal{O}$ that assigns a real value $S(\zeta)$ to each outcome $\zeta \in \mathcal{O}$ of a random experiment

- Define: **Cumulative distribution function** (cdf) of a random variable S:

$$\boxed{F_S(s) = P(S \leq s) = P(\,\{\zeta\colon\ S(\zeta) \leq s\}\,)} \tag{19}$$

- Properties of cdfs:
  - $F_S(s)$ is non-decreasing
  - $F_S(-\infty) = 0$
  - $F_S(\infty) = 1$

Joint Cumulative Distribution Function

- **Joint cdf** or **joint distribution** of two random variables X and Y

$$F_{XY}(x, y) = P(X \leq x, Y \leq y) \tag{20}$$

- N-dimensional random vector $\boldsymbol{S} = (S_0, \cdots, S_{N-1})^{\mathrm{T}}$:

  Vector of random variables S_0, S_1, $\cdots$, S_{N-1}

- **N-dimensional cdf**, **joint cdf**, or **joint distribution**:

$$\boxed{F_{\boldsymbol{S}}(\boldsymbol{s}) = P(\boldsymbol{S} \leq \boldsymbol{s}) = P(S_0 \leq s_0, \cdots, S_{N-1} \leq s_{N-1})} \tag{21}$$

  with $\boldsymbol{S} = (S_0, \cdots, S_{N-1})^{\mathrm{T}}$ being a random vector

- Joint cdf of two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$

$$F_{\boldsymbol{XY}}(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{X} \leq \boldsymbol{x}, \boldsymbol{Y} \leq \boldsymbol{y}) \tag{22}$$

Conditional Cumulative Distribution Function

- **Conditional cdf** of random variable S given event $\mathcal{B}$ with $P(\mathcal{B}) > 0$

$$F_{S|\mathcal{B}}(s \,|\, \mathcal{B}) = P(S \leq s \,|\, \mathcal{B}) = \frac{P(\{S \leq s\} \cap \mathcal{B})}{P(\mathcal{B})} \tag{23}$$

- Conditional cdf of a random variable X given another random variable Y

$$F_{X|Y}(x|y) = \frac{F_{XY}(x,y)}{F_Y(y)} = \frac{P(X \leq x, Y \leq y)}{P(Y \leq y)} \tag{24}$$

- Conditional cdf of a random vector $\boldsymbol{X}$ given another random vector $\boldsymbol{Y}$

$$F_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{F_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})}{F_{\boldsymbol{Y}}(\boldsymbol{y})} \tag{25}$$

Continuous Random Variables

- A random variables S is called a **continuous random variable**, if and only if its cdf $F_S(s)$ is a continuous function

- Define: **Probability density function** (pdf) for continuous random variables

$$\boxed{f_S(s) = \frac{\mathrm{d}F_S(s)}{\mathrm{d}s} \quad \Longleftrightarrow \quad F_S(s) = \int_{-\infty}^{s} f_S(t)\,\mathrm{d}t} \tag{26}$$

- Properties of pdfs:
  - $f_S(s) \geq 0, \forall s$
  - $\int_{-\infty}^{\infty} f_S(t)\,\mathrm{d}t = 1$

Examples for Pdfs

- **Uniform pdf**:

$$f_S(s) = \begin{cases} 1/A & : \quad -A/2 \leq s \leq A/2 \\ 0 & : \quad \text{otherwise} \end{cases}, \qquad A > 0 \tag{27}$$

- **Laplacian pdf**:

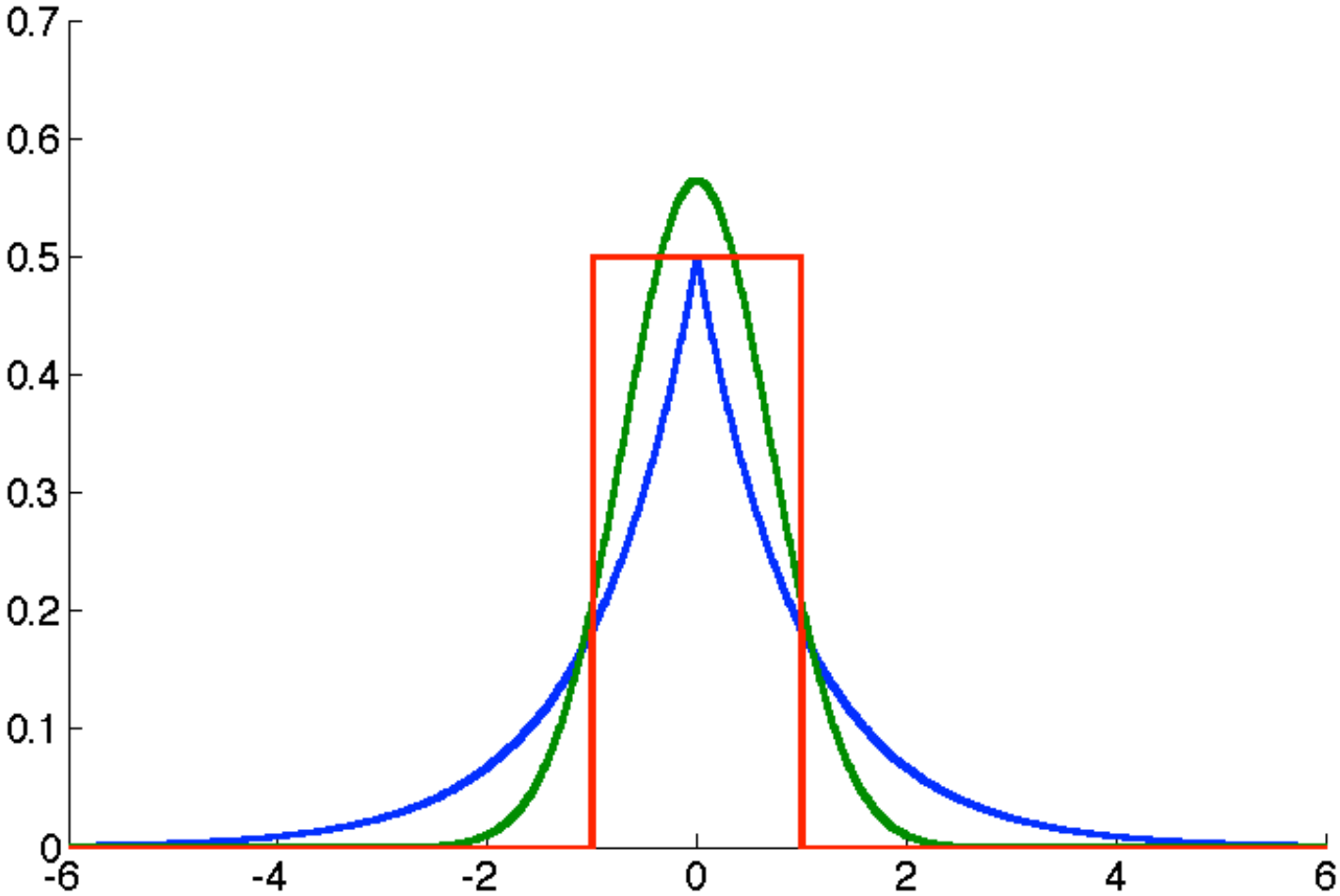
$$f_S(s) = \frac{1}{\sigma_S \sqrt{2}} \, e^{-|s-\mu_S|\sqrt{2}/\sigma_S}, \qquad \sigma_S > 0 \tag{28}$$

- **Gaussian pdf**:

$$f_S(s) = \frac{1}{\sigma_S \sqrt{2\pi}} \, e^{-(s-\mu_S)^2/(2\sigma_S^2)}, \qquad \sigma_S > 0 \tag{29}$$

Generalized Gaussian Distribution

$$f_S(s) = \frac{\beta}{2\,\alpha\,\Gamma(1/\beta)} \cdot e^{-(|x-\mu|/\alpha)^{\beta}} \qquad \Gamma(z) = \int_0^{\infty} e^{-t}\, t^{z-1}\, \mathrm{d}t \qquad (30)$$



Joint and Conditional Pdfs

- **N-dimensional pdf**, **joint pdf**, or **joint density**

$$\boxed{f_{\boldsymbol{S}}(\boldsymbol{s}) = \frac{\partial^N F_{\boldsymbol{S}}(\boldsymbol{s})}{\partial s_0 \cdots \partial s_{N-1}}} \tag{31}$$

- **Conditional pdf** or **conditional density** $f_{S|\mathcal{B}}(s|\mathcal{B})$ of a random variable S given an event $\mathcal{B}$

$$f_{S|\mathcal{B}}(s|\mathcal{B}) = \frac{\mathrm{d}\, F_{S|\mathcal{B}}(s|\mathcal{B})}{\mathrm{d}\, s} \tag{32}$$

- Conditional density of a random variable X given another random variable Y

$$f_{X|Y}(x|y) = \frac{f_{XY}(x,y)}{f_Y(y)} \tag{33}$$

- Conditional density of a random vector $\boldsymbol{X}$ given another random vector $\boldsymbol{Y}$

$$f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})}{f_{\boldsymbol{Y}}(\boldsymbol{y})} \tag{34}$$

Discrete Random Variables

- A random variable S is called a **discrete random variable**, if and only if its cdf $F_S(s)$ represents a staircase function
- Discrete random variable S takes values of countable set $\mathcal{A} = \{a_0, a_1, \ldots\}$
- Define: **Probability mass function** (pmf) for discrete random variables:

$$\boxed{p_S(a) = P(S = a) = P(\,\{\zeta\colon\ S(\zeta) = a\}\,)} \tag{35}$$

- Cdf of discrete random variable

$$\boxed{F_S(s) = \sum_{a \leq s} p(a)} \tag{36}$$

- Pdf can be constructed using the Dirac delta function δ

$$f_S(s) = \sum_{a \in \mathcal{A}} \delta(s - a)\, p_S(a) \tag{37}$$

Examples for Pmfs

- **Binary pmf**:

$$\mathcal{A} = \{a_0, a_1\} \qquad p_S(a_0) = p, \qquad p_S(a_1) = 1 - p \tag{38}$$

- **Uniform pmf**:

$$\mathcal{A} = \{a_0, a_1, \cdots, a_{M-1}\} \qquad p_S(a_i) = 1/M \qquad \forall\, a_i \in \mathcal{A} \tag{39}$$

- **Geometric pmf**:

$$\mathcal{A} = \{a_0, a_1, \cdots\} \qquad p_S(a_i) = (1-p)\, p^i \qquad \forall\, a_i \in \mathcal{A} \tag{40}$$

Joint and Conditional Pmfs

- **N-dimensional pmf** or **joint pmf** for a random vector $\boldsymbol{S} = (S_0, \cdots, S_{N-1})^{\mathrm{T}}$

$$\boxed{p_{\boldsymbol{S}}(\boldsymbol{a}) = P(\boldsymbol{S} = \boldsymbol{a}) = P(S_0 = a_0, \cdots, S_{N-1} = a_{N-1})} \tag{41}$$

- Joint pmf of two random vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$: $p_{\boldsymbol{XY}}(\boldsymbol{a_x}, \boldsymbol{a_y})$
- **Conditional pmf** $p_{S|\mathcal{B}}(a \,|\, \mathcal{B})$ of a random variable S given an event $\mathcal{B}$, with $P(\mathcal{B}) > 0$

$$p_{S|\mathcal{B}}(a \,|\, \mathcal{B}) = P(S = a \,|\, \mathcal{B}) \tag{42}$$

- Conditional pmf of a random variable X given another random variable Y

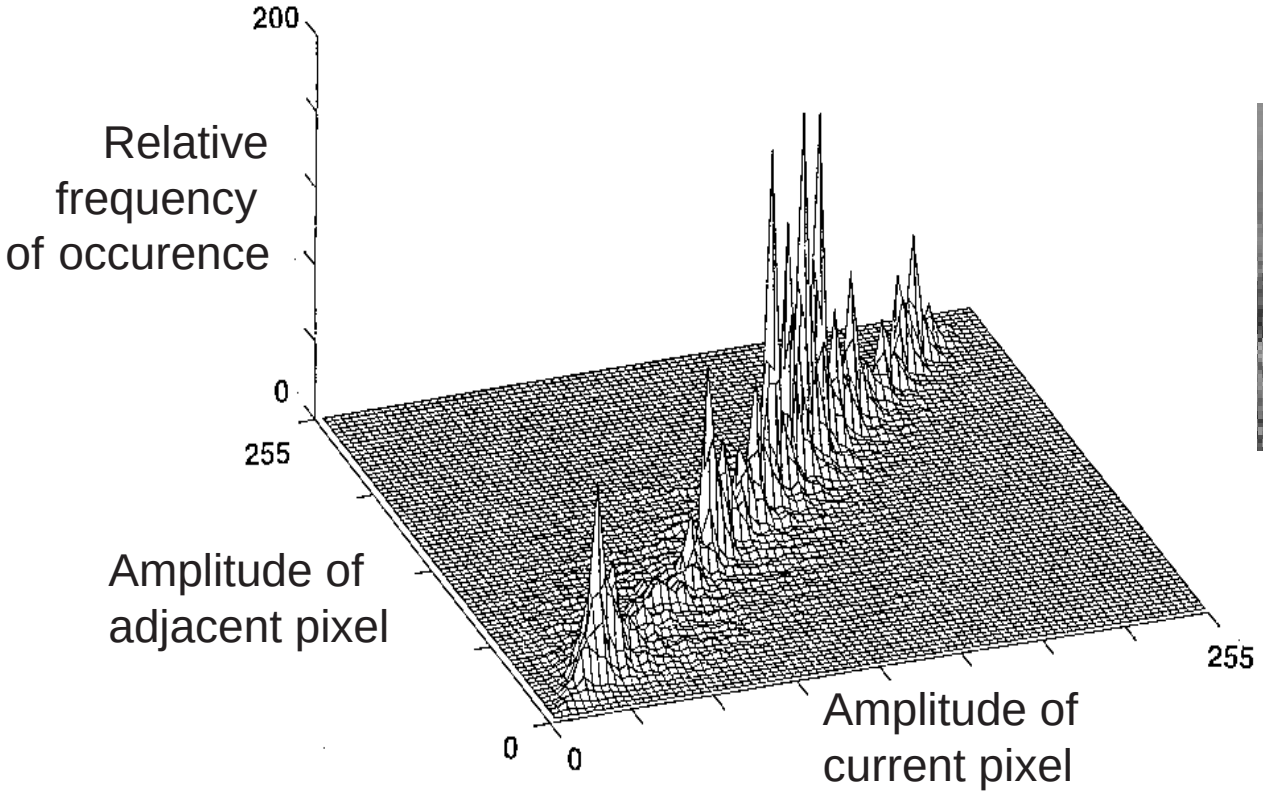
$$p_{X|Y}(a_x|a_y) = \frac{p_{XY}(a_x, a_y)}{p_Y(a_y)} \tag{43}$$

- Conditional pmf of a random vector $\boldsymbol{X}$ given another random vector $\boldsymbol{Y}$

$$p_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{a_x}|\boldsymbol{a_y}) = \frac{p_{\boldsymbol{XY}}(\boldsymbol{a_x}, \boldsymbol{a_y})}{p_{\boldsymbol{Y}}(\boldsymbol{a_y})} \tag{44}$$

Example for a Joint Pmf

- For example, samples in picture and video signals typically show strong statistical dependencies
- Below: Histogram of two horizontally adjacent sampels for the picture 'Lena'



Expectation

- **Expectation value** or **expected value** of a continuous random variable S

$$\boxed{E\{g(S)\} = \int_{-\infty}^{\infty} g(s)\, f_S(s)\, \mathrm{d}s} \tag{45}$$

of a discrete random variable S

$$\boxed{E\{g(S)\} = \sum_{a \in \mathcal{A}} g(a)\, p_S(a)} \tag{46}$$

- Important expectation values are **mean** μ_S and **variance** σ_S^2

$$\boxed{\mu_S = E\{S\}} \quad \text{and} \quad \boxed{\sigma_S^2 = E\{(S - \mu_s)^2\}} \tag{47}$$

- Expectation value of a function $g(\boldsymbol{S})$ of a set of N random variables $\boldsymbol{S} = \{S_0, \cdots, S_{N-1}\}$

$$\boxed{E\{g(\boldsymbol{S})\} = \int_{\mathcal{R}^N} g(\boldsymbol{s})\, f_{\boldsymbol{S}}(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s}} \tag{48}$$

Conditional Expectation

- **Conditional expectation value** of function $g(S)$ given an event $\mathcal{B}$, with $P(\mathcal{B}) > 0$

$$E\{g(S) \,|\, \mathcal{B}\} = \int_{-\infty}^{\infty} g(s)\, f_{S|\mathcal{B}}(s \,|\, \mathcal{B})\, \mathrm{d}s \tag{49}$$

- Conditional expectation value of function $g(X)$ given a particular value y for another random variable Y

$$E\{g(X) \,|\, y\} = E\{g(X) \,|\, Y = y\} = \int_{-\infty}^{\infty} g(x)\, f_{X|Y}(x, y)\, \mathrm{d}x \tag{50}$$

- Note: $E\{g(X) \,|\, y\}$ is a deterministic function of y

- Conditional expectation value of function $g(X)$ given a random variable Y,

$$E\{g(X) \,|\, Y\} = \int_{-\infty}^{\infty} g(x)\, f_{X|Y}(x, Y)\, \mathrm{d}x, \tag{51}$$

  is another random variable

Iterative Expectation Rule

- Expectation value $E\{Z\}$ of a random variable $Z = E\{g(X)|Y\}$

$$\begin{aligned} E\{E\{g(X)|Y\}\} &= \int_{-\infty}^{\infty} E\{g(X)|y\}\; f_Y(y)\,\mathrm{d}y \\ &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} g(x)\, f_{X|Y}(x,y)\,\mathrm{d}x \right) f_Y(y)\,\mathrm{d}y \\ &= \int_{-\infty}^{\infty} g(x) \left(\int_{-\infty}^{\infty} f_{X|Y}(x,y)\, f_Y(y)\,\mathrm{d}y \right) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} g(x)\, f_X(x)\,\mathrm{d}x \\ &= E\{g(X)\} \end{aligned} \tag{52}$$

$\Longrightarrow$ $E\{E\{g(X)|Y\}\}$ does not depend on the statistical properties of the random variable Y, but only on those of X

Random Processes

- Series of random experiments at time instants t_n, with $n = 0, 1, 2, \ldots$
- Outcome of experiment: Random variable $S_n = S(t_n)$
- **Discrete-time random process**: Series of random variables $\boldsymbol{S} = \{S_n\}$
- Statistical properties of discrete-time random process $\boldsymbol{S}$: N-th order joint cdf

$$F_{\boldsymbol{S}_k}(\boldsymbol{s}) = P(\boldsymbol{S}_k^{(N)} \leq \boldsymbol{s}) = P(S_k \leq s_0, \cdots, S_{k+N-1} \leq s_{N-1}) \tag{53}$$

- **Continuous random process**

$$f_{\boldsymbol{S}_k}(\boldsymbol{s}) = \frac{\partial^N}{\partial s_0 \cdots \partial s_{N-1}} F_{\boldsymbol{S}_k}(\boldsymbol{s}) \tag{54}$$

- **Discrete random process**

$$F_{\boldsymbol{S}_k}(\boldsymbol{s}) = \sum_{\boldsymbol{a} \in \mathcal{A}^N} p_{\boldsymbol{S}_k}(\boldsymbol{a}) \tag{55}$$

$\mathcal{A}^N$ product space of the alphabets $\mathcal{A}_n$ and

$$p_{\boldsymbol{S}_k}(\boldsymbol{a}) = P(S_k = a_0, \cdots, S_{k+N-1} = a_{N-1}) \tag{56}$$

Autocovariance and Autocorrelation Matrix

- N-th order **autocovariance matrix**

$$\boxed{\boldsymbol{C}_N(t_k) = E\left\{ \left(\boldsymbol{S}_k^{(N)} - \boldsymbol{\mu}_N(t_k) \right) \left(\boldsymbol{S}_k^{(N)} - \boldsymbol{\mu}_N(t_k) \right)^{\mathrm{T}} \right\}} \tag{57}$$

- N-th order **autocorrelation matrix**

$$\boxed{\boldsymbol{R}_N(t_k) = E\left\{ \left(\boldsymbol{S}_k^{(N)} \right) \left(\boldsymbol{S}_k^{(N)} \right)^{\mathrm{T}} \right\}} \tag{58}$$

- Note the following relationship

$$\begin{aligned} \boldsymbol{C}_N(t_k) &= E\left\{ \left(\boldsymbol{S}_k^{(N)} - \boldsymbol{\mu}_N(t_k) \right) \left(\boldsymbol{S}_k^{(N)} - \boldsymbol{\mu}_N(t_k) \right)^{\mathrm{T}} \right\} \\ &= E\left\{ \left(\boldsymbol{S}_k^{(N)} \right) \left(\boldsymbol{S}_k^{(N)} \right)^{\mathrm{T}} \right\} - E\left\{ \boldsymbol{S}_k^{(N)} \right\} \boldsymbol{\mu}_N(t_k)^{\mathrm{T}} \\ &\quad - \boldsymbol{\mu}_N(t_k)\, E\left\{ \boldsymbol{S}_k^{(N)} \right\}^{\mathrm{T}} + \boldsymbol{\mu}_N(t_k)\, \boldsymbol{\mu}_N(t_k)^{\mathrm{T}} \\ &= \boldsymbol{R}_N(t_k) - \boldsymbol{\mu}_N(t_k)\, \boldsymbol{\mu}_N(t_k)^{\mathrm{T}} \end{aligned} \tag{59}$$

Stationary Random Process

- **Stationary random process**:
  Statistical properties are invariant to a shift in time

  $\Longrightarrow$ $F_{\boldsymbol{S}_k}(\boldsymbol{s})$, $f_{\boldsymbol{S}_k}(\boldsymbol{s})$ and $p_{\boldsymbol{S}_k}(\boldsymbol{a})$ are independent of t_k
  and are denoted by $F_{\boldsymbol{S}}(\boldsymbol{s})$, $f_{\boldsymbol{S}}(\boldsymbol{s})$ and $p_{\boldsymbol{S}}(\boldsymbol{a})$, respectively

  $\Longrightarrow$ $\boldsymbol{\mu}_N(t_k)$, $\boldsymbol{C}_N(t_k)$ and $\boldsymbol{R}_N(t_k)$ are independent of t_k
  and are denoted by $\boldsymbol{\mu}_{\boldsymbol{N}}$, $\boldsymbol{C}_N$ and $\boldsymbol{R}_N$, respectively

- N-th order autocovariance matrix

$$\boldsymbol{C}_N = E\Big\{(\boldsymbol{S}^{(N)} - \boldsymbol{\mu}_N)(\boldsymbol{S}^{(N)} - \boldsymbol{\mu}_N)^{\mathrm{T}}\Big\} \tag{60}$$

  is a symmetric **Toeplitz matrix**

$$\boldsymbol{C}_N = \sigma_S^2 \begin{pmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{N-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{N-2} \\ \rho_2 & \rho_1 & 1 & \cdots & \rho_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{N-1} & \rho_{N-2} & \rho_{N-3} & \cdots & 1 \end{pmatrix} \tag{61}$$

  with

$$\rho_k = \frac{1}{\sigma_S^2} E\Big\{(S_\ell - \mu_S)\,(S_{\ell+k} - \mu_S)\Big\} \tag{62}$$

Memoryless and IID Random Processes

- **Memoryless random process**:
  Random process $\boldsymbol{S} = \{S_n\}$ for which the random variables S_n are independent
- **Independent and identical distributed** (iid) random process:
  Stationary and memoryless random process
- N-th order cdf $F_{\boldsymbol{S}}(\boldsymbol{s})$, pdf $f_{\boldsymbol{S}}(\boldsymbol{s})$, and pmf $p_{\boldsymbol{S}}(\boldsymbol{a})$ for iid processes, with $\boldsymbol{s} = (s_0, \cdots, s_{N-1})^T$ and $\boldsymbol{a} = (a_0, \cdots, a_{N-1})^T$

$$F_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{k=0}^{N-1} F_S(s_k) \tag{63}$$

$$f_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{k=0}^{N-1} f_S(s_k) \tag{64}$$

$$p_{\boldsymbol{S}}(\boldsymbol{a}) = \prod_{k=0}^{N-1} p_S(a_k) \tag{65}$$

$F_S(s)$, $f_S(s)$, and $p_S(a)$ are the marginal cdf, pdf, and pmf, respectively

Markov Processes

- **Markov process**: Future outcomes do not depend on past outcomes, but only on the present outcome,

$$P(S_n \leq s_n \mid S_{n-1} = s_{n-1}, \cdots) = P(S_n \leq s_n \mid S_{n-1} = s_{n-1}) \tag{66}$$

- Discrete Markov processes

$$p_{S_n}(a_n \mid a_{n-1}, \cdots) = p_{S_n}(a_n \mid a_{n-1}) \tag{67}$$

- Example for a discrete Markov process

a	a_0	a_1	a_2
$p(a\|a_0)$	0.90	0.05	0.05
$p(a\|a_1)$	0.15	0.80	0.05
$p(a\|a_2)$	0.25	0.15	0.60
$p(a)$			

Continuous Markov Processes

- Continuous Markov processes

$$f_{S_n}(s_n \mid s_{n-1}, \cdots) = f_{S_n}(s_n \mid s_{n-1}) \tag{68}$$

- Construction of continuous stationary Markov process $\boldsymbol{S} = \{S_n\}$ with mean μ_S, given a zero-mean iid process $\boldsymbol{Z} = \{Z_n\}$

$$S_n = Z_n + \rho\,(S_{n-1} - \mu_S) + \mu_S, \quad \text{with} \quad |\rho| < 1 \tag{69}$$

$\Longrightarrow$ Variance σ_S^2 of stationary Markov process $\boldsymbol{S}$

$$\sigma_S^2 = E\big\{(S_n - \mu_S)^2\big\} = E\big\{\big(Z_n + \rho\,(S_{n-1} - \mu_S)\,\big)^2\big\} = \frac{\sigma_Z^2}{1-\rho^2} \tag{70}$$

$\Longrightarrow$ Autocovariance function of stationary Markov process $\boldsymbol{S}$

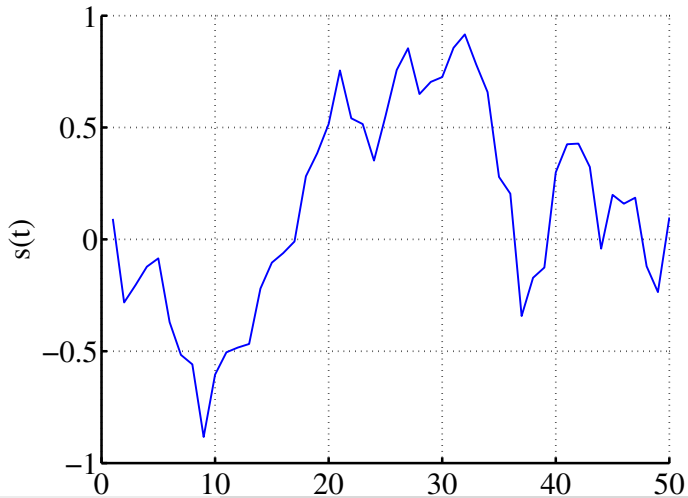
$$\phi_{k,\ell} = \phi_{|k-\ell|} = E\{(S_k - \mu_S)(S_\ell - \mu_S)\} = \sigma_S^2\,\rho^{|k-\ell|} \tag{71}$$

Gaussian Processes

- **Gaussian process**: Continuous process $\boldsymbol{S} = \{S_n\}$ with the property that all finite collections of random variables S_n represent Gaussian random vectors
- N-th order pdf of stationary Gaussian process with N-th order autocorrelation matrix $\boldsymbol{C}_N$ and mean μ_S

$$f_{\boldsymbol{S}}(\boldsymbol{s}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{C}_N|}} e^{-\frac{1}{2}(\boldsymbol{s}-\boldsymbol{\mu}_S)^{\mathrm{T}} \boldsymbol{C}_N^{-1} (\boldsymbol{s}-\boldsymbol{\mu}_S)} \quad \text{with} \quad \boldsymbol{\mu}_S = \begin{bmatrix} \mu_S \\ \vdots \\ \mu_S \end{bmatrix} \tag{72}$$

- **Stationary Gauss-Markov process**:
  Stationary process that is a Gaussian process and a Markov process
  - IID process $\boldsymbol{Z} = \{Z_n\}$ in (69) has a Gaussian pdf
  - Statistical properties are completely determined by
    - mean μ_S
    - variance σ_S^2
    - correlation factor ρ



Chapter Summary

Random variables

- Discrete and continuous random variables
- Cumulative distribution function (cdf)
- Probability density function (pdf)
- Probability mass function (pmf)
- Joint and conditional cdfs, pdfs, pmfs
- Expectation values and conditional expectation values

Random processes

- Stationary processes
- Memoryless processes
- IID processes
- Markov processes
- Gaussian processes
- Gauss-Markov processes

Exercise 1

Given is a stationary discrete Markov process with the alphabet $\mathcal{A} = \{a_0, a_1, a_2\}$ and the conditional pmfs listed in the table below

a	a_0	a_1	a_2
$p(a\|a_0)$	0.90	0.05	0.05
$p(a\|a_1)$	0.15	0.80	0.05
$p(a\|a_2)$	0.25	0.15	0.60
$p(a)$			

Determine the marginal pmf $p(a)$.

Exercise 2

Investigate the relationship between independence and correlation.

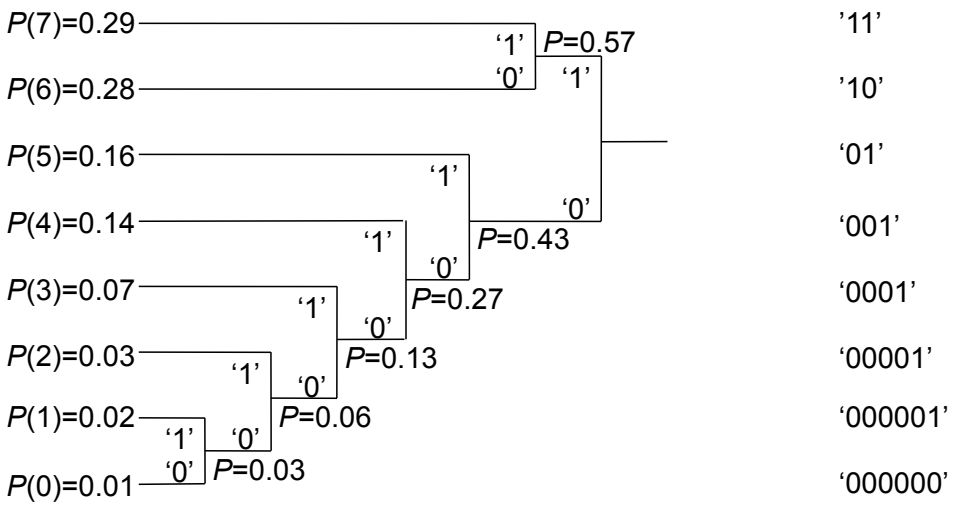
(a) Two random variables X and Y are said to be *correlated* if and only if their covariance C_{XY} is not equal to 0.
Can two independent random variables X and Y be correlated?

(b) Let X be a continuous random variable with a variance $\sigma_X^2 > 0$ and a pdf $f_X(x)$. The pdf shall be non-zero for all real numbers, $f_X(x) > 0$, $\forall x \in \mathbb{R}$. Furthermore, the pdf $f_X(x)$ shall be symmetric around zero, $f_X(x) = f_X(-x)$, $\forall x \in \mathbb{R}$. Let Y be a random variable given by $Y = a\,X^2 + b\,X + c$ with $a, b, c \in \mathbb{R}$.
For which values of a, b, and c are X and Y uncorrelated?
For which values of a, b, and c are X and Y independent?

(c) Which of the following statements for two random variables X and Y are true?

- If X and Y are uncorrelated, they are also independent.
- If X and Y are independent, $E\{XY\} = 0$.
- If X and Y are correlated, they are also dependent.

Lossless Coding



Outline

Part I: Source Coding Fundamentals

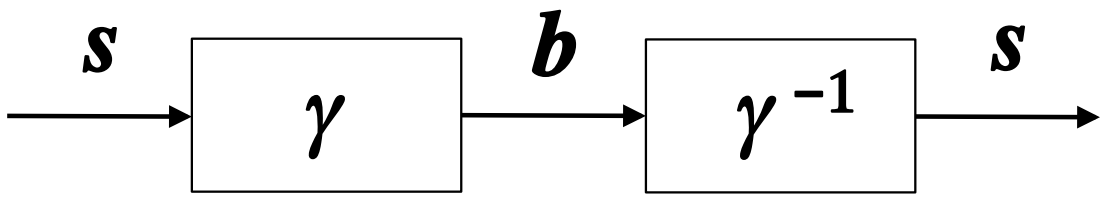
- Probability, Random Variables and Random Processes
- **Lossless Source Coding**
  - Introduction
  - Variable-Length Coding for Scalars
  - Variable-Length Coding for Vectors
  - Elias and Arithmetic Coding
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

Lossless Source Coding – Overview

- **Reversible mapping** of sequence of discrete source symbols into sequences of codewords
- Other names:
  - Noiseless coding
  - Entropy coding
- Original source sequence can be exactly reconstructed (Note: Not the case in lossy coding)
- Bit rate reduction possible, if and only if source data have statistical properties that are exploitable for data compression



Lossless Source Coding – Terminology

- Message $\boldsymbol{s}^{(L)} = \{s_0, \cdots, s_{L-1}\}$ drawn from stochastic process $\boldsymbol{S} = \{S_n\}$
- Sequence $\boldsymbol{b}^{(K)} = \{b_0, \cdots, b_{K-1}\}$ of K bits ($b_k \in \mathcal{B} = \{0, 1\}$)
- Process of lossless coding: Message $\boldsymbol{s}^{(L)}$ is converted to $\boldsymbol{b}^{(K)}$
- Assume:
  - Subsequence $\boldsymbol{s}^{(N)} = \{s_n, \cdots, s_{n+N-1}\}$ with $1 \leq N \leq L$ and
  - Bits $\boldsymbol{b}^{(\ell)}(\boldsymbol{s}^{(N)}) = \{b_0, \cdots, b_{\ell-1}\}$ assigned to it

- **Lossless source code**
  - Encoder mapping:

$$\boldsymbol{b}^{(\ell)} = \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \tag{73}$$

  - Decoder mapping:

$$\boldsymbol{s}^{(N)} = \gamma^{-1}\big(\, \boldsymbol{b}^{(\ell)} \,\big) = \gamma^{-1}\big(\, \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \,\big) \tag{74}$$

Classification of Lossless Source Codes

- **Lossless source code**
  - Encoder mapping:

$$\boldsymbol{b}^{(\ell)} = \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \tag{75}$$

  - Decoder mapping:

$$\boldsymbol{s}^{(N)} = \gamma^{-1}\big(\, \boldsymbol{b}^{(\ell)} \,\big) = \gamma^{-1}\big(\, \gamma\big(\, \boldsymbol{s}^{(N)} \,\big) \,\big) \tag{76}$$

- **Fixed-to-fixed mapping**: N and ℓ are both fixed
  - Will be discussed as special case of fixed-to-variable
- **Fixed-to-variable mapping**: N fixed and ℓ variable
  - Huffman algorithm for scalars and vectors (discussed in lecture)
- **Variable-to-fixed mapping**: N variable and ℓ fixed
  - Tunstall codes (not discussed in lecture)
- **Variable-to-variable mapping**: ℓ and N are both variable
  - Elias and arithmetic codes (discussed in lecture)

Variable-Length Coding for Scalars

- Assign a separate codeword to each scalar symbol s_n of a message $\boldsymbol{s}^{(L)}$
- Assume:
  Message $\boldsymbol{s}^{(L)}$ generated by stationary discrete random process $\boldsymbol{S} = \{S_n\}$
- Random variables $S_n = S$ with symbol alphabet $\mathcal{A} = \{a_0, \cdots, a_{M-1}\}$ and marginal pmf $p(a) = P(S = a)$
- Lossless source code:
  Assign to each a_i a binary codeword $\boldsymbol{b}_i = \{b_0^i, \cdots, b_{\ell(a_i)-1}^i\}$, length $\ell(a_i) \geq 1$
- Example:
  - Alphabet $\mathcal{A} = \{x, y, z\}$
  - Encoder mapping $\gamma(a) = \begin{cases} 0 & : \quad a = x \\ 10 & : \quad a = y \\ 11 & : \quad a = z \end{cases}$
  - Message $\boldsymbol{s}$ = "xyxxzyx"
  - Bit sequence $\boldsymbol{b}$ = "0100011100"

Optimization Problem

- **Average codeword length** is given as

$$\bar{\ell} = E\{\ell(S)\} = \sum_{i=0}^{M-1} p(a_i) \cdot \ell(a_i) \tag{77}$$

- **The goal of the lossless code design problem is to minimize the average codeword length $\bar{\ell}$ while being able to uniquely decode**

a_i	$p(a_i)$	code A	code B	code C	code D	code E
a_0	0.5	0	0	0	00	0
a_1	0.25	10	01	01	01	10
a_2	0.125	11	010	011	10	110
a_3	0.125	11	011	111	110	111
$\bar{\ell}$		1.5	1.75	1.75	2.125	1.75

Unique Decodability and Prefix Codes

- For unique decodability, we need to generate a code $\gamma : a_i \rightarrow \boldsymbol{b}_i$ such that

$$\text{if} \qquad a_k \neq a_j \qquad \text{then} \qquad \boldsymbol{b}_k \neq \boldsymbol{b}_j \tag{78}$$

  Codes that don't have that property are called **singular codes**

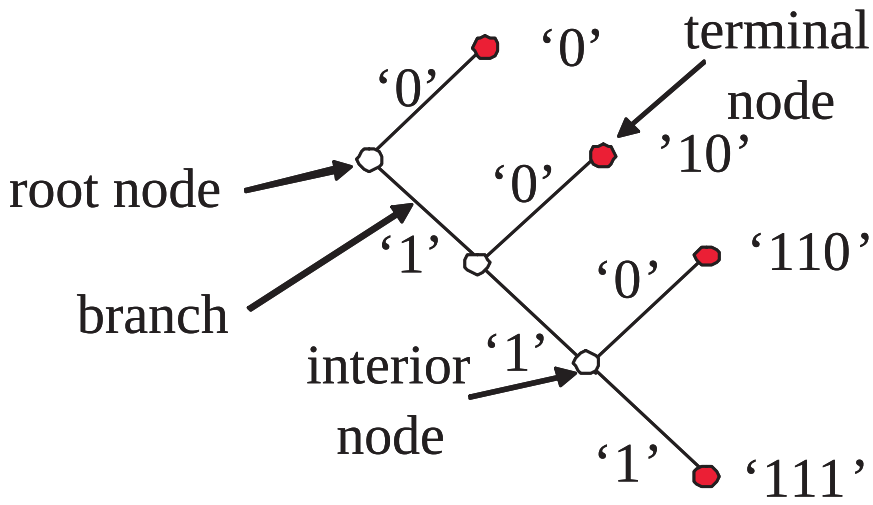
- For sequences of symbols, above constraint needs to be extended to the concatenation of multiple symbols

$\Longrightarrow$ **For a uniquely decodable code, a sequence of codewords can only be generated by one possible sequence of source symbols.**

- **Prefix codes**: One class of codes that satisfies the constraint of unique decodability
  - A code is called a prefix code if no codeword for an alphabet letter represents the codeword or a prefix of the codeword for any other alphabet letter
  - It is obvious that if the code is a prefix code, then any concatenation of symbols can be uniquely decoded

Binary Code Trees

- Prefix codes can be represented by trees



- A binary tree contains nodes with two branches (labelled as '0' and '1') leading to other nodes starting from a root node
- A node from which branches depart is called an interior node while a node from which no branches depart is called a terminal node
- A **prefix code** can be constructed by assigning letters of the alphabet $\mathcal{A}$ to terminal nodes of a binary tree

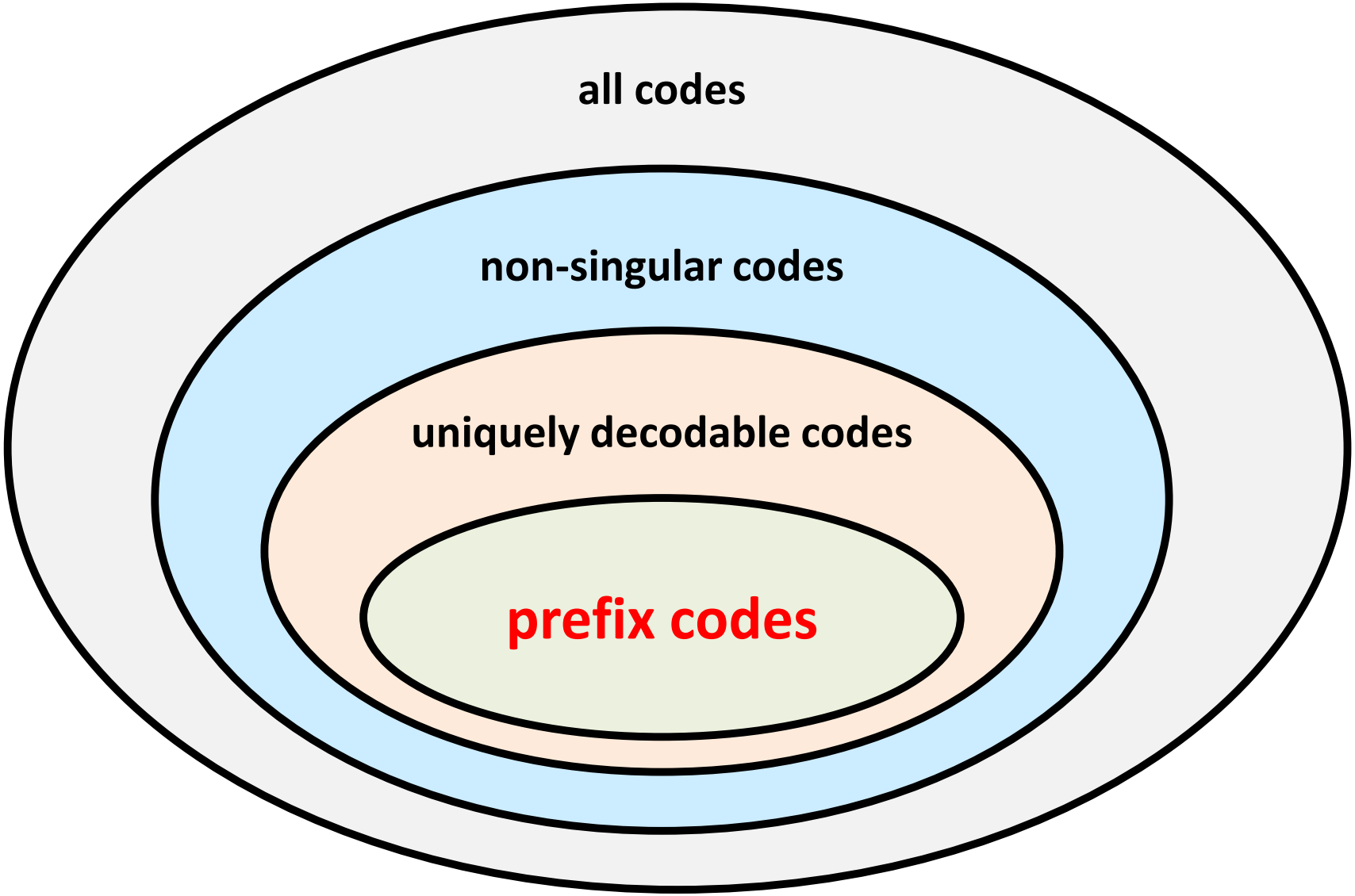
Parsing of Prefix Codes

- Given the code word assignment to terminal nodes of the binary tree, the parsing rule for this prefix code is given as follows

  1. Set the current node n_i equal to the root node
  2. Read the next bit b from the bitstream
  3. Follow the branch labelled with the value of b from the current node n_i to the descendant node n_j
  4. If n_j is a terminal node, return the associated alphabet letter and proceed with step 1.
     Otherwise, set the current node n_i equal to n_j and repeat the previous two steps

- Important properties of prefix codes:
  - Prefix codes are uniquely decodable
  - Prefix codes are **instantaneously decodable**

Classification of Codes



Unique Decodability: Kraft Inequality

- Assume fully balanced tree with depth $\ell_{\max}$ (length of longest codeword)
- Codewords are assigned to nodes with codeword length $\ell(a_k) \leq \ell_{\max}$
- Each choice with $\ell(a_k) \leq \ell_{\max}$ eliminates $2^{\ell_{\max}-\ell(a_k)}$ other possibilities of codeword assignment at level $\ell_{\max}$, example:
  - $\rightarrow$ $\ell_{\max} - \ell(a_k) = 0$, one option is covered
  - $\rightarrow$ $\ell_{\max} - \ell(a_k) = 1$, two options are covered
- Number of removed terminal nodes must be less than or equal to number of terminal nodes in balanced tree with depth $\ell_{\max}$, which is $2^{\ell_{\max}}$

$$\sum_{i=0}^{M-1} 2^{\ell_{\max}-\ell(a_i)} \leq 2^{\ell_{\max}} \tag{79}$$

- A code γ may be uniquely decodable (McMillan) if

$$\text{Kraft inequality:} \qquad \zeta(\gamma) = \sum_{i=0}^{M-1} 2^{-\ell(a_i)} \leq 1 \tag{80}$$

Proof of the Kraft Inequality

- Consider

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{i_0=0}^{M-1} \sum_{i_1=0}^{M-1} \cdots \sum_{i_{L-1}=0}^{M-1} 2^{-\left(\ell(a_{i_0}) + \ell(a_{i_1}) + \cdots + \ell(a_{i_{L-1}}) \right)} \tag{81}$$

- $\ell_L = \ell(a_{i_0}) + \ell(a_{i_1}) + \cdots + \ell(a_{i_{L-1}})$ represents the combined codeword length for coding L symbols
- Let $A(\ell_L)$ denote the number of distinct symbol sequences that produce a bit sequence with the same length ℓ_L
- Let $\ell_{\max}$ be the maximum codeword length
- Hence, we can write

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{\ell_L=L}^{L \cdot \ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \tag{82}$$

Proof of the Kraft Inequality

- We have

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{\ell_L=L}^{L \cdot \ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \tag{83}$$

- For a uniquely decodable code, $A(\ell_L)$ must be less than or equal to 2^{ℓ_L}, since there are only 2^{ℓ_L} distinct bit sequences of length ℓ_L
- Hence, a uniquely decodable code must fulfill the inequality

$$\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \right)^L = \sum_{\ell_L=L}^{L \cdot \ell_{\max}} A(\ell_L)\, 2^{-\ell_L} \leq \sum_{\ell_L=L}^{L \cdot \ell_{\max}} 2^{\ell_L}\, 2^{-\ell_L} = L\,(\ell_{\max} - 1) + 1 \tag{84}$$

- The left side of this inequality grows exponentially with L, while the right side grows only linearly with L
- If the Kraft inequality is not fulfilled, we can always find a value of L for which the condition (84) is violated

$\Longrightarrow$ Kraft inequality specifies a necessary condition for uniquely decodable codes

Prefix Codes and the Kraft Inequality

- Given is a set of codeword lengths $\{\ell_0, \ell_1, \cdots, \ell_{M-1}\}$ that satisfies the Kraft inequality, with $\ell_0 \leq \ell_1 \leq \cdots \leq \ell_{M-1}$
- Construction of prefix code
  - Start with fully balanced code tree of infinite depth (or depth ℓ_{M-1})
  - Choose a node of depth ℓ_0 for first codeword and prune tree at this node
  - Choose a node of depth ℓ_1 for second codeword and prune tree at this node
  - Continue this procedure until all codeword length are assigned
- Question: Is that always possible?
  - Selection of codeword ℓ_k removes $2^{\ell_i - \ell_k}$ codewords with length $\ell_i \geq \ell_k$

$\Longrightarrow$ For codeword of length ℓ_i, number of available choices is given by

$$n(\ell_i) = 2^{\ell_i} - \sum_{k=0}^{i-1} 2^{\ell_i - \ell_k} = 2^{\ell_i} \left(1 - \sum_{k=0}^{i-1} 2^{-\ell_k} \right) \tag{85}$$

Prefix Codes and the Kraft Inequality

- Number of available choices for codeword length ℓ_i

$$n(\ell_i) = 2^{\ell_i} - \sum_{k=0}^{i-1} 2^{\ell_i - \ell_k} = 2^{\ell_i} \left(1 - \sum_{k=0}^{i-1} 2^{-\ell_k} \right) \tag{86}$$

- Kraft inequality is fulfilled

$$\sum_{k=0}^{M-1} 2^{-\ell_k} \leq 1 \tag{87}$$

- This yields

$$n(\ell_i) \geq 2^{\ell_i} \left(\sum_{k=0}^{M-1} 2^{-\ell_k} - \sum_{k=0}^{i-1} 2^{-\ell_k} \right) = 2^{\ell_i} \sum_{k=i}^{M-1} 2^{-\ell_k} = 1 + \sum_{k=i+1}^{M-1} 2^{\ell_i - \ell_k} \geq 1 \tag{88}$$

$\Longrightarrow$ **Can construct prefix code for any set of codeword lengths that fulfills Kraft inequality**

Practical Importance of Prefix Codes

- We have shown:
  - All uniquely decodable codes fulfill Kraft inequality
  - Possible to construct prefix code for any set of codeword lengths that fulfills Kraft inequality

$\Longrightarrow$ **There are no uniquely decodable codes that have a smaller average codeword length than the best prefix code**

- Prefix codes have further desirable properties
  - Instantaneous decodability
  - Easy to construct

$\Longrightarrow$ **All variable-length codes used in practice are prefix codes**

Lower Bound for Average Codeword Length

- Average codeword length

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i)\,\ell(a_i) = -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{2^{-\ell(a_i)}}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (89)$$

- With the definition $q(a_i) = 2^{-\ell(a_i)} \Big/ \left(\sum_{k=0}^{M-1} 2^{-\ell(a_k)}\right)$, we obtain

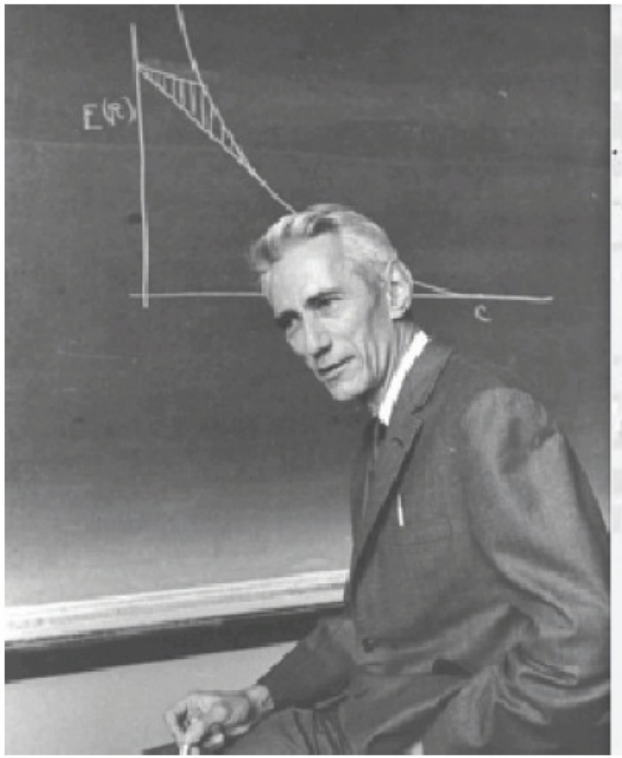
$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (90)$$

- We will show that

$$\bar{\ell} \geq -\sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) = H(S) \quad \text{(Entropy)} \quad (91)$$

Historical Reference

- C. E. SHANNON introduced entropy as an uncertainty measure for random experiments and derived it based on three postulates
- Published 1 year later as: "The Mathematical Theory of Communication"



The Bell System Technical Journal

Vol. XXVII *July, 1948* *No. 3*

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

Lower Bound for Average Codeword Length

- Average codeword length

$$\bar{\ell} = -\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) - \sum_{i=0}^{M-1} p(a_i)\,\log_2 p(a_i) \quad (92)$$

- Kraft inequality $\sum_{i=0}^{M-1} 2^{-\ell(a_i)} \leq 1$ applied to first term

$$-\log_2\left(\sum_{i=0}^{M-1} 2^{-\ell(a_i)}\right) \geq 0 \quad (93)$$

- Inequality $\ln x \leq x - 1$ (with equality if and only if $x = 1$), applied to second term, yields

$$\begin{aligned} -\sum_{i=0}^{M-1} p(a_i)\,\log_2\left(\frac{q(a_i)}{p(a_i)}\right) &\geq \frac{1}{\ln 2}\sum_{i=0}^{M-1} p(a_i)\,\left(1 - \frac{q(a_i)}{p(a_i)}\right) \\ &= \frac{1}{\ln 2}\left(\sum_{i=0}^{M-1} p(a_i) - \sum_{i=0}^{M-1} q(a_i)\right) = 0 \end{aligned} \quad (94)$$

- Called **divergence inequality**

Entropy and Redundancy

- Average codeword length $\bar{\ell}$ for uniquely decodable codes is bounded

$$\bar{\ell} \geq H(S) = E\{-\log_2 p(S)\} = -\sum_{i=0}^{M-1} p(a_i)\, \log_2 p(a_i) \tag{95}$$

- The measure $H(S)$ is called the **entropy** of a random variable S
- The entropy is a measure of the **uncertainty** of a random variable
- **Redundancy** of a code is given by the difference

$$\varrho = \bar{\ell} - H(S) = \sum_{i=0}^{M-1} p(a_i)\big(\ell(a_i) - \log_2 p(a_i)\big) \geq 0 \tag{96}$$

- Redundancy is zero only, if and only if
  - Kraft inequality is fulfilled with equality (codeword at all terminal nodes)
  - All probability masses are negative integer powers of two

An Upper Bound for the Minimum Average Codeword Length

- Upper bound of $\bar{\ell}$: Choose $\ell(a_i) = \lceil -\log_2 p(a_i) \rceil$, $\forall a_i \in \mathcal{A}$
- Codewords satisfy Kraft inequality (show using $\lceil x \rceil \geq x$)

$$\sum_{i=0}^{M-1} 2^{-\lceil -\log_2 p(a_i) \rceil} \leq \sum_{i=0}^{M-1} 2^{\log_2 p(a_i)} = \sum_{i=0}^{M-1} p(a_i) = 1 \tag{97}$$

- Obtained average codeword length (use $\lceil x \rceil < x + 1$)

$$\bar{\ell} = \sum_{i=0}^{M-1} p(a_i) \lceil -\log_2 p(a_i) \rceil < \sum_{i=0}^{M-1} p(a_i) \left(1 - \log_2 p(a_i)\right) = H(S) + 1 \tag{98}$$

$\Longrightarrow$ Bounds on minimum average codeword length $\bar{\ell}_{\min}$

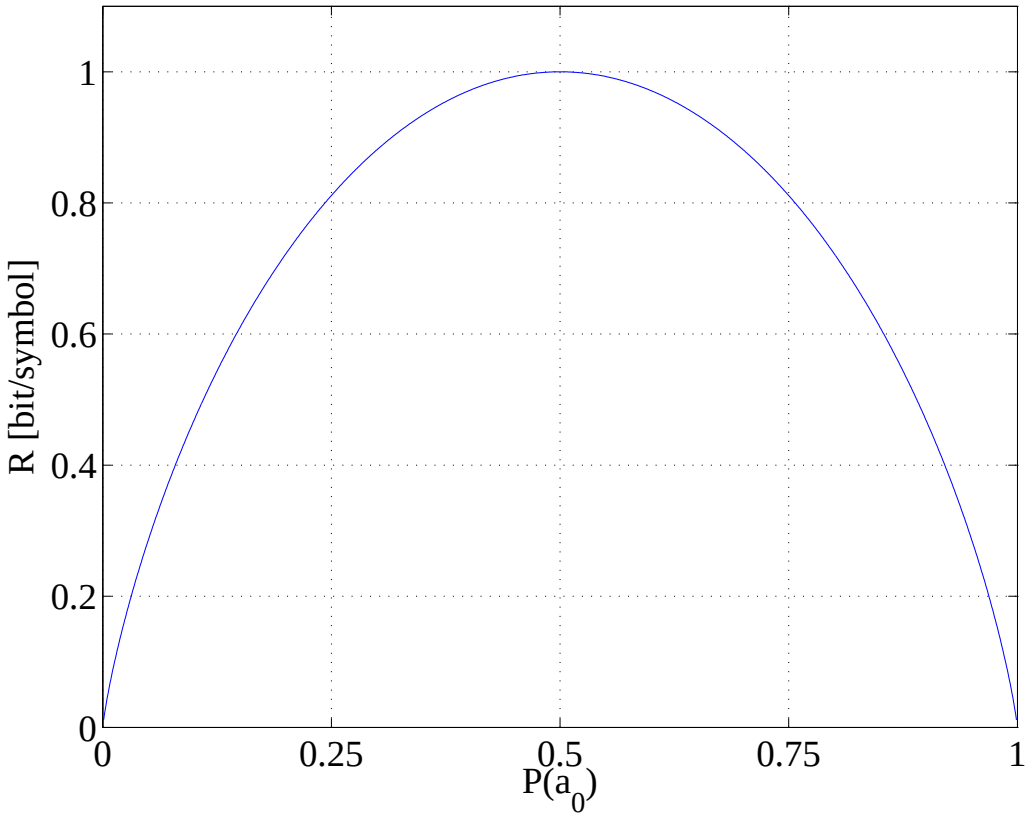
$$\boxed{H(S) \leq \bar{\ell}_{\min} < H(S) + 1} \tag{99}$$

Entropy of a Binary Source

- A binary source has probabilities $p(0) = p$ and $p(1) = 1 - p$
- The entropy of the binary source is given as

$$H(S) = -p \log_2 p - (1-p) \log_2 (1-p) = H_b(p) \tag{100}$$

with $H_b(x)$ being the so-called binary entropy function



Relative Entropy or Kullback-Leibler Divergence

- Defined as

$$D(p||q) = \sum_{i=0}^{M-1} p(a_i) \log_2 \frac{p(a_i)}{q(a_i)} \tag{101}$$

- Divergence inequality (we have already proofed it)

$$D(p||q) \geq 0 \qquad \text{(equality if and only if } p = q) \tag{102}$$

- Note: $D(p||q) \neq D(q||p)$

- What does the measure $D(p||q)$ tell us?
  - Assume we have an optimal code with an average codeword length equal to the entropy for a pmf q and apply it to a pmf p
  - $D(p||q)$ is the difference between average codeword length and entropy

$$\begin{aligned} D(p||q) &= -\sum_{i=0}^{M-1} p(a_i) \log_2 q(a_i) + \sum_{i=0}^{M-1} p(a_i) \log_2 p(a_i) \\ &= \sum_{i=0}^{M-1} p(a_i) \ell_q(a_i) - H(S) = \bar{\ell}_q - H(S) \end{aligned} \tag{103}$$

Optimal Prefix Codes

Conditions for optimal prefix codes

1. For any two symbols $a_i, a_j \in \mathcal{A}$ with $p(a_i) > p(a_j)$, the associated codeword lengths satisfy $\ell(a_i) \leq \ell(a_j)$
2. There are always two codewords that have the maximum codeword length and differ only in the final bit

Justification

1. Otherwise, an exchange of the codewords for the symbols a_i and a_j would decrease the average codeword length while preserving the prefix property
2. Otherwise, the removal of the last bit of the codeword with maximum length would preserve the prefix property and decrease the average codeword length

The Huffman Algorithm

How can we generate an optimal prefix code?

- The answer to this question was given by D. A. Huffman in 1952
- The so-called Huffman algorithm always finds a prefix-free code with minimum redundancy
- For a proof that Huffman codes are optimal instantaneous codes (with minimum expected length), see Cover and Thomas

General idea

- Both optimality conditions are obeyed if the two codewords with maximum length (that differ only in the final bit) are assigned to the letters a_i and a_j with minimum probabilities
- The two letters are then treated as a new letter with probability $p(a_i) + p(a_j)$
- Procedure is repeated for the new alphabet

The Huffman Algorithm

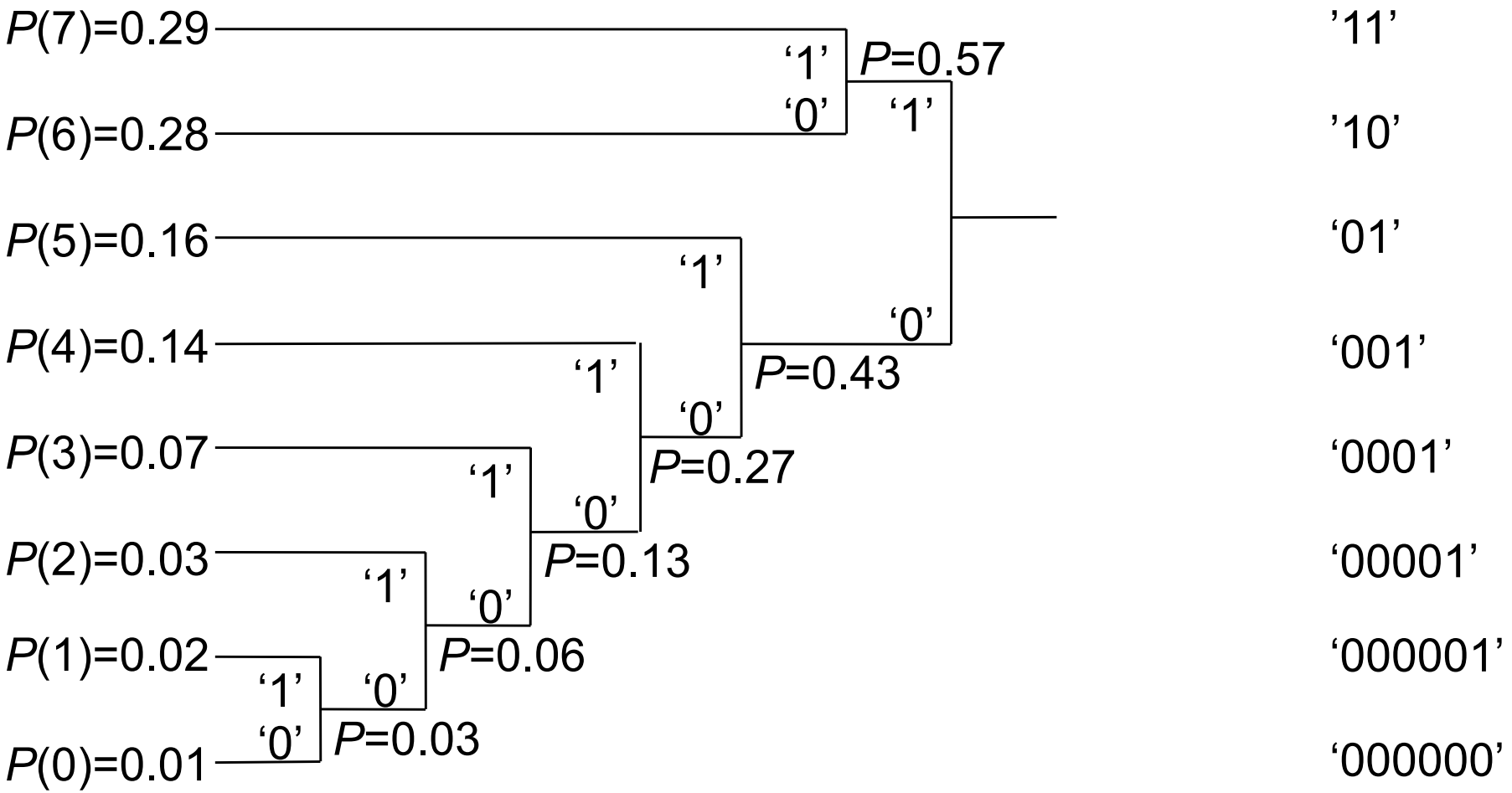
Huffman algorithm for given alphabet $\mathcal{A}$ with marginal pmf p

1. Select the two letters a_i and a_j with the smallest probabilities and create a parent node for the nodes that represent these two letters in the binary code tree
2. Replace the letters a_i and a_j by a new letter with an associated probability of $p(a_i) + p(a_j)$
3. If more than one letter remains, repeat the previous steps
4. Convert the binary code tree into a prefix code

Note: There are multiple optimal prefix codes

- Assigment of "0" and "1" to tree branches is arbitrary
- If some letters have the same probability (at some stage of the algorithm), there might be multiple ways to select two letters with smallest probabilities
- But, all optimal prefix codes have the same average codeword length

Example for the Design of a Huffman code



Conditional Huffman Codes

- Random process $\{S_n\}$ with memory: Design VLC for conditional pmf
- Example:
  - Stationary discrete Markov process, $\mathcal{A} = \{a_0, a_1, a_2\}$
  - Conditional pmfs $p(a|a_k) = P(S_n = a \mid S_{n-1} = a_k)$ with $k = 0, 1, 2$

a	a_0	a_1	a_2	entropy
$p(a\|a_0)$	0.90	0.05	0.05	$H(S_n\|a_0) = 0.5690$
$p(a\|a_1)$	0.15	0.80	0.05	$H(S_n\|a_1) = 0.8842$
$p(a\|a_2)$	0.25	0.15	0.60	$H(S_n\|a_2) = 1.3527$
$p(a)$	$0.6\overline{4}$	$0.2\overline{4}$	$0.\overline{1}$	$H(S) = 1.2575$

- Design Huffman code for conditional pmfs

a_i	Huffman codes for conditional pmfs			Huffman code for marginal pmf
	$S_{n-1} = a_0$	$S_{n-1} = a_1$	$S_{n-1} = a_2$	
a_0	1	00	00	1
a_1	00	1	01	00
a_2	01	01	1	01
	$\bar{\ell}_0 = 1.1$	$\bar{\ell}_1 = 1.2$	$\bar{\ell}_2 = 1.4$	$\bar{\ell} = 1.3556$

Average Codeword Length for Conditional Huffman Codes

- We know: Average codeword length $\bar{\ell}_k = \bar{\ell}(S_{n-1}=a_k)$ is bounded by

$$H(S_n|a_k) \leq \bar{\ell}_k < H(S_n|a_k) + 1 \tag{104}$$

with **conditional entropy** of S_n given the event $\{S_{n-1}=a_k\}$

$$H(S_n|a_k) = H(S_n|S_{n-1}=a_k) = -\sum_{i=0}^{M-1} p(a_i|a_k)\, \log_2 p(a_i|a_k) \tag{105}$$

- Resulting average codeword length

$$\bar{\ell} = \sum_{k=0}^{M-1} p(a_k)\, \bar{\ell}_k \tag{106}$$

- Resulting bounds

$$\sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) \leq \bar{\ell} < \sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) + 1 \tag{107}$$

Conditional Entropy

- Lower bound is called **conditional entropy** $H(S_n|S_{n-1})$ of the random vriable S_n given random variable S_{n-1}

$$\begin{aligned} H(S_n|S_{n-1}) &= E\{-\log_2 p(S_n|S_{n-1})\} \\ &= = \sum_{k=0}^{M-1} p(a_k)\, H(S_n|S_{n-1}=a_k) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i,a_k)\, \log_2 p(a_i|a_k), \end{aligned} \tag{108}$$

- Minimum average codeword length for conditional code is bounded by

$$\boxed{H(S_n|S_{n-1}) \le \bar{\ell}_{\min} < H(S_n|S_{n-1}) + 1} \tag{109}$$

Conditioning May Reduce Minimum Average Codeword Length

- Minimum average codeword length $\bar{\ell}_{\min}$

$$H(S_n|S_{n-1}) \leq \bar{\ell}_{\min} < H(S_n|S_{n-1}) + 1 \tag{110}$$

- Conditioning may reduce minimum average codeword length (use divergence inequality)

$$\begin{aligned} H(S) - H(S_n|S_{n-1}) &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i, a_k)\Big(\log_2 p(a_i) - \log_2 p(a_i|a_k)\Big) \\ &= -\sum_{i=0}^{M-1}\sum_{k=0}^{M-1} p(a_i, a_k)\, \log_2 \frac{p(a_i)\, p(a_k)}{p(a_i, a_k)} \\ &\geq 0 \end{aligned} \tag{111}$$

- Note: Equality for iid process, $p(a_i, a_k) = p(a_i)\, p(a_k)$
- For the example Markov source:
  - No conditioning: $H(S) = 1.2575$, $\ell_{min} = 1.3556$
  - Conditioning: $H(S_n|S_{n-1}) = 0.7331$, $\ell_{min} = 1.1578$

Huffman Coding of Fixed-Length Vectors

- Consider stationary discrete random sources $\boldsymbol{S} = \{S_n\}$ with an M-ary alphabet $\mathcal{A} = \{a_0, \cdots, a_{M-1}\}$
- N symbols are coded jointly (code vector instead of scalar)
- Design Huffman code for joint pmf
  $p(a_0, \cdots, a_{N-1}) = P(S_n = a_0, \cdots, S_{n+N-1} = a_{N-1})$
- Average codeword length $\bar{\ell}_{\min}$ per symbol is bounded

$$\frac{H(S_n, \cdots, S_{n+N-1})}{N} \leq \bar{\ell}_{\min} < \frac{H(S_n, \cdots, S_{n+N-1})}{N} + \frac{1}{N} \tag{112}$$

- Define **block entropy**

$$\begin{aligned} H(S_n, \cdots, S_{n+N-1}) &= E\{-\log_2 p(S_n, \cdots, S_{n+N-1})\} \\ &= -\sum_{a_0} \cdots \sum_{a_{N-1}} p(a_0, \cdots, a_{N-1}) \log_2 p(a_0, \cdots, a_{N-1}) \end{aligned} \tag{113}$$

Entropy Rate

Entropy rate

- The following limit is called **entropy rate**

$$\boxed{\bar{H}(\boldsymbol{S}) = \lim_{N \to \infty} \frac{H(S_0, \cdots, S_{N-1})}{N}} \tag{114}$$

- The limit in (114) always exists for stationary sources

Fundamental lossless source coding theorem

- Entropy rate $\bar{H}(\boldsymbol{S})$: Greatest lower bound for the average codeword length $\bar{\ell}$ per symbol

$$\boxed{\bar{\ell} \geq \bar{H}(\boldsymbol{S})} \tag{115}$$

- Always asymptotically achievable with block Huffman coding for $N \to \infty$

Entropy Rate for Special Sources

- Entropy rate for iid processes

$$\begin{aligned}\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{\sum_{n=0}^{N-1} E\{-\log_2 p(S_n)\}}{N} \\ &= \lim_{N\to\infty} E\{-\log_2 p(S_n)\} \\ &= H(S) \end{aligned} \tag{116}$$

- Entropy rate for stationary Markov processes

$$\begin{aligned}\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0, S_1, \cdots, S_{N-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0)\} + \sum_{n=1}^{N-1} E\{-\log_2 p(S_n|S_{n-1})\}}{N} \\ &= \lim_{N\to\infty} \frac{E\{-\log_2 p(S_0)\}}{N} + E\{-\log_2 p(S_n|S_{n-1})\} \\ &= H(S_n|S_{n-1}) \end{aligned} \tag{117}$$

Example for Block Huffman Coding

Example:

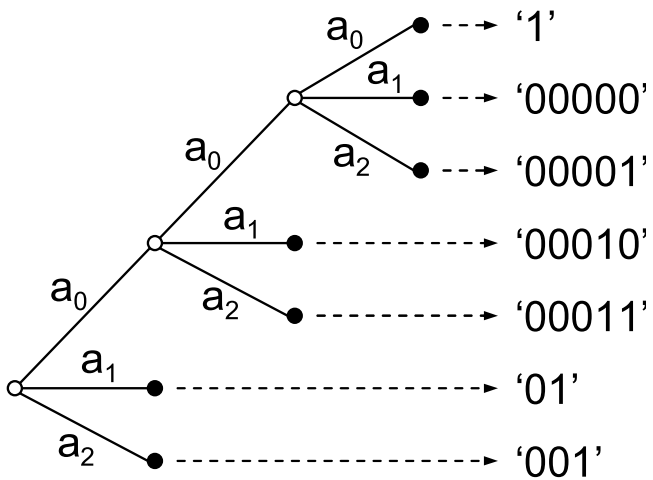
- Joint Huffman coding of 2 symbols for example Markov source
- Efficiency of block Huffman coding for different N
- Table sizes for different N

$a_i a_k$	$p(a_i, a_k)$	codewords
$a_0 a_0$	0.58	1
$a_0 a_1$	$0.03\overline{2}$	00001
$a_0 a_2$	$0.03\overline{2}$	00010
$a_1 a_0$	$0.03\overline{6}$	0010
$a_1 a_1$	$0.19\overline{5}$	01
$a_1 a_2$	$0.01\overline{2}$	000000
$a_2 a_0$	$0.02\overline{7}$	00011
$a_2 a_1$	$0.01\overline{7}$	000001
$a_2 a_2$	$0.0\overline{6}$	0011

N	$\bar{\ell}$	N_C
1	1.3556	3
2	1.0094	9
3	0.9150	27
4	0.8690	81
5	0.8462	243
6	0.8299	729
7	0.8153	2187
8	0.8027	6561
9	0.7940	19683

Huffman Codes for Variable-Length Vectors

- Assign codewords to variable-length vectors: V2V codes



- Associate each leaf node $\mathcal{L}_k$ of the symbol tree with a codeword
- Use pmf of leaf nodes $p(\mathcal{L}_k)$ for Huffman design
- Average number of bits per alphabet letter

$$\bar{\ell} = \frac{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, \ell_k}{\sum_{k=0}^{N_{\mathcal{L}}-1} p(\mathcal{L}_k)\, N_k} \tag{118}$$

where N_k denotes number of alphabet letters associated with $\mathcal{L}_k$

V2V Code Performance

- Example Markov process: $\bar{H}(\boldsymbol{S}) = H(S_n|S_{n-1}) = 0.7331$
- Faster reduction of $\bar{\ell}$ with increasing N_C compared to fixed-length vector Huffman coding

$\boldsymbol{a}_k$	$p(\mathcal{L}_k)$	codewords
a_0a_0	0.5799	1
a_0a_1	0.0322	00001
a_0a_2	0.0322	00010
a_1a_0	0.0277	00011
$a_1a_1a_0$	0.0222	000001
$a_1a_1a_1$	0.1183	001
$a_1a_1a_2$	0.0074	0000000
a_1a_2	0.0093	0000001
a_2	0.1708	01

N_C	$\bar{\ell}$
5	1.1784
7	1.0551
9	1.0049
11	0.9733
13	0.9412
15	0.9293
17	0.9074
19	0.8980
21	0.8891

Summary on Variable-Length Coding

Uniquely decodable codes

- Necessary condition: Kraft inequality
- Prefix codes: Instantaneously decodable
- Can construct prefix codes for codeword lengths that fulfill Kraft inequality

Bounds for lossless coding

- Entropy
- Conditional entropy
- Block entropy
- Entropy rate

Optimal prefix codes

- Huffman algorithm for given pmf
- Scalar Huffman codes
- Conditional Huffman codes
- Block Huffman codes
- V2V codes

Exercise 3

A fair coin is tossed an infinite number of times. Let Y_n be a random variable, with $n \in \mathbb{Z}$, that describes the outcome of the n-th coin toss. If the outcome of the n-th coin toss is head, Y_n is equal to 1; if it is tail, Y_n is equal to 0. Now consider the random process $\mathbf{X} = \{X_n\}$. The random variables X_n are determined by $X_n = Y_n + Y_{n-1}$, and thus describe the total number of heads in the n-th and $(n-1)$-th coin tosses.

(a) Determine the marginal pmf $p_{X_n}(x_n)$ and the marginal entropy $H(X_n)$. Is it possible to design a uniquely decodable code with one codeword per possible outcome of X_n that has an average codeword length equal to the marginal entropy?

(b) Determine the conditional pmf $p_{X_n|X_{n-1}}(x_n|x_{n-1})$ and the conditional entropy $H(X_n|X_{n-1})$. Design a conditional Huffman code. What is the average codeword length of the conditional Huffman code?

(c) Is the random process $\mathbf{X}$ a Markov process?

(d) Derive a general formula for the N-th order block entropy $H_N = H(X_n, \cdots, X_{n-N+1})$. How many symbols have to be coded jointly at minimum for obtaining a code that is more efficient than the conditional Huffman code developed in (b)?

(e) Calculate the entropy rate $\bar{H}(\mathbf{X})$ of the random process $\mathbf{X}$. Is it possible to design a variable length code with finite complexity and an average codeword length equal to the entropy rate? If yes, what requirement has to be fulfilled?

Exercise 4

Given is a discrete iid process $\mathbf{X}$ with the alphabet $\mathcal{A} = \{a, b, c, d, e, f, g\}$. The pmf $p_X(x)$ and 6 example codes are listed in the following table.

x	$p_X(x)$	A	B	C	D	E	F
a	$1/3$	1	0	00	01	000	1
b	$1/9$	0001	10	010	101	001	100
c	$1/27$	000000	110	0110	111	010	100000
d	$1/27$	00001	1110	0111	010	100	10000
e	$1/27$	000001	11110	100	110	111	000000
f	$1/9$	001	111110	101	100	011	1000
g	$1/3$	01	111111	11	00	001	10

(a) Develop a Huffman code for the given pmf $p_X(x)$, calculate its average codeword length and its absolute and relative redundancy.

(b) For all codes A, B, C, D, E, and F, do the following:

- Calculate the average codeword length per symbol;
- Determine whether the code is a singular code;
- Determine whether the code is uniquely decodable;
- Determine whether the code is a prefix code;
- Determine whether the code is an optimal prefix code.

(c) Briefly describe a process for decoding a symbol sequence given a finite sequence of K bits that is coded with code F.

Exercise 5

Given is a Bernoulli process $\mathbf{X}$ with the alphabet $\mathcal{A} = \{a, b\}$ and the pmf $p_X(a) = p$, $p_X(b) = 1 - p$. Consider the three codes in the following table.

Code A		Code B		Code C	
symbols	codeword	symbols	codeword	symbol	codeword
aa	1	aa	0001	a	0
ab	01	ab	001	b	1
b	00	ba	01		
		bb	1		

(a) Calculate the average codeword length per symbol for the three codes.

(b) For which probabilities p is the code A more efficient than code B?

(c) For which probabilities p is the simple code C more efficient than both code A and code B?

Exercise 6

Given is a Bernoulli process $\mathbf{B} = \{B_n\}$ with the alphabet $\mathcal{A}_B = \{0, 1\}$, the pmf $p_B(0) = p$, $p_B(1) = 1 - p$, and $0 \leq p < 1$. Consider the random variable X that specifies the number of random variables B_n that have to be observed to get exactly one "1".

Calculate the entropies $H(B_n)$ and $H(X)$.

For which value of p, with $0 < p < 1$, is $H(X)$ four times as large as $H(B_n)$?

Hint:

$$\forall_{|a|<1}, \sum_{k=0}^{\infty} a^k = \frac{1}{1-a}$$

$$\forall_{|a|<1}, \sum_{k=0}^{\infty} k\, a^k = \frac{a}{(1-a)^2}$$

Exercise 7

Proof the chain rule for the joint entropy,

$$H(X,Y) = H(X) + H(Y|X).$$

Exercise 8

Investigate the entropy of a function of a random variable X. Let X be a discrete random variable with the alphabet $\mathcal{A}_X = \{0, 1, 2, 3, 4\}$ and the binomial pmf

$$p_X(x) = \begin{cases} 1/16 & : \quad x = 0 \vee x = 4 \\ 1/4 & : \quad x = 1 \vee x = 3 \\ 3/8 & : \quad x = 2 \end{cases} .$$

(a) Calculate the entropy $H(X)$.

(b) Consider the functions $g_1(x) = x^2$ and $g_2(x) = (x-2)^2$.
Calculate the entropies $H(g_1(X))$ and $H(g_2(X))$.

(c) Proof that the entropy $H(g(X))$ of a function $g(x)$ of a random variable X is not greater than the entropy of the random variable X,

$$H(g(X)) \leq H(X)$$

Determine the condition under which equality is achieved.

Outline

Part I: Source Coding Fundamentals

- Probability, Random Variables and Random Processes
- Lossless Source Coding
  - Introduction
  - Variable-Length Coding for Scalars
  - Variable-Length Coding for Vectors
  - **Elias and Arithmetic Coding**
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

Elias Coding and Arithmetic Coding

- Scalar and conditional Huffman codes can be very inefficient
- Main drawback of block Huffman codes: Large table sizes
- Another class of uniquely decodable codes are **Elias and Arithmetic codes**
- Mapping of a string of N symbols $\boldsymbol{s} = \{s_0, s_1, ..., s_{N-1}\}$ onto a string of K bits $\boldsymbol{b} = \{b_0, b_1, ..., b_{K-1}\}$

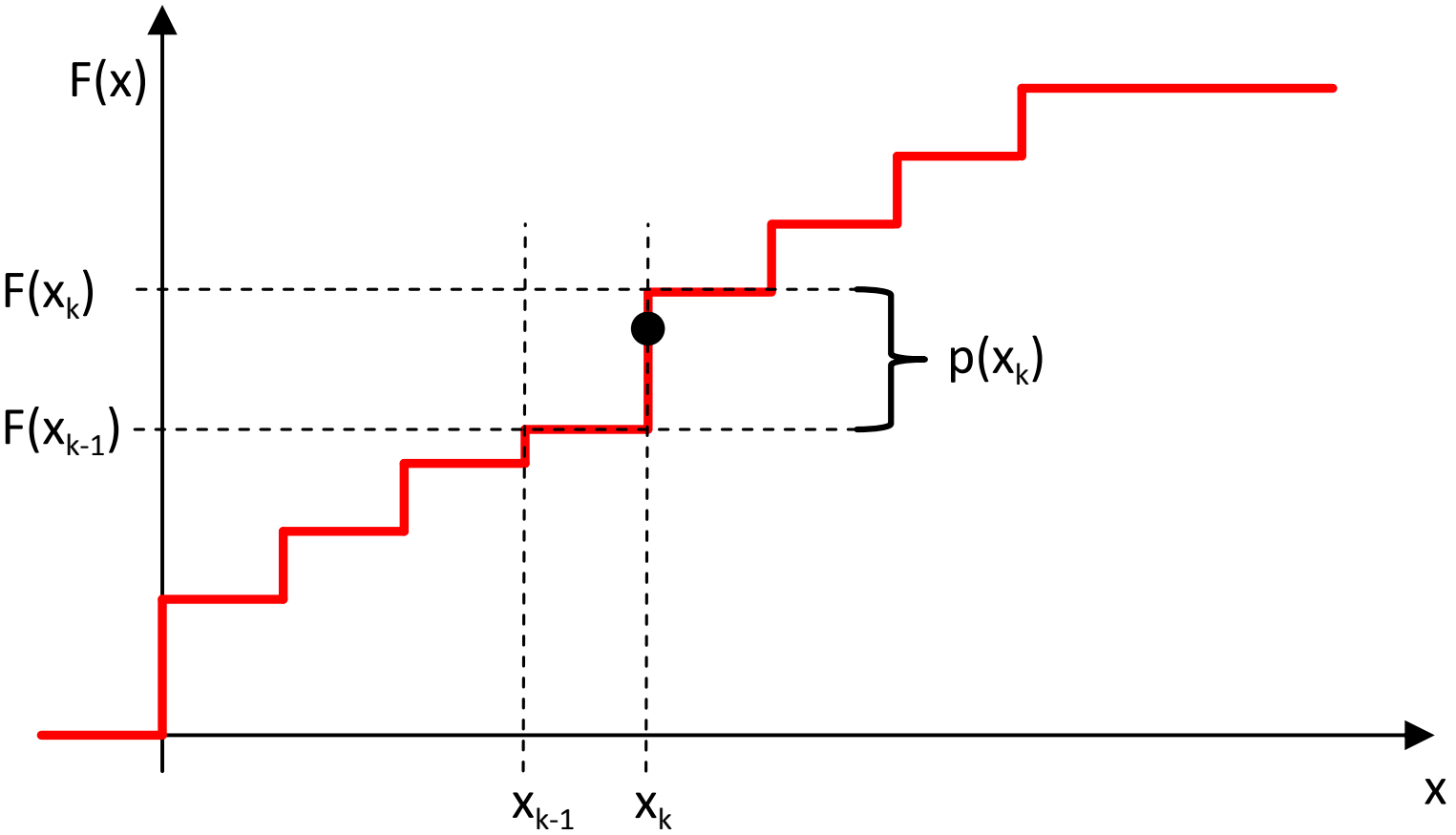
$$\gamma : \boldsymbol{s} \rightarrow \boldsymbol{b} \tag{119}$$

- Decoding or parsing maps the bit string onto the string of symbols

$$\gamma^{-1} : \boldsymbol{b} \rightarrow \boldsymbol{s} \tag{120}$$

- Complexity of code construction: Linear per symbol
- Suitable for adapting pmfs to instationary statistics

Idea of Elias Coding



- Order symbols or messages
- Transmit number in interval $[0, 1)$ which characterizes the symbol or message
- Number of transmitted bits depends on probability of the message

Define an Order of Symbol Sequences

- Consider coding of symbol sequences $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{N-1}\}$
- Realization of sequence of random variables $\boldsymbol{S} = \{S_0, S_1, \ldots, S_{N-1}\}$
- Number N of symbols is known at encoder and decoder
- Each random variable S_n is characterized by an alphabet $\mathcal{A}_n$ of M_n symbols

- Statistical properties are characterized by joint pmf

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(S_0=s_0, S_1=s_1, \cdots, S_{N-1}=s_{N-1}) \tag{121}$$

- Need to define an order for symbol sequences
- For example: Symbol sequence $\boldsymbol{s}_a = \{s_0^a, s_1^a, \cdots, s_{N-1}^a\}$ is less than another symbol sequence $\boldsymbol{s}_b = \{s_0^b, s_1^b, \cdots, s_{N-1}^b\}$ if and only if there exists an integer n, with $0 \leq n < N$ such that

$$s_k^a = s_k^b \quad \text{for } k = 0, \cdots, n-1 \quad \text{and} \quad s_n^a < s_n^b \tag{122}$$

Mapping of Symbol Sequences to Intervals

- Joint pmf

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(S_0=s_0, S_1=s_1, \cdots, S_{N-1}=s_{N-1}) \tag{123}$$

- Using the defined order for symbol sequences, the pmf of $\boldsymbol{s}$ can be written

$$p(\boldsymbol{s}) = P(\boldsymbol{S}=\boldsymbol{s}) = P(\boldsymbol{S}\leq\boldsymbol{s}) - P(\boldsymbol{S}<\boldsymbol{s}) \tag{124}$$

- Mapping of $\boldsymbol{s} = \{s_0, s_1, \ldots, s_{N-1}\}$ to half-open interval $\mathcal{I}_N \subset [0,1)$

$$\boxed{\mathcal{I}_N(\boldsymbol{s}) = [L_N, L_N+W_N) = \big[P(\boldsymbol{S}<\boldsymbol{s}),\ P(\boldsymbol{S}\leq\boldsymbol{s})\big)} \tag{125}$$

with

$$L_N = P(\boldsymbol{S}<\boldsymbol{s}) \tag{126}$$

$$W_N = P(\boldsymbol{S}=\boldsymbol{s}) = p(\boldsymbol{s}) \tag{127}$$

Unique Identification: The Intervals are Disjoint

- Consider two symbol sequences $\boldsymbol{s}_a$ and $\boldsymbol{s}_b$, with $\boldsymbol{s}_a < \boldsymbol{s}_b$
- Intervals are disjoint if and only if $L_N^b \geq L_N^a + W_N^a$
- Proof:

$$\begin{aligned} L_N^b &= P(\boldsymbol{S} < \boldsymbol{s}_b) \\ &= P(\{\boldsymbol{S} \leq \boldsymbol{s}_a\} \cup \{\boldsymbol{s}_a < \boldsymbol{S} < \boldsymbol{s}_b\}) \\ &= P(\boldsymbol{S} \leq \boldsymbol{s}_a) + \underbrace{P(\boldsymbol{s}_a < \boldsymbol{S} < \boldsymbol{s}_b)}_{\geq 0} \\ &\geq P(\boldsymbol{S} \leq \boldsymbol{s}_a) \\ &= L_N^a + W_N^a \end{aligned} \tag{128}$$

$\Longrightarrow$ Intervals $\mathcal{I}_N^a$ and $\mathcal{I}_N^b$ do not overlap

$\Longrightarrow$ Any number v in the interval $\mathcal{I}_N(\boldsymbol{s})$ uniquely identifies $\boldsymbol{s}$

How Many Bits for Identifying an Interval?

- Identify an interval $\mathcal{I}_N(\boldsymbol{s})$ for a sequence $\boldsymbol{s}$ by a number v
- Number v can be represented as a binary fraction with K bits

$$v = \sum_{i=0}^{K-1} b_i \cdot 2^{i-1} = 0.b_0 b_1 \cdots b_{K-1} \in \mathcal{I}_N(\boldsymbol{s}) \tag{129}$$

- For identifying $\boldsymbol{s}$: Transmit bit sequence $\boldsymbol{b} = \{b_0, b_1, \cdots, b_{K-1}\}$
- Elias code: Assignment of bit sequences $\boldsymbol{b}$ to symbol sequences $\boldsymbol{s}$

- Question: How many bits do we need to uniquely identify an interval $\mathcal{I}_N(\boldsymbol{s})$?
- Intuitively: Size of interval, given by $p(\boldsymbol{s})$, governs number K of bits that are needed to identify the interval

$$\begin{aligned}
p(\boldsymbol{s}) = 1/2 \quad &\rightarrow \quad \mathcal{B} = \{.0, .1\} \\
p(\boldsymbol{s}) = 1/4 \quad &\rightarrow \quad \mathcal{B} = \{.00, .01, .10, .11\} \\
p(\boldsymbol{s}) = 1/8 \quad &\rightarrow \quad \mathcal{B} = \{.000, .001, .010, .011, .100, .101, .110, .111\}
\end{aligned}$$

How Many Bits for Identifying an Interval?

- Goal: **Choose real number $v \in \mathcal{I}_N$ that can be represented with the minimum amount of bits**
- Distance between successive binary fractions of K bits is 2^{-K}
- To be sure that a binary fraction of K bits falls inside an interval of width W_N, we need

$$\begin{aligned} 2^{-K} &\leq W_N \\ K &\geq -\log_2 W_N \end{aligned} \tag{130}$$

- Hence, we choose

$$K = K(\boldsymbol{s}) = \lceil -\log_2 W_N \rceil = \lceil -\log_2 p(\boldsymbol{s}) \rceil \tag{131}$$

- The binary number v identifying the interval $\mathcal{I}_N$ can be determined by

$$v = \lceil L_N \cdot 2^K \rceil \cdot 2^{-K} \tag{132}$$

Verification of Selection of Bits and Bitstring

- Binary number v identifying the interval $\mathcal{I}_n$

$$v = \lceil L_N \cdot 2^K \rceil \cdot 2^{-K} \qquad \text{with} \qquad K = \lceil -\log_2 W_N \rceil \tag{133}$$

- With $x \leq \lceil x \rceil$ and $\lceil x \rceil < x + 1$, we obtain

$$L_N \leq v < L_N + 2^{-K} \tag{134}$$

- Using the expression for the required number of bits

$$K = \lceil -\log_2 p(\boldsymbol{s}) \rceil \geq -\log_2 p(\boldsymbol{s}) \quad \Longrightarrow \quad 2^{-K} \leq p(\boldsymbol{s}) = W_N \tag{135}$$

yields

$$L_N \leq v < L_N + W_N \tag{136}$$

- The representative $v = 0.b_0 b_1 \ldots b_{K-1}$ always lies inside the interval $\mathcal{I}_N(\boldsymbol{s})$

$\Longrightarrow$ Message $\boldsymbol{s}$ can be uniquely decoded from the transmitted bit string $\boldsymbol{b} = \{b_0, b_1, \ldots, b_{K-1}\}$ of $K(\boldsymbol{s}) = \lceil -\log_2 p(\boldsymbol{s}) \rceil$ bits

Redundancy of Elias Coding

- Average codeword length per symbol

$$\bar{\ell} = \frac{E\{K(\boldsymbol{S})\}}{N} = \frac{E\{\lceil -\log_2 p(\boldsymbol{S}) \rceil\}}{N} \tag{137}$$

- Applying inequalities $x \leq \lceil x \rceil$ and $\lceil x \rceil < x + 1$, we obtain

$$\frac{E\{-\log_2 p(\boldsymbol{S})\}}{N} \leq \bar{\ell} < \frac{E\{1 - \log_2 p(\boldsymbol{S})\}}{N} \tag{138}$$

- Average codeword length is bounded

$$\boxed{\frac{H_N(\boldsymbol{S})}{N} \leq \bar{\ell} \leq \frac{H_N(\boldsymbol{S})}{N} + \frac{1}{N}} \tag{139}$$

- Note: Same bounds as for block Huffman codes
- For specific application: One additional bit required (see exercise)
- Question: What is the advantage?

Derivation of Iterative Algorithm for Elias Coding

Iterative construction of codewords

- Consider sub-sequences $\boldsymbol{s}^{(n)} = \{s_0, s_1, \cdots, s_{n-1}\}$ with $1 \leq n \leq N$
- Interval width W_{n+1} for the sub-sequence $\boldsymbol{s}^{(n+1)} = \{\boldsymbol{s}^{(n)}, s_n\}$

$$\begin{aligned} W_{n+1} &= P\big(\boldsymbol{S}^{(n+1)} = \boldsymbol{s}^{(n+1)}\big) \\ &= P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)},\ S_n = s_n\big) \\ &= P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \cdot P\big(S_n = s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \end{aligned}$$

- Iteration rule for interval width

$$\boxed{W_{n+1} = W_n \cdot p(s_n \,|\, s_0, s_1, \ldots, s_{n-1})} \tag{140}$$

- Since $p(s_n \,|\, s_0, s_1, \ldots, s_{n-1}) \leq 1$, it follows

$$W_{n+1} \leq W_n \tag{141}$$

Derivation of Iterative Algorithm for Elias Coding

- Derivation for lower interval border L_{n+1} for the sub-sequence $\boldsymbol{s}^{(n+1)} = \{\boldsymbol{s}^{(n)}, s_n\}$

$$\begin{aligned} L_{n+1} &= P\big(\boldsymbol{S}^{(n+1)} < \boldsymbol{s}^{(n+1)}\big) \\ &= P\big(\boldsymbol{S}^{(n)} < \boldsymbol{s}^{(n)}\big) + P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)},\, S_n < s_n\big) \\ &= P\big(\boldsymbol{S}^{(n)} < \boldsymbol{s}^{(n)}\big) + P\big(\boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \cdot P\big(S_n < s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \end{aligned}$$

- Iteration rule of lower interval boundary

$$\boxed{L_{n+1} = L_n + W_n \cdot c(s_n \,|\, s_0, s_1, \ldots, s_{n-1})} \tag{142}$$

with the cmf $c(\cdot)$ being defined as

$$c(s_n \,|\, s_0, s_1, \ldots, s_{n-1}) = \sum_{\forall a \in \mathcal{A}_n :\, a < s_n} p(a \,|\, s_0, s_1, \ldots, s_{n-1}) \tag{143}$$

- Note: The function $c(\cdot)$ excludes the current symbol
- Since $W_n \cdot c(s_n \,|\, s_0, s_1, \ldots, s_{n-1}) \geq 0$, it follows

$$L_{n+1} \geq L_n \tag{144}$$

Intervals Are Nested

- Iteration rules:

$$\begin{aligned} W_{n+1} &= W_n \cdot P\big(S_n = s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ L_{n+1} &= L_n + W_n \cdot P\big(S_n < s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \end{aligned}$$

- We have already shown: $L_{n+1} \geq L_n$
- Now, we consider upper interval boundary $L_{n+1} + W_{n+1}$

$$\begin{aligned} L_{n+1} + W_{n+1} &= L_n + W_n \cdot P\big(S_n < s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &\quad + W_n \cdot P\big(S_n = s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &= L_n + W_n \cdot P\big(S_n \leq s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big) \\ &= L_n + W_n - \underbrace{W_n \cdot P\big(S_n > s_n \,\big|\, \boldsymbol{S}^{(n)} = \boldsymbol{s}^{(n)}\big)}_{\geq 0} \\ &\leq L_n + W_n \end{aligned} \tag{145}$$

$\Longrightarrow$ Intervals are nested: $\mathcal{I}_{n+1} \subset \mathcal{I}_n$

Iterative Algorithm for IID and Markov Sources

- Derivation above for general case of dependent and differently distributed random variables (may even have different alphabets)
- Initialization

$$W_0 = 1 \tag{146}$$
$$L_0 = 0 \tag{147}$$

- For **iid sources**, interval refinement can be simplified

$$W_{n+1} = W_n \cdot p(s_n) \tag{148}$$
$$L_{n+1} = L_n + W_n \cdot c(s_n) \tag{149}$$

- For **Markov sources**: Conditional pmf $p(s_n|s_{n-1})$ and cmf $c(s_n|s_{n-1})$

$$W_{n+1} = W_n \cdot p(s_n|s_{n-1}) \tag{150}$$
$$L_{n+1} = L_n + W_n \cdot c(s_n|s_{n-1}) \tag{151}$$

- Non-stationary sources: Probabilities $p(\cdot)$ can be adapted during coding

Elias Coding Example: IID Source

- Example for an iid source for which an optimum Huffman code exists

symbol a_k	pmf $p(a_k)$	Huffman code	cmf $c(a_k)$
a_0='A'	$0.25 = 2^{-2}$	00	$0.00 = 0$
a_1='B'	$0.25 = 2^{-2}$	01	$0.25 = 2^{-2}$
a_2='C'	$0.50 = 2^{-1}$	1	$0.50 = 2^{-1}$

- Suppose we intend to send the symbol string $\boldsymbol{s}$ = "CABAC"
- Using the Huffman code, the bit string would be $\boldsymbol{b} = 10001001$ (8 bits)
- An alternative to Huffman coding is Elias coding
- Probability of the symbol string "CABAC" is given by

$$p(\boldsymbol{s}) = p('C') \cdot p('A') \cdot p('B') \cdot p('A') \cdot p('C') = \frac{1}{2}\frac{1}{4}\frac{1}{4}\frac{1}{4}\frac{1}{2} = \frac{1}{256}$$

- The size of the bit string is

$$K = \lceil -\log_2 p(\boldsymbol{s}) \rceil = 8 \text{ bits}$$

Encoding Algorithm for Elias Codes

Encoding algorithm:

1. Given is a sequence $\{s_0, \cdots, s_{N-1}\}$ of N symbols
2. Initialization of the iterative process by $W_0 = 1$, $L_0 = 0$
3. For each $n = 0, 1, \cdots, N-1$, determine the interval $\mathcal{I}_{n+1}$ by

$$\begin{aligned} W_{n+1} &= W_n \cdot p(s_n | s_0, \cdots, s_{n-1}) \\ L_{n+1} &= L_n + W_n \cdot c(s_n | s_0, \cdots, s_{n-1}) \end{aligned}$$

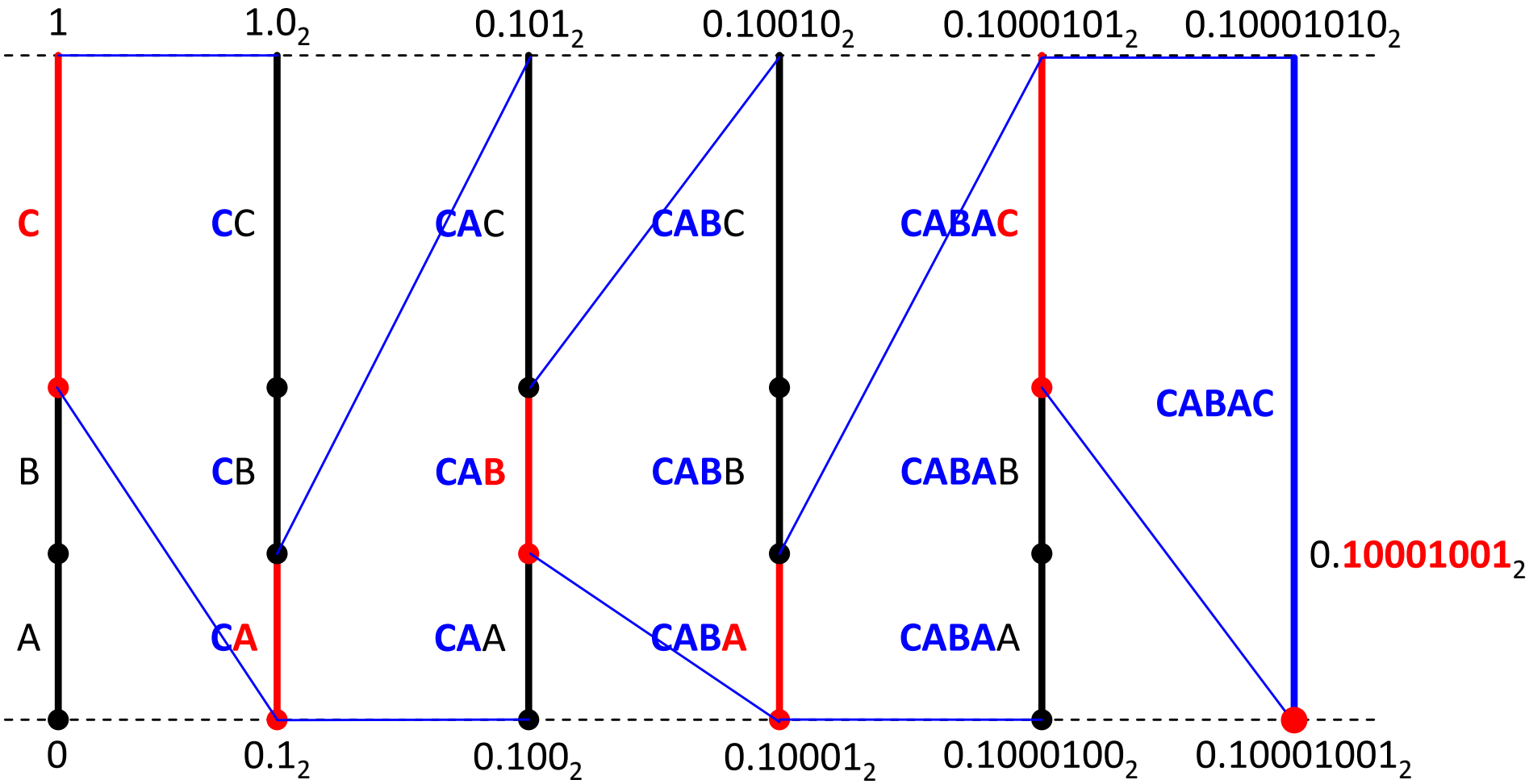
4. Determine the codeword length by $K = \lceil -\log_2 W_N \rceil$
5. Transmit the codeword $\boldsymbol{b}^{(K)}$ of K bits that represents the fractional part of $v = \lceil L_N \, 2^K \rceil \, 2^{-K}$

Example for Elias Encoding

s_0='C'	s_1='A'	s_2='B'
$W_1 = W_0 \cdot p(\text{'C'}) = 1 \cdot 2^{-1} = 2^{-1} = (0.1)_2$	$W_2 = W_1 \cdot p(\text{'A'}) = 2^{-1} \cdot 2^{-2} = 2^{-3} = (0.001)_2$	$W_3 = W_2 \cdot p(\text{'B'}) = 2^{-3} \cdot 2^{-2} = 2^{-5} = (0.00001)_2$
$L_1 = L_0 + W_0 \cdot c(\text{'C'}) = 0 + 1 \cdot 2^{-1} = 2^{-1} = (0.1)_2$	$L_2 = L_1 + W_1 \cdot c(\text{'A'}) = 2^{-1} + 2^{-1} \cdot 0 = 2^{-1} = (0.100)_2$	$L_3 = L_2 + W_2 \cdot c(\text{'B'}) = 2^{-1} + 2^{-3} \cdot 2^{-2} = 2^{-1} + 2^{-5} = (0.10001)_2$

s_3='A'	s_4='C'	termination
$W_4 = W_3 \cdot p(\text{'A'}) = 2^{-5} \cdot 2^{-2} = 2^{-7} = (0.0000001)_2$	$W_5 = W_4 \cdot p(\text{'C'}) = 2^{-7} \cdot 2^{-1} = 2^{-8} = (0.00000001)_2$	$K = \lceil -\log_2 W_5 \rceil = 8$ $v = \lceil L_5 \, 2^K \rceil \, 2^{-K} = 2^{-1} + 2^{-5} + 2^{-8}$ $\boldsymbol{b} = \text{'}\mathbf{10001001}\text{'}$
$L_4 = L_3 + W_3 \cdot c(\text{'A'}) = (2^{-1} + 2^{-5}) + 2^{-5} \cdot 0 = 2^{-1} + 2^{-5} = (0.1000100)_2$	$L_5 = L_4 + W_4 \cdot c(\text{'C'}) = (2^{-1} + 2^{-5}) + 2^{-7} \cdot 2^{-1} = 2^{-1} + 2^{-5} + 2^{-8} = (0.10001001)_2$	

Illustration of Iteration



Decoding Algorithm for Elias Codes

Decoding algorithm:

1. Given is the number N of symbols to be decoded and a codeword $\boldsymbol{b}^{(K)} = \{b_0, \cdots, b_{K-1}\}$ of K_N bits
2. Determine the interval representative v according to

$$v = \sum_{i=0}^{K-1} b_i \, 2^{-i}$$

3. Initialization of the iterative process by $W_0 = 1$, $L_0 = 0$
4. For each $n = 0, 1, \cdots, N-1$, do the following:
   1. For each $a_i \in \mathcal{A}_n$, determine the interval $\mathcal{I}_{n+1}(a_i)$ by

$$\begin{aligned} W_{n+1}(a_i) &= W_n \cdot p(a_i | s_0, \ldots, s_{n-1}) \\ L_{n+1}(a_i) &= L_n + W_n \cdot c(a_i | s_0, \ldots, s_{n-1}) \end{aligned}$$

   2. Select the letter $a_i \in \mathcal{A}_n$ for which $v \in \mathcal{I}_{n+1}(a_i)$ and set $s_n = a_i$, $W_{n+1} = W_{n+1}(a_i)$, $L_{n+1} = L_{n+1}(a_i)$

Properties of Elias Codes

Efficiency

- Bounds on average codeword length are the same as for block Huffman codes
- Concatenations cannot always be decoded using iterative procedure
- Issue can be solved by adding one bit per message (see exercise)
- Block Huffman codes of same size N are optimal

$\Longrightarrow$ **No higher efficiency than block Huffman codes of same size**

Code construction

- Can iteratively construct the codeword for a given message
- Don't need to construct and store the entire code table
- Easy to incorporate adaptation of probabilities to source statistics

$\Longrightarrow$ **That's the advantage of Elias codes**

Are there any issues?

- Require extremely high precision arithmetic for calculating interval size and lower interval boundary for long messages
- Solution: **Approximation using fixed-precision integer arithmetic**

Arithmetic Coding

Elias coding

- Very efficient for long symbol sequences (if suitable probabilities are used)
- Simple codeword construction
- Do not need to store codeword tables
- Problem: Precision requirement for W_n and L_n

Arithmetic coding

- Fixed-precision variant of Elias coding
- Can be realized with standard precision integer arithmetic
- Loss in efficiency due to fixed-precision arithmetic is negligible
- Following approximations are used

  $\Longrightarrow$ Represent probabilities with fixed-precision integers

  $\Longrightarrow$ Represent interval width with fixed-precision integers

  $\Longrightarrow$ Output bits as soon as they cannot be modified in following steps

Quantization of Pmf and Cmf

- Represent pmfs $p(a)$ and cmfs $c(a)$ by V-bit integers $p_V(a)$ and $c_V(a)$

$$p(a) = p_V(a) \cdot 2^{-V} \tag{152}$$

$$c(a) = c_V(a) \cdot 2^{-V} = \sum_{a_i < a} p_V(a_i) \cdot 2^{-V} \tag{153}$$

- Following condition has to be fulfilled

$$\left(\sum_{\forall a_i} p_V(a_i) \right) \cdot 2^{-V} \leq 1 \tag{154}$$

- Simple (but not necessarily best) approach

$$p_V(a_i) = \left\lfloor p(a_i) \cdot 2^V \right\rfloor \tag{155}$$

Note: V must be so large that $p_V(a_i) > 0$ for all a_i

Quantization of Interval Width

- Observation: Elias code remains decodable if intervals are always nested

$$0 < W_{n+1} \leq W_n \cdot p(s_n) \tag{156}$$

$\Longrightarrow$ Rounding down of $W_n \cdot p(s_n)$ at each iteration (for fixed-precision rep.)

- Represent W_n by U-bit integer A_n and integer $z_n \geq U$

$$W_n = A_n \cdot 2^{-z_n} \tag{157}$$

- Initialization: Approximate $W_0 = 1$ by

$$A_0 = 2^U - 1 \qquad \text{and} \qquad z_0 = U \tag{158}$$

- Restriction for A_n

$$\underbrace{2^{U-1}}_{\text{max. precision}} \leq A_n < \underbrace{2^U}_{U\text{-bit integer}} \tag{159}$$

Rounding in Interval Refinement

- Representation of interval width

$$W_n = A_n \cdot 2^{-z_n} \qquad \text{with} \qquad 2^{U-1} \leq A_n < 2^U \tag{160}$$

- Interval refinement for arbitrary precision

$$\begin{aligned} W_{n+1} &= W_n \cdot p(s_n) \\ A_{n+1} \cdot 2^{-z_{n+1}} &= A_n \cdot 2^{-z_n} \cdot p_V(s_n) \cdot 2^{-V} \\ &= \left(A_n \cdot p_V(s_n) \cdot 2^{-y_{n+1}}\right) \cdot 2^{-z_n - V + y_{n+1}} \end{aligned} \tag{161}$$

- Interval refinement for fixed-precision arithmetic

$$A_{n+1} = \left\lfloor A_n \cdot p_V(s_n) \cdot 2^{-y_{n+1}} \right\rfloor \qquad \text{(simple right shift by } y_{n+1} \text{ bits)} \tag{162}$$

$$z_{n+1} = z_n + V - y_{n+1} \tag{163}$$

- Choose y_{n+1} so that $2^{U-1} \leq A_{n+1} < 2^U$ (some comparison operations)

$$y_{n+1} = \underbrace{\left\lceil \log_2(A_n \cdot p_V(s_n) + 1) \right\rceil}_{\text{pos. of most-sign. bit in } A_n p_V(s_n)} - U \tag{164}$$

Analysis of Binary Representations

- Binary representation of interval width $W_n = A_n \cdot 2^{-z_n}$:

$$W_n = 0.\overbrace{\underbrace{00000\cdots0}_{z_n - U \text{ bits}}\underbrace{1xx\cdots x}_{U \text{ bits}}}^{z_n \text{ bits}}000\cdots$$

- Binary representation of cmf $c(s_n) = c_V(s_n) \cdot 2^{-V}$:

$$c(s_n) = 0.\underbrace{xxx\cdots x}_{V \text{ bits}}000\cdots$$

- Binary representation of product $W_n \cdot c(s_n)$ (added to L_n in update):

$$W_n \cdot c(s_n) = 0.\overbrace{\underbrace{00000\cdots0}_{z_n - U \text{ bits}}\underbrace{xxx\cdots x}_{U+V \text{ bits}}}^{z_n + V \text{ bits}}000\cdots$$

Effect on Lower Interval Boundary

- Remember: Update of lower interval boundary $L_{n+1} = L_n + W_n \cdot c(s_n)$
- Binary representation of the product $W_n \cdot c(s_n)$:

$$W_n \cdot c(s_n) = 0.\overbrace{\underbrace{00000\cdots0}_{z_n - U \text{ bits}}\underbrace{xxx\cdots x}_{U+V \text{ bits}}}^{z_n+V \text{ bits}}000\cdots$$

- What is the effect on lower interval boundary

$$L_n = 0.\overbrace{\underbrace{aaaaa\cdots a}_{\substack{z_n - c_n - U \\ \textit{settled bits}}}\ \underbrace{0111111\cdots1}_{\substack{c_n \\ \textit{outstanding bits}}}}^{z_n - U \text{ bits}}\ \underbrace{xxxxx\cdots x}_{\substack{U+V \\ \textit{active bits}}}\ \underbrace{00000\cdots}_{\textit{trailing bits}}$$

  - Trailing bits: Equal to 0, but maybe changed later
  - Active bits: Directly modified by the update $L_{n+1} = L_n + W_n \cdot c(s_n)$
  - Outstanding bits: May be modified by a carry from the active bits
  - Settled bits: Not modified in any following interval update

Representation of Lower Interval Boundary

- Lower interval boundary L_n

$$L_n = 0.\overbrace{\underbrace{aaaaa\cdots a}_{z_n - c_n - U}\ \underbrace{0111111\cdots 1}_{c_n}}^{z_n - U \text{ bits}}\ \underbrace{xxxx\cdots x}_{U+V}\ \underbrace{00000\cdots}_{}$$

settled bits | *outstanding bits* | *active bits* | *trailing bits*

- Active bits can be represented by an $(U+V)$-bit integer B_n
- Outstanding bits can be represented by a counter c_n
- Settled bits are output as soon as they become settled
- Total number of bits to output is

$$K = \lceil -\log_2 W_N \rceil = z_n - \lfloor \log_2 A_N \rfloor = z_n - U + 1 \qquad (165)$$

- Termination of arithmetic coding
  $\Longrightarrow$ Output all outstanding bits
  $\Longrightarrow$ Output most significant bit of $(U+V)$-integer B_n

Overview of Process of Arithmetic Coding

Initialization:

- Initialize integer representation of interval width: $A_0 = 2^U - 1$
- Initialize $U + V$ active bits: $B_0 = 0$
- Initialize number of outstandig bits: $c_0 = 0$

Iterative coding (for $n = 0$ to $n = N - 1$):

- Calculate product $A^*_{n+1} = A_n \cdot p_V(s_n)$
- Determine bit shift parameter y_{n+1} (check first bit equal to "1" in A^*_{n+1})
- Update interval width: $A_{n+1} = A^*_{n+1} \gg y_{n+1}$
- Output settled bits
- Update active bits B_{n+1} and counter c_{n+1} for outstanding bits

Termination

- Output outstanding bits
- Output most significant bit of $(U + V)$-integer B_N

Example for Arithmetic Coding

- Consider iid process with symbol alphabet $\mathcal{A} = \{\mathsf{M}, \mathsf{I}, \mathsf{S}, \mathsf{P}\}$
- Marginal pmf given by $p(a_i) = \{1/11, 4/11, 4/11, 2/11\}$
- Consider arithmetic coding with $V = 4$ and $U = 4$
- Consider coding of symbol sequence "MISSISSIPPI"

- Preparation: Quantization of pmf (and cmf) with $V = 4$ bits

a_i	$p(a_i)$	$p(a_i) \cdot 2^4$	$p_V(a_i)$	$c_V(a_i)$
M	$1/11$	$16/11 \approx 1.45$	1	0
I	$4/11$	$64/11 \approx 5.82$	6	1
S	$4/11$	$64/11 \approx 5.82$	6	7
P	$2/11$	$32/11 \approx 2.91$	3	13

- Note: Quantized pmf $p_Q(a_n)$ fulfills the requirement $\sum p_Q(a_n) \leq 1$

$$\sum_{\forall a_i} p_Q(a_i) = \sum_{\forall a_i} p_V(a_i) \cdot 2^{-4} = 16 \cdot 2^{-4} = 1$$

Example for Arithmetic Coding – Step 1

s_n	p_V	c_V	parameter updates & output
initialization			$A_0 = 15 =$ '**1111**' $c_0 = 0$ ('') $B_0 = 0 =$ '**0000 0000**' bitstream = **“”**
“M”	1	0	$A_0 \cdot p_V = 15 \cdot 1 = 15 =$ '**0000 1111**' $B_0 + A_0 \cdot c_V = 0 + 15 \cdot 0 = 0 =$ '**0 0000 0000**' $y_1 = 0$
			$A_1 =$ '**1111**' $= 15$ $c_1 = 1$ ('**0**') $B_1 =$ '**0000 0000**' $= 0$ **output** = **“000”** bitstream = **“000”**

Example for Arithmetic Coding – Step 2

s_n	p_V	c_V	parameter updates & output
after step 1			$A_1 = 15 = \text{'}\mathbf{1111}\text{'}$ $c_1 = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_1 = 0 = \text{'}\mathbf{0000\ 0000}\text{'}$ bitstream = **"000"**
"I"	6	1	$A_1 \cdot p_V = 15 \cdot 6 = 90 = \text{'}\mathbf{0101\ 1010}\text{'}$ $B_1 + A_1 \cdot c_V = 0 + 15 \cdot 1 = 15 = \text{'}\mathbf{0\ 0000\ 1111}\text{'}$ $y_2 = 3$
			$A_2 = \text{'}\mathbf{1011}\text{'} = 11$ $c_2 = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_2 = \text{'}\mathbf{0001\ 1110}\text{'} = 30$ **output** = **"0"** bitstream = **"0000"**

Example for Arithmetic Coding – Step 3

s_n	p_V c_V	parameter updates & output
after step 2		$A_2 = 11 =$ '**1011**' $c_2 = 1$ ('**0**') $B_2 = 30 =$ '**0001 1110**' bitstream = **"0000"**
"S"	6 7	$A_2 \cdot p_V = 11 \cdot 6 = 66 =$ '**0100 0010**' $B_2 + A_2 \cdot c_V = 30 + 11 \cdot 7 = 107 =$ '**0 0110 1011**' $y_3 = 3$
		$A_3 =$ '**1000**' $= 8$ $c_3 = 1$ ('**0**') $B_3 =$ '**1101 0110**' $= 214$ **output** = **"0"** bitstream = **"0000 0"**

Example for Arithmetic Coding – Step 4

s_n	p_V c_V	parameter updates & output
after step 3		$A_3 = 8 =$ '**1000**' $c_3 = 1$ ('**0**') $B_3 = 214 =$ '**1101 0110**' bitstream = **"0000 0"**
"S"	6 7	$A_3 \cdot p_V = 8 \cdot 6 = 48 =$ '**0011 0000**' $B_3 + A_3 \cdot c_V = 214 + 8 \cdot 7 = 270 =$ '**1 0000 1110**' $y_4 = 2$
		$A_4 =$ '**1100**' $= 12$ $c_4 = 1$ ('**0**') $B_4 =$ '**0011 1000**' $= 56$ **output** = **"10"** (invert outstanding bits) bitstream = **"0000 010"**

Example for Arithmetic Coding – Step 5

s_n	p_V c_V	parameter updates & output
after step 4		$A_4 = 12 =$ '**1100**' $c_4 = 1$ ('**0**') $B_4 = 56 =$ '**0011 1000**' bitstream = **"0000 010"**
"l"	6 1	$A_4 \cdot p_V = 12 \cdot 6 = 72 =$ '**0100 1000**' $B_4 + A_4 \cdot c_V = 56 + 12 \cdot 1 = 68 =$ '**0 0100 0100**' $y_5 = 3$
		$A_5 =$ '**1001**' $= 9$ $c_5 = 1$ ('**0**') $B_5 =$ '**1000 1000**' $= 136$ **output** = **"0"** bitstream = **"0000 0100"**

Example for Arithmetic Coding – Step 6

s_n	p_V c_V	parameter updates & output
after step 5		$A_5 = 9 =$ '**1001**' $c_5 = 1$ ('**0**') $B_5 = 136 =$ '**1000 1000**' bitstream = **"0000 0100"**
"S"	6 7	$A_5 \cdot p_V = 9 \cdot 6 = 54 =$ '**0011 0110**' $B_5 + A_5 \cdot c_V = 136 + 9 \cdot 7 = 68 =$ '**0 1100 0111**' $y_6 = 2$
		$A_6 =$ '**1101**' $= 13$ $c_6 = 3$ ('**011**') $B_6 =$ '**0001 1100**' $= 28$ **output** = **""** bitstream = **"0000 0100"**

Example for Arithmetic Coding – Step 7

s_n	p_V c_V	parameter updates & output
after step 6		$A_6 = 13 =$ '**1101**' $c_6 = 3$ ('**011**') $B_6 = 28 =$ '**0001 1100**' bitstream = **"0000 0100"**
"S"	6 7	$A_6 \cdot p_V = 13 \cdot 6 = 78 =$ '**0100 1110**' $B_6 + A_6 \cdot c_V = 28 + 13 \cdot 7 = 119 =$ '**0 0111 0111**' $y_7 = 3$
		$A_7 =$ '**1001**' $= 9$ $c_7 = 1$ ('**0**') $B_7 =$ '**1110 1110**' $= 238$ **output** = **"011"** bitstream = **"0000 0100 011"**

Example for Arithmetic Coding – Step 8

s_n	p_V c_V	parameter updates & output
after step 7		$A_7 = 9 = \text{'}\mathbf{1001}\text{'}$ $c_7 = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_7 = 238 = \text{'}\mathbf{1110\ 1110}\text{'}$ bitstream = **"0000 0100 011"**
"l"	6 1	$A_7 \cdot p_V = 9 \cdot 6 = 54 = \text{'}\mathbf{0011\ 0110}\text{'}$ $B_7 + A_7 \cdot c_V = 238 + 9 \cdot 1 = 247 = \text{'}\mathbf{0\ 1111\ 0111}\text{'}$ $y_8 = 2$
		$A_8 = \text{'}\mathbf{1101}\text{'} = 13$ $c_8 = 3 \quad (\text{'}\mathbf{011}\text{'})$ $B_8 = \text{'}\mathbf{1101\ 1100}\text{'} = 220$ **output** = **""** bitstream = **"0000 0100 011"**

Example for Arithmetic Coding – Step 9

s_n	p_V	c_V	parameter updates & output
after step 8			$A_8 = 13 = \text{'}\mathbf{1101}\text{'}$ $c_8 = 3 \quad (\text{'}\mathbf{011}\text{'})$ $B_8 = 220 = \text{'}\mathbf{1101\ 1100}\text{'}$ bitstream = **"0000 0100 011"**
"P"	3	13	$A_8 \cdot p_V = 13 \cdot 3 = 39 = \text{'}\mathbf{0010\ 0111}\text{'}$ $B_8 + A_8 \cdot c_V = 220 + 13 \cdot 13 = 389 = \text{'}\mathbf{1\ 1000\ 0101}\text{'}$ $y_9 = 2$
			$A_9 = \text{'}\mathbf{1001}\text{'} = 9$ $c_9 = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_9 = \text{'}\mathbf{0001\ 0100}\text{'} = 20$ **output** = **"1001"** (invert outstanding bits) bitstream = **"0000 0100 0111 001"**

Example for Arithmetic Coding – Step 10

s_n	p_V c_V	parameter updates & output
after step 9		$A_9 = 9 = \text{'}\mathbf{1001}\text{'}$ $c_9 = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_9 = 20 = \text{'}\mathbf{0001\ 0100}\text{'}$ bitstream = **"0000 0100 0111 001"**
"P"	3 13	$A_9 \cdot p_V = 9 \cdot 3 = 27 = \text{'}\mathbf{0001\ 1011}\text{'}$ $B_9 + A_9 \cdot c_V = 20 + 9 \cdot 13 = 137 = \text{'}\mathbf{0\ 1000\ 1001}\text{'}$ $y_{10} = 1$
		$A_{10} = \text{'}\mathbf{1101}\text{'} = 13$ $c_{10} = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_{10} = \text{'}\mathbf{0100\ 1000}\text{'} = 72$ **output** = **"010"** bitstream = **"0000 0100 0111 0010 10"**

Example for Arithmetic Coding – Step 11

s_n	p_V c_V	parameter updates & output
after step 10		$A_{10} = 13 = \text{'}\mathbf{1101}\text{'}$ $c_{10} = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_{10} = 72 = \text{'}\mathbf{0100\ 1000}\text{'}$ bitstream = **"0000 0100 0111 0010 10"**
"l"	6 1	$A_{10} \cdot p_V = 13 \cdot 6 = 78 = \text{'}\mathbf{0100\ 1110}\text{'}$ $B_{10} + A_{10} \cdot c_V = 72 + 13 \cdot 1 = 85 = \text{'}\mathbf{0\ 0101\ 0101}\text{'}$ $y_{11} = 3$
		$A_{11} = \text{'}\mathbf{1001}\text{'} = 9$ $c_{11} = 1 \quad (\text{'}\mathbf{0}\text{'})$ $B_{11} = \text{'}\mathbf{1010\ 1010}\text{'} = 170$ **output = "0"** bitstream = **"0000 0100 0111 0010 100"**

Example for Arithmetic Coding – Termination

s_n	p_V c_V	parameter updates & output
after step 11		$A_{11} = 9 =$ '**1001**' $c_{11} = 1$ ('**0**') $B_{11} = 170 =$ '**1010 1010**' bitstream = **"0000 0100 0111 0010 100"**
termination		**output** = **"01"** (outstanding + first bit of B_{11}) bitstream = **"0000 0100 0111 0010 1000 1"**

- Transmitted bitstream: "0000 0100 0111 0010 1000 1"
- Number of transmitted bits: $K = 21$

Example for Arithmetic Coding – Efficiency

- Number of written bits using arithmetic coding: $K_{AC} = 21$

- Number of bits for Elias coding

$$\begin{aligned} K_{EC} &= \Big\lceil -\log_2 p(\text{“MISSISSIPPI”}) \Big\rceil \\ &= \left\lceil -\log_2 \left(\frac{1}{11} \cdot \frac{4}{11} \cdot \frac{4}{11} \cdot \frac{4}{11} \cdot \frac{4}{11} \cdot \frac{4}{11} \cdot \frac{4}{11} \cdot \frac{4}{11} \cdot \frac{2}{11} \cdot \frac{2}{11} \cdot \frac{4}{11} \right) \right\rceil \\ &= \big\lceil -\log_2 (0.000\ 000\ 918\ \cdots) \big\rceil = \big\lceil 20.053 \cdots \big\rceil = 21 \end{aligned}$$

- Scalar Huffman coding

a_i	$p(a_i)$	codeword
M	1/11	**000**
I	4/11	**01**
S	4/11	**1**
P	2/11	**001**

- Bitstream:
  “**0000 1110 1110 1001 0010 1**”

- Number of written bits: $K_{HC} = 21$

Efficiency of Arithmetic Coding

- Excess rate due to down rounding of interval width

$$\Delta\ell = \left\lceil -\log_2 W_N \right\rceil - \left\lceil -\log_2 p(\boldsymbol{s}) \right\rceil < 1 + \log_2 \frac{p(\boldsymbol{s})}{W_N} \tag{166}$$

- Upper bound for increase in codeword length per symbol of arithmetic coding relative to infinite precision Elias coding

$$\Delta\bar{\ell} < \frac{1}{N} + \log_2 \left(1 + 2^{1-U}\right) - \log_2 \left(1 - \frac{2^{-V}}{p_{\min}}\right) \tag{167}$$

  For a derivation see WIEGAND, SCHWARZ, page 51-52

- Example:
  - number of coded symbols $N = 1000$,
  - precision for representing probabilities $V = 16$,
  - precision for representing intervals $U = 12$,
  - minimum probablity $p_{\min} = 0.02$

  $\Longrightarrow$ Increase in codeword length is less than 0.003 bit per symbol

Binary Arithmetic Coding

- Complexity reduction: Most popular type of arithmetic coding
  - MQ-coder in JPEG 2000
  - M-coder in H.264/AVC and H.265/HEVC
- Binarization of $S \in \{a_0, a_1, \ldots, a_{M-1}\}$ produces $C \in \{0, 1\}$
- Any prefix code can be used for binarization
- Example in table: Unary truncated binarization

S_n	number of bins B	C_0	C_1	C_2	$\cdots$	C_{M-2}	C_{M-1}
a_0	1	1					
a_1	2	0	1				
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$			
a_{M-2}	$M-2$	0	0	0	$\cdots$	0	1
a_{M-1}	$M-2$	0	0	0	$\cdots$	0	0

- Entropy and efficiency of coding unchanged due to binarization $\boldsymbol{S} \mapsto \boldsymbol{C}$

$$H(\boldsymbol{S}) = E\{-\log_2 p(\boldsymbol{S})\} = E\{-\log_2 p(\boldsymbol{C})\} = H(\boldsymbol{C})$$

Comparison of Lossless Coding Techniques

Experimental determination of average codeword length

- Coding of 1 000 000 realizations of our example stationary Markov source
- Determine average codeword length for sequences of 1 to 1000 symbols

Tested lossless coding techniques

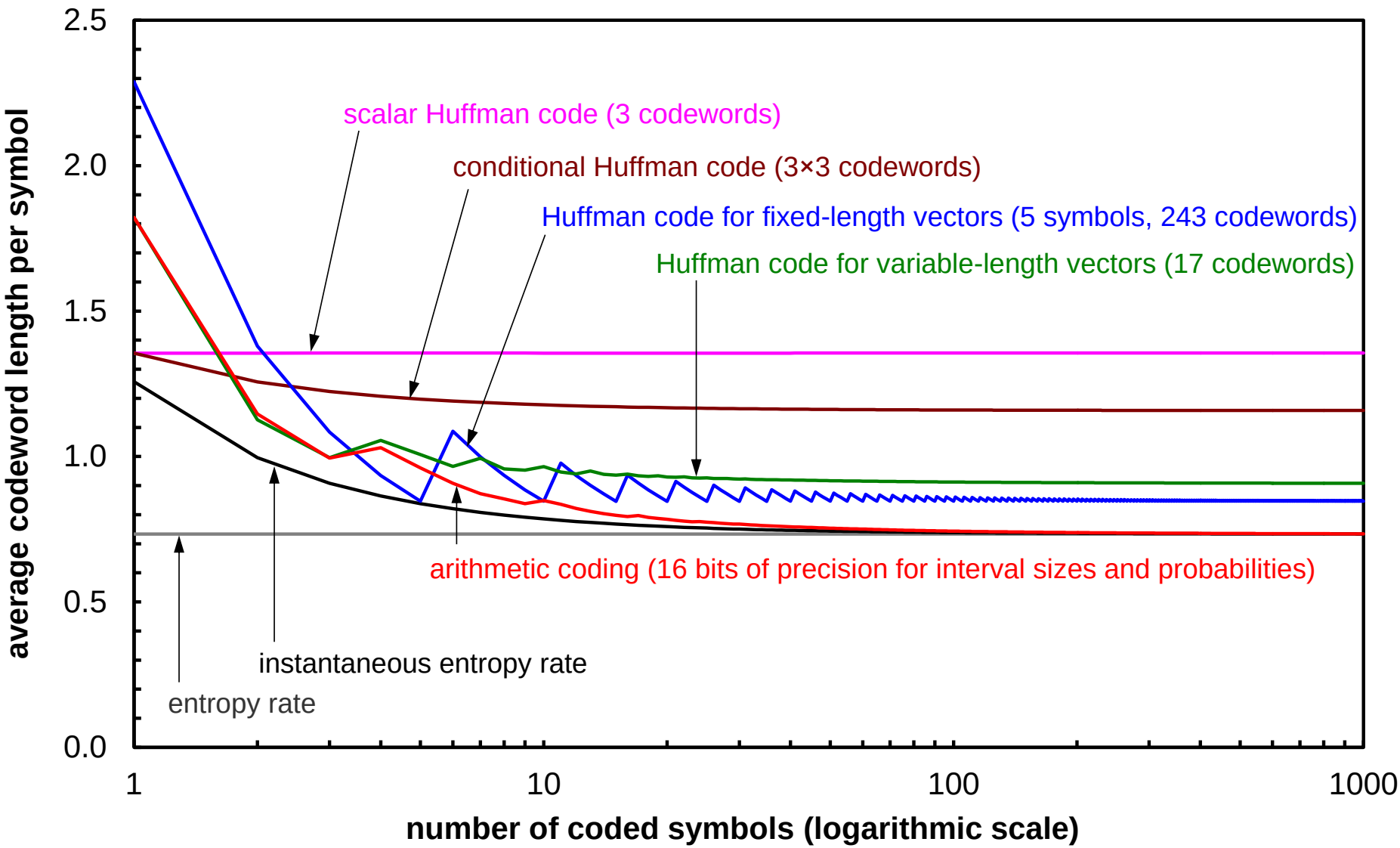
- Scalar Huffman coding (3 codewords)
- Conditional Huffman coding (9 codewords)
- Block Huffman coding of 5 symbols (243 codewords)
- Huffman coding for variable-length vectors (17 codewords)
- Arithmetic coding with $U = V = 16$

Bounds for lossless coding

- Marginal entropy $H(S)$ for coding of a single symbol
- Entropy rate $\bar{H}(\boldsymbol{S})$ for coding of infinite many symbols
- Instantaneous entropy rate $\bar{H}_{\text{inst}}(\boldsymbol{S}, L)$ for coding L symbols

$$\bar{H}_{\text{inst}}(\boldsymbol{S}, L) = \frac{1}{L} H(S_0, S_1, \ldots, S_{L-1}) \tag{168}$$

Comparison of Lossless Coding Techniques



Conditional and Adaptive Codes

- Question: How can we efficiently code sources with memory and/or with varying statistics
- Conditional Huffman coding (adaptive for instationary sources)
  - The resulting number of code tables is often too large in practice
  - Adaptation of Huffman code tables is often considered as to complex
- Block Huffman coding (adaptive for instationary sources)
  - Code tables are typically too large to be used in practice
  - Adaptation of Huffman code tables is often considered as to complex
- Conditional and adaptive arithmetic coding
  - Easy to incorporate conditional probabilities as well as varying probabilities
  - In adaptive arithmetic coding, probabilities $p(a_k)$ are estimated/adapted simultaneously at encoder and decoder
  - Statistical dependencies can be exploited using so-called context modeling techniques: Conditional probabilities $p(a_k|\boldsymbol{z}_k)$ with $\boldsymbol{z}_k$ being a context/state that is simultaneously computed at encoder and decoder based on already transmitted symbols

Forward and Backward Adaptation

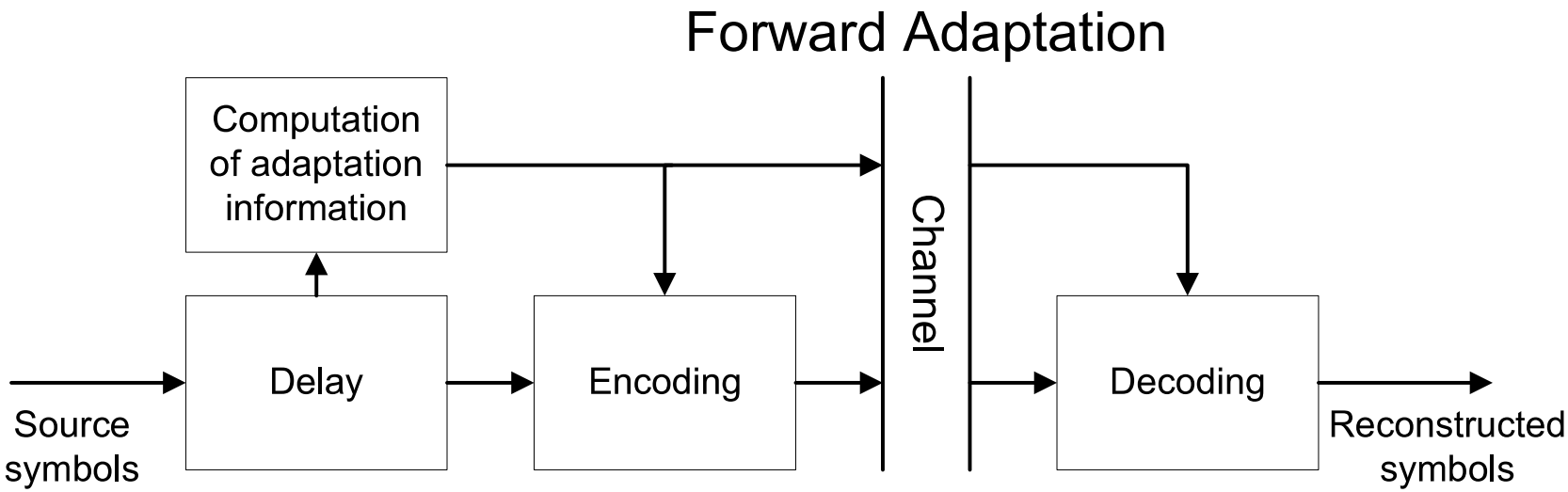
The two basic approaches for adaptation are

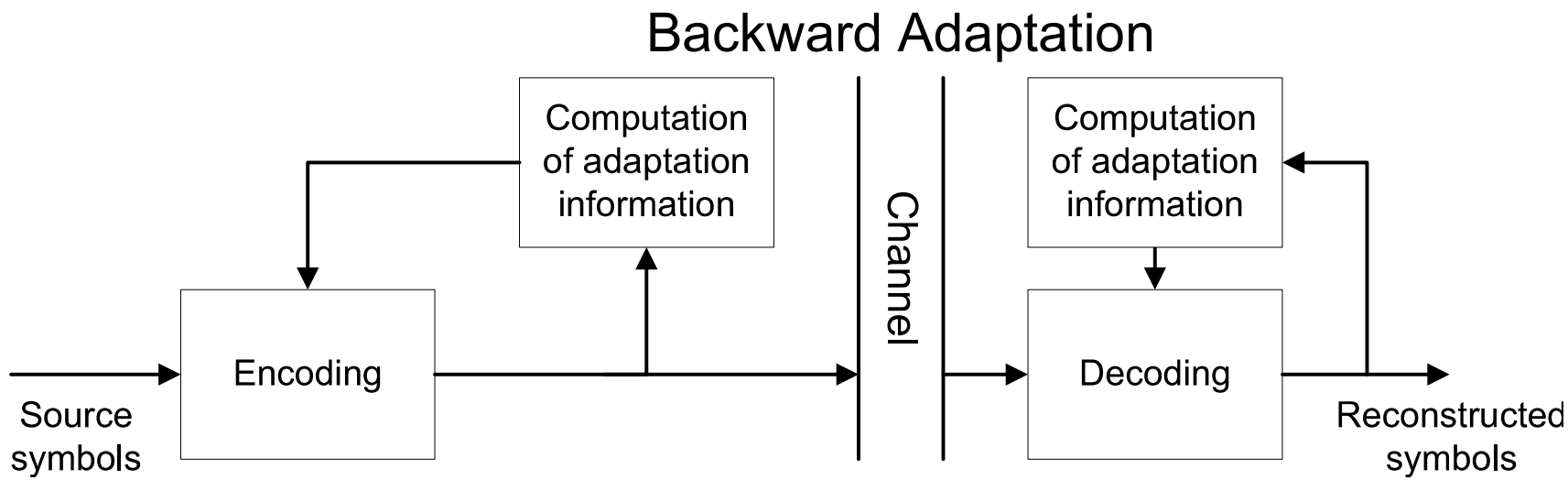
- **Forward adaptation**:
  - Gather statistics for a large enough block of source symbols
  - Transmit adaptation signal to decoder as side information
  - Disadvantage: Increased bit rate due to side information
- **Backward adaptation**:
  - Gather statistics simultaneously at coder and decoder
  - Drawback: Error resilience

With today's packet-switched transmission systems, an efficient combination of the two adaptation approaches can be achieved:

1. Gather statistics for the entire packet and provide initialization of entropy code at the beginning of the packet
2. Conduct backwards adaptation for each symbol inside the packet in order to minimize the size of the packet

Forward and Backward Adaptation





Chapter Summary

Uniquely decodable codes and bounds for lossless coding

- Kraft inequality, prefix codes
- Entropy, conditional entropy, block entropy
- Entropy rate, instantaneous entropy rate

Huffman codes for scalars and vectors (optimal prefix codes)

- Efficient and simple entropy coding method
- Needs a code table
- Can be inefficient for certain probabilities
- Difficult to use for sources with memory or time-varying statistics

Arithmetic coding (fixed-precision variant of Elias coding)

- Universal method for encoding strings of symbols
- Does not need a code table, but a table for storing probabilities
- Approaches entropy for long symbol sequences
- Well suited for exploiting statistical dependencies and coding of instationary sources (probability updates)

Exercise 9 – Part 1/2

Given is a Bernoulli process $\mathbf{X} = \{X_n\}$ with the alphabet $\mathcal{A} = \{a, b\}$ and the pmf $p_X(a) = 1/4$ and $p_X(b) = 3/4$.

(a) Consider Elias coding and derive the codeword for the symbol sequence "$abba$" using the iterative encoding procedure.

(b) Develop the complete code for an Elias coding of 3 symbols (i.e., determine the codewords for all symbol sequences that consist of 3 symbols).

Determine the average codeword length per symbol and compare it to the entropy rate and the average codeword length per symbol for a joint Huffman code for sequences of 3 symbols.

Is the Elias code more efficient than the Huffman code for the same number of jointly coded symbols?

(c) Decode the 3-symbol sequence $\{s_0, s_1, s_2\}$ represented by the bit string "100" using the iterative Elias decoding algorithm.

Exercise 9 – Part 2/2

(d) Consider the case in which the developed Elias code is used for coding multiple 3-symbol sequences. The codewords for the 3-symbol sequences are concatenated. Given is a bit string "10011100".

Decode the symbol sequence using the developed code table.

Decode the first three symbols (i.e., the first 3-symbol sequence) using the iterative decoding algorithm.

What do you observe?

(e) How many bits have to be used for a codeword in order to make an Elias code uniquely decodable using the iterative decoding algorithm for a sequence of codewords.

Derive a lower and upper bound for the average codeword length per symbol for the corresponding Elias code if a codeword is generated for a sequences of N symbols.

Exercise 10 – Part 1/3 (Optional Self Study)

Given is a discrete iid process $\mathbf{X} = \{X_n\}$ with the symbol alphabet $\mathcal{A} = \{\text{'M'}, \text{'I'}, \text{'S'}, \text{'P'}\}$ and the pmf

$$p_X(x) = \begin{cases} 0.1 & : \quad x = \text{'M'} \\ 0.3 & : \quad x = \text{'I'} \\ 0.4 & : \quad x = \text{'S'} \\ 0.2 & : \quad x = \text{'P'} \end{cases}$$

Consider binary arithmetic coding of the given source.

(a) Use a fixed-length code for binarizing the given source $\mathbf{X}$.

Verify on the given example that binarization does not have any impact on the coding efficiency (assuming a successive coding that achieves the entropy rate).

What binarization schemes can be used in the context of binary arithmetic coding?

Show that binarization does not change the lower bound for the average codeword length per symbol.

Exercise 10 – Part 2/3 (Optional Self Study)

(b) For arithmetic coding, the probability masses have to be represented with finite precision. Round the pmfs for the binary symbols to a precision of $V = 4$ bit.

What conditions need to be fulfilled for the rounded version of a pmf? Are these conditions fulfilled for the example?

Investigate the impact on the average codeword length per symbol of the given source $\mathbf{X}$ (assuming that the following arithmetic coding process does not have any negative impact on the coding efficiency).

(c) For arithmetic coding, the interval width has to be represented with finite precision.

Show that, for a coding of N symbols, the corresponding increase in average codeword length per arithmetically coded symbol is less than $1/N + \log_2(1 + 2^{1-U})$ bits, if U is the number of bits used for representing the interval width.

Determine the minimum precision U that is required to guarantee that the coding efficiency loss due to rounding the interval width is less than 0.1% when more than 10000 symbols X are transmitted.

Exercise 10 – Part 3/3 (Optional Self Study)

(d) Consider arithmetic coding that uses U bits for representing the interval width and V bits for representing the probability masses.

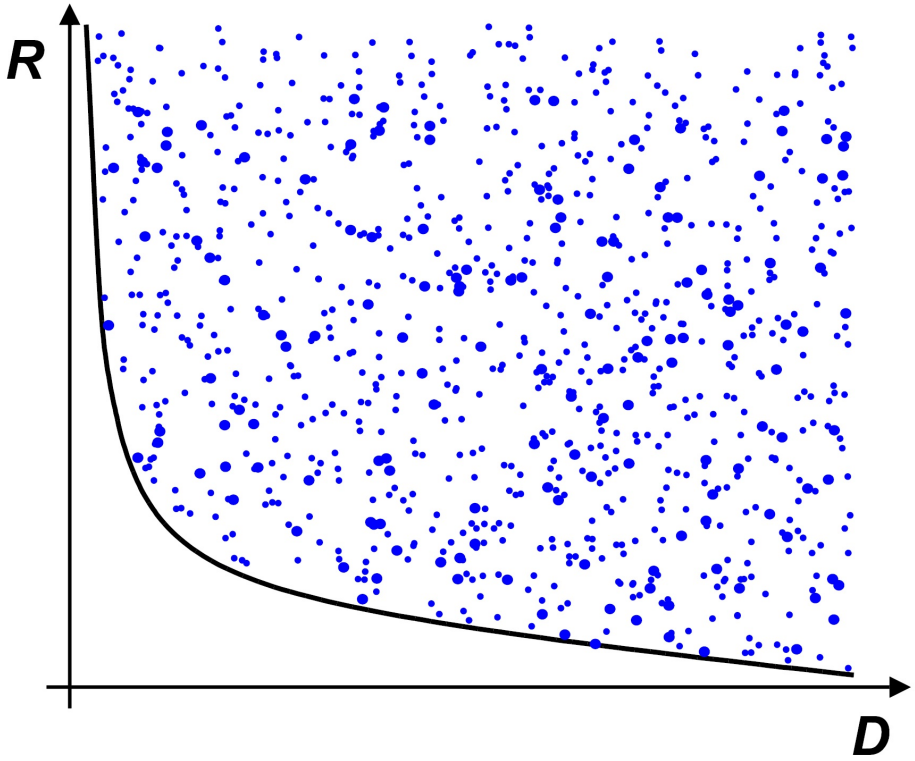
Show that the lower interval boundaries can be represented by a counter and an integer value of $U + V$ bits.

(e) Consider binary arithmetic coding for the given source $\mathbf{X}$ with fixed-length binarization, $V = 4$ bits for representing the probability masses, and $U = 4$ bits for representing the interval width.

Generate the arithmetic codeword for the symbol sequence "MISS".

Compare the length of the arithmetic codeword with the length of the codeword that would be generated by Elias coding.

Rate-Distortion Theory



Outline

Part I: Source Coding Fundamentals

- Probability, Random Variables and Random Processes
- Lossless Source Coding
- **Rate-Distortion Theory**
  - Operational Rate-Distortion Function
  - Information Rate-Distortion Function
  - Shannon Lower Bound
  - Rate-Distortion Function for Gaussian Sources
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

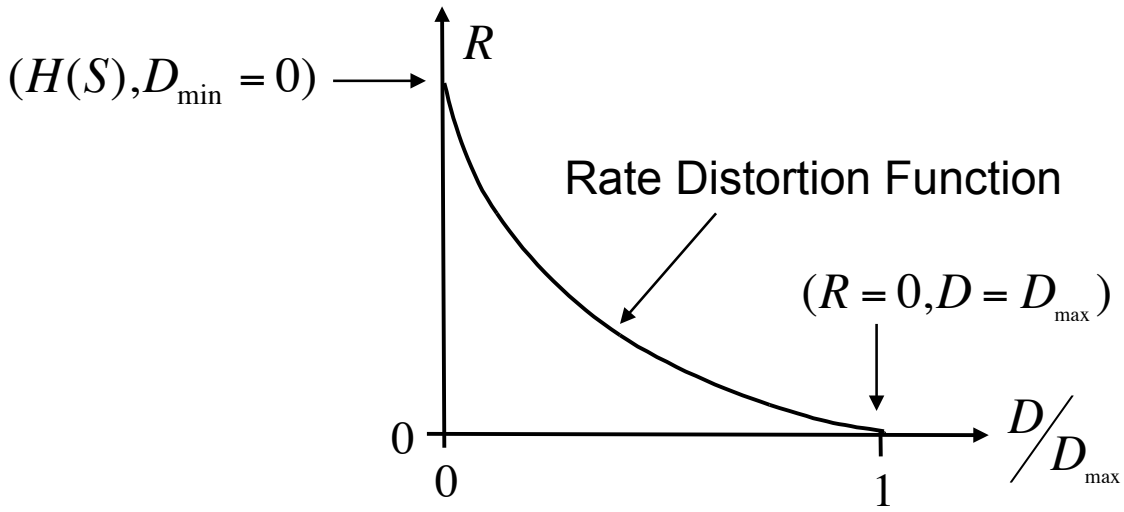
Rate-Distortion Theory – Motivation

Lossy coding: Decoded signal is an approximation of original

Rate-distortion theory: Information theoretical bounds for lossy compression

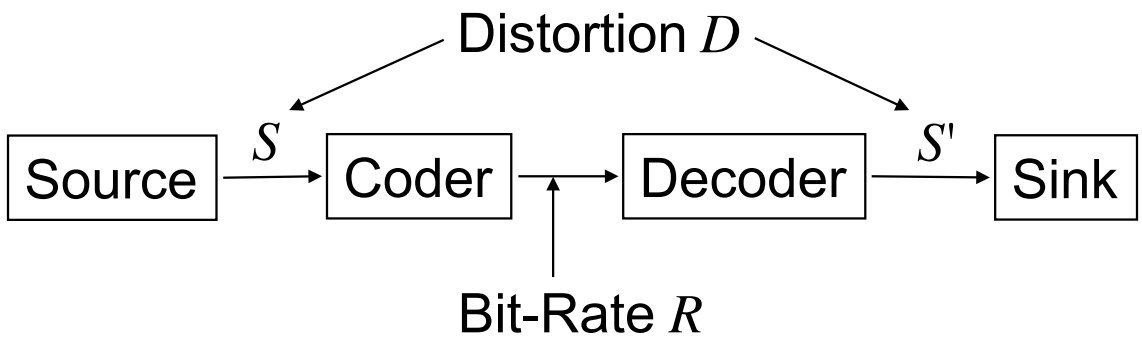
- Results are obtained without consideration of a specific coding method
- Goal of rate-distortion theory is to calculate the minimum transmission bit rate for a given distortion and source

Example for a rate-distortion function of a discrete iid source



Transmission System and Variables

Transmission system

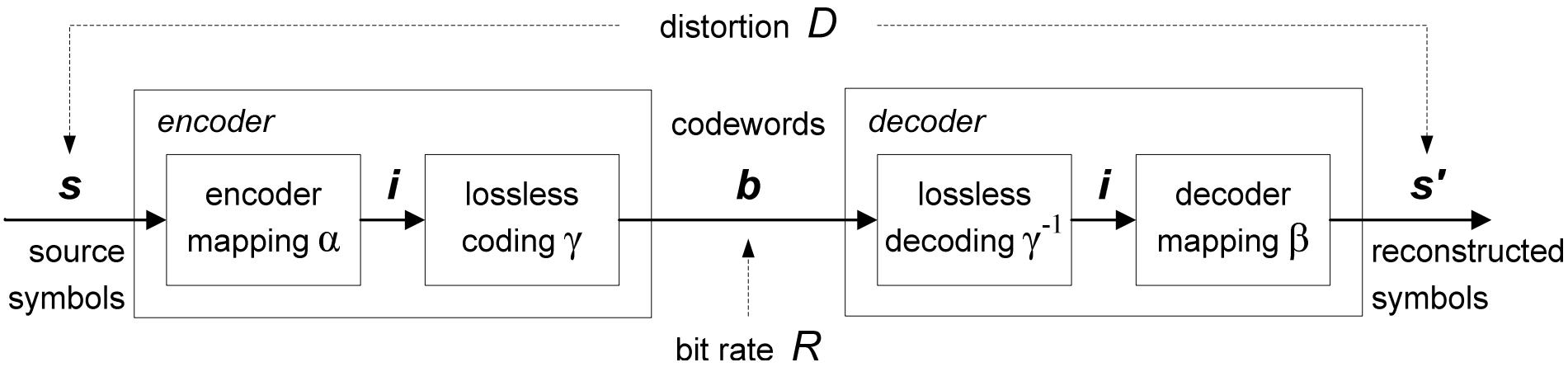


- Derivation in two steps
  - Define S, S', coder/decoder, distortion D and rate R
  - Establish a functional relationship between S, S', D, and R

- For two types of random variables
  - Discrete random variables
  - Continuous-amplitude random variables (Gaussian, Laplacian, etc.)

General Structure of Lossy Source Codecs

- **Encoder**:
  - Irreversible encoder mapping $\alpha : \boldsymbol{s} \rightarrow \boldsymbol{i}$
  - Lossless mapping $\gamma : \boldsymbol{i} \rightarrow \boldsymbol{b}$
- **Decoder**:
  - Lossless mapping $\gamma^{-1} : \boldsymbol{b} \rightarrow \boldsymbol{i}$
  - Decoder mapping $\beta : \boldsymbol{i} \rightarrow \boldsymbol{s'}$



Source Codes

- A **source code** $Q = (\alpha, \beta, \gamma)$ is given by an encoder mapping α, a decoder mapping β and a lossless mapping γ
- Special case: N-dimensional block source code $Q_N = \{\alpha_N, \beta_N, \gamma_N\}$
  - Blocks of N consecutive input samples are independently coded
  - Each block of input samples $\boldsymbol{s}^{(N)} = \{s_0, \cdots, s_{N-1}\}$ is mapped to a vector of K quantization indexes

$$\boldsymbol{i}^{(K)} = \alpha_N(\boldsymbol{s}^{(N)}) \tag{169}$$

  - Resulting vector of indexes $\boldsymbol{i}^{(N)}$ is converted into a bit sequence

$$\boldsymbol{b}^{(\ell)} = \gamma_N(\boldsymbol{i}^{(K)}) = \gamma_N(\alpha_N(\boldsymbol{s}^{(N)})) \tag{170}$$

  - At decoder side, index vector is recovered

$$\boldsymbol{i}^{(K)} = \gamma_N^{-1}(\boldsymbol{b}^{(\ell)}) = \gamma_N^{-1}(\gamma_N(\boldsymbol{i}^{(K)})) \tag{171}$$

  - Index vector is mapped to a block of reconstructed samples $\boldsymbol{s'}^{(N)} = \{s'_0, \cdots, s'_{N-1}\}$

$$\boldsymbol{s'}^{(N)} = \beta_N(\boldsymbol{i}^{(K)}) = \beta_N(\alpha_N(\boldsymbol{s}^{(N)})) \tag{172}$$

Distortion

- **Distortion**: Measure of difference between a block of N input samples $\boldsymbol{s}^{(N)} = \{s_0, s_1, \ldots, s_{N-1}\}$ and the corresponding block of reconstructed samples $\boldsymbol{s'}^{(N)} = \{s'_0, s'_1, \ldots, s'_{N-1}\}$,

$$d_N\left(\boldsymbol{s}^{(N)}, \boldsymbol{s'}^{(N)}\right)$$

- Typically: **Additive distortion measures**

$$d_N(\boldsymbol{s}^{(N)}, \boldsymbol{s'}^{(N)}) = \frac{1}{N} \sum_{i=0}^{N-1} d_1(s_i, s'_i) \tag{173}$$

with the single symbol distortion $d_1(s, s') \geq 0$ (equality, if and only if $s = s'$)

- In this lecture: Mean squared error $d_1(s, s') = (s - s')^2$

$$d_N\left(\boldsymbol{s}^{(N)}, \boldsymbol{s'}^{(N)}\right) = \frac{1}{N} \sum_{i=0}^{N-1} d_1(s_i, s'_i) = \frac{1}{N} \sum_{i=0}^{N-1} (s_i - s'_i)^2 \tag{174}$$

Average Distortion for Source Codes

- Average distortion for a stationary random process $\boldsymbol{S} = \{S_n\}$ and an N-dimensional block source code $Q_N = \{\alpha_N, \beta_N, \gamma_N\}$

$$\delta(Q_N) = E\Big\{d_N\big(\boldsymbol{S}^{(N)},\, \beta_N(\alpha_N(\boldsymbol{S}^{(N)}))\big)\Big\} \tag{175}$$

$$= \int_{\mathcal{R}^N} f(\boldsymbol{s})\, d_N\big(\boldsymbol{s},\, \beta_N(\alpha_N(\boldsymbol{s}))\big)\, \mathrm{d}\boldsymbol{s} \tag{176}$$

- For arbitrary random process $\boldsymbol{S} = \{S_n\}$ and arbitrary code Q

$$\delta(Q) = \lim_{N\to\infty} E\Big\{d_N\big(\boldsymbol{S}^{(N)},\, \beta_N(\alpha_N(\boldsymbol{S}^{(N)}))\big)\Big\} \tag{177}$$

- For additive distortion measures (such as the MSE distortion)

$$\delta(Q) = \delta(S, S') = E\{d_1(S, S')\} = \int_s \int_{s'} f_{SS'}(s, s')\, d_1(s, s')\, \mathrm{d}s\, \mathrm{d}s' \tag{178}$$

Average Rate for Source Codes

- Average number of bits per input symbol ($|\cdot|$ denotes the number of bits)

$$r_N(\boldsymbol{s}^{(N)}) = \frac{1}{N} \left| \gamma_N(\alpha_N(\boldsymbol{s}^{(N)})) \right| \qquad \text{with} \qquad \boldsymbol{b}^{(\ell)} = \gamma_N(\alpha_N(\boldsymbol{s}^{(N)})) \tag{179}$$

- Stationary random process $\boldsymbol{S} = \{S_n\}$ and N-dimensional block source code $Q_N = \{\alpha_N, \beta_N, \gamma_N\}$

$$r(Q_N) = \frac{1}{N} E\Big\{ \left| \gamma_N(\alpha_N(\boldsymbol{S}^{(N)})) \right| \Big\} \tag{180}$$

$$= \frac{1}{N} \int_{\mathcal{R}^N} f(\boldsymbol{s}) \left| \gamma_N(\alpha_N(\boldsymbol{s})) \right| \mathrm{d}\boldsymbol{s} \tag{181}$$

- For arbitrary random process $\boldsymbol{S} = \{S_n\}$ and arbitrary code Q

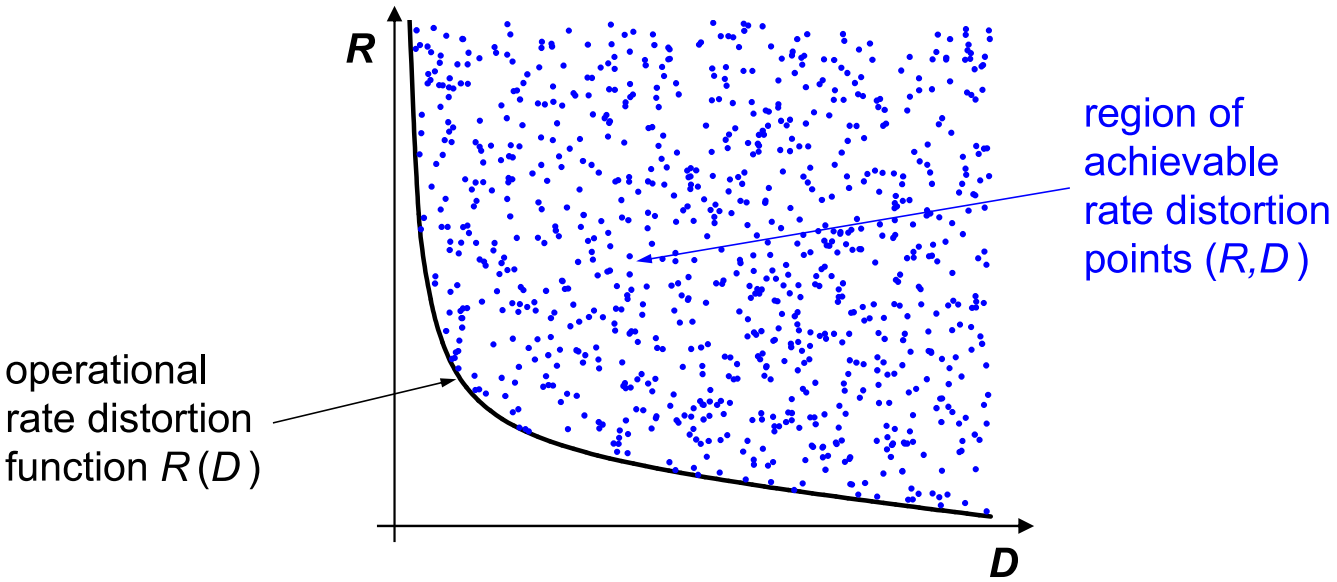
$$r(Q) = \lim_{N \to \infty} \frac{1}{N} E\Big\{ \left| \gamma_N(\alpha_N(\boldsymbol{S}^{(N)})) \right| \Big\} \tag{182}$$

Operational Rate-Distortion Function

For given source S:

- Each code Q is associated with a rate distortion point $(R, D) = (r(Q), \delta(Q))$
- A rate distortion point is **achievable**, if there exist a code Q such that $r(Q) \leq R$ and $\delta(Q) \leq D$
- The **operational rate-distortion function** $R(D)$ and its inverse, the **operational distortion-rate function** $D(R)$ are defined by

$$\boxed{R(D) = \inf_{Q:\, \delta(Q) \leq D} r(Q)} \qquad \boxed{D(R) = \inf_{Q:\, r(Q) \leq R} \delta(Q)} \tag{183}$$



Motivation for Information Rate-Distortion Function

Operational rate-distortion function

- Defined by

$$R(D) = \inf_{Q:\, \delta(Q) \leq D} r(Q) \tag{184}$$

- Specifies a fundamental performance bound for lossy source coding
- Difficulty to evaluate (minimization over all possible codes)

Information rate-distortion function

- Introduced by SHANNON in [Shannon 1948; Shannon1959]
- Obtain expression of rate-distortion bound that involves the distribution of the source using **mutual information**
- Show that information rate-distortion function is achievable

Mutual Information for Discrete Random Variables

- **Mutual information** between two discrete random variables A and B is defined by

$$\boxed{I(A;B) = H(A) - H(A|B)} \tag{185}$$

- Entropy $H(A)$ is a measure of uncertainty about random variable A
- Conditional entropy $H(A|B)$ is a measure of uncertainty about random variable A after observing random variable B
- Mutual information is a measure for the reduction of uncertainty about A due to the observation of B

$\Longrightarrow$ **Average amount of information that A carries about B**

- Mutual information for discrete random variables $A \in \mathcal{A}$ and $B \in \mathcal{B}$

$$I(A;B) = H(A) - H(A|B) = \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} p(a,b) \log_2 \frac{p(a|b)}{p(a)} \tag{186}$$

Mutual Information for Discrete Random Variables

- Mutual information rewritten using Bayes' rule

$$\begin{aligned} I(A;B) &= \sum_{a\in\mathcal{A}}\sum_{b\in\mathcal{B}} p(a,b)\log_2\frac{p(a|b)}{p(a)} \\ &= \sum_{a\in\mathcal{A}}\sum_{b\in\mathcal{B}} p(a,b)\log_2\frac{p(a,b)}{p(a)\,p(b)} \\ &= \sum_{a\in\mathcal{A}}\sum_{b\in\mathcal{B}} p(a,b)\log_2\frac{p(b|a)}{p(b)} \\ &= H(B) - H(B|A) \end{aligned} \tag{187}$$

- Mutual information between two random variables A and B represents the average amount of information that
  - the random variable A carries about the random variable B, or
  - the random variable B carries about the random variable A

Mutual Information for Discrete Random Vectors

- Mutual information between two random variables A and B

$$\begin{aligned} I(A;B) &= H(A) - H(A|B) \\ &= H(B) - H(B|A) \\ &= \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} p(a,b) \log_2 \frac{p(a,b)}{p(a)\,p(b)} \end{aligned} \tag{188}$$

- Extension to N-dimensional random vectors $\boldsymbol{A} = (A_0, A_1, \cdots, A_{N-1})^{\mathrm{T}}$ and $\boldsymbol{B} = (B_0, B_1, \cdots, B_{N-1})^{\mathrm{T}}$

$$\begin{aligned} I_N(\boldsymbol{A};\boldsymbol{B}) &= H_N(\boldsymbol{A}) - H_N(\boldsymbol{A}|\boldsymbol{B}) \\ &= H_N(\boldsymbol{B}) - H_N(\boldsymbol{B}|\boldsymbol{A}) \\ &= \sum_{\boldsymbol{a} \in \mathcal{A}^N} \sum_{\boldsymbol{b} \in \mathcal{B}^N} p(\boldsymbol{a},\boldsymbol{b}) \log_2 \frac{p(\boldsymbol{a},\boldsymbol{b})}{p(\boldsymbol{a})\,p(\boldsymbol{b})} \end{aligned} \tag{189}$$

Properties of Mutual Information for Discrete RV

- Mutual information between discrete random vectors $\boldsymbol{A}$ and $\boldsymbol{B}$

$$I_N(\boldsymbol{A};\boldsymbol{B}) = H_N(\boldsymbol{A}) - H_N(\boldsymbol{A}|\boldsymbol{B}) \tag{190}$$
$$= H_N(\boldsymbol{B}) - H_N(\boldsymbol{B}|\boldsymbol{A}) \tag{191}$$

- Since the conditional entropies are non-negative

$$I_N(\boldsymbol{A};\boldsymbol{B}) \leq H_N(\boldsymbol{A}) \tag{192}$$
$$I_N(\boldsymbol{A};\boldsymbol{B}) \leq H_N(\boldsymbol{B}) \tag{193}$$

- For independent random vectors $\boldsymbol{A}$ and $\boldsymbol{B}$

$$I_N(\boldsymbol{A};\boldsymbol{B}) = 0 \tag{194}$$

- If the random vector $\boldsymbol{B}$ is a deterministic function of the random vector $\boldsymbol{A}$,

$$\boldsymbol{B} = f(\boldsymbol{A}) \quad \Longrightarrow \quad I_N(\boldsymbol{A};\boldsymbol{B}) = H_N(\boldsymbol{B}) \tag{195}$$

Mutual Information for Coding of Discrete Sources

- Consider mutual information $I_N(\boldsymbol{S};\boldsymbol{S'})$ between a vector of N successive input samples $\boldsymbol{S}$ and the corresponding vector of N reconstructed samples $\boldsymbol{S'}$

$$\begin{aligned} I_N(\boldsymbol{S};\boldsymbol{S'}) &= H_N(\boldsymbol{S'}) - H_N(\boldsymbol{S'}|\boldsymbol{S}) \\ &\leq H_N(\boldsymbol{S'}) \end{aligned} \tag{196}$$

  where equality is achieved if and only if the vector $\boldsymbol{S'}$ of reconstructed samples is a deterministic function of the input vector $\boldsymbol{S}$

- Recall: Fundamental bound for lossless coding

$$r(Q) \geq \bar{H}(\boldsymbol{S'}) = \lim_{N\to\infty} \frac{H_N(\boldsymbol{S'})}{N} \tag{197}$$

- Rate of for code Q

$$\boxed{r(Q) \geq \lim_{N\to\infty} \frac{H_N(\boldsymbol{S'})}{N} \geq \lim_{N\to\infty} \frac{I_N(\boldsymbol{S'};\boldsymbol{S})}{N}} \tag{198}$$

Mutual Information for Continuous Random Variables

Remember: Discrete random variables

- Mutual information for discrete random variables A and B

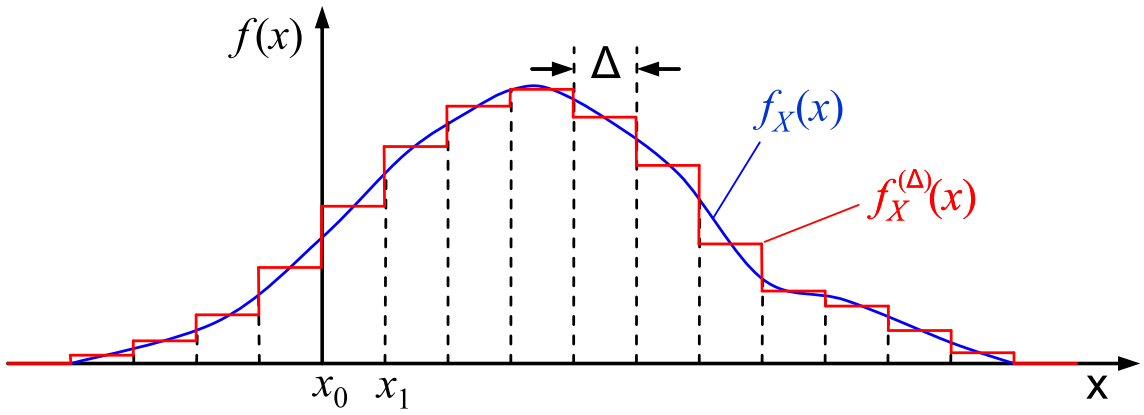
$$I(A;B) = H(A) - H(A|B) \tag{199}$$
$$= H(B) - H(B|A) \tag{200}$$

- For continuous random variables, the discrete entropies are not defined (they approach infinity)

Definition of mutual information for continuous random variables

- Quantize pdfs with a quantization step size Δ
- Calculate mutual information for resulting discrete random variables
- Consider limit for quantization step size Δ approaching zero

Discretization of Continuous Random Variables



- Approximation $f_X^{(\Delta)}$ of pdf f_X

$$\forall x : x_i \leq x < x_{i+1} \qquad f_X^{(\Delta)}(x) = \frac{1}{\Delta} \int_{x_i}^{x_{i+1}} f_X(x')\,\mathrm{d}x' \tag{201}$$

- Pmf p_{X_Δ} for random variable X_Δ

$$\boxed{p_{X_\Delta}(x_i) = \int_{x_i}^{x_{i+1}} f_X(x')\,\mathrm{d}x' = f_X^{(\Delta)}(x_i) \cdot \Delta} \tag{202}$$

- Joint pmf of two discrete approximations X_Δ and Y_Δ

$$\boxed{p_{X_\Delta Y_\Delta}(x_i, y_j) = f_{XY}^{(\Delta)}(x_i, y_j) \cdot \Delta^2} \tag{203}$$

Mutual Information for Continuous Random Variables

- Mutual information for discrete random variables $X_\Delta \in \mathcal{A}_{X_\Delta}$ and $Y_\Delta \in \mathcal{A}_{Y_\Delta}$

$$I(X_\Delta; Y_\Delta) = \sum_{x_i \in \mathcal{A}_{X_\Delta}} \sum_{y_i \in \mathcal{A}_{Y_\Delta}} p_{X_\Delta Y_\Delta}(x_i, y_j) \log_2 \frac{p_{X_\Delta Y_\Delta}(x_i, y_i)}{p_{X_\Delta}(x_i)\, p_{Y_\Delta}(y_j)} \tag{204}$$

$$= \sum_{x_i \in \mathcal{A}_{X_\Delta}} \sum_{y_i \in \mathcal{A}_{Y_\Delta}} f_{XY}^{(\Delta)}(x_i, y_j) \cdot \log_2 \frac{f_{XY}^{(\Delta)}(x_i, y_j)}{f_X^{(\Delta)}(x_i)\, f_Y^{(\Delta)}(y_j)} \cdot \Delta^2$$

- The mutual information $I(X;Y)$ for the continuous random variables X and Y is obtained for Δ approaching zero,

$$I(X;Y) = \lim_{\Delta \to 0} I(X_\Delta; Y_\Delta) \tag{205}$$

- For $\Delta \to 0$, the piecewise constant pdf approximations $f_{XY}^{(\Delta)}$, $f_X^{(\Delta)}$, and $f_Y^{(\Delta)}$ approach the pdfs f_{XY}, f_X, and f_Y, and we obtain

$$\boxed{I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{XY}(x,y) \log_2 \frac{f_{XY}(x,y)}{f_X(x)\, f_Y(y)} \,\mathrm{d}x \,\mathrm{d}y} \tag{206}$$

Mutual Information for Continuous Random Vectors

- Mutual information for continuous random variables X and Y

$$I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{XY}(x,y) \, \log_2 \frac{f_{XY}(x,y)}{f_X(x) \, f_Y(y)} \, \mathrm{d}x \, \mathrm{d}y \tag{207}$$

- Consider extension to N-dimensional random vectors $\boldsymbol{X} = (X_0, X_1, \cdots, X_{N-1})^{\mathrm{T}}$ and $\boldsymbol{Y} = (Y_0, Y_1, \cdots, Y_{N-1})^{\mathrm{T}}$

$$I_N(\boldsymbol{X};\boldsymbol{Y}) = \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y}) \, \log_2 \frac{f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})}{f_{\boldsymbol{X}}(\boldsymbol{x}) \, f_{\boldsymbol{Y}}(\boldsymbol{y})} \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} \tag{208}$$

- Using $f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y}) = f_{\boldsymbol{X}}(\boldsymbol{x}) f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{y})$, we can also write

$$I_N(\boldsymbol{X};\boldsymbol{Y}) = \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{y}) \, \log_2 \frac{f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{y})}{f_{\boldsymbol{Y}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} \tag{209}$$

Mutual Information between Discrete and Continuous RV

- Let $\boldsymbol{Y}$ be a discrete random vector with alphabet $\mathcal{A}_Y^N$

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \sum_{\boldsymbol{a} \in \mathcal{A}_Y^N} \delta(\boldsymbol{y} - \boldsymbol{a})\, p_{\boldsymbol{Y}}(\boldsymbol{a}) \tag{210}$$

$$f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \mathcal{A}_Y^N} \delta(\boldsymbol{y} - \boldsymbol{a})\, p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \tag{211}$$

- Rewriting mutual information using above pmfs yields

$$\begin{aligned} I_N(\boldsymbol{X};\boldsymbol{Y}) &= \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{y})\, \log_2 \frac{f_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{y})}{f_{\boldsymbol{Y}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\ &= \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \sum_{\boldsymbol{a} \in \mathcal{A}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x})\, \log_2 \frac{p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x})}{p_{\boldsymbol{Y}}(\boldsymbol{a})}\, \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \sum_{\boldsymbol{a} \in \mathcal{A}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \Big(\log_2 p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) - \log_2 p_{\boldsymbol{Y}}(\boldsymbol{a}) \Big)\, \mathrm{d}\boldsymbol{x} \end{aligned} \tag{212}$$

Mutual Information between Discrete and Continuous RV

- Continue reformulation of mutual information $I_N(\boldsymbol{X};\boldsymbol{Y})$

$$
\begin{aligned}
I_N(\boldsymbol{X};\boldsymbol{Y}) &= \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \sum_{\boldsymbol{a}\in\mathcal{A}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \Big(\log_2 p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) - \log_2 p_{\boldsymbol{Y}}(\boldsymbol{a}) \Big) \, \mathrm{d}\boldsymbol{x} \\
&= -\sum_{\boldsymbol{a}\in\mathcal{A}_Y^N} \left(\int_{\mathcal{R}^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) f_{\boldsymbol{X}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right) \log_2 p_{\boldsymbol{Y}}(\boldsymbol{a}) \\
&\quad + \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \left(\sum_{\boldsymbol{a}\in\mathcal{A}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \log_2 p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} \\
&= -\sum_{\boldsymbol{a}\in\mathcal{A}_Y^N} p_{\boldsymbol{Y}}(\boldsymbol{a}) \log_2 p_{\boldsymbol{Y}}(\boldsymbol{a}) - \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \, H_N(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\
&= H_N(\boldsymbol{Y}) - \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x}) \, H_N(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \qquad (213)
\end{aligned}
$$

Mutual Information between Discrete and Continuous RV

- Mutual information between a discrete random vector $\boldsymbol{Y}$ and a continuous random vector $\boldsymbol{X}$

$$I_N(\boldsymbol{X};\boldsymbol{Y}) = H_N(\boldsymbol{Y}) - \int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x})\, H_N(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{214}$$

  where $H_N(\boldsymbol{Y})$ is the entropy of the discrete random vector $\boldsymbol{Y}$ and

$$H_N(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) = -\sum_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}_Y^N} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x})\, \log_2 p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{a}|\boldsymbol{x}) \tag{215}$$

  is the conditional entropy of $\boldsymbol{Y}$ given the event $\{\boldsymbol{X}=\boldsymbol{x}\}$

- Since the conditional entropy $H_N(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x})$ is always nonnegative, we have

$$\boxed{I_N(\boldsymbol{X};\boldsymbol{Y}) \leq H_N(\boldsymbol{Y})} \tag{216}$$

  with equality if and only if $\boldsymbol{Y}$ is a deterministic function of $\boldsymbol{X}$

- If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent, we have $I_N(\boldsymbol{X};\boldsymbol{Y}) = 0$

Mutual Information for Coding of Continuous Sources

- Consider mutual information $I_N(\boldsymbol{S};\boldsymbol{S'})$ between a vector of N successive input samples $\boldsymbol{S}$ and the corresponding vector of N reconstructed samples $\boldsymbol{S'}$
- Since vectors of reconstructed samples are discrete, we can write

$$I_N(\boldsymbol{S};\boldsymbol{S'}) \;=\; H_N(\boldsymbol{S'}) - \int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s})\, H_N(\boldsymbol{S'}|\boldsymbol{S}=\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \;\leq\; H_N(\boldsymbol{S'}) \tag{217}$$

  where equality is achieved if and only if the vector $\boldsymbol{S'}$ of reconstructed samples is a deterministic function of the input vector $\boldsymbol{S}$

- Using the fundamental bound for lossless coding, we have for the average rate of a source code Q,

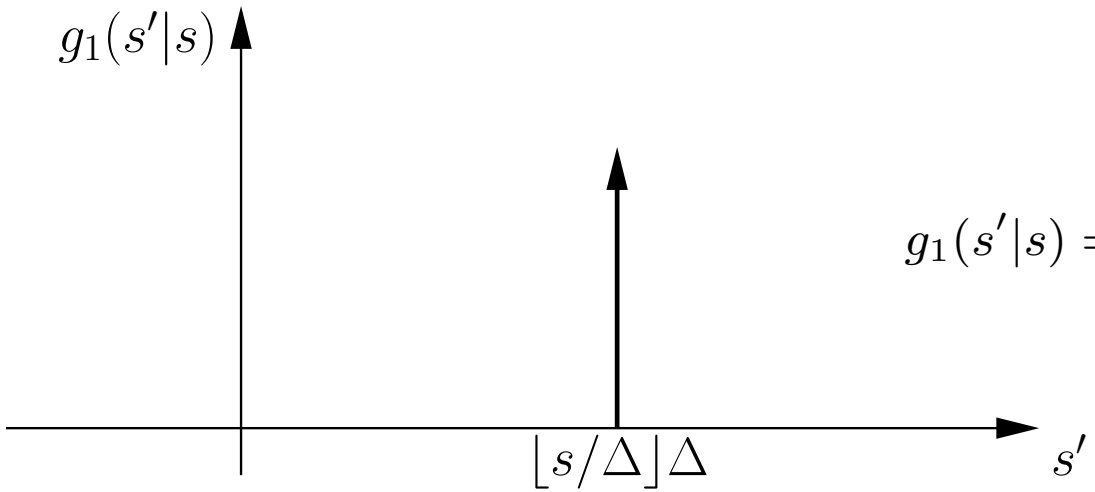
$$\boxed{r(Q) \geq \lim_{N\to\infty} \frac{H_N(\boldsymbol{S'})}{N} \geq \lim_{N\to\infty} \frac{I_N(\boldsymbol{S'};\boldsymbol{S})}{N}} \tag{218}$$

$\Longrightarrow$ Same expression as for coding of discrete sources

Description of a Source Code using a Conditional Pdf

- Statistical properties of a mapping $\boldsymbol{s'} = \beta(\alpha(\boldsymbol{s}))$ can be described by an N-th order conditional pdf $g_N(\boldsymbol{s'}|\boldsymbol{s})$

- **Example 1**: Mapping $s \rightarrow s' : \; s' = \lfloor s/\Delta \rfloor \cdot \Delta$



$$g_1(s'|s) = \delta(s' - \lfloor s/\Delta \rfloor \cdot \Delta)$$

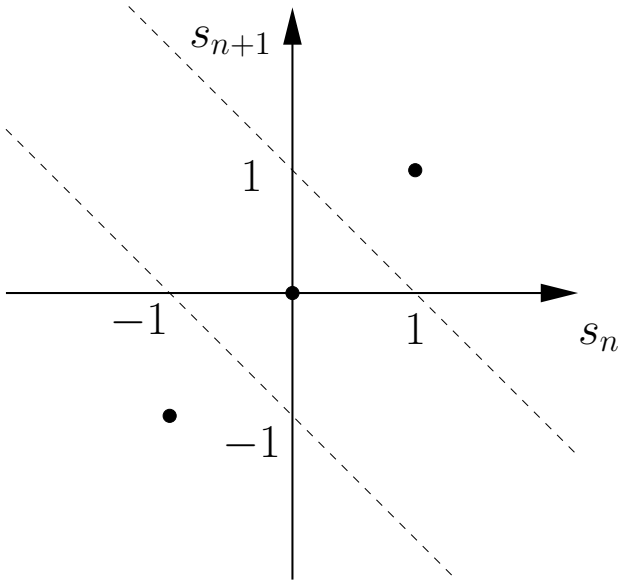
- For $N > 1$, $g_N(\boldsymbol{s'}|\boldsymbol{s})$ are multivariate conditional pdfs

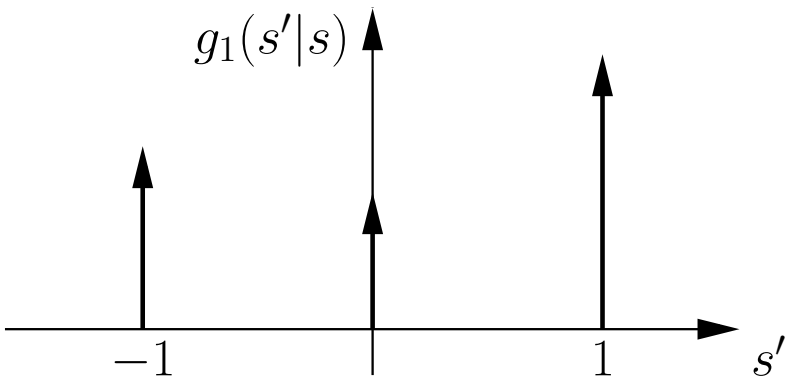
- The pdfs $g_N(\boldsymbol{s'}|\boldsymbol{s})$ obtained by a deterministic mapping (codes) are a subset of the set of all conditional pmfs

Description of a Source Code using a Conditional Pdf

- **Example 2**: Mapping $(s_n, s_{n+1}) \rightarrow (s'_n, s'_{n+1})$

$$(s'_n, s'_{n+1}) = \begin{cases} (1,1) & : \quad s_n + s_{n+1} > 1 \\ (-1,-1) & : \quad s_n + s_{n+1} < -1 \\ (0,0) & : \quad \text{otherwise} \end{cases}$$





$$g_1(s'|s) = x \cdot \delta(s'+1) + y \cdot \delta(s') + z \cdot \delta(s'-1)$$

with $x + y + z = 1$

Distortion for a Source Code using Conditional Pdf

- Let $g_N^Q(\boldsymbol{s'}|\boldsymbol{s})$ be the N-th order conditional pdf of a source code Q with $\boldsymbol{s'} \in \mathcal{R}^N$ and $\boldsymbol{s} \in \mathcal{R}^N$
- N-th order distortion

$$\begin{aligned}\delta_N(g_N) &= E\{d_N(\boldsymbol{S}, \boldsymbol{S'})\} \\ &= \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{S}\boldsymbol{S'}}(\boldsymbol{s}, \boldsymbol{s'}) \cdot d_N(\boldsymbol{s}, \boldsymbol{s'}) \, \mathrm{d}\boldsymbol{s} \, \mathrm{d}\boldsymbol{s'} \\ &= \int_{\mathcal{R}^N} \int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s}) \cdot g_N^Q(\boldsymbol{s'}|\boldsymbol{s}) \cdot d_N(\boldsymbol{s}, \boldsymbol{s'}) \, \mathrm{d}\boldsymbol{s} \, \mathrm{d}\boldsymbol{s'} \end{aligned} \tag{219}$$

- Recall: General expression for distortion $\delta(Q)$ of a source code Q

$$\delta(Q) = \lim_{N \to \infty} E\{d_N(\boldsymbol{S}, \boldsymbol{S'})\} \tag{220}$$

- Distortion for a source code Q can be written as

$$\delta(Q) = \lim_{N \to \infty} \delta_N(g_N^Q) \tag{221}$$

Mutual Information for a Source Code using Conditional Pdf

- N-th order mutual information

$$\begin{aligned} I_N(g_N) &= E\left\{\log_2 \frac{f_{\boldsymbol{S}\boldsymbol{S}'}(\boldsymbol{S},\boldsymbol{S}')}{f_{\boldsymbol{S}}(\boldsymbol{S}) f_{\boldsymbol{S}'}(\boldsymbol{S}')}\right\} \\ &= \int_{\mathcal{R}^N}\int_{\mathcal{R}^N} f_{\boldsymbol{S}\boldsymbol{S}'}(\boldsymbol{s},\boldsymbol{s}') \cdot \log_2 \frac{f_{\boldsymbol{S}\boldsymbol{S}'}(\boldsymbol{S},\boldsymbol{S}')}{f_{\boldsymbol{S}}(\boldsymbol{S}) f_{\boldsymbol{S}'}(\boldsymbol{S}')}\,\mathrm{d}\boldsymbol{s}\,\mathrm{d}\boldsymbol{s}' \\ &= \int_{\mathcal{R}^N}\int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s}) \cdot g_N(\boldsymbol{s}'|\boldsymbol{s}) \cdot \log_2 \frac{g_N(\boldsymbol{s}'|\boldsymbol{s})}{f_{\boldsymbol{S}'}(\boldsymbol{s}')}\,\mathrm{d}\boldsymbol{s}\,\mathrm{d}\boldsymbol{s}' \end{aligned} \tag{222}$$

with

$$f_{\boldsymbol{S}'}(\boldsymbol{s}') = \int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s}) \cdot g_N(\boldsymbol{s}'|\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}. \tag{223}$$

- For a given source $\boldsymbol{S}$, both the N-th order distortion δ_N and the N-th order mutual information I_N are completely determined by the N-th order conditional pdf $g_N^Q(\boldsymbol{s}'|\boldsymbol{s})$

Information Rate-Distortion Function

- Consider any source code Q with a distortion $\delta(Q) \leq D$
- Associated rate is denoted by $r(Q)$
- Output $\boldsymbol{S'}$ of source codec is a discrete random process
- Remember: Fundamental theorem for lossless coding

$$r(Q) \geq \bar{H}(\boldsymbol{S'}) = \lim_{N \to \infty} \frac{H_N(\boldsymbol{S'})}{N} \tag{224}$$

- Using mutual information, we can write

$$r(Q) \geq \lim_{N \to \infty} \frac{H_N(\boldsymbol{S'})}{N} \geq \lim_{N \to \infty} \frac{I_N(\boldsymbol{S}; \boldsymbol{S'})}{N} = \lim_{N \to \infty} \frac{I_N(g_N^Q)}{N} \tag{225}$$

- Deterministic mapping g_N^Q as given by a source code Q is a special case of a random mapping g_N

$$I_N(g_N^Q) \geq \inf_{g_N : \delta_N(g_N) \leq D} I_N(g_N) \tag{226}$$

Information Rate-Distortion Function

- Hence, we have

$$r(Q) \geq \lim_{N\to\infty} \frac{I_N(g_N^Q)}{N} \geq \lim_{N\to\infty} \inf_{g_N:\delta_N(g_N)\leq D} \frac{I_N(g_N)}{N} \tag{227}$$

- **Information rate-distortion function**

$$\boxed{R^{(I)}(D) = \lim_{N\to\infty} \inf_{g_N:\delta_N(g_N)\leq D} \frac{I_N(g_N)}{N}} \tag{228}$$

- **Fundamental source coding theorem**

$$\boxed{\forall Q : \delta(Q) \leq D, \quad r(Q) \geq R^{(I)}(D)} \tag{229}$$

$\Longrightarrow$ For a given maximum distortion D, the rate $r(Q)$ for each source code Q that yields a distortion $\delta(Q) \leq D$ is greater than or equal to the information rate-distortion function $R^{(I)}(D)$

Information vs Operational Rate-Distortion Function

- We have shown that information rate-distortion function $R^{(I)}(D)$ represents a lower bound for all source codes Q

$\Longrightarrow$ Lower bound for operational rate-distortion function

- It can also be shown that $R^{(I)}(D)$ is asymptotically achievable
  - For any $\epsilon > 0$, there exists a code Q with

$$\begin{aligned} \delta(Q) &\leq D && \text{and} \\ r(Q) &\leq R^{(I)}(D) + \epsilon \end{aligned}$$

  (see proof in [Cover and Thomas])

$\Longrightarrow$ **Information rate-distortion function is equal to operational rate-distortion function**

- Use the term **rate-distortion function** $R(D)$ for both in the following

(Information) Distortion-Rate Function

- Fundamental source coding theorem

$$\boxed{\forall Q : \delta(Q) \leq D, \quad r(Q) \geq R(D)} \tag{230}$$

with (information) rate-distortion function

$$\boxed{R(D) = \lim_{N\to\infty} \inf_{g_N : \delta_N(g_N) \leq D} \frac{I_N(g_N)}{N}} \tag{231}$$

- Alternative formulation by interchanging roles of rate and distortion

$$\boxed{\forall Q : r(Q) \leq R, \quad \delta(Q) \geq D^{(I)}(R)} \tag{232}$$

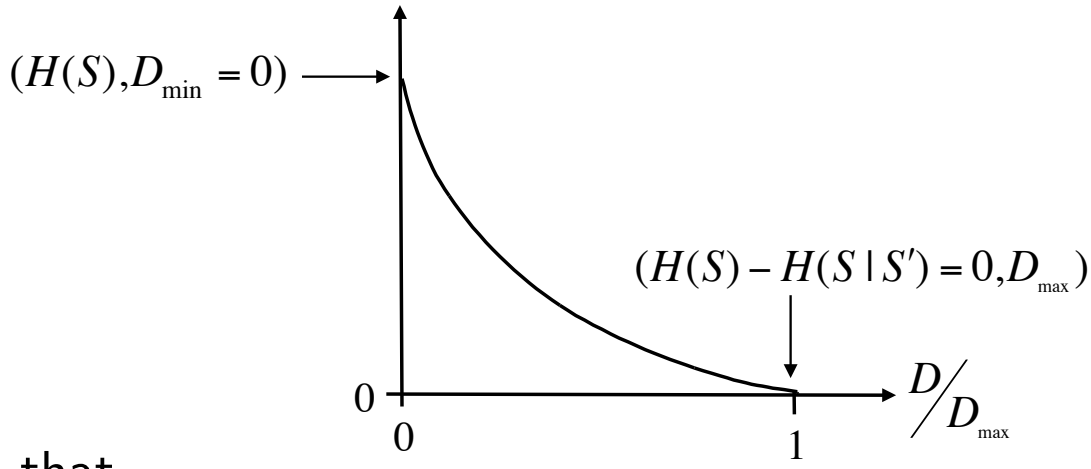
with (information) **distortion-rate function**

$$\boxed{D(R) = \lim_{N\to\infty} \inf_{g_N :\, I_N(g_N)/N \leq R} \delta_N(g_N)} \tag{233}$$

- Distortion-rate function $D(R)$ is inverse of rate-distortion function $R(D)$

$R(D)$ for Discrete Sources and Additive Distortion Measures

- Example of $R(D)$ for a discrete iid source
- $R(D)$ is a **non-increasing** and **convex** function of D



- There exists a value $D_{\max}$, so that

$$\forall D \geq D_{\max} \qquad R(D) = 0 \tag{234}$$

$\Longrightarrow$ For MSE distortion measure: $D_{\max}$ is equal to the variance σ^2 of the source

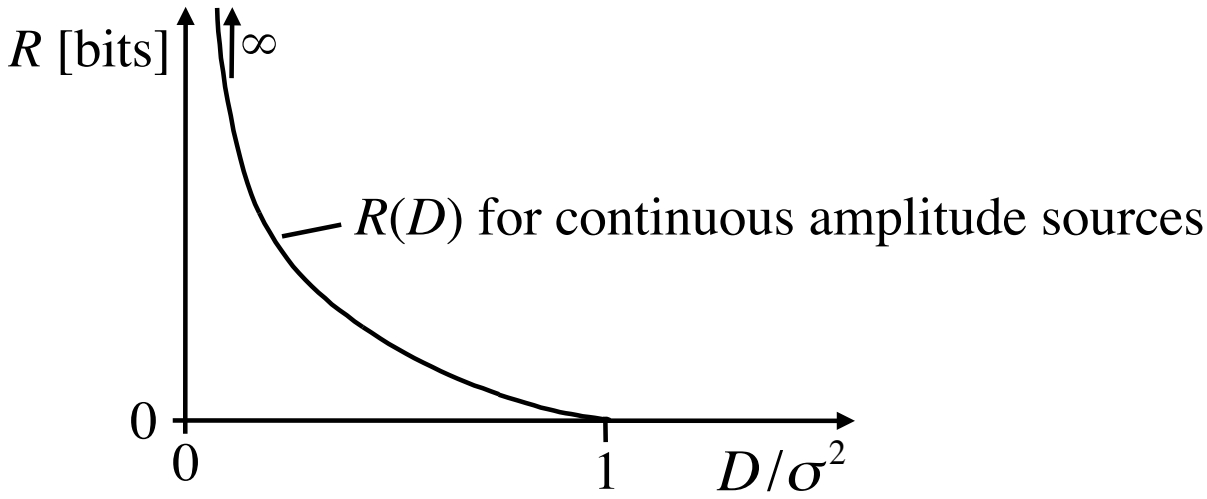
- Minimum rate required for lossless transmission of a discrete source is equal to the entropy rate

$$D_{\min} = 0 \qquad R(0) = \bar{H}(\boldsymbol{S}) \tag{235}$$

$\Longrightarrow$ Fundamental bound for lossless coding:
Special case of the fundamental bound for lossy coding

$R(D)$ for Continuous Sources and Additive Distortion Meas.

- Example of $R(D)$ for an amplitude-continuous source



- $R(D)$ is a **non-increasing** and **convex** function of D
- There exists a value $D_{\max}$, so that

$$\forall D \geq D_{\max} \qquad R(D) = 0 \tag{236}$$

$\Longrightarrow$ For MSE distortion measure: $D_{\max}$ is equal to the variance σ^2 of the source

- $R(D)$ approaches infinity as D approaches zero

Rate-Distortion Function for IID Sources

- N-th order distortion $\delta_N(g_N)$ for additive distortion measures

$$\delta_N(g_N) = E\{d_N(\boldsymbol{S}, \boldsymbol{S'})\} = E\left\{\frac{1}{N}\sum_{i=0}^{N-1} d_1(S_i, S'_i)\right\} = E\{d_1(S, S')\}$$

$$= \int_{-\infty}^{\infty} f_S(s) \cdot g_1(s'|s) \cdot d_1(s, s') \, \mathrm{d}s = \delta_1(g) \tag{237}$$

- N-th order mutual information for iid sources
  (Note: If the source $\boldsymbol{S}$ is iid, the reconstruction $\boldsymbol{S'}$ is also iid)

$$I_N(g_N) = E\left\{\log_2 \frac{f_{\boldsymbol{S}\boldsymbol{S'}}(\boldsymbol{S}, \boldsymbol{S'})}{f_{\boldsymbol{S}}(\boldsymbol{S})\, f_{\boldsymbol{S'}}(\boldsymbol{S'})}\right\} = E\left\{\log_2 \left(\frac{f_{SS'}(S, S')}{f_S(S)\, f_{S'}(S')}\right)^N\right\}$$

$$= N \cdot E\left\{\log_2 \frac{f_{SS'}(S, S')}{f_S(S)\, f_{S'}(S')}\right\}$$

$$= N \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_S(s)\, g_1(s'|s)\, \log_2 \frac{g_1(s'|s)}{f_{S'}(s')} \, \mathrm{d}s \, \mathrm{d}s' = N \cdot I_1(g) \tag{238}$$

Rate-Distortion Function for IID Sources

- For iid sources and additive distortion measures, we have

$$\delta_N(g_N^Q) = \delta_1(g^Q) \qquad \text{and} \qquad I_N(g_N^Q) = N \cdot I_1(g^Q) \tag{239}$$

- Rate-distortion function for iid sources and additive distortion measures

$$R(D) = \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\le D} \frac{I_N(g_N)}{N} = \inf_{g_1:\, \delta_1(g_1)\le D} I_1(g_1) \tag{240}$$

$\Longrightarrow$ Also called first-order rate-distortion function $R_1(D)$

- Distortion-rate function for iid sources and additive distortion measures

$$D(R) = \lim_{N\to\infty} \inf_{g_N:\, I_N(g_N)/N\le R} \delta_N(g_N) = \inf_{g_1:\, I_1(g_1)\le R} \delta_1(g_1) \tag{241}$$

$\Longrightarrow$ Also called first-order distortion-rate function $D_1(R)$

N-th Order Rate-Distortion Functions

- Can define N-th order rate-distortion and distortion-rate functions

$$R_N(D) = \inf_{g_N:\, \delta_N(g_N) \le D} \frac{I_N(g_N)}{N} \tag{242}$$

$$D_N(R) = \inf_{g_N:\, I_N(g_N)/N \le R} \delta_N(g_N) \tag{243}$$

- In general, the rate-distortion and distortion-rate functions can be written as

$$\boxed{R(D) = \lim_{N\to\infty} R_N(D)} \quad \text{and} \quad \boxed{D(R) = \lim_{N\to\infty} D_N(R)} \tag{244}$$

- For **iid sources and additive distortion measures**, we have

$$\boxed{R(D) = R_1(D)} \quad \text{and} \quad \boxed{D(R) = D_1(R)} \tag{245}$$

Discussion of Rate-Distortion Functions

Operational rate-distortion function

$$R(D) = \inf_{Q:\, \delta(Q) \le D} r(Q) \tag{246}$$

- Minimization over all possible source codes
- Easy to define, but impossible to evaluate

Information rate-distortion function

$$R(D) = \lim_{N \to \infty} \inf_{g_N : \delta_N(g_N) \le D} \frac{I_N(g_N)}{N} \tag{247}$$

- Property of source: Don't need to consider all possible codes
- Still impossible to evaluate directly (minimization over all conditional pdfs)
- Numerical minimization for discrete sources: Blahut-Arimoto algorithm

How can we proceed?

- Can derive lower bound for (information) rate-distortion function
- For some sources and distortion measures (e.g., Gaussian and MSE):
  $\Longrightarrow$ Can show that lower bound is achievable

Differential Entropy

- Mutual information between a continuous random vector $\boldsymbol{X}$ and a continuous or discrete random vector $\boldsymbol{Y}$

$$\begin{aligned} I(\boldsymbol{X};\boldsymbol{Y}) &= E\left\{\log_2 \frac{f_{\boldsymbol{XY}}(\boldsymbol{X},\boldsymbol{Y})}{f_{\boldsymbol{X}}(\boldsymbol{X})\, f_{\boldsymbol{Y}}(\boldsymbol{Y})}\right\} = E\left\{\log_2 \frac{f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{X}|\boldsymbol{Y})}{f_{\boldsymbol{X}}(\boldsymbol{X})}\right\} \\ &= E\{-\log_2 f_{\boldsymbol{X}}(\boldsymbol{X})\} - E\left\{-\log_2 f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{X}|\boldsymbol{Y})\right\} \\ &= h(\boldsymbol{X}) - h(\boldsymbol{X}|\boldsymbol{Y}) \end{aligned} \tag{248}$$

- Define: **Differential entropy** of a continuous random vector $\boldsymbol{X}$

$$h(\boldsymbol{X}) = E\{-\log_2 f_{\boldsymbol{X}}(\boldsymbol{X})\} = -\int_{\mathcal{R}^N} f_{\boldsymbol{X}}(\boldsymbol{x})\, \log_2 f_{\boldsymbol{X}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{249}$$

- Define: **Conditional differential entropy** of $\boldsymbol{X}$ given $\boldsymbol{Y}$

$$\begin{aligned} h(\boldsymbol{X}|\boldsymbol{Y}) &= E\left\{-\log_2 f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{X}|\boldsymbol{Y})\right\} \\ &= -\int_{\mathcal{R}^N}\int_{\mathcal{R}^N} f_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})\, \log_2 f_{\boldsymbol{X}|\boldsymbol{Y}}(\boldsymbol{x}|\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \end{aligned} \tag{250}$$

Example: Differential Entropy for an Uniform IID Source

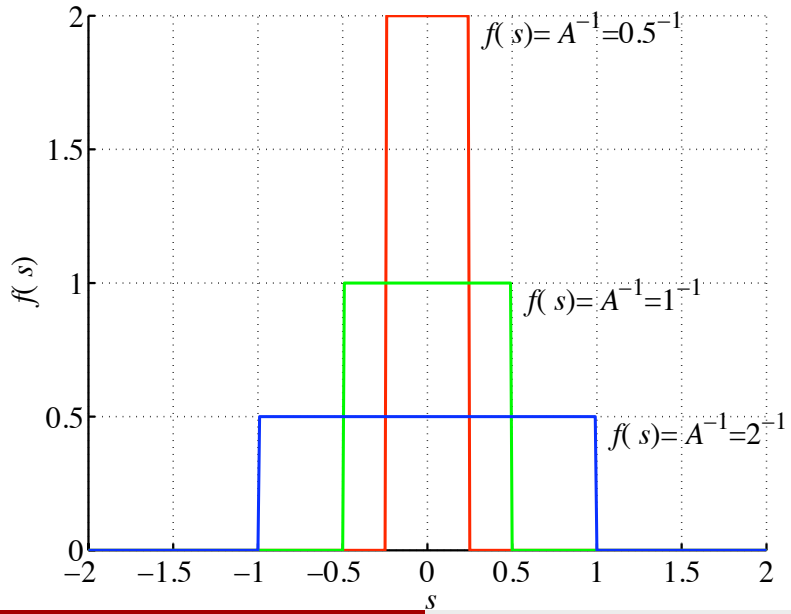
- For an continuous iid source S, differential entropy is defined as

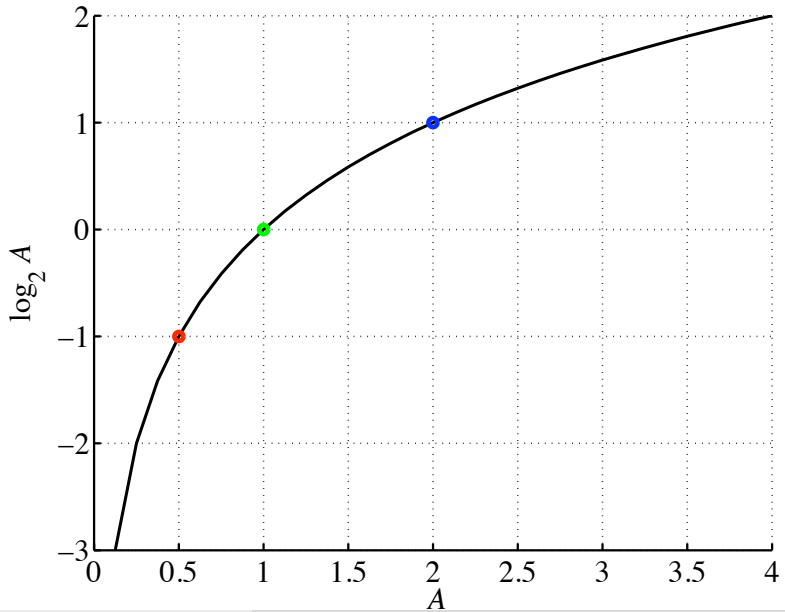
$$h(S) = E\{-\log_2 f(S)\} = -\int_{-\infty}^{\infty} f(s) \log_2 f(s)\, \mathrm{d}s \tag{251}$$

- $h(S)$ for uniform distribution $f(s) = 1/A$ for $-A/2 \leq s \leq A/2$

$$h(S) = -\int_{-A/2}^{A/2} \frac{1}{A} \log_2 \frac{1}{A} \mathrm{d}s = \frac{1}{A} \log_2 A \int_{-A/2}^{A/2} \mathrm{d}s = \log_2 A \tag{252}$$

- Differential entropy can become negative (in contrast to discrete entropy)





Differential Entropy for an Gaussian IID Source

- Gaussian iid process

$$f_S(s) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(s-\mu)^2}{2\sigma^2}} \tag{253}$$

- Differential entropy

$$\begin{aligned}
h(S) &= -\int_{-\infty}^{\infty} f_S(s) \log_2 f_S(s) \mathrm{d}s \\
&= -\int_{-\infty}^{\infty} f_S(s) \left[-\frac{(s-\mu)^2}{2\sigma^2} \log_2 e - \log_2 \sqrt{2\pi\sigma^2} \right] \mathrm{d}s \\
&= \frac{E\{(S-\mu)^2\}}{2\sigma^2} \cdot \log_2 e + \frac{1}{2} \log_2 (2\pi\sigma^2) \\
&= \frac{1}{2} \log_2 e + \frac{1}{2} \log_2 (2\pi\sigma^2) \\
&= \frac{1}{2} \log_2 (2\pi e \sigma^2)
\end{aligned} \tag{254}$$

N-th Order Differential Entropy

- **N-th order differential entropy**

$$h_N(\boldsymbol{S}) = h(\boldsymbol{S}^{(N)}) = h(S_0, \cdots, S_{N-1}) = E\Big\{ -\log_2 f_{\boldsymbol{S}}(\boldsymbol{S}^{(N)}) \Big\} \tag{255}$$

- **Differential entropy rate**

$$\bar{h}(\boldsymbol{S}) = \lim_{N\to\infty} \frac{h_N(\boldsymbol{S})}{N} = \lim_{N\to\infty} \frac{h(S_0, \cdots, S_{N-1})}{N} \tag{256}$$

- N-th order pdf of a stationary Gaussian process

$$f_G(\boldsymbol{s}) = \frac{1}{(2\pi)^{N/2}\,|\boldsymbol{C}_N|^{1/2}}\; e^{-\frac{1}{2}(\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)} \tag{257}$$

- N-th order differential entropy of stationary Gaussian process

$$\begin{aligned} h_N^{(G)}(\boldsymbol{S}) &= -\int_{\mathcal{R}^N} f_G(\boldsymbol{s})\, \log_2 f_G(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \\ &= \frac{1}{2}\log_2\big((2\pi)^N |\boldsymbol{C}_N| \big) \\ &\quad + \frac{\log_2 e}{2} \int_{\mathcal{R}^N} f_G(\boldsymbol{s})\, (\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1} (\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s} \end{aligned} \tag{258}$$

N-th order Differential Entropy of Stationary Gaussian Process

- General stationary process with pdf $f_{\boldsymbol{S}}(\boldsymbol{s})$, mean $\boldsymbol{\mu}_N$, covariance matrix $\boldsymbol{C}_N$

$$\begin{aligned}
&\int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s})\, (\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1} (\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s} \\
&\qquad = E\big\{(\boldsymbol{S}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1} (\boldsymbol{S}-\boldsymbol{\mu}_N)\big\} \\
&\qquad = E\left\{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (S_i-\mu_i)(C^{-1})_{i,j}(S_j-\mu_j)\right\} \\
&\qquad = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} E\{(S_i-\mu_i)(S_j-\mu_j)\}\,(C^{-1})_{i,j} \\
&\qquad = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C_{j,i}(C^{-1})_{i,j} \\
&\qquad = \sum_{i=0}^{N-1} (CC^{-1})_{j,j} \\
&\qquad = N
\end{aligned} \tag{259}$$

N-th order Differential Entropy of Stationary Gaussian Process

- Showed for general pdf $f_{\boldsymbol{S}}(\boldsymbol{s})$

$$\int_{\mathcal{R}^N} f_{\boldsymbol{S}}(\boldsymbol{s})\, (\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1} (\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s} = N \tag{260}$$

- Continue derivation for stationary Gaussian source

$$\begin{aligned} h_N^{(G)}(\boldsymbol{S}) &= \frac{1}{2} \log_2\big((2\pi)^N |\boldsymbol{C}_N| \big) \\ &\qquad + \frac{\log_2 e}{2} \int_{\mathcal{R}^N} f_G(\boldsymbol{s})\, (\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1} (\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s} \\ &= \frac{1}{2} \log_2\big((2\pi)^N |\boldsymbol{C}_N| \big) + \frac{N}{2} \log_2 e \\ &= \frac{1}{2} \log_2\big((2\pi)^N |\boldsymbol{C}_N| \big) + \frac{1}{2} \log_2 e^N \\ &= \frac{1}{2} \log_2\big((2\pi e)^N |\boldsymbol{C}_N| \big) \end{aligned} \tag{261}$$

N-th order Differential Entropy of Stat. Non-Gaussian Process

- Consider stationary non-Gaussian process with N-th order pdf $f(\boldsymbol{s})$
- Let $f_G(\boldsymbol{s})$ be the N-th order pdf of a Gaussian process with the same N-th order autocovariance matrix $\boldsymbol{C}_N$
- By applying the divergence inequality for pdfs, we obtain

$$\begin{aligned} h_N(\boldsymbol{S}) &= -\int_{\mathcal{R}^N} f(\boldsymbol{s})\, \log_2 f(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \\ &\leq -\int_{\mathcal{R}^N} f(\boldsymbol{s})\, \log_2 f_G(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \\ &= \frac{1}{2} \log_2 \big((2\pi)^N |\boldsymbol{C}_N| \big) + \frac{\log_2 e}{2} \int_{\mathcal{R}^N} f(\boldsymbol{s}) (\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1} (\boldsymbol{s}-\boldsymbol{\mu}_N)\, \mathrm{d}\boldsymbol{s} \\ &= \frac{1}{2} \log_2 \big((2\pi e)^N |\boldsymbol{C}_N| \big) = h_N^{(G)}(\boldsymbol{S}) \end{aligned} \tag{262}$$

$\Longrightarrow$ **Gaussian process with a given N-th order autocovariance matrix C_N has the largest N-th order differential entropy among all processes with the same autocovariance matrix C_N**

Eigendecomposition of the Covariance Matrix

- Determinant $|\boldsymbol{C}_N|$: Product of the eigenvalues ξ_i of the matrix $\boldsymbol{C}_N$,

$$\boldsymbol{C}_N = \boldsymbol{A}_N \boldsymbol{\Xi}_N \boldsymbol{A}_N^T \quad \rightarrow \quad |\boldsymbol{C}_N| = \underbrace{|\boldsymbol{A}_N|}_{=1} \cdot |\boldsymbol{\Xi}_N| \cdot \underbrace{|\boldsymbol{A}_N^T|}_{=1} = \prod_{i=0}^{N-1} \xi_i^{(N)} \tag{263}$$

- $\boldsymbol{A}_N$: Orthogonal matrix with the N unit-norm eigenvectors as columns

$$\boldsymbol{A}_N = \left(\boldsymbol{v}_0^{(N)}, \boldsymbol{v}_1^{(N)}, \cdots, \boldsymbol{v}_{N-1}^{(N)} \right) \tag{264}$$

- $\boldsymbol{\Xi}_N$: Diagonal matrix with the N eigenvalues of $\boldsymbol{C}_N$ on its main diagonal

$$\boldsymbol{\Xi}_N = \begin{pmatrix} \xi_0^{(N)} & 0 & \cdots & 0 \\ 0 & \xi_1^{(N)} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \xi_{N-1}^{(N)} \end{pmatrix} \tag{265}$$

Maximum Differential Entropy

- Determinant of a matrix is the product of its eigenvalues

$$|\boldsymbol{C}_N| = \prod_{i=0}^{N-1} \xi_i^{(N)} \tag{266}$$

- Trace of a matrix is the sum of its eigenvalues (trace is similarity invariant)

$$\mathrm{tr}\left(|\boldsymbol{C}_N|\right) = \sum_{i=0}^{N-1} \xi_i^{(N)} = N \cdot \sigma^2 \tag{267}$$

- Inequality of arithmetic and geometric means:

$$\left(\prod_{i=0}^{N-1} x_i\right)^{\frac{1}{N}} \leq \frac{1}{N} \sum_{i=0}^{N-1} x_i, \tag{268}$$

with equality if and only if $x_0 = x_1 = \ldots = x_{N-1}$
(when geometric mean is maximized)

Maximum Differential Entropy

- Apply inequality to determinant of autocovariance matrix

$$|\boldsymbol{C}_N| = \prod_{i=0}^{N-1} \xi_i \leq \left(\frac{1}{N} \sum_{i=0}^{N-1} \xi_i \right)^N = \sigma^{2N} \tag{269}$$

$\Longrightarrow$ Equality if and only if source is iid (all eigenvalues are the same)

- For N-th order differential entropy of any source $\boldsymbol{S}$, we get

$$\begin{aligned} h_N(\boldsymbol{S}) &\leq \frac{1}{2} \log_2 \left((2\pi e)^N \, |\boldsymbol{C}_N| \right) && \text{(equality for Gaussian)} \\ &\leq \frac{N}{2} \log_2 \left(2\pi e \sigma^2 \right) && \text{(equality for iid)} \end{aligned} \tag{270}$$

$\Longrightarrow$ Equality if and only if source is Gaussian iid

$\Longrightarrow$ **For a given variance σ^2, the N-th order differential entropy is maximized for Gaussian iid processes**

$$\boxed{h_N(\boldsymbol{S}) \leq \frac{N}{2} \log_2 \left(2\pi e \sigma^2 \right)} \tag{271}$$

Shannon Lower Bound

- Lower bound for rate-distortion function $R(D)$

$$\begin{aligned} R(D) &= \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\le D} \frac{I_N(\boldsymbol{S};\boldsymbol{S'})}{N} \\ &= \lim_{N\to\infty} \inf_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S}) - h_N(\boldsymbol{S}|\boldsymbol{S'})}{N} \\ &= \lim_{N\to\infty} \frac{h_N(\boldsymbol{S})}{N} - \lim_{N\to\infty} \sup_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S}|\boldsymbol{S'})}{N} \\ &= \bar{h}(\boldsymbol{S}) - \lim_{N\to\infty} \sup_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S}-\boldsymbol{S'}|\boldsymbol{S'})}{N} \end{aligned} \tag{272}$$

- Define: **Shannon Lower Bound**

$$\boxed{R_L(D) = \bar{h}(\boldsymbol{S}) - \lim_{N\to\infty} \sup_{g_N:\, \delta_N(g_N)\le D} \frac{h_N(\boldsymbol{S}-\boldsymbol{S'})}{N}} \tag{273}$$

- Since conditioning does not increase differential entropy, we have

$$\boxed{R(D) \ge R_L(D)} \qquad \text{(equality if } \boldsymbol{S}-\boldsymbol{S'} \text{ is independent of } \boldsymbol{S'}\text{)} \tag{274}$$

Shannon Lower Bound for MSE Distortion

- For MSE distortion: Distortion is given by variance of differences

$$\delta_N(g_N) = \sigma_Z^2 \qquad \text{with} \qquad \boldsymbol{Z} = \boldsymbol{S} - \boldsymbol{S'} \qquad \text{and} \qquad \mu_Z = 0 \tag{275}$$

- Remember: Maximum differential entropy

$$h_N(\boldsymbol{S} - \boldsymbol{S'}) = h_N(\boldsymbol{Z}) \leq \frac{N}{2} \log_2 \left(2\pi e \sigma_Z^2\right) = \frac{N}{2} \log_2 \left(2\pi e D\right) \tag{276}$$

- **Shannon lower bound for MSE distortion**

$$\boxed{R_L(D) = \bar{h}(\boldsymbol{S}) - \frac{1}{2} \log_2 \left(2\pi e D\right)} \tag{277}$$

  $\Longrightarrow$ **For given C_N or $\Phi_{SS}(\omega)$, maximized for Gaussian processes**
  $\Longrightarrow$ **For given σ^2, maximized for Gaussian iid processes**

- When is the Shannon lower bound for MSE achievable?
  $\Longrightarrow$ Difference process $\boldsymbol{Z} = \boldsymbol{S} - \boldsymbol{S'}$ has to be zero-mean Gaussian iid
  $\Longrightarrow$ Difference process $\boldsymbol{Z} = \boldsymbol{S} - \boldsymbol{S'}$ has to be independent of $\boldsymbol{S'}$:

$$g_{\boldsymbol{Z}|\boldsymbol{S'}}(\boldsymbol{z}|\boldsymbol{s'}) = g_{\boldsymbol{Z}}(\boldsymbol{z})$$

Shannon Lower Bound for IID Sources MSE Distortion

- **Shannon lower bound for MSE distortion**

$$\boxed{R_L(D) = \bar{h}(\boldsymbol{S}) - \frac{1}{2}\log_2\big(2\pi e D\big)} \qquad \boxed{D_L(R) = \frac{1}{2\pi e}\cdot 2^{2\,\bar{h}(\boldsymbol{S})}\cdot 2^{-2R}} \tag{278}$$

- For iid sources $\boldsymbol{S}$, we have

$$\begin{aligned}
\bar{h}(\boldsymbol{S}) &= \lim_{N\to\infty}\frac{h_N(\boldsymbol{S})}{N} = \lim_{N\to\infty}\frac{1}{N}E\{-\log_2 f_{\boldsymbol{S}}(\boldsymbol{S})\} \\
&= \lim_{N\to\infty}\frac{1}{N}\sum_{i=0}^{N-1}E\{-\log_2 f_S(S_i)\} = \lim_{N\to\infty}\frac{N}{N}E\{-\log_2 f_S(S)\} \\
&= E\{-\log_2 f_S(S)\} = h(S)
\end{aligned} \tag{279}$$

- **Shannon lower bound for MSE distortion and iid sources**

$$\boxed{R_L(D) = h(S) - \frac{1}{2}\log_2\big(2\pi e D\big)} \qquad \boxed{D_L(R) = \frac{1}{2\pi e}\cdot 2^{2\,h(S)}\cdot 2^{-2R}} \tag{280}$$

Shannon Lower Bound Selected IID Sources

- **Uniform pdf**:

$$h(S) = \frac{1}{2}\log_2(12\sigma^2) \quad \Longrightarrow \quad D_L(R) = \underbrace{\frac{6}{\pi e}}_{\approx 0.7}\,\sigma^2 \cdot 2^{-2R} \tag{281}$$

- **Laplacian pdf**:

$$h(S) = \frac{1}{2}\log_2(2e^2\sigma^2) \quad \Longrightarrow \quad D_L(R) = \underbrace{\frac{e}{\pi}}_{\approx 0.865}\,\sigma^2 \cdot 2^{-2R} \tag{282}$$

- **Gaussian pdf**:

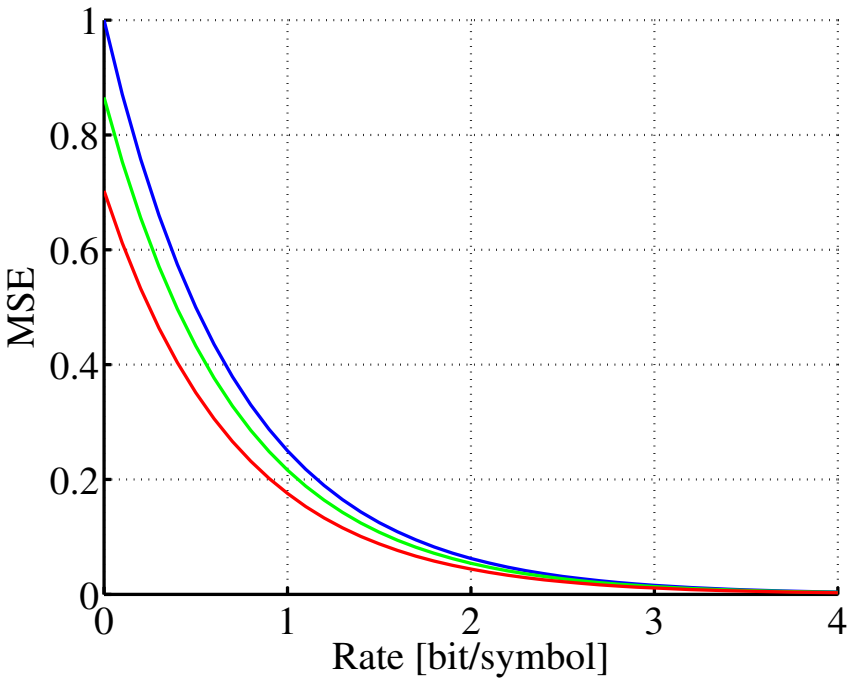
$$h(S) = \frac{1}{2}\log_2(2\pi e\sigma^2) \quad \Longrightarrow \quad D_L(R) = \sigma^2 \cdot 2^{-2R} \tag{283}$$

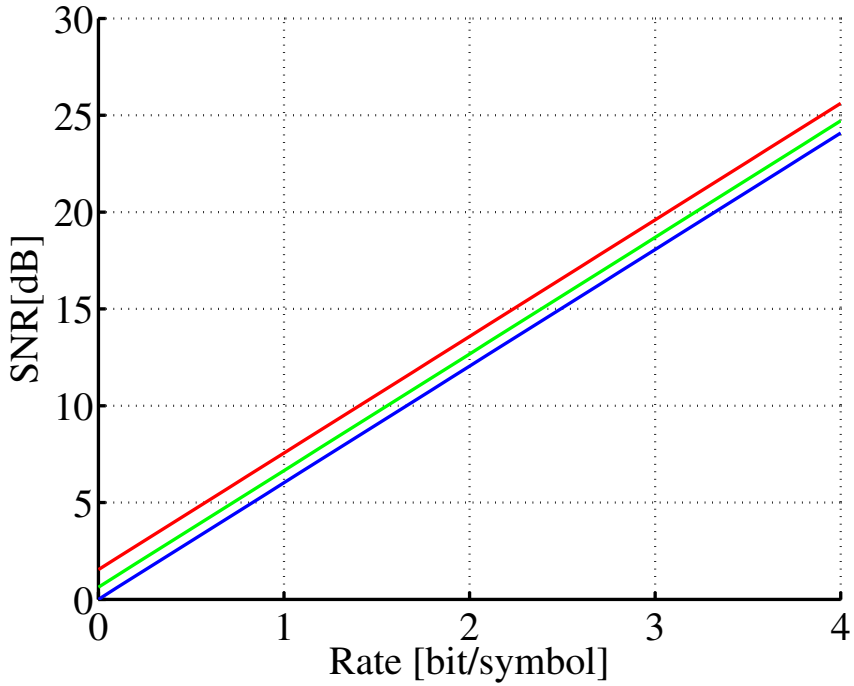
Shannon Lower Bound Selected IID Sources

Shannon lower bound using MSE and SNR

$$\mathrm{SNR} = 10 \log_{10} \frac{\sigma^2}{\mathrm{MSE}} \tag{284}$$

- Uniform iid process: red
- Laplace iid process: green
- Gauss iid process: blue



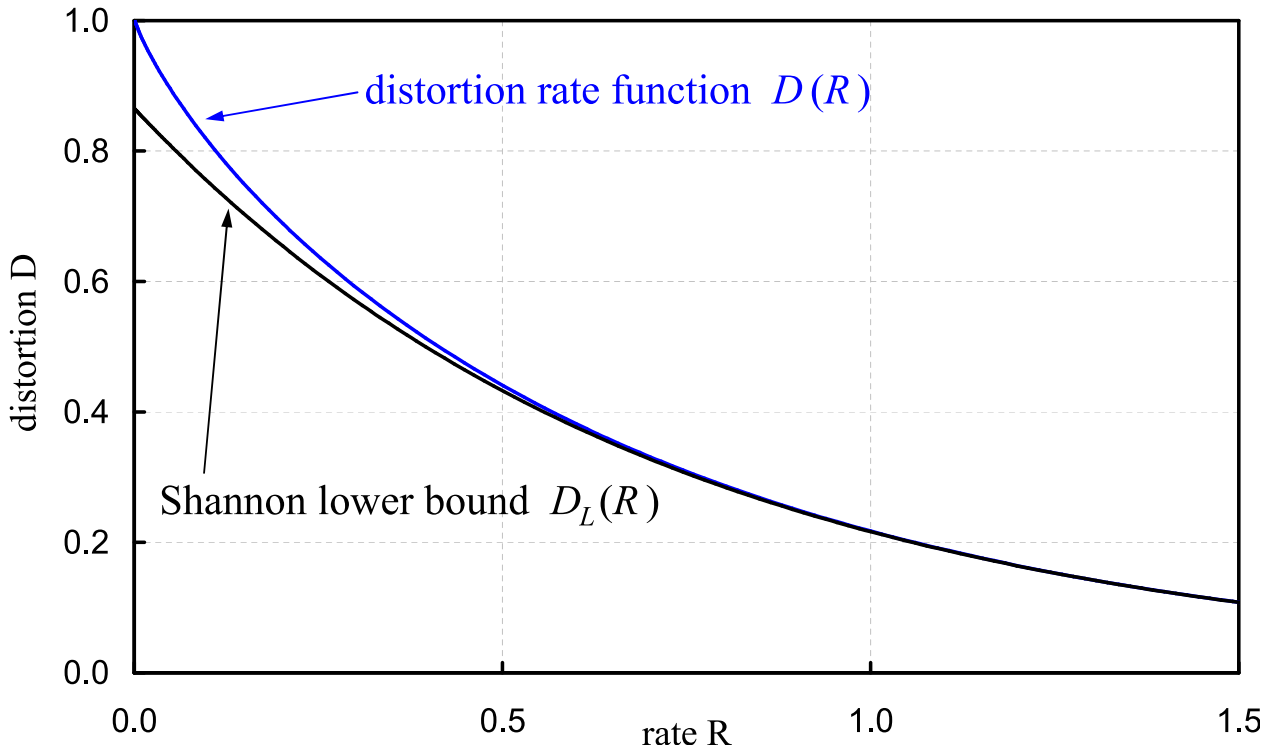


Asymptotic Tightness of the Shannon Lower Bound

- Shannon lower bound approaches distortion rate function for small distortions or high rates

$$\lim_{D \to 0} R(D) - R_L(D) = 0. \tag{285}$$

- Comparison of $D(R)$ with $D_L(R)$ for the Laplacian iid source



Shannon Lower Bound for Gaussian Sources with Memory

- Differential entropy for Gaussian sources

$$h_N^{(G)}(\boldsymbol{S}) = \frac{1}{2}\log_2\big((2\pi e)^N |\boldsymbol{C}_N| \big) \tag{286}$$

- Shannon lower bound for MSE distortion

$$\begin{aligned}
R_L(D) &= \lim_{N\to\infty} \frac{h_N^{(G)}(\boldsymbol{S})}{N} - \frac{1}{2}\log_2\big(2\pi e D \big) \\
&= \lim_{N\to\infty} \frac{\log_2((2\pi e)^N |\boldsymbol{C}_N|)}{2N} - \frac{1}{2}\log_2\big(2\pi e D \big) \\
&= \frac{1}{2}\log_2(2\pi e) + \lim_{N\to\infty} \frac{\log_2(|\boldsymbol{C}_N|)}{2N} - \frac{1}{2}\log_2\big(2\pi e D \big) \\
&= \lim_{N\to\infty} \frac{\log_2 |\boldsymbol{C}_N|}{2N} - \frac{1}{2}\log_2 D \\
&= \lim_{N\to\infty} \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \xi_i^{(N)} - \frac{1}{2}\log_2 D
\end{aligned} \tag{287}$$

Grenander and Szegö's theorem

- Assume zero-mean process: $\boldsymbol{C}_N = \boldsymbol{R}_N$
- Given the conditions
  - $\boldsymbol{R}_N$ is a sequence of Hermitian Toeplitz matrices with elements ϕ_k on the k-th diagonal
  - The infimum $\Phi_{\inf} = \inf_\omega \Phi(\omega)$ and supremum $\Phi_{\sup} = \sup_\omega \Phi(\omega)$ of the Fourier series are finite

$$\Phi(\omega) = \sum_{k=-\infty}^{\infty} \phi_k \, e^{-j\omega k} \tag{288}$$

  - The function G is continuous in the interval $[\Phi_{\inf}, \Phi_{\sup}]$
- The following expression holds

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} G\left(\xi_i^{(N)}\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G\left(\Phi(\omega)\right) \, \mathrm{d}\omega \tag{289}$$

where $\xi_i^{(N)}$, for $i = 0, 1, \ldots, N-1$, denote the eigenvalues of the N-th matrix $\boldsymbol{R}_N$

Shannon Lower Bound for Gaussian Sources with Memory

- We have already derived

$$R_L(D) = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \xi_i^{(N)} - \frac{1}{2} \log_2 D \tag{290}$$

- Applying Grenander and Szegö's theorem

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} G\left(\xi_i^{(N)}\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G\left(\Phi(\omega)\right) \, \mathrm{d}\omega \tag{291}$$

yields

$$\begin{aligned} R_L(D) &= \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) \, \mathrm{d}\omega - \frac{1}{2} \log_2 D \\ &= \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) \, \mathrm{d}\omega - \frac{1}{4\pi} \log_2 D \int_{-\pi}^{\pi} \mathrm{d}\omega \\ &= \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \frac{\Phi_{SS}(\omega)}{D} \, \mathrm{d}\omega \end{aligned} \tag{292}$$

Power Spectral Density of a Gauss-Markov Process

- Zero-mean Gauss-Markov process with $|\rho| < 1$

$$S_n = Z_n + \rho \cdot S_{n-1} \tag{293}$$

- Auto-correlation function

$$\phi[k] = \sigma^2 \cdot \rho^{|k|} \tag{294}$$

- Using the relationship

$$\sum_{k=1}^{\infty} a^k \, e^{-jkx} = \frac{a}{e^{-jx} - a} \tag{295}$$

we obtain

$$\begin{aligned}
\Phi_{SS}(\omega) &= \sum_{k=-\infty}^{\infty} \phi[k] \cdot e^{-j\omega k} \\
&= \sum_{k=-\infty}^{\infty} \sigma^2 \cdot \rho^{|k|} \cdot e^{-j\omega k} \\
&= \sigma^2 \cdot \left(1 + \frac{\rho}{e^{-j\omega} - \rho} + \frac{\rho}{e^{j\omega} - \rho} \right) \\
&= \sigma^2 \cdot \frac{1 - \rho^2}{1 - 2\rho \cos\omega + \rho^2}
\end{aligned} \tag{296}$$

Shannon Lower Bound for Gaussian-Markov Processes

- Shannon lower bound for a zero-mean Gauss-Markov process with $|\rho| < 1$

$$\begin{aligned} R_L(D) &= \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \frac{\Phi_{SS}(\omega)}{D} \, \mathrm{d}\omega \\ &= \frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2 \frac{\sigma^2(1-\rho^2)}{D} \, \mathrm{d}\omega - \\ &\qquad \underbrace{\frac{1}{4\pi} \int_{-\pi}^{\pi} \log_2(1 - 2\rho\cos\omega + \rho^2) \, \mathrm{d}\omega}_{=0} \end{aligned}$$

$$R_L(D) = \frac{1}{2} \log_2 \frac{\sigma^2 \, (1-\rho^2)}{D} \tag{297}$$

where we used

$$\int_0^{\pi} \ln(a^2 - 2ab\cos x + b^2) \, \mathrm{d}x = 2\pi \ln a \tag{298}$$

- Shannon lower bound as distortion-rate function

$$\boxed{D_L(R) = (1-\rho^2) \, \sigma^2 \, 2^{-2R}} \tag{299}$$

Rate-Distortion Function for Gaussian IID Sources

- Consider Gaussian iid source

$$f_S(s) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{(s-\mu)^2}{2\sigma^2}} \tag{300}$$

- Shannon lower bound for Gaussian iid sources

$$\boxed{D_L(R) = \sigma^2 \cdot 2^{-2R}} \quad \Longleftrightarrow \quad \boxed{R_L(D) = \begin{cases} \frac{1}{2}\log_2 \frac{\sigma^2}{D} & : \quad D \leq \sigma^2 \\ 0 & : \quad D > \sigma^2 \end{cases}} \tag{301}$$

- For Gaussian iid sources: Rate-distortion function = Shannon lower bound
- How can we proof it?
  - Could show that Shannon lower bound is achievable

  $\Longrightarrow$ Need to find $g_{S'|S}(s'|s)$ for which the Shannon lower bound is achieved
- Remember: Discussed that Shannon lower bound is achievable if
  - Difference signal $Z = S - S'$ is independent of S'
  - Difference signal $Z = S - S'$ has a zero-mean Gaussian distribution

Rate-Distortion Function for Gaussian IID Sources

Consider conditional pdf $g_{Z|S'}(z|s') = g_{S-S'|S'}(s-s'|s')$ instead of $g_{S'|S}(s'|s)$

- Given $g_{Z|S'}(z|s')$, conditional pdf $g_{S'|S}(s'|s)$ can be derived by

$$g_{S'|S}(s'|s) = g_{S|S'}(s|s') \cdot \frac{f_{S'}(s')}{f_S(s)} \qquad \text{with} \qquad g_{S|S'}(s|s') = g_{Z|S'}(z+s'|s') \tag{302}$$

- Shannon lower bound coincides with rate-distortion function, only if the difference signal $Z = S - S'$ fulfills the conditions:
  - Difference signal $Z = S - S'$ is independent of S'
  - Difference signal $Z = S - S'$ has a zero-mean Gaussian distribution
- Hence, $g_{Z|S'}(z|s')$ has to have the form

$$g_{Z|S'}(z|s') = \frac{1}{\sqrt{2\pi\sigma_Z^2}}\, e^{-\frac{z^2}{2\sigma_Z^2}} = \frac{1}{\sqrt{2\pi D}}\, e^{-\frac{z^2}{2D}} = f_Z(z) \tag{303}$$

- Need to verify that this is a valid choice!

Rate-Distortion Function for Gaussian IID Sources

- Question: Is the conditional pdf $g_{Z|S'}(z|s')$ a valid choice?

$$g_{Z|S'}(z|s') = f_Z(z) = \frac{1}{\sqrt{2\pi D}}\, e^{-\frac{z^2}{2D}} \tag{304}$$

- Source S is the sum of two independent random variables $Z = S - S'$ and S'
- Hence, $f_S(s)$ is given by the convolution

$$f_S(s) = f_Z(z) * f_{S'}(s') \tag{305}$$

- Note: Convolution of two Gaussians $f(\mu_1, \sigma_1^2)$ and $f(\mu_2, \sigma_2^2)$ is a Gaussian with $\mu = \mu_1 + \mu_2$ and $\sigma = \sigma_1^2 + \sigma_2^2$
- Hence, the pdf of the reconstructed samples is

$$f_{S'}(s') = \frac{1}{\sqrt{2\pi\,(\sigma^2 - D)}}\, e^{-\frac{(s'-\mu)^2}{2(\sigma^2 - D)}} \tag{306}$$

- This is a valid pdf for S' (no negative values)

$\Longrightarrow$ Our choice for $g_{Z|S'}(z|s')$ is valid

Rate-Distortion Function for Gaussian IID Sources

Check distortion and rate (mutual information)

- Distortion given by variance of difference process $Z = S - S'$

$$\delta(g) = E\{(S - S')^2\} = E\{Z^2\} = D \tag{307}$$

- Mutual information

$$\begin{aligned} I(g) &= h(S) - h(S|S') \\ &= h(S) - h(S - S'|S') \\ &= h(S) - h(Z|S') \\ &= h(S) - h(Z) \\ &= \frac{1}{2}\log_2\big(2\pi e\sigma^2\big) - \frac{1}{2}\log_2\big(2\pi eD\big) \\ &= R(D) = \frac{1}{2}\log_2\frac{\sigma^2}{D} \end{aligned} \tag{308}$$

$\Longrightarrow$ **For Gaussian iid processes and MSE distortion, the rate-distortion function coincides with the Shannon lower bound**

Rate-Distortion Function for Gaussian IID Sources

- Considered Gaussian iid source with a variance σ^2 and MSE distortion
- Shannon lower bound coincides with the rate-distortion function
- The rate-distortion function $R(D)$ is given by

$$\boxed{R(D) = \begin{cases} \frac{1}{2}\log_2 \frac{\sigma^2}{D}, & 0 \le D \le \sigma^2 \\ 0, & D > \sigma^2 \end{cases}} \tag{309}$$

- The distortion-rate function is given as

$$\boxed{D(R) = \sigma^2 \cdot 2^{-2R}} \tag{310}$$

- The **signal-to-noise ratio** (SNR) is given as

$$\boxed{\mathrm{SNR}(R) = 10 \cdot \log_{10} \frac{\sigma^2}{D(R)} = 10 \cdot \log_{10} 2^{2R} \approx 6R \quad [\mathrm{dB}]} \tag{311}$$

- **For MSE distortion and a given variance σ^2, the rate-distortion function $R(D)$ is maximized for Gaussian iid processes**

  $\Longrightarrow$ **Gaussian iid processes are the hardest to code**

Rate-Distortion Function for Gaussian Sources with Memory

- N-th order pdf of stationary Gaussian random process

$$f_{\boldsymbol{S}}^{(G)}(\boldsymbol{s}) = \frac{1}{(2\pi)^{N/2}\,|\boldsymbol{C}_N|^{1/2}}\; e^{-\frac{1}{2}(\boldsymbol{s}-\boldsymbol{\mu}_N)^T \boldsymbol{C}_N^{-1}(\boldsymbol{s}-\boldsymbol{\mu}_N)} \tag{312}$$

- Eigendecomposition of covariance matrix $\boldsymbol{C}_N$,

$$\boldsymbol{C}_N = \boldsymbol{A}_N \cdot \boldsymbol{\Xi}_N \cdot \boldsymbol{A}_N^T \tag{313}$$

- $\boldsymbol{A}_N$: Matrix with columns are equal to the N unit-norm eigenvectors

$$\boldsymbol{A}_N = \left(\boldsymbol{v}_0^{(N)}, \boldsymbol{v}_1^{(N)}, \cdots, \boldsymbol{v}_{N-1}^{(N)} \right) \tag{314}$$

- $\boldsymbol{\Xi}_N$: Diagonal matrix with eigenvalues of $\boldsymbol{C}_N$ on its main diagonal

$$\boldsymbol{\Xi}_N = \begin{pmatrix} \xi_0^{(N)} & 0 & \cdots & 0 \\ 0 & \xi_1^{(N)} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \xi_{N-1}^{(N)} \end{pmatrix} \tag{315}$$

Signal Space Rotation

- Given stationary Gaussian source $\{S_n\}$: Construct source $\{U_n\}$ by decomposing $\{S_n\}$ into vectors $\boldsymbol{S}$ of size N and applying the transform

$$\boldsymbol{U} = \boldsymbol{A}_N^{-1} \left(\boldsymbol{S} - \boldsymbol{\mu}_N \right) = \boldsymbol{A}_N^T \left(\boldsymbol{S} - \boldsymbol{\mu}_N \right) \tag{316}$$

- Linear transformation of a Gaussian random vector results in another Gaussian random vector
- The chosen transform yields independent random variables U_i

$$f_{\boldsymbol{U}}(\boldsymbol{u}) = \frac{1}{(2\pi)^{N/2} \left| \boldsymbol{\Xi}_N \right|^{1/2}} \, e^{-\frac{1}{2} \boldsymbol{u}^T \boldsymbol{\Xi}_N^{-1} \boldsymbol{u}} = \prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi \xi_i^{(N)}}} \, e^{-\frac{u_i^2}{2\,\xi_i^{(N)}}} \tag{317}$$

- Mean

$$E\{\boldsymbol{U}\} = \boldsymbol{A}_N^T \left(E\{\boldsymbol{S}\} - \boldsymbol{\mu}_N \right) = \boldsymbol{A}_N^T \left(\boldsymbol{\mu}_N - \boldsymbol{\mu}_N \right) = \boldsymbol{0} \tag{318}$$

- Covariance

$$\begin{aligned} E\Big\{ \boldsymbol{U}\, \boldsymbol{U}^T \Big\} &= \boldsymbol{A}_N^T \, E\Big\{ \left(\boldsymbol{S} - \boldsymbol{\mu}_N \right) \left(\boldsymbol{S} - \boldsymbol{\mu}_N \right)^T \Big\} \, \boldsymbol{A}_N \\ &= \boldsymbol{A}_N^T \, \boldsymbol{C}_N \, \boldsymbol{A}_N = \boldsymbol{\Xi}_N \end{aligned} \tag{319}$$

Distortion and Mutual Information

- Inverse transform after compression identical to forward transform

$$\boldsymbol{S'} = \boldsymbol{A}_N \, \boldsymbol{U'} + \boldsymbol{\mu}_N, \tag{320}$$

- With

$$\big(\boldsymbol{U'} - \boldsymbol{U}\big) = \boldsymbol{A}_N^T \big(\boldsymbol{S'} - \boldsymbol{S}\big) \quad \Longleftrightarrow \quad \big(\boldsymbol{S'} - \boldsymbol{S}\big) = \boldsymbol{A}_N \big(\boldsymbol{U'} - \boldsymbol{U}\big) \tag{321}$$

- MSE distortion between any realization $\boldsymbol{s}$ of $\boldsymbol{S}$ and its reconstruction $\boldsymbol{s'}$

$$\begin{aligned} d_N(\boldsymbol{s}; \boldsymbol{s'}) &= \frac{1}{N} \sum_{i=0}^{N-1} (s_i - s'_i)^2 = \frac{1}{N} (\boldsymbol{s} - \boldsymbol{s'})^T (\boldsymbol{s} - \boldsymbol{s'}) \\ &= \frac{1}{N} (\boldsymbol{u} - \boldsymbol{u'})^T \boldsymbol{A}_N^T \boldsymbol{A}_N (\boldsymbol{u} - \boldsymbol{u'}) = \frac{1}{N} (\boldsymbol{u} - \boldsymbol{u'})^T (\boldsymbol{u} - \boldsymbol{u'}) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} (u_i - u'_i)^2 = d_N(\boldsymbol{u}; \boldsymbol{u'}) \end{aligned} \tag{322}$$

- Since coordinate transform is invertible,

$$I_N(\boldsymbol{S}; \boldsymbol{S'}) = I_N(\boldsymbol{U}; \boldsymbol{U'}) \tag{323}$$

Distortion-Rate Function

- Mutual information and average distortion considering independence of the components U_i

$$I_N(g_N^Q) = \sum_{i=0}^{N-1} I_1(g_i^Q) \qquad \text{and} \qquad \delta_N(g_N^Q) = \frac{1}{N} \sum_{i=0}^{N-1} \delta_1(g_i^Q) \tag{324}$$

- N-th order distortion rate function $D_N(R)$

$$D_N(R) = \frac{1}{N} \sum_{i=0}^{N-1} D_i(R_i) \qquad \text{with} \qquad R = \frac{1}{N} \sum_{i=0}^{N-1} R_i \tag{325}$$

- $D_i(R_i)$: Distortion-rate function for Gaussian iid processes for component U_i

$$D_i(R_i) = \sigma_i^2 \, 2^{-2R_i} = \xi_i^{(N)} \, 2^{-2R_i} \tag{326}$$

with $\xi_i^{(N)}$ being the eigenvalues of $\boldsymbol{C}_N$

Optimal Bit Allocation

- Have to distribute the bit rate in an optimal way

$$\min_{R_0,R_1,\ldots,R_{N-1}} D_N(R) = \frac{1}{N}\sum_{i=0}^{N-1} \xi_i^{(N)}\, 2^{-2R_i} \qquad \text{such that} \qquad R \geq \frac{1}{N}\sum_{i=0}^{N-1} R_i$$

- Comparison on different types of mean computations

$$D_N(R) = \frac{1}{N}\sum_{i=0}^{N-1} \xi_i^{(N)}\, 2^{-2R_i} \geq \left(\prod_{i=0}^{N-1} \xi_i^{(N)}\, 2^{-2R_i}\right)^{\frac{1}{N}} = \underbrace{\left(\prod_{i=0}^{N-1} \xi_i^{(N)}\right)^{\frac{1}{N}}}_{=|\boldsymbol{C}_N|^{\frac{1}{N}}=\tilde{\xi}^{(N)}} \cdot\, 2^{-2R}$$

  with $\prod_{i=0}^{N-1} 2^{-2R_i} = 2^{-2R_0}\cdot 2^{-2R_1}\cdots 2^{-2R_{N-1}} = 2^{-\sum_{i=0}^{N-1} 2R_i} = 2^{-2RN}$

- Expression on the right-hand side of above inequality is constant: equality achieved when all terms $\xi_i^{(N)}\, 2^{-2R_i} = \tilde{\xi}^{(N)} 2^{-2R}$

$$R_i = R + \frac{1}{2}\log_2 \frac{\xi_i^{(N)}}{\tilde{\xi}^{(N)}} = \frac{1}{2}\log_2 \frac{\xi_i^{(N)}}{\tilde{\xi}^{(N)} 2^{-2R}} \qquad \text{with} \qquad \tilde{\xi}^{(N)} = \left(\prod_{i=0}^{N-1} \xi_i^{(N)}\right)^{\frac{1}{N}}$$

Condition for Partial Bit Rates

- So far, we have ignored that R_i cannot be less than 0

$$R_i = \frac{1}{2}\log_2 \frac{\xi_i^{(N)}}{\tilde{\xi}^{(N)} 2^{-2R}} \geq 0 \quad \Longrightarrow \quad R_i = 0 \quad \text{if} \quad \xi_i^{(N)} \leq \tilde{\xi}^{(N)} 2^{-2R} \tag{327}$$

- Introducing the parameter θ, with $0 \leq \theta \leq D$, yields

$$R_i = \begin{cases} \frac{1}{2}\log_2 \frac{\xi_i^{(N)}}{\theta} & : \quad \theta \leq \xi_i^{(N)} \\ 0 & : \quad \theta > \xi_i^{(N)} \end{cases} \tag{328}$$

and

$$D_i = \begin{cases} \theta & : \quad \theta \leq \xi_i^{(N)} \\ \xi_i^{(N)} & : \quad \theta > \xi_i^{(N)} \end{cases} \tag{329}$$

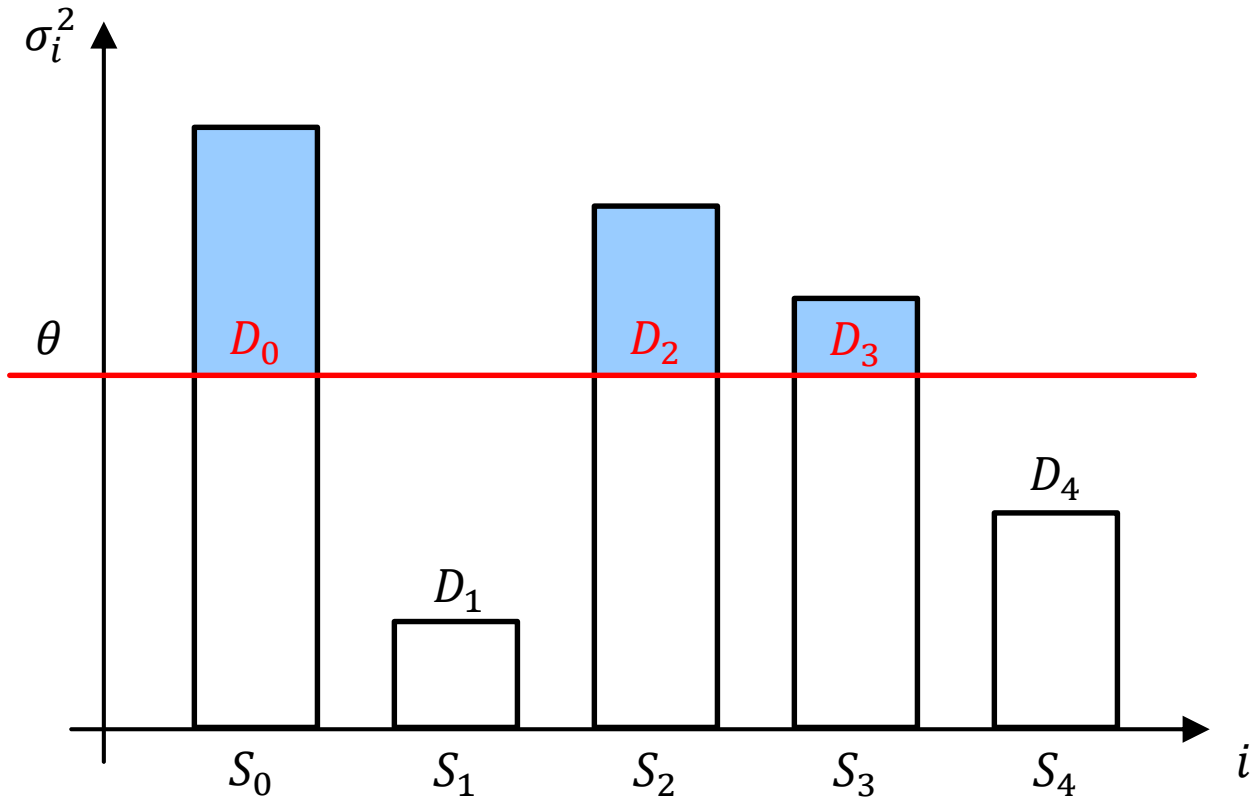
- Can also be written as

$$R_i(\theta) = \max\left(0, \frac{1}{2}\log_2 \frac{\xi_i^{(N)}}{\theta}\right) \quad \text{and} \quad D_i(\theta) = \min\left(\xi_i^{(N)}, \theta\right) \tag{330}$$

- This rate allocation concept is also referred to as **reverse water filling**

Reverse Water Filling for Independents Gaussian RV

$$D_i = \min\left(\sigma_i^2, \theta\right)$$



- Optimal rate allocation for independent Gaussian RV and MSE distortion
- Code random variable with $\sigma_i^2 > \theta$ so that the same distortion is obtained
- Do not assign any rate to random variables with $\sigma_i^2 \leq \theta$

N-th Order Rate-Distortion Function

- N-th order distortion-rate function $D_N(R)$

$$D_N(R) = \frac{1}{N} \sum_{i=0}^{N-1} D_i(R_i) \qquad \text{with} \qquad R = \frac{1}{N} \sum_{i=0}^{N-1} R_i \tag{331}$$

- Optimal rate allocation

$$R_i(\theta) = \max\left(0, \frac{1}{2} \log_2 \frac{\xi_i^{(N)}}{\theta}\right) \quad \text{and} \quad D_i(\theta) = \min\left(\xi_i^{(N)}, \theta\right) \tag{332}$$

- Parametric expressions for N-th order rate-distortion function

$$D_N(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} D_i = \frac{1}{N} \sum_{i=0}^{N-1} \min\left(\xi_i^{(N)}, \theta\right) \tag{333}$$

$$R_N(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} R_i = \frac{1}{N} \sum_{i=0}^{N-1} \max\left(0, \frac{1}{2} \log_2 \frac{\xi_i^{(N)}}{\theta}\right) \tag{334}$$

Parametric Rate-Distortion Function

- Rate-distortion function is given by limit for $N \to \infty$

$$D(\theta) = \lim_{N\to\infty} D_N(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \min\left(\xi_i^{(N)}, \theta\right) \tag{335}$$

$$R(\theta) = \lim_{N\to\infty} R_N(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \max\left(0, \frac{1}{2}\log_2 \frac{\xi_i^{(N)}}{\theta}\right) \tag{336}$$

- Recall: Grenander and Szegös theorem for infinite Toeplitz matrices

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} G(\xi_i^{(N)}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G(\Phi(\omega)) d\omega \tag{337}$$

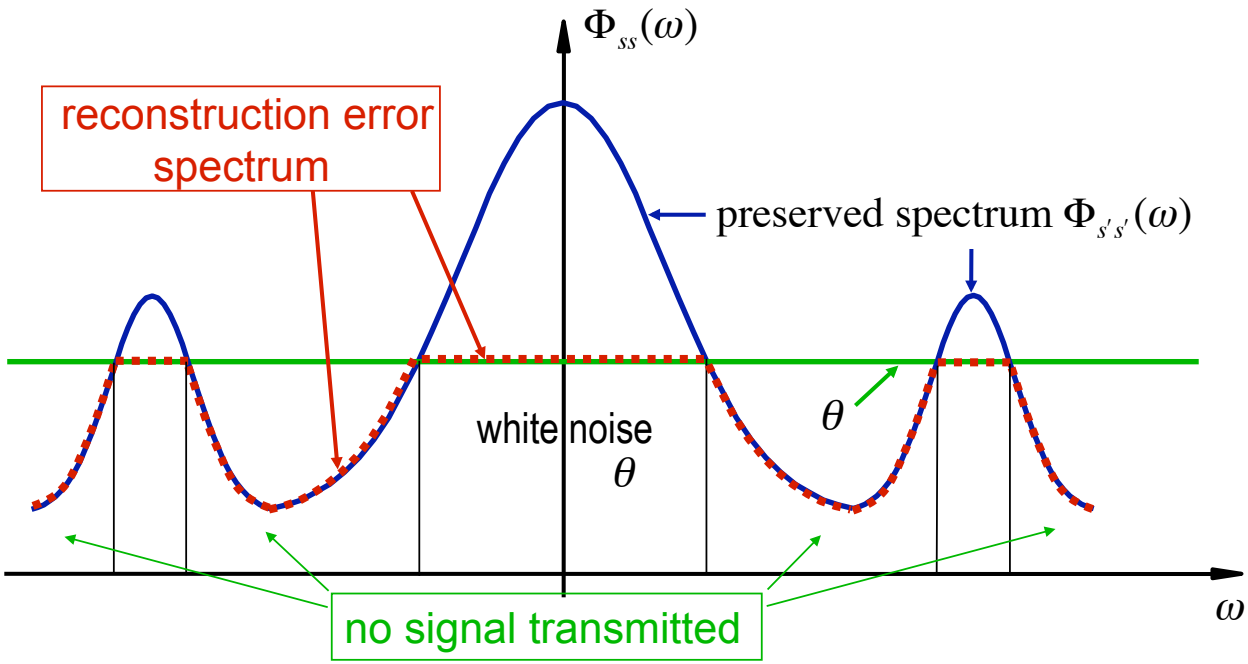
$\Longrightarrow$ **Rate-distortion function $R(D)$ for Gaussian sources with memory**

$$D(\theta) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \min\{\Phi_{SS}(\omega), \theta\} d\omega$$

$$R(\theta) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \max\left\{0, \frac{1}{2}\log_2 \frac{\Phi_{SS}(\omega)}{\theta}\right\} d\omega \tag{338}$$

$\Longrightarrow$ **Specifies upper bound for $R(D)$ of all processes with the same $\Phi_{SS}(\omega)$**

Illustration of Minimization Approach



Similar to reverse water filling

- At each frequency, the variance of the frequency component as given by the spectral density $\Phi_{SS}(\omega)$ is compared to the parameter θ, which represents the target mean squared error of that frequency component
- When $\Phi_{SS}(\omega)$ is found to be larger than θ, the rate $\frac{1}{2}\log_2\frac{\Phi_{ss}(\omega)}{\theta}$ is assigned, otherwise zero rate is assigned to that frequency component

Rate-Distortion Function for Gauss-Markov Sources

- $R(D)$ for zero-mean Gauss-Markov process with $|\rho| < 1$ and variance σ^2

$$S_n = Z_n + \rho \cdot S_{n-1} \tag{339}$$

- Auto-correlation function and spectral density function are given as

$$\phi[k] = \sigma^2 |\rho|^k \qquad \Phi(\omega) = \sum_{k=-\infty}^{\infty} \phi[k] e^{-jk\omega} = \frac{\sigma^2(1-\rho^2)}{1 - 2\rho\cos\omega + \rho^2} \tag{340}$$

- If we choose

$$\theta \geq \min_{\forall\omega} \Phi_{SS}(\omega) = \sigma^2 \frac{1-\rho^2}{1-2\rho+\rho^2} = \sigma^2 \frac{1-\rho}{1+\rho} \tag{341}$$

we obtain

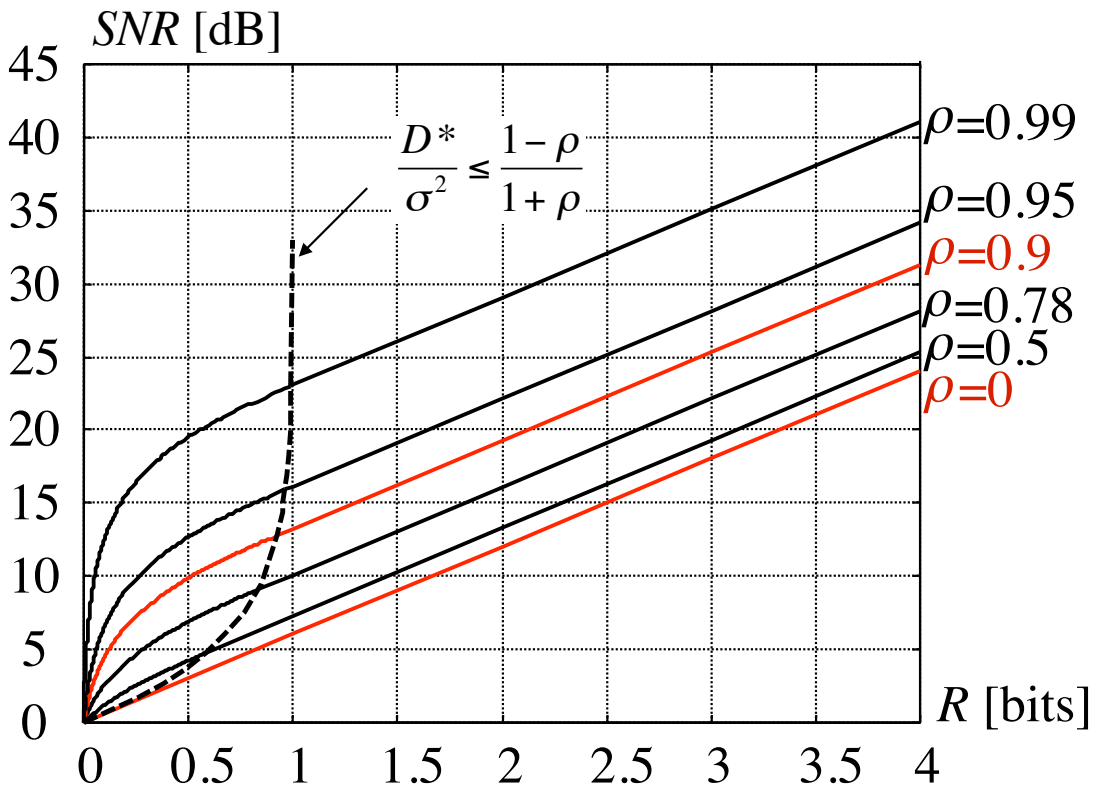
$$\boxed{R(D) = \frac{1}{2} \log_2 \frac{\sigma^2(1-\rho^2)}{D}} \tag{342}$$

Rate-Distortion Function for Gauss-Markov Sources

- Corresponding distortion rate function for $R \geq \log_2(1+\rho)$ is given by

$$\boxed{D(R) = (1-\rho^2) \cdot \sigma^2 \cdot 2^{-2R}} \tag{343}$$

- Includes result for Gaussian iid sources ($\rho = 0$)



Chapter Summary

Rate-distortion theory

- Determine minimum rate R for a given distortion D and source
- Determine minimum distortion D for a given distortion R and source

Operational rate-distortion function

- Fundamental bound as minimum over all possible source codes

Information rate-distortion function

- Minimum over all conditional pdfs $g_{S'|S}(s'|s)$
- Coincides with operational rate-distortion function
- Use term rate-distortion function $R(D)$ for both
- Fundamental bound for lossless coding is given by $R(0)$
- Discrete sources: $R(D)$ is a convex function with $R(0) = \bar{H}(\boldsymbol{S})$
- Continuous sources: $R(D)$ is a convex function with $R(0) \to \infty$
- MSE distortion measure: $D(0) = \sigma^2$

Chapter Summary

Shannon lower bound

- Lower bound of rate-distortion function
- Asymptotically tight for high rates
- Suitable reference for performance evaluation at high rates
- Shannon lower bound $R_L(D)$ can often be computed analytically
- Computed $R_L(D)$ for several iid sources and Gaussian source with memory

Rate-distortion function for Gaussian sources and MSE distortion

- $R(D)$ for Gaussian iid sources coincides with Shannon lower bound
- Any other source than the Gaussian iid source with the same variance requires less bits for same MSE distortion
- $R(D)$ for Gaussian source with memory can be specified as parametric expression using the power spectral density $\Phi_{SS}(\omega)$
- Derived analytic expression for Gauss-Markov source and $R \geq \log_2(1+\rho)$
- $R(D)$ for Gaussian source with memory and a spectral density $\Phi_{SS}(\omega)$ specifies an upper bound for all other sources with the same spectral density

$\Longrightarrow$ Gaussian sources are the most difficult to code

Exercise 11

A fair die is rolled at the same time as a fair coin is tossed. Let A be the number on the upper surface of the die and let B describe the outcome of the coin toss, where B is equal to 1 if the result is "head" and it is equal to 0 if the result if "tail". The random variables X and Y are given by $X = A + B$ and $Y = A - B$, respectively.

Calculate:

- the joint entropy $H(X, Y)$,
- the marginal entropies $H(X)$ and $H(Y)$,
- the conditional entropies $H(X|Y)$ and $H(Y|X)$,
- the mutual information $I(X; Y)$.

Exercise 12

Consider a stationary Gauss-Markov process $\mathbf{X} = \{X_n\}$ with mean μ, variance σ^2, and the correlation coefficient ρ (correlation coefficient between two successive random variables).

Determine the mutual information $I(X_k; X_{k+N})$ between two random variables X_k and X_{k+N}, where the distance between the random variables is N times the sampling interval.

Interpret the results for the special cases $\rho = -1$, $\rho = 0$, and $\rho = 1$.

Hint: In the lecture, we showed

$$E\left\{(\mathbf{X} - \mu_N)^{\mathrm{T}} \cdot \mathbf{C}_N^{-1} \cdot (\mathbf{X} - \mu_N)\right\} = N,$$

which can be useful for the problem.

Exercise 13

Show that for discrete random processes the fundamental bound for lossless coding is a special case of the fundamental bound for lossy coding.

Exercise 14

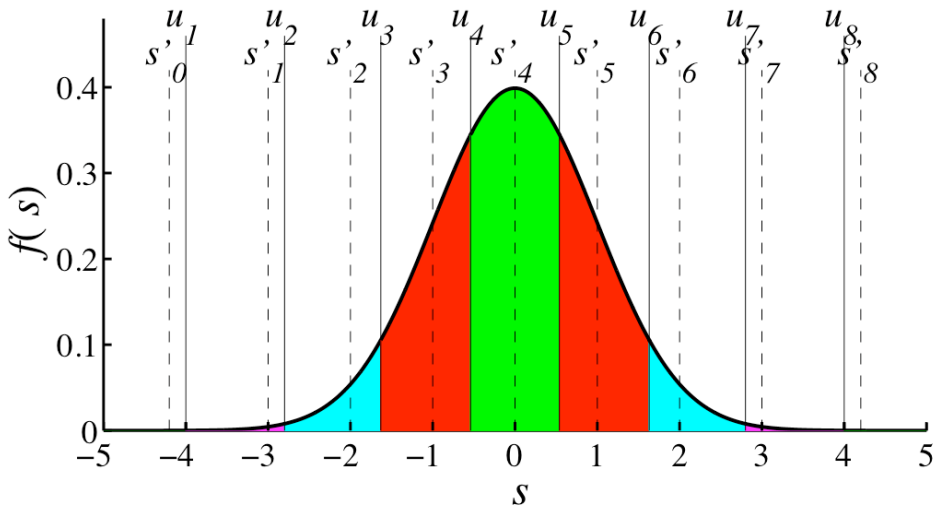
Determine the Shannon lower bound with MSE distortion, as distortion-rate function, for iid processes with the following pdfs:

- The exponential pdf $f_E(x) = \lambda \cdot e^{-\lambda \cdot x}$, with $x \geq 0$
- The zero-mean Laplace pdf $f_L(x) = \frac{\lambda}{2} \cdot e^{-\lambda \cdot |x|}$

Express the distortion-rate function for the Shannon lower bound as a function of the variance σ^2.

Which of the given pdfs is easier to code (if the variance is the same)?

Quantization



Outline

Part I: Source Coding Fundamentals

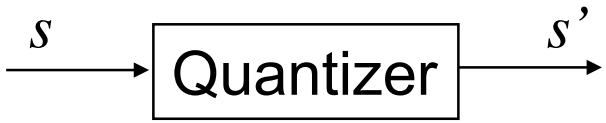
- Probability, Random Variables and Random Processes
- Lossless Source Coding
- Rate-Distortion Theory
- **Quantization**
  - Scalar Quantization
    - Centroid Quantizer and Lloyd Quantizer
    - Entropy-Constrained Scalar Quantization
    - High-Rate Approximations for Scalar Quantizers
  - Vector Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

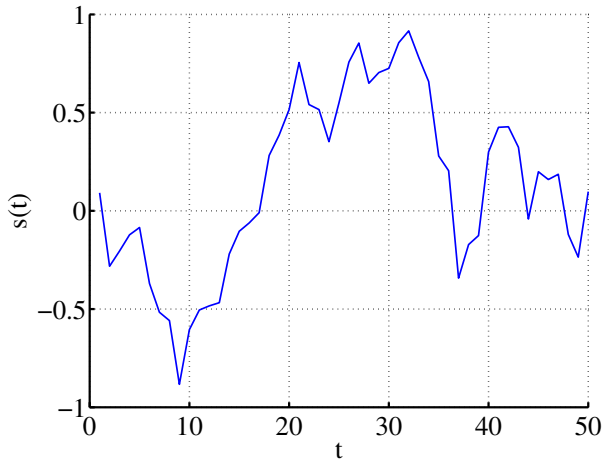
Quantization – Introduction

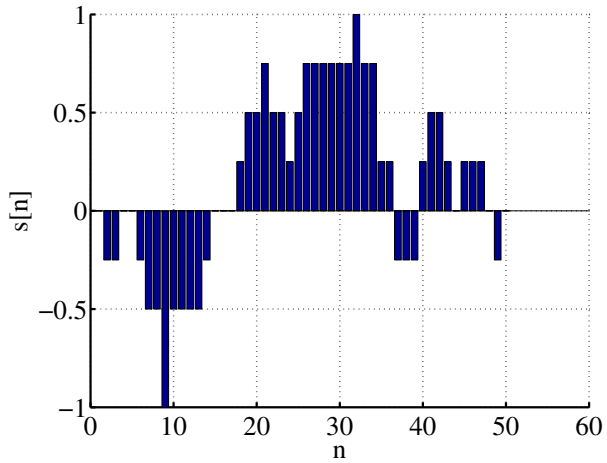
- Quantization is the realization of the "lossy part" of source coding
- Typically allows for a trade-off between signal fidelity and bit rate



- Quantization is a functional mapping of an input point to an output point
  - the input can be discrete or continuous scalars or vectors
  - the set of obtainable output points is countable
  - less obtainable output points than input points

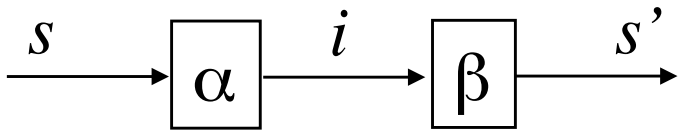
$\Longrightarrow$ Non-reversible loss in signal fidelity





Structure of Quantizers

- Quantizer description is split into encoder α and decoder β, between which a quantization index i is transmitted



- Adding lossless coding γ of quantization indices

- Quantization procedure
  1. Encoder α maps one or more samples of input signal s to indices i
  2. Lossless mapping γ codes the indices i into a bit stream b
  3. Channel outputs transmitted bit stream b' (error-free: $b' = b$)
  4. Inverse lossless mapping γ^{-1} reproduces quantization indices i
  5. Decoder β maps index i to one or more samples of decoded signal s'

Quantizer Mappings

- Encoder mappings α, γ have their counterparts at decoder β, γ^{-1}
- Decoder mappings must be either implemented at receiver and/or transmitted
- General case: Mapping for N-dimensional vectors

$$\boxed{Q: \ \mathbb{R}^N \to \{\boldsymbol{s'}_0, \boldsymbol{s'}_1, \cdots, \boldsymbol{s'}_{K-1}\}} \tag{344}$$

- Quantization cells: Subsets $\mathcal{C}_i$ of the N-dimensional Euclidean space $\mathbb{R}^N$

$$\mathcal{C}_i = \{\boldsymbol{s} \in \mathbb{R}^N : \ Q(\boldsymbol{s}) = \boldsymbol{s'}_i\} \tag{345}$$

- Quantization cells $\mathcal{C}_i$ form partition of the N-dimensional Euclidean space $\mathbb{R}^N$

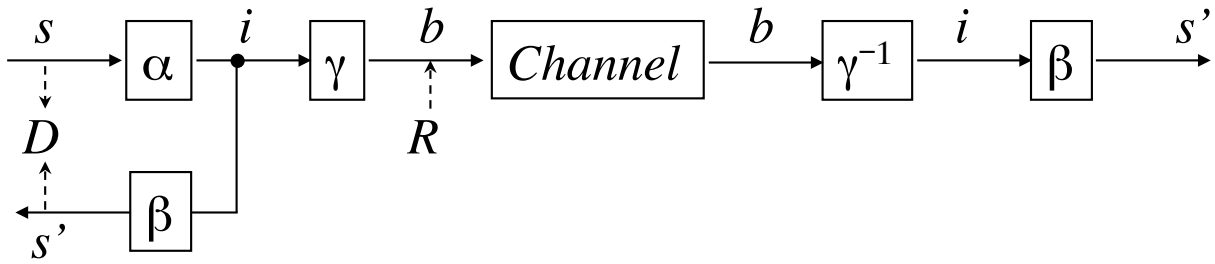
$$\bigcup_{i=0}^{K-1} \mathcal{C}_i = \mathbb{R}^N \qquad \text{with} \qquad \forall i \neq j: \ \ \mathcal{C}_i \cap \mathcal{C}_j = \emptyset \tag{346}$$

- Specify quantization mapping

$$\boxed{Q(\boldsymbol{s}) = \boldsymbol{s'}_i \qquad \forall \boldsymbol{s} \in \mathcal{C}_i} \tag{347}$$

Performance of Quantizers

- Encoder mapping $\alpha : \mathbb{R}^N \to \mathcal{I}$ introduces distortion



- Assume random process $\{\boldsymbol{S}_n\}$ to be stationary: Distortion and rate

$$D = E\{d_N(\boldsymbol{S}_n, Q(\boldsymbol{S}_n))\} = \frac{1}{N} \sum_{i=0}^{K-1} \int_{\mathcal{C}_i} d_N(\boldsymbol{s}, Q(\boldsymbol{s}))\ f_{\boldsymbol{S}}(\boldsymbol{s})\, \mathrm{d}\boldsymbol{s} \tag{348}$$

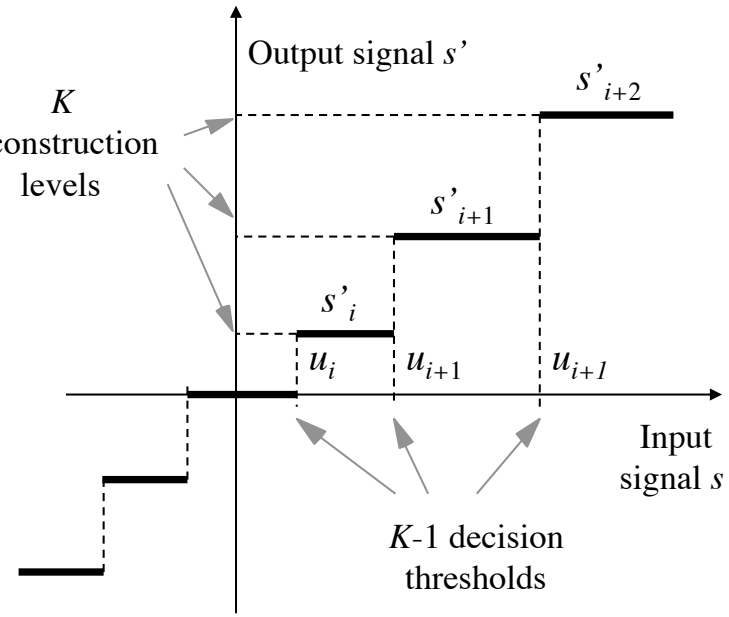
$$R = \frac{1}{N} E\{\, |\gamma(\, Q(\boldsymbol{S}_n)\,)|\,\} = \frac{1}{N} \sum_{i=0}^{N-1} p_i \cdot |\gamma(\boldsymbol{s}'_i)| = \frac{1}{N} \sum_{i=0}^{N-1} p_i \cdot \ell_i \tag{349}$$

where $|\gamma(\boldsymbol{s}'_i)|$ denotes codeword length ℓ_i and p_i denotes the pmf for $\boldsymbol{s}'_i$

$$p_i = p(\boldsymbol{s}'_i) = \int_{\mathcal{C}_i} f_{\boldsymbol{S}}(\boldsymbol{s})\ \mathrm{d}\boldsymbol{s} \tag{350}$$

Scalar Quantization

- Input/output function of a scalar quantizer



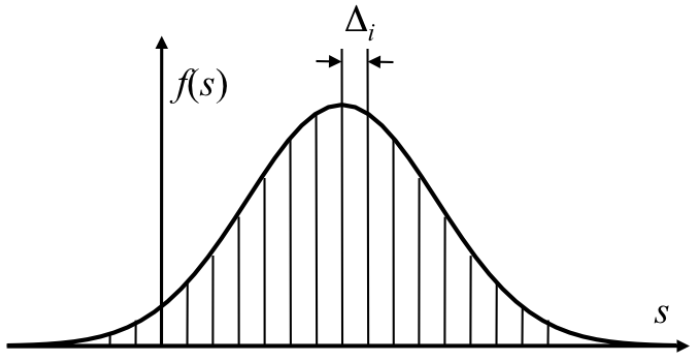
- A scalar (one-dimensional) quantizer is a mapping

$$\boxed{Q : \mathbb{R} \to \{s'_0, s'_1, \ldots, s'_{K-1}\}} \tag{351}$$

- Quantization cells $\mathcal{C}_i = [u_i, u_{i+1})$ with $u_0 = -\infty$ and $u_K = \infty$
- Step size for reconstruction level i is denoted as $\Delta_i = u_{i+1} - u_i$

Performance of Scalar Quantizers

- Scalar quantization of an amplitude-continuous random variable S can be viewed as a discretization of its continuous pdf $f(s)$



- Average MSE distortion is given as

$$D = E\{d_1(S, Q(S))\} = E\{d_1(S, S')\} = \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} (s - s'_i)^2 \cdot f(s)\, \mathrm{d}s \tag{352}$$

- Average rate is given by the expectation value of the codeword length

$$R = E\{|\gamma(Q(S))|\} = \sum_{i=0}^{N-1} p_i \cdot |\gamma(s'_i)| = \sum_{i=0}^{N-1} p_i \cdot \ell_i \tag{353}$$

- Goal of design: Optimize mappings α (i.e. u_i), β (i.e. s'_i), and γ

Scalar Quantization with Fixed-Length Codes

- Consider restriction on lossless mapping γ:
  $\Longrightarrow$ Assign codeword of same length to all quantization indices
- Quantizer of size K:
  $\Longrightarrow$ Codeword length must be greater than or equal to $\lceil \log_2 K \rceil$
- If K is not a power of 2, quantizer requires the same minimum codeword length as a quantizer of size $K' = 2^{\lceil \log_2 K \rceil}$
- Since $K < K'$, quantizer of size K' can achieve a smaller distortion
- Define rate according to

$$R = \log_2 K, \tag{354}$$

  while only considering quantizer sizes K that represent integer powers of 2

Simplest Case: Pulse-Code-Modulation (PCM)

- PCM: Uniform mappings α and β
  - All quantization intervals have same size Δ
  - Reconstruction values s'_i lie in the middle of the intervals
- PCM for random processes with amplitude range $[s_{\min}, s_{\max}]$

$$A = s_{\max} - s_{\min} \qquad \Longrightarrow \qquad \Delta = \frac{A}{K} = A \cdot 2^{-R} \tag{355}$$

- Quantization mapping

$$Q(s) = \text{round}\left(\frac{s - s_{\min}}{\Delta} + 0.5\right) \cdot \Delta + s_{\min} \tag{356}$$

- Example: Uniform distribution $f(s) = \frac{1}{A}$ for $-\frac{A}{2} \leq s \leq \frac{A}{2}$

$$D = \sum_{i=0}^{K-1} \int_{s_{\min}+i\Delta}^{s_{\min}+(i+1)\Delta} \frac{1}{A} \left(s - s_{\min} - \left(i + \frac{1}{2}\right) \cdot \Delta\right)^2 \, \mathrm{d}s \tag{357}$$

- Resulting operational rate distortion function

$$D_{\text{PCM,uniform}}(R) = \frac{A^2}{12} \cdot 2^{-2R} = \sigma^2 \cdot 2^{-2R} \tag{358}$$

PCM for Sources with Infinite Support

- In general, interval limits u_i can be chosen as

$$u_0 = -\infty, \quad u_K = \infty, \quad u_{i+1} - u_i = \Delta \text{ for } 1 \leq i \leq K-1 \tag{359}$$

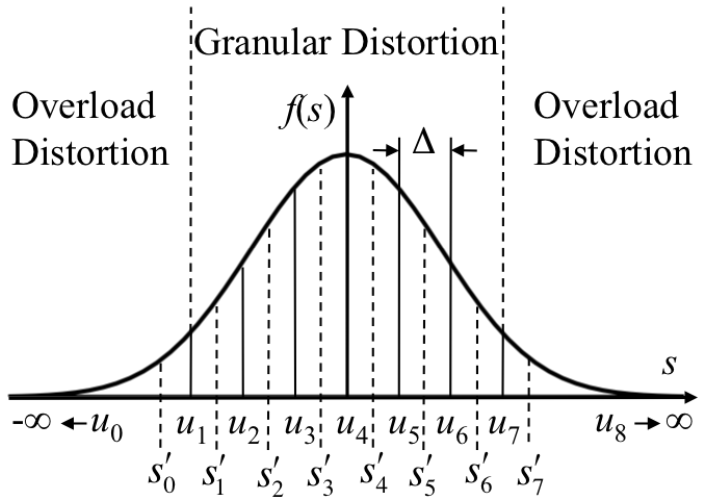
- Symmetric pdfs: Reconstruction symbols s_i with $0 \leq i < K$ and interval boundaries u_i with $0 < i < K$

$$s'_i = (i - \frac{K-1}{2}) \cdot \Delta \qquad u_i = \left(i - \frac{K}{2}\right) \cdot \Delta \tag{360}$$

- Distortion D is split into granular distortion D_G and overload distortion D_O

$$D(\Delta) = D_G(\Delta) + D_O(\Delta)$$

- Optimum Δ for given rate R?



- Distortion minimization by balancing granular and overload distortion

$$\min_{\Delta} D(\Delta) = \min_{\Delta} \left[D_G(\Delta) + D_O(\Delta)\right] \tag{361}$$

Overload and Granular Distortion

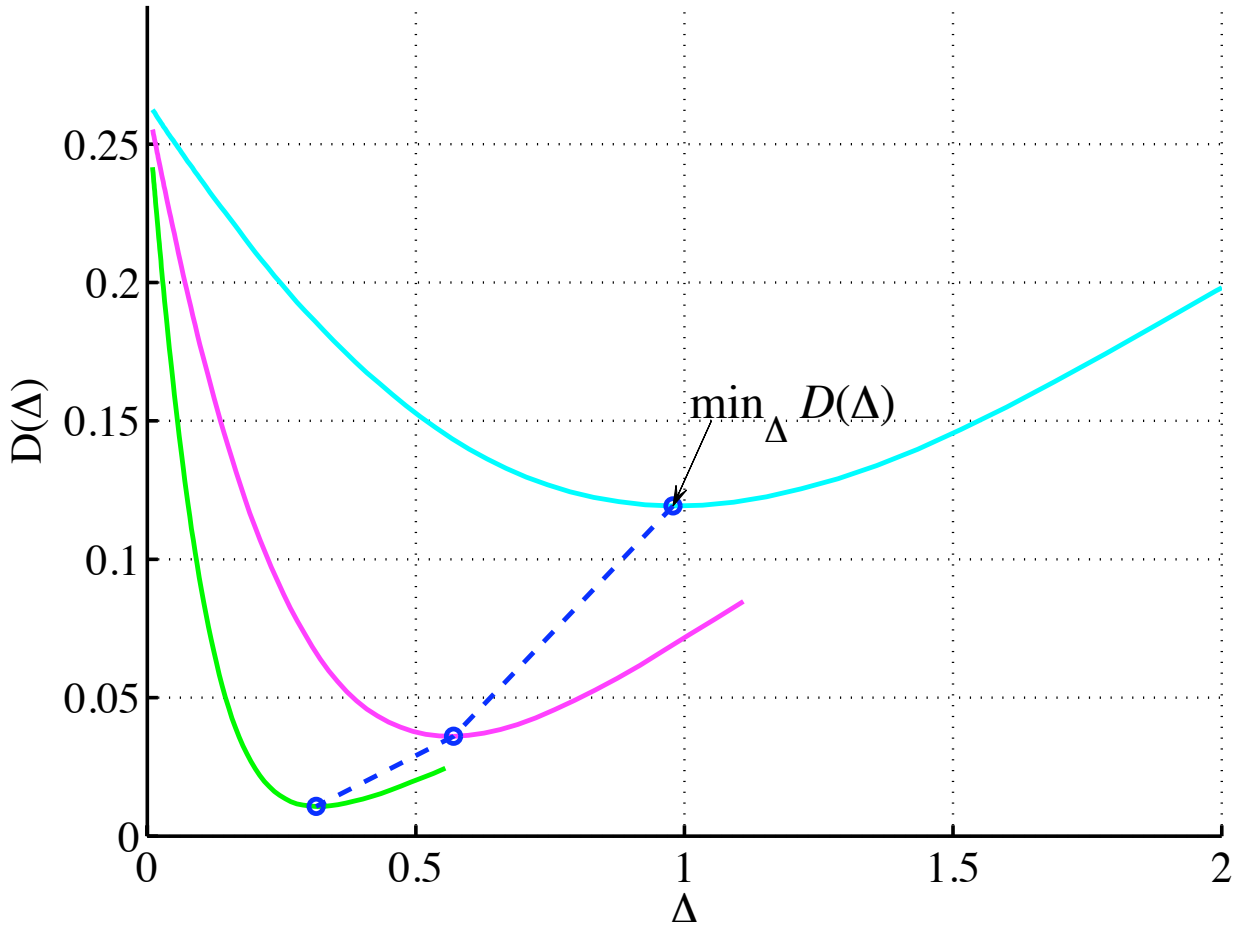
- Average distortion for PCM for sources with infinite support

$$
\begin{aligned}
D(\Delta) &= \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} (s - s'_i)^2 \cdot f(s)\,\mathrm{d}s \\
&= \underbrace{\int_{-\infty}^{(-\frac{K}{2}+1)\Delta} (s - s'_0)^2 f(s)\mathrm{d}s}_{\text{overload distortion}} \\
&\quad + \underbrace{\sum_{i=1}^{K-2} \int_{(i-\frac{K}{2})\Delta}^{(i+1-\frac{K}{2})\Delta} (s - s'_i)^2 f(s)\,\mathrm{d}s}_{\text{granular distortion}} \\
&\quad + \underbrace{\int_{(\frac{K}{2}-1)\Delta}^{\infty} (s - s'_{K-1})^2 f(s)\,\mathrm{d}s}_{\text{overload distortion}}
\end{aligned}
\tag{362}
$$

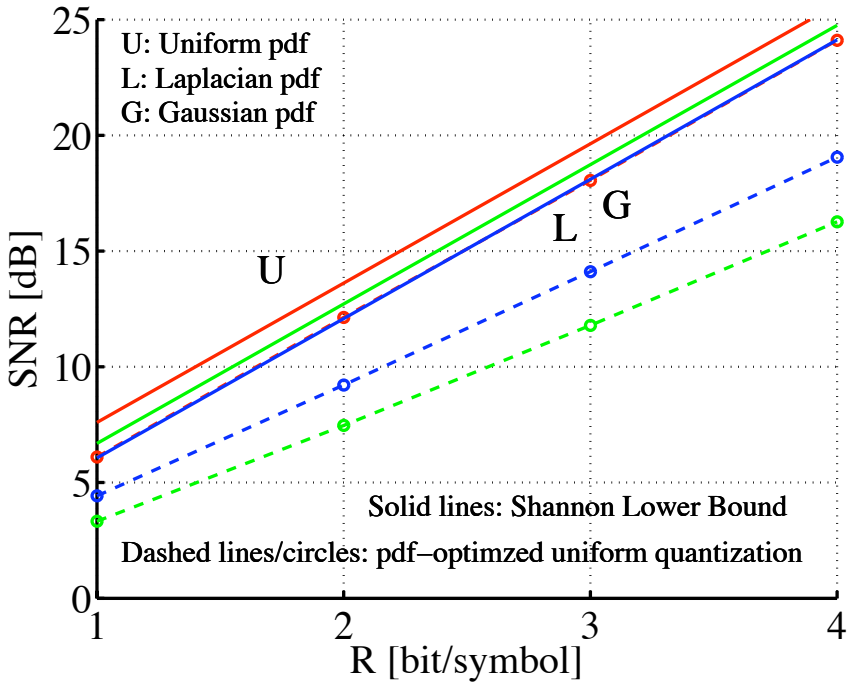
- In general: Optimum step size Δ_{opt} cannot be analytically calculated
  $\Longrightarrow$ Numerical optimization

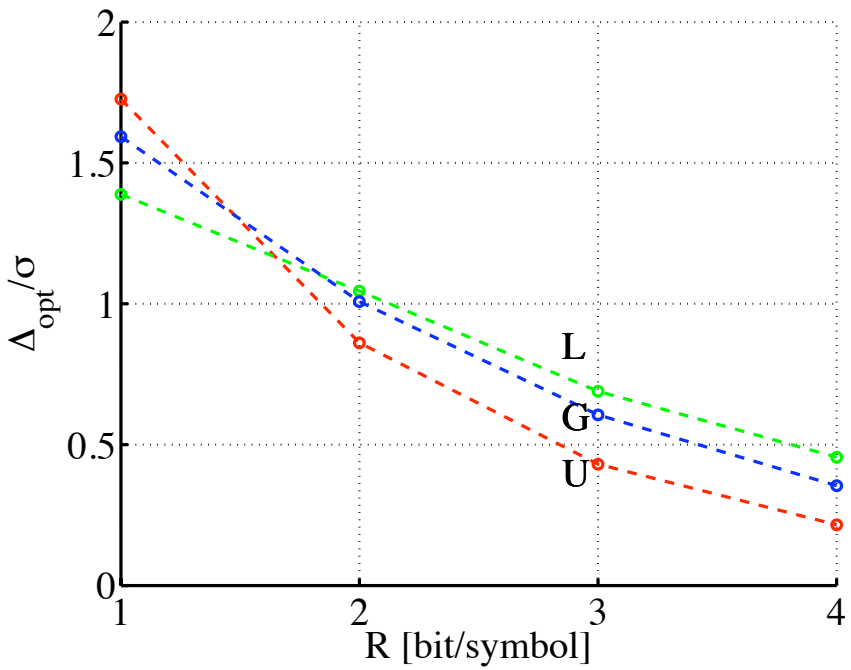
Optimum Step Size for PCM

- Distortion $D(\Delta)$ vs. step size Δ for a Gaussian pdf with unit variance
- Cyan: $R = 2$, Magenta: $R = 3$, Green: $R = 4$ bit/sample



Numerical Optimization Results for PCM Quantization





- Numerical minimization of distortion by varying Δ
- Loss in SNR is large and increases towards higher rates
- Improvement through pdf-optimized quantizers
  - $\Longrightarrow$ Make quantization step sizes Δ_i variable?
  - $\Longrightarrow$ Modify placement of s'_i inside a quantization interval?
  - $\Longrightarrow$ Use variable length codes?

Optimality for Decoding Mapping: Centroid Condition

- Assume given decision thresholds and consider optimal reconstruction values
- Distortion D_i inside a quantization interval $\mathcal{C}_i$

$$D_i = \int_{u_i}^{u_{i+1}} d_1(s, s'_i) \cdot f(s|s'_i)\, \mathrm{d}s = E\{d_1(S, s'_i) |\, S \in \mathcal{C}_i\} \tag{363}$$

- Probability that a source symbol falls inside quantization interval $\mathcal{C}_i$

$$p_i = \int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s \tag{364}$$

- Average distortion

$$D = \sum_{i=0}^{K-1} p_i \cdot D_i = \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} d_1(s, s'_i) \cdot f(s)\, \mathrm{d}s \tag{365}$$

- Since p_i does not depend on s'_i, the optimality criterion is

$$\boxed{s'^{*}_i = \arg\min_{s' \in \mathbb{R}} E\{d_1(S, s') |\, S \in \mathcal{C}_i\}} \tag{366}$$

$\Longrightarrow$ **General centroid condition**

Centroid Condition for MSE Distortion

- Given a random variable X, the value of y that minimizes $E\{(X-y)^2\}$ is

$$y = E\{X\} \tag{367}$$

which can be shown by

$$\begin{aligned} E\{(X-y)^2\} &= E\{(X-E\{X\}+E\{X\}-y)^2\} \\ &= E\{(X-E\{X\})^2\} + (E\{X\}-y)^2 \\ &\geq E\{(X-E\{X\})^2\} \end{aligned} \tag{368}$$

- Consequently, given an event $\mathcal{A}$, the value y that minimizes

$$E\{(X-y)^2 | X \in \mathcal{A}\} \tag{369}$$

is

$$y = E\{X | X \in \mathcal{A}\} \tag{370}$$

Centroid Condition for MSE Distortion

- General centroid condition

$$s_i'^{*} = \arg\min_{s' \in \mathbb{R}} \; E\{d_1(S, s') |\, S \in \mathcal{C}_i\} \tag{371}$$

- MSE distortion

$$d_1(x, y) = (x - y)^2 \tag{372}$$

- The value of s_i' that minimizes the centroid condition is

$$s_i'^{*} = E\{S |\, S \in \mathcal{C}_i\} = \int_{u_i}^{u_{i+1}} s \cdot f(s|s_i')\, \mathrm{d}s = \int_{u_i}^{u_{i+1}} s \cdot \frac{f(s)}{p_i}\, \mathrm{d}s \tag{373}$$

$\Longrightarrow$ Centroid condition for MSE distortion

$$\boxed{s_i' = \frac{1}{p_i} \int_{u_i}^{u_{i+1}} s\, f(s)\, \mathrm{d}s = \frac{\int_{u_i}^{u_{i+1}} s\, f(s)\, \mathrm{d}s}{\int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s}} \tag{374}$$

Properties of Centroid Quantizers

- Quantization does not change the mean

$$E\{S\} = \sum_i \int_{u_i}^{u_{i+1}} s\, f(s)\, \mathrm{d}s = \sum_i p_i\, s'_i = E\{S'\} = E\{Q(S)\} \tag{375}$$

- Mean of quantization error

$$E\{e(S)\} = E\{S - Q(S)\} = E\{S\} - E\{Q(S)\} = 0 \tag{376}$$

- Distortion D (2nd moment and variance of quantization error)

$$\begin{aligned} D = E\{e(S)^2\} &= \sum_i \int_{u_i}^{u_{i+1}} (s - s'_i)^2\, f(s)\, \mathrm{d}s \\ &= \sum_i \left(\int_{u_i}^{u_{i+1}} s^2\, f(s)\, \mathrm{d}s - 2s'_i \int_{u_i}^{u_{i+1}} s\, f(s)\, \mathrm{d}s + s'^2_i \int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s \right) \\ &= \int_{-\infty}^{\infty} s^2\, f(s)\, \mathrm{d}s - \sum_i \left(2s'_i \cdot s'_i \cdot p_i - s'^2_i \cdot p_i \right) \\ &= E\{S^2\} - E\{Q(S)^2\} \qquad (377) \\ \Longrightarrow \quad \sigma^2_{e(S)} &= \sigma^2_S - \sigma^2_{Q(S)} \qquad (378) \end{aligned}$$

Properties of Centroid Quantizers

- Correlation between quantizer input S and quantizer output $Q(S)$

$$\begin{aligned} E\{S \cdot Q(S)\} &= \sum_i \int_{u_i}^{u_{i+1}} s\, s'_i\, f(s)\, g(s'_i|s)\, \mathrm{d}s \\ &= \sum_i s'_i \int_{u_i}^{u_{i+1}} s\, f(s)\, \mathrm{d}s = \sum_i s'^2_i\, p_i = E\{Q(S)^2\} \end{aligned} \tag{379}$$

- Correlation between quantizer input S and quantization error $e(S)$

$$\begin{aligned} E\{S \cdot e(S)\} &= E\{S\,(S - Q(S))\} = E\{S^2\} - E\{S\,Q(S)\} \\ &= E\{e(S)^2\} + E\{Q(S)^2\} - E\{Q(S)^2\} \\ &= E\{e(S)^2\} = D \end{aligned} \tag{380}$$

- Correlation between quantizer output $Q(S)$ and quantization error $e(S)$

$$\begin{aligned} E\{Q(S) \cdot e(S)\} &= E\{Q(S)\,(S - Q(S))\} = E\{Q(S)\,S\} - E\{Q(S)^2\} \\ &= E\{Q(S)^2\} - E\{Q(S)^2\} = 0 \end{aligned} \tag{381}$$

$\Longrightarrow$ **Quantizer output and quantization error are uncorrelated**

Optimality for Encoding Mapping: Nearest Neighbor Condition

- Assume fixed-length coding and given reconstruction levels s'_i
- Choose decision thresholds u_i so that distortion D is minimized

$$D = \sum_{i=0}^{K-1} p_i \, D_i = \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} d_1(s, s'_i) \cdot f(s) \, \mathrm{d}s \tag{382}$$

- Each decision thresholds u_i influences only the distortions D_{i-1} and D_i of the neighboring intervals $\mathcal{C}_{i-1}$ and $\mathcal{C}_i$, respectively
- Distortion is minimized if the following condition is obeyed

$$\boxed{d_1(u_i, s'_{i-1}) = d_1(u_i, s'_i)} \tag{383}$$

- For MSE distortion, optimal decision thresholds u_i^* are given by

$$\boxed{u_i^* = \frac{s'_{i-1} + s'_i}{2}} \tag{384}$$

Lloyd Quantizer

Optimal scalar quantizer with fixed-length codes

- Do not consider entropy coding of quantization indices
- Minimize distortion for given number K of quantization intervals
- Rate can be represented by

$$R = \log_2 K \tag{385}$$

- Preferable to choose K as an integer power of 2

Necessary conditions for optimality

- General centroid condition (for reconstruction levels s'_i)

$$s'^{*}_i = \arg\min_{s' \in \mathbb{R}} E\{d_1(S, s') \,|\, S \in \mathcal{C}_i\} \tag{386}$$

- General nearest neighbor condition (for decision threshold u_i)

$$d_1(u_i, s'_{i-1}) = d_1(u_i, s'_i) \tag{387}$$

Lloyd Quantizer

Optimality conditions for MSE distortion

- Centroid condition

$$s'_i = \frac{\int_{u_i}^{u_{i+1}} s\, f(s)\, \mathrm{d}s}{\int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s} \tag{388}$$

- Nearest neighbor condition (for decision threshold u_i)

$$u_i = \frac{s'_{i-1} + s'_i}{2} \tag{389}$$

Design of Lloyd quantizers

- In general, cannot be derived analytically
- Iterative algorithm consisting of
  - Optimize decision thresholds u_i given reconstruction levels s'_i
  - Optimize reconstruction levels s'_i given decision thresholds u_i
- Iterative design can be based on
  - Given probability density function (perhaps using numerical integration)
  - Sufficiently large training set for considered source

Lloyd Algorithm for a Training Set

Given is

- a sufficiently large realization $\{s_n\}$ of considered source
- the number K of reconstruction levels $\{s'_i\}$

Iterative quantizer design

1. Choose an initial set of reconstruction levels $\{s'_i\}$
2. Associate all samples of the training set $\{s_n\}$ with one of the quantization intervals $\mathcal{C}_i$ according to

$$\alpha(s_n) = \arg\min_{\forall i} \; d_1(s_n, s'_i) \qquad \text{(nearest neighbor condition)}$$

   and update the decision thresholds $\{u_i\}$ accordingly
3. Update the reconstruction levels $\{s'_i\}$ according to

$$s'_i = \arg\min_{s' \in \mathbb{R}} \; E\{d_1(S, s') \,|\, \alpha(S) = i\} \qquad \text{(centroid condition)}$$

   where the expectation value is taken over the training set
4. Repeat the previous two steps until convergence

Example: Lloyd Algorithm for a Gaussian Source

- Gaussian distribution with zero mean and unit variance

$$f(s) = \frac{1}{\sigma\sqrt{2\pi}} e^{-s^2/(2\sigma^2)} \tag{390}$$

- Draw a sufficiently large number of samples (> 10000) from $f(s)$
- Design Lloyd quantizer with rate $R = 2$ bit/symbol ($K = 4$)
- Result of Lloyd algorithm
  - Decision thresholds u_i

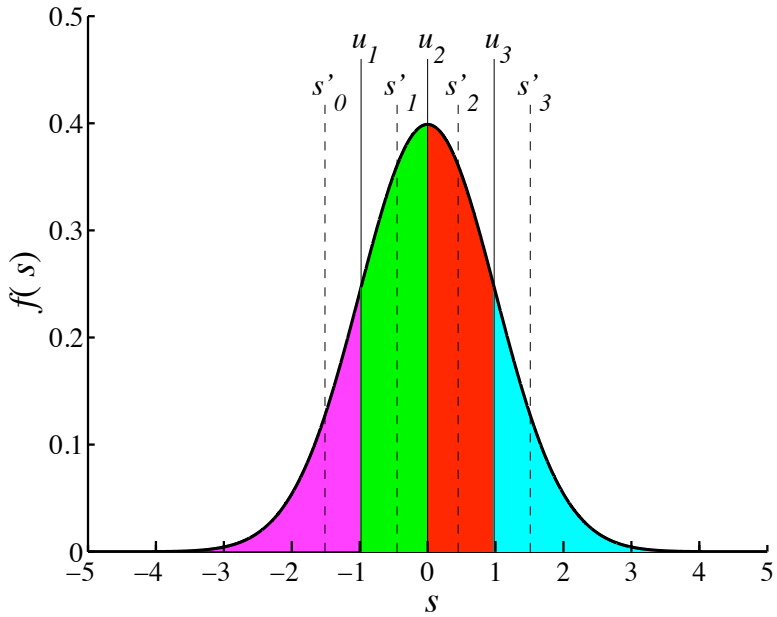
    $u_1 = -0.98, \quad u_2 = 0, \quad u_3 = 0.98$

  - Decoding symbols s'_i

    $s'_0 = -1.51, \quad s'_1 = -0.45$

    $s'_2 = 0.45, \quad s'_3 = 1.51$

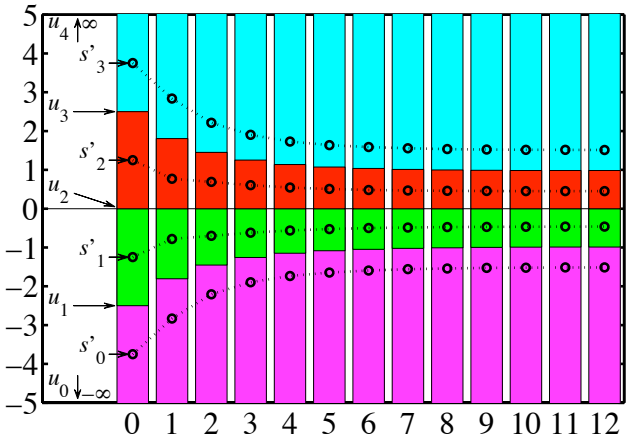
  - Minimum distortion:
    $D^*_F = 0.12 = 9.3$ dB

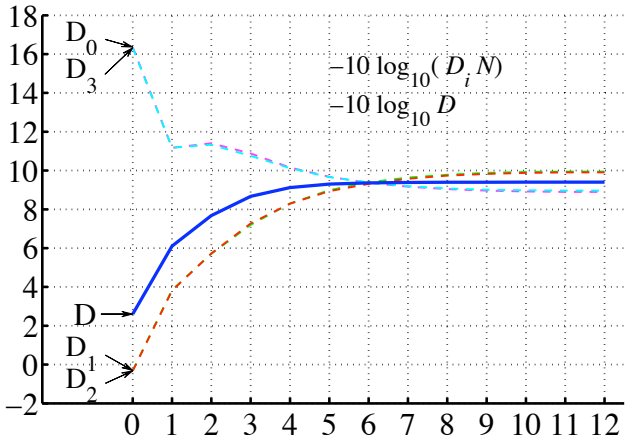


Convergence of Lloyd Algorithm for Gaussian Source Example

Initialization A:

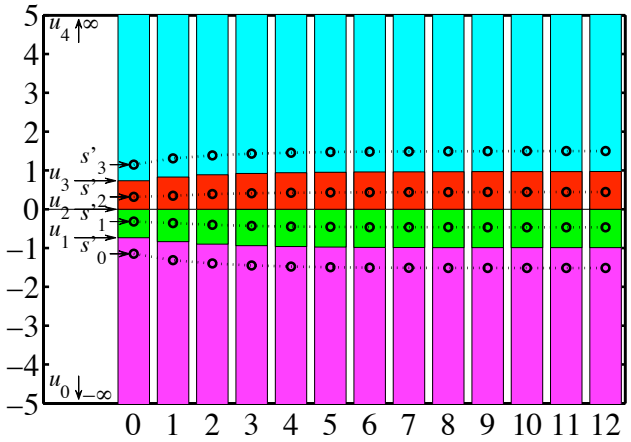
$$s'_i = -3.75 + 2.5 \cdot i$$

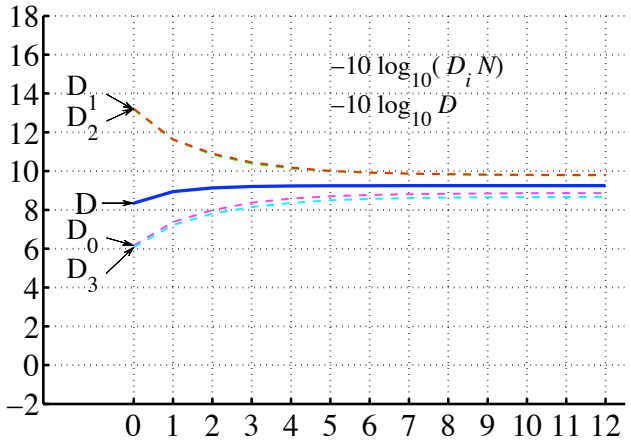




Initialization B:

$$s'_{3/0} = +/- 1.15, \; s'_{2/1} = +/- 0.32$$





- For both initializations, $(D - D_F^*)/D_F^* < 1\%$ after 6 iterations

Example: Lloyd Algorithm for a Laplacian Source

- Laplacian distribution with zero mean and unit variance

$$f(s) = \frac{1}{\sigma\sqrt{2}} e^{-|s|\sqrt{2}/\sigma} \tag{391}$$

- Draw a sufficiently large number of samples (> 10000) from $f(s)$
- Design Lloyd quantizer with rate $R = 2$ bit/symbol ($K = 4$)
- Result of Lloyd algorithm
  - Decision thresholds u_i

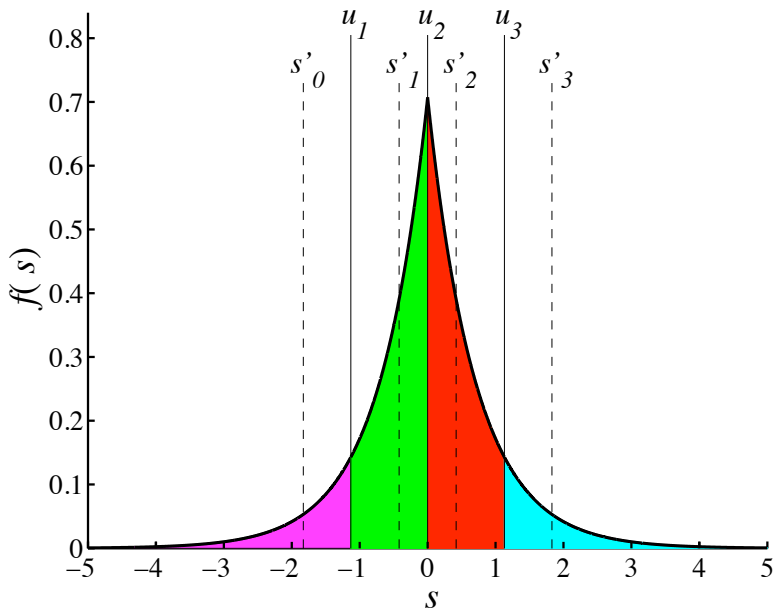
    $u_1 = -1.13, \quad u_2 = 0, \quad u_3 = 1.13$

  - Decoding symbols s'_i

    $s'_0 = -1.83, \quad s'_1 = -0.42$

    $s'_2 = 0.42, \quad s'_3 = 1.83$

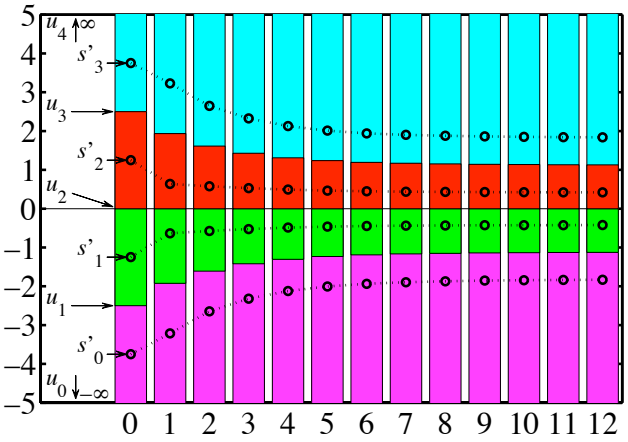
  - Minimum distortion:
    $D^*_F = 0.18 = 7.55$ dB

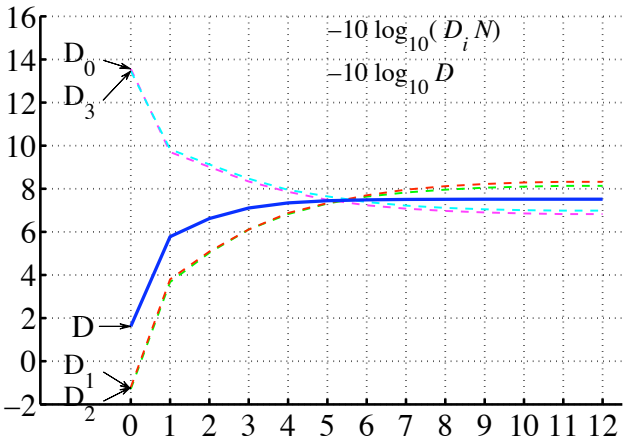


Convergence of Lloyd Algorithm for Laplacian Source Example

Initialization A:

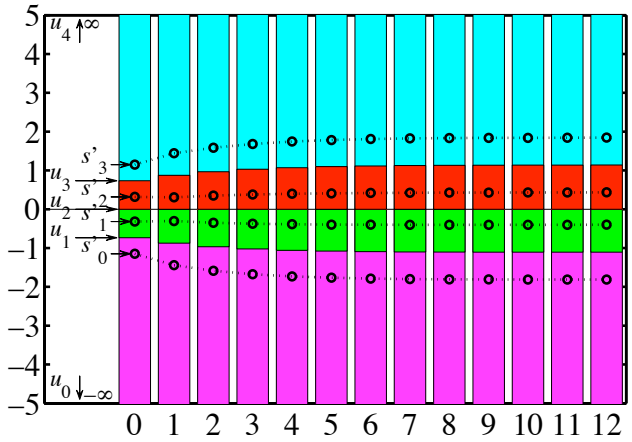
$$s'_i = -3.75 + 2.5 \cdot i$$

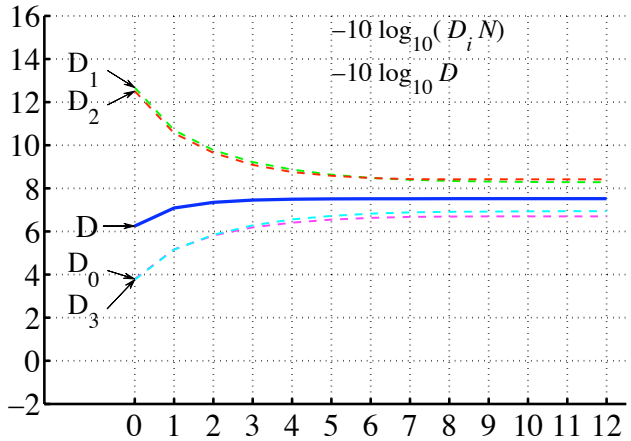




Initialization B:

$$s'_{3/0} = +/-1.15, \; s'_{2/1} = +/-0.32$$





- For both initializations, $(D - D_F^*)/D_F^* < 1\%$ after 6 iterations

Entropy-Constrained Scalar Quantization (ECSQ)

- Lloyd quantizer: Minimize distortion for given number K of intervals
- Now: Consider quantizer design with variable-length coding of indices

- Average rate (without exploiting dependencies between quantization indices)

$$R = \sum_{i=0}^{N-1} p_i \cdot \ell_i \;\geq\; H(S') = -\sum_{i=0}^{K-1} p_i \log_2 p_i \tag{392}$$

with

$$p_i = \int_{u_i}^{u_{i+1}} f(s)\,\mathrm{d}s \tag{393}$$

$\Longrightarrow$ Consider entropy instead of the rate of an actual code

- Average MSE distortion

$$D = E\{d_1(S, S')\} = \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} (s - s'_i)^2 \cdot f(s)\,\mathrm{d}s \tag{394}$$

Joint Minimization of Rate and Distortion

- We look for solutions of constrained minimization problems

$$\min D \quad \text{subject to} \quad R \leq R_C \tag{395}$$

$$\text{or equivalently} \quad \min R \quad \text{subject to} \quad D \leq D_C \tag{396}$$

- Instead of the constrained minimization, minimize a **Lagrangian function**

$$\boxed{J = D + \lambda \cdot R = E\{d_1(S, S')\} + \lambda \cdot E\{\ell(S')\}} \tag{397}$$

- The chosen λ corresponds to a rate constraint R_C (distortion constraint D_C)

- Minimization of J with respect to reconstruction levels s'_i is the same as the minimization of the distortion D with respect to the reconstruction levels s'_i

$\Longrightarrow$ Centroid condition still optimal for reconstruction levels (decoder $\beta(i)$)

$$\text{MSE:} \qquad s'^{*}_i = E\{S | s \in \mathcal{C}_i\} = \frac{\int_{u_i}^{u_{i+1}} s \cdot f(s)\, ds}{\int_{u_i}^{u_{i+1}} f(s)\, ds} \tag{398}$$

Necessary Conditions for Optimality

- Optimal quantizer design: Minimize Lagrange cost J for given λ

$$J = D + \lambda \cdot R = E\{d_1(S, S')\} + \lambda \cdot E\{\ell(S')\} \tag{399}$$

- Optimal reconstruction levels only depend on decision thresholds u_i

$$\boxed{s_i'^{*} = E\{S|s \in \mathcal{C}_i\} = \frac{\int_{u_i}^{u_{i+1}} s \cdot f(s)\, ds}{\int_{u_i}^{u_{i+1}} f(s)\, ds}} \qquad \text{(for MSE)} \tag{400}$$

- Optimal codeword lengths also depends only on decision thresholds u_i

$$\boxed{\ell_i = -\log_2 p_i = -\log_2 \left(\int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s \right)} \tag{401}$$

- How to derive optimal decision thresholds?

Optimal Decision Thresholds

- Want to minimize J given optimal decoder β and entropy coding γ

$$\begin{aligned} J &= D + \lambda \cdot R \\ &= = \sum_{\forall i} \int_{u_i}^{u_{i+1}} d_1(s, s'_i)\, f(s)\, \mathrm{d}s + \lambda \sum_{\forall i} \ell_i \int_{u_i}^{u_{i+1}} f(s)\, \mathrm{d}s \end{aligned} \tag{402}$$

- For given reconstruction levels s'_i and codeword lengths ℓ_i:

  $\Longrightarrow$ Each decision threshold u_i only influences distortion of neighboring intervals $\mathcal{C}_{i-1}$ and $\mathcal{C}_i$

- Optimal threshold u_i:
  Each value s is assigned to the interval for which $D + \lambda R$ is minimized

$$\boxed{\alpha(s) = \arg\min_{\forall s'_i} \; d_1(s, s'_i) + \lambda\, \ell_i} \tag{403}$$

Optimal Decision Thresholds

- Lagrangian function is minimized for encoding

$$\alpha(s) = \arg\min_{\forall s'_i} \; d_1(s, s'_i) + \lambda \, \ell_i \tag{404}$$

- Optimal decision threshold u_i fulfils condition

$$d_1(u_i, s'_{i-1}) + \lambda \cdot \ell_{i-1} = d_1(u_i, s'_i) + \lambda \cdot \ell_i \tag{405}$$

- For MSE distortion, we have

$$(u_i - s'_{i-1})^2 + \lambda \cdot \ell_{i-1} = (u_i - s'_i)^2 + \lambda \cdot \ell_i \tag{406}$$

yielding

$$\boxed{u_i^* = \frac{s'_i + s'_{i-1}}{2} + \frac{\lambda}{2} \cdot \frac{\ell_i - \ell_{i-1}}{s'_i - s'_{i-1}}} \tag{407}$$

- The decision threshold is shifted from the middle between the reconstruction values toward the reconstruction value with the longer codeword

Entropy-Constrained Lloyd Algorithm for a Training Set

Given is

- a sufficiently large realization $\{s_n\}$ of considered source
- a Lagrange parameter λ

Iterative quantizer design

1. Choose initial set of reconstruction levels $\{s'_i\}$ and codeword lengths $\{\ell_i\}$
2. Associate all samples of the traing set $\{s_n\}$ with one of the quantization intervals $\mathcal{C}_i$ according to

$$\alpha(s_n) = \arg\min_{\forall s'_i} d_1(s_n, s'_i) + \lambda\, \ell_i \tag{408}$$

   and update the decision thresholds $\{u_i\}$ accordingly
3. Update the reconstruction levels $\{s'_i\}$ according to

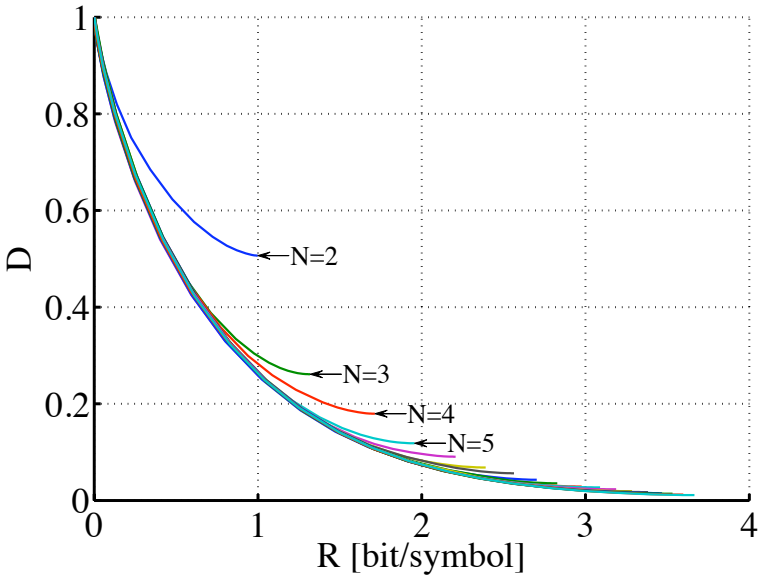
$$s'_i = \arg\min_{s' \in \mathcal{R}} E\{d_1(S, s') \mid \alpha(S) = i\} \tag{409}$$

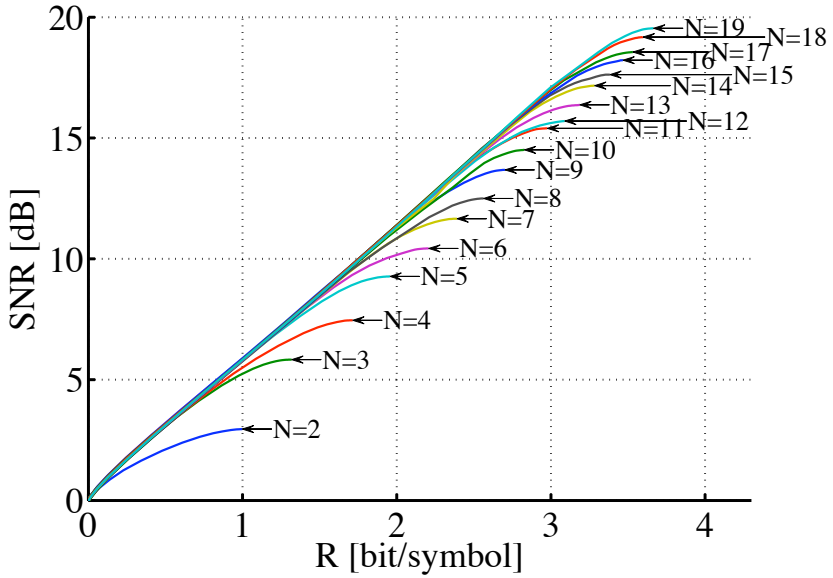
4. Update the codeword lengths ℓ_i according to

$$\ell_i = -\log_2 p_i \tag{410}$$

5. Repeat previous three steps until convergence

Number of Initial Intervals for EC Lloyd Algorithm

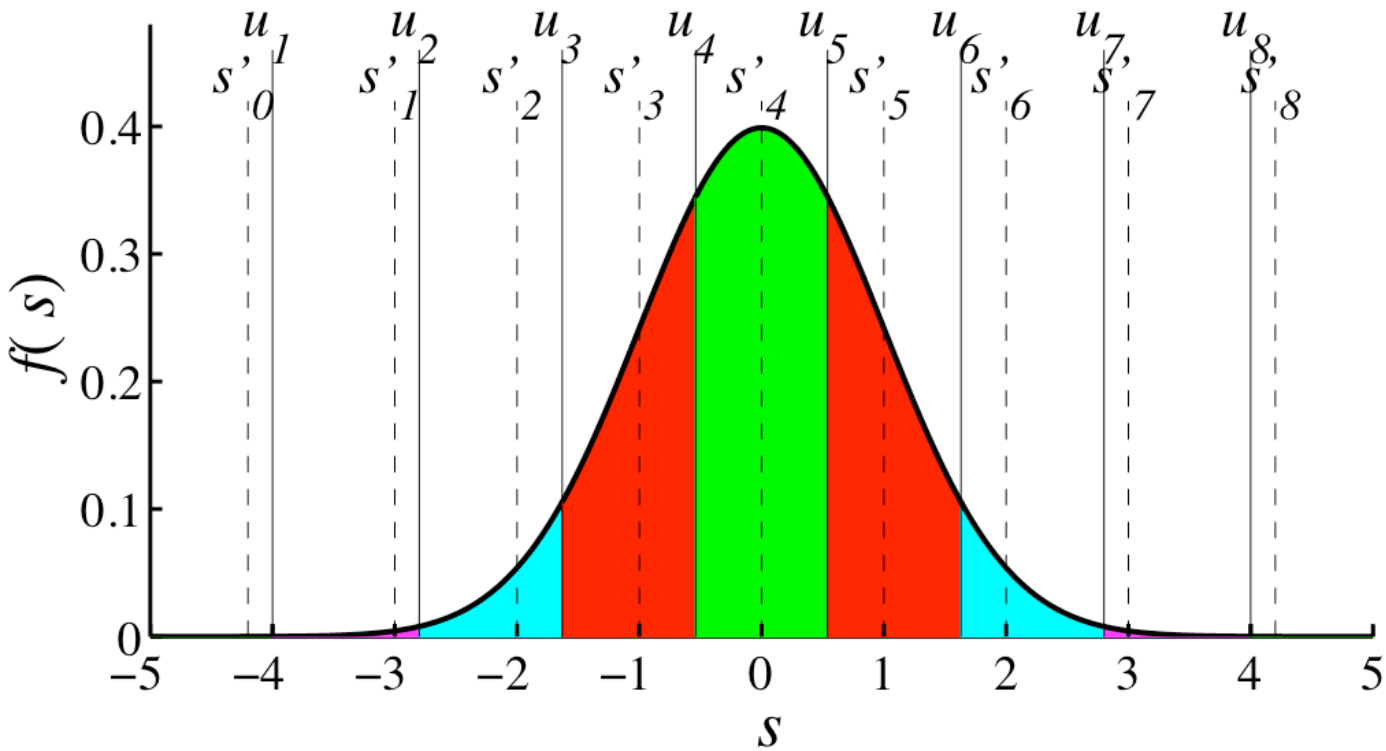




- Entropy constraint in EC Lloyd algorithm causes shift of costs
- If two level s'_i and s'_k are competing, the symbol with larger popularity has higher chance of being chosen
- Level which is not chosen further reduces its associated conditional probability
- As a consequence, symbols get "removed" and the EC Lloyd algorithm can be initialized with more symbols than the final result

Entropy-Constrained Lloyd Algorithm for Gaussian Source

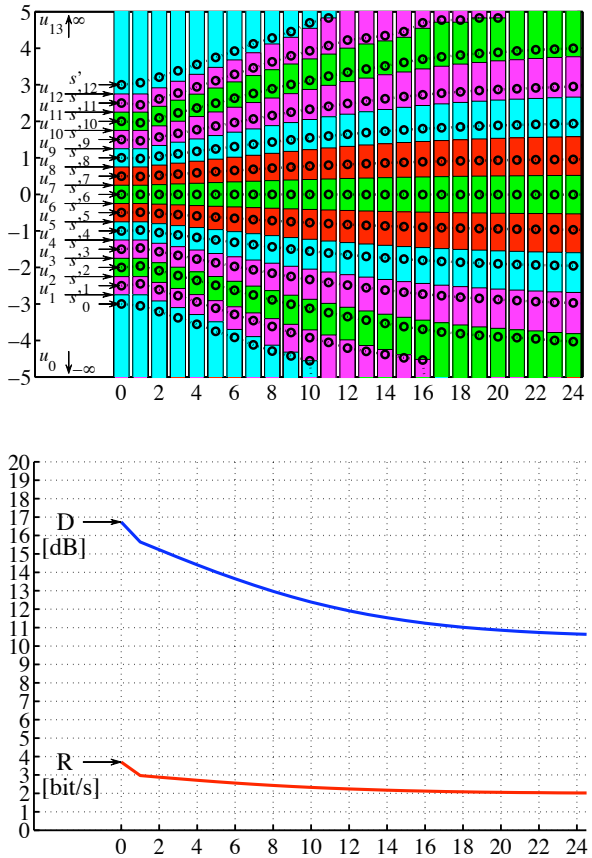
- Consider Gaussian source with zero mean and unit variance
- Design optimal entropy-constrained quantizer with rate $R = 2$ bit/symbol
- Optimum average distortion: $D_F^* = 0.09 = 10.45$ dB
- Results for optimal decision thresholds u_i and decoding symbols s'_i are



Convergence EC Lloyd Algorithm for Gaussian Source

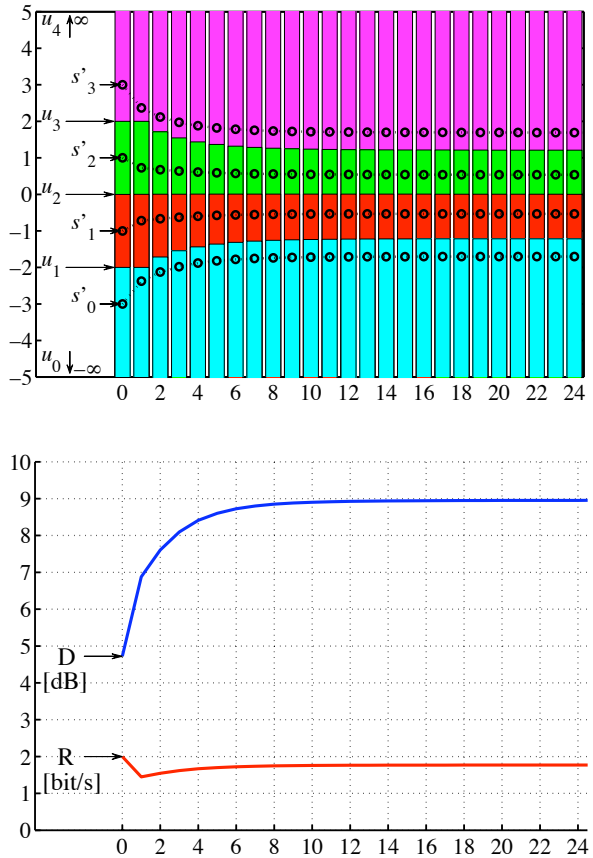
Initialization A:

$$s'_i = -3 + 0.5 \cdot i$$



Initialization B:

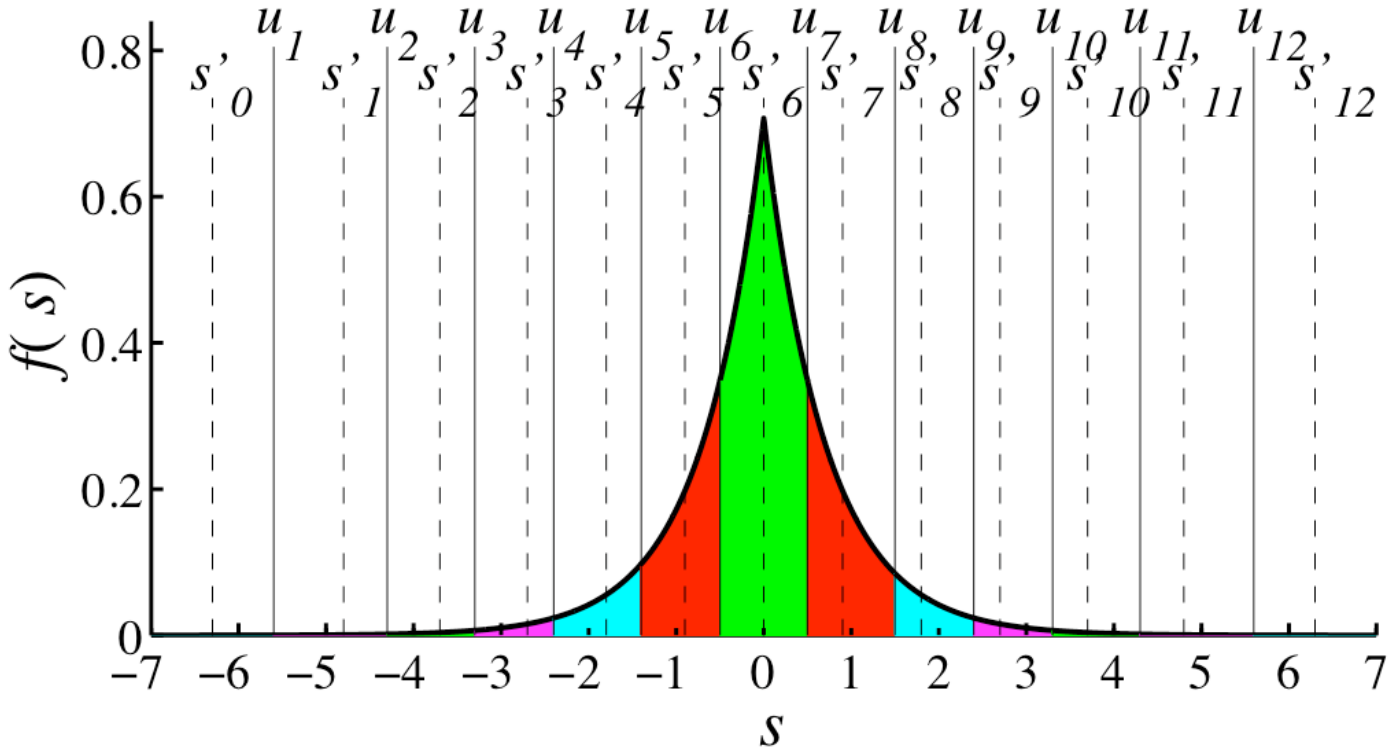
$$s'_i = -3 + 2 \cdot i$$



- For initialization A, decoding bins get discarded
- For initialization B, desired quantizer performance is not achieved

Entropy-Constrained Lloyd Algorithm for Laplacian Source

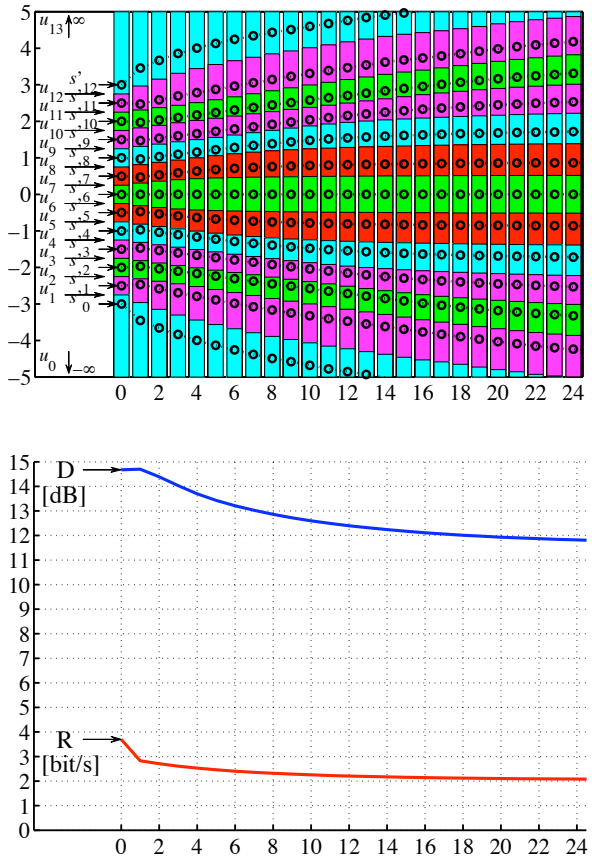
- Consider Laplacian source with zero mean and unit variance
- Design optimal entropy-constrained quantizer with rate $R = 2$ bit/symbol
- Optimum average distortion: $D_V^* = 0.07 = 11.46$ dB
- Results for optimal decision thresholds u_i and decoding symbols s'_i are



Convergence of EC Lloyd Algorithm for Laplacian Source

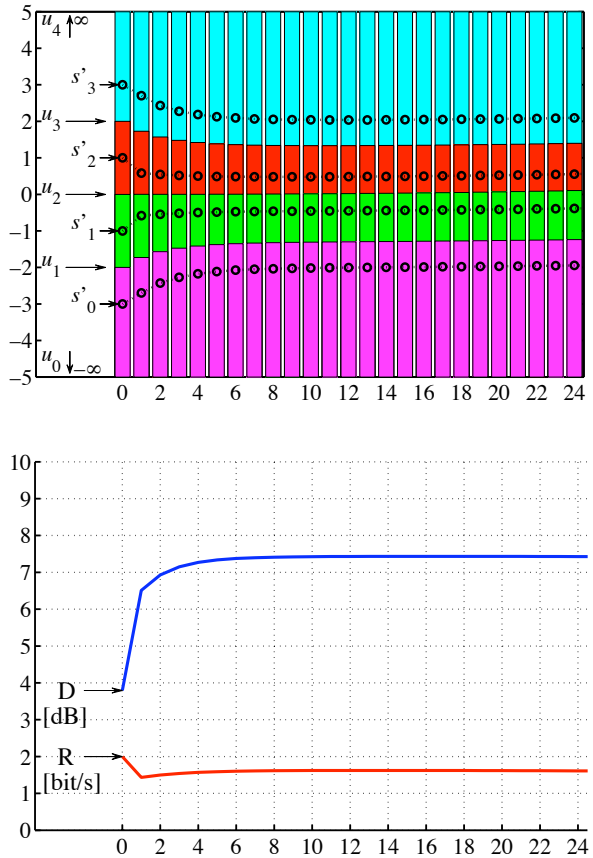
Initialization A:

$$s'_i = -3 + 0.5 \cdot i$$



Initialization B:

$$s'_i = -3 + 2 \cdot i$$



- For initialization A, faster convergence of costs than thresholds
- For initialization B, desired quantizer performance is not achieved

High-Rate Approximation for Scalar Quantizers

- Assumption: Small sizes Δ_i of quantization intervals $[u_i, u_{i+1})$
- Then: Marginal pdf $f(s)$ nearly constant inside each interval

$$f(s) \approx f(s'_i) \qquad \text{for} \qquad s \in [u_i, u_{i+1}) \tag{411}$$

- Approximation

$$p_i = \int_{u_i}^{u_{i+1}} f(s)\, ds \approx (u_{i+1} - u_i) f(s'_i) = \Delta_i \cdot f(s'_i) \tag{412}$$

- Average distortion

$$\begin{aligned} D &= E\{d(S, Q(S))\} \\ &= \sum_{i=0}^{K-1} \int_{u_i}^{u_{i+1}} (s - s'_i)^2 f(s)\, \mathrm{d}s \\ &\approx \sum_{i=0}^{K-1} f(s'_i) \int_{u_i}^{u_{i+1}} (s - s'_i)^2\, \mathrm{d}s \end{aligned} \tag{413}$$

High-Rate Approximation for Scalar Quantizers

- Average distortion

$$D \approx \sum_{i=0}^{K-1} f(s'_i) \int_{u_i}^{u_{i+1}} (s - s'_i)^2 \, ds \tag{414}$$

$$= \frac{1}{3} \sum_{i=0}^{K-1} f(s'_i) \left((u_{i+1} - s'_i)^3 - (u_i - s'_i)^3 \right) \tag{415}$$

- By differentiation with respect to s'_i, we find that for minimum distortion,

$$(u_{i+1} - s'_i)^2 = (u_i - s'_i)^2 \qquad \Longrightarrow \qquad s'_i = \frac{1}{2}(u_i + u_{i+1}) \tag{416}$$

- Average distortion at high rates

$$\boxed{D \approx \frac{1}{12} \sum_{i=0}^{K-1} f(s'_i) \, \Delta_i^3 = \frac{1}{12} \sum_{i=0}^{K-1} p_i \, \Delta_i^2} \tag{417}$$

- Average distortion at high rates for constant $\Delta = \Delta_i$

$$\boxed{D \approx \frac{\Delta^2}{12}} \tag{418}$$

High-Rate Approximation for Scalar Quantizers with FLC

- Using $\sum_{i=0}^{K-1} K^{-1} = 1$

$$D = \frac{1}{12} \sum_{i=0}^{K-1} f(s'_i) \Delta_i^3 = \frac{1}{12} \left(\left(\sum_{i=0}^{K-1} f(s'_i) \Delta_i^3 \right)^{\frac{1}{3}} \cdot \left(\sum_{i=0}^{K-1} \frac{1}{K} \right)^{\frac{2}{3}} \right)^3 \tag{419}$$

- Using Hölders inequality

$$\alpha + \beta = 1 \quad \Longrightarrow \quad \left(\sum_{i=a}^{b} x_i \right)^{\alpha} \cdot \left(\sum_{i=a}^{b} y_i \right)^{\beta} \geq \sum_{i=a}^{b} x_i^{\alpha} \cdot y_i^{\beta} \tag{420}$$

with equality if and only if x_i is proportional to y_i, it follows

$$D \geq \frac{1}{12} \left(\sum_{i=0}^{K-1} f(s'_i)^{\frac{1}{3}} \cdot \Delta_i \cdot \left(\frac{1}{K} \right)^{\frac{2}{3}} \right)^3 = \frac{1}{12\,K^2} \left(\sum_{i=0}^{K-1} \sqrt[3]{f(s'_i)}\, \Delta_i \right)^3 \tag{421}$$

- Reason for $\alpha = 1/3$: Obtain expression in which Δ_i has no exponent

High-Rate Approximations for Scalar Quantizers with FLC

- Inequality for average distortion

$$D \geq \frac{1}{12\,K^2} \left(\sum_{i=0}^{K-1} \sqrt[3]{f(s'_i)}\,\Delta_i \right)^3 \tag{422}$$

  becomes equality if all terms $f(s'_i)\,\Delta_i^3$ are the same

- Approximation asymptotically valid for small intervals Δ_i

$$D = \frac{1}{12K^2} \left(\int_{-\infty}^{\infty} \sqrt[3]{f(s)}\,\mathrm{d}s \right)^3 \tag{423}$$

- With $1/K^2 = 2^{-\log_2 K^2} = 2^{-2R}$: Operational distortion rate function for optimal scalar quantizers with fixed-length codes

$$\boxed{D_F(R) = \sigma^2 \cdot \varepsilon_F^2 \cdot 2^{-2R} \quad \text{with} \quad \varepsilon_F^2 = \frac{1}{12\sigma^2} \left(\int_{-\infty}^{\infty} \sqrt[3]{f(s)}\,\mathrm{d}s \right)^3} \tag{424}$$

$\Longrightarrow$ Published by Panter and Dite in [Panter and Dite, 1951] and is also referred to as the **Panter and Dite formula**

Efficiency of Optimum High-Rate Quantizers with FLCs

- $D_F(R)$ for optimum high-rate scalar quantization with fixed-length codes

$$\boxed{D_F(R) = \varepsilon_F^2 \cdot \sigma^2 \cdot 2^{-2R}} \tag{425}$$

- **Uniform pdf**:

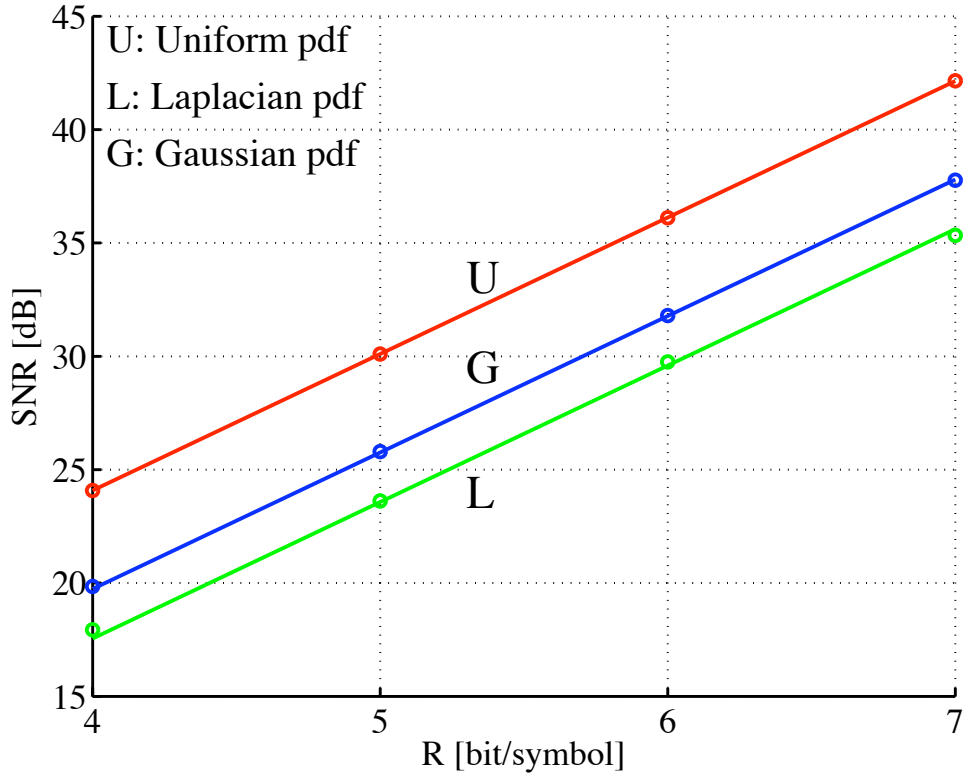
$$\varepsilon_F^2 = 1 \quad (0 \text{ dB})$$

- **Laplacian pdf**:

$$\varepsilon_F^2 = 4.5 \quad (6.53 \text{ dB})$$

- **Gaussian pdf**:

$$\varepsilon_F^2 = \frac{\sqrt{3}\pi}{2} \approx 2.721 \quad (4.35 \text{ dB})$$



High-Rate Approximation for Quantizers with VLC

- Use variable length coding for the quantizer indexes
- Again, assume pmf p_i of quantized output signal s' as $p_i = f(s'_i)\Delta_i$
- The average rate is given as

$$\begin{aligned} R = H(S') &= -\sum_{i=0}^{K-1} p_i \log_2 p_i = -\sum_{i=0}^{K-1} f(s'_i)\Delta_i \log_2 (f(s'_i)\Delta_i) \\ &= -\sum_{i=0}^{K-1} f(s'_i) \log_2 (f(s'_i)) \cdot \Delta_i - \sum_{i=0}^{K-1} f(s'_i)\Delta_i \log_2 \Delta_i \\ &\approx \underbrace{-\int f(s) \log_2 f(s)\, \mathrm{d}s}_{\text{differential entropy } h(S)} - \frac{1}{2} \sum_{i=0}^{K-1} p_i \log_2 \Delta_i^2 \\ &= h(S) - \frac{1}{2} \sum_{i=0}^{K-1} p_i \log_2 \Delta_i^2 \end{aligned} \tag{426}$$

High-Rate Approximation for Quantizers with VLC

- JENSEN'S inequality for convex functions $\varphi(x_i)$ such as $\varphi(x_i) = -\log_2 x_i$

$$\varphi\left(\sum_{i=0}^{K-1} a_i\, x_i\right) \leq \sum_{i=0}^{K-1} a_i\, \varphi(x_i) \qquad \text{for} \qquad \sum_{i=0}^{K-1} a_i = 1 \tag{427}$$

  with equality for constant x_i

- Jensen's inequality and the high-rate distortion approximation

$$\begin{aligned} R &= h(S) - \frac{1}{2}\sum_{i=0}^{K-1} p(s'_i)\log_2 \Delta_i^2 \geq h(S) - \frac{1}{2}\log_2\left(\sum_{i=0}^{K-1} p(s'_i)\Delta_i^2\right) \\ &= h(S) - \frac{1}{2}\log_2(12D) \end{aligned} \tag{428}$$

  with equality if and only if all $\Delta_i = \Delta$, i.e. for uniform quantization

$\Longrightarrow$ **For MSE distortion and high rates, optimal quantizers with variable length codes have uniform step sizes**

Comparison of High-Rate Distortion-Rate Functions

- **Optimum high-rate scalar quantizers with variable-length codes**

$$\boxed{D_V(R) = \frac{1}{12} \cdot 2^{2h(S)} \cdot 2^{-2R}} \tag{429}$$

is factor $\pi e/6 \approx 1.42$ or ≈ 1.53 dB from the Shannon Lower Bound (SLB)

$$D_L(R) = \frac{1}{2\pi e} \cdot 2^{2h(S)} \cdot 2^{-2R} \tag{430}$$

- Recall: **Optimum high-rate scalar quantizers with fixed-length codes**

$$D_F(R) = \frac{1}{12} \left[\int_{-\infty}^{\infty} \sqrt[3]{f(s)}\, \mathrm{d}s \right]^3 \cdot 2^{-2R} \tag{431}$$

- The $D_X(R)$ functions $(X = L, F, V)$ can be expressed in general form as

$$\boxed{D_X(R) = \varepsilon_X^2 \cdot \sigma^2 \cdot 2^{-2R}} \tag{432}$$

with ε_X^2 being a factor that depends on pdf $(f(s))$ of the source and properties of the quantizer (fixed-length vs. variable length vs. SLB)

Comparison of High-Rate Distortion-Rate Functions

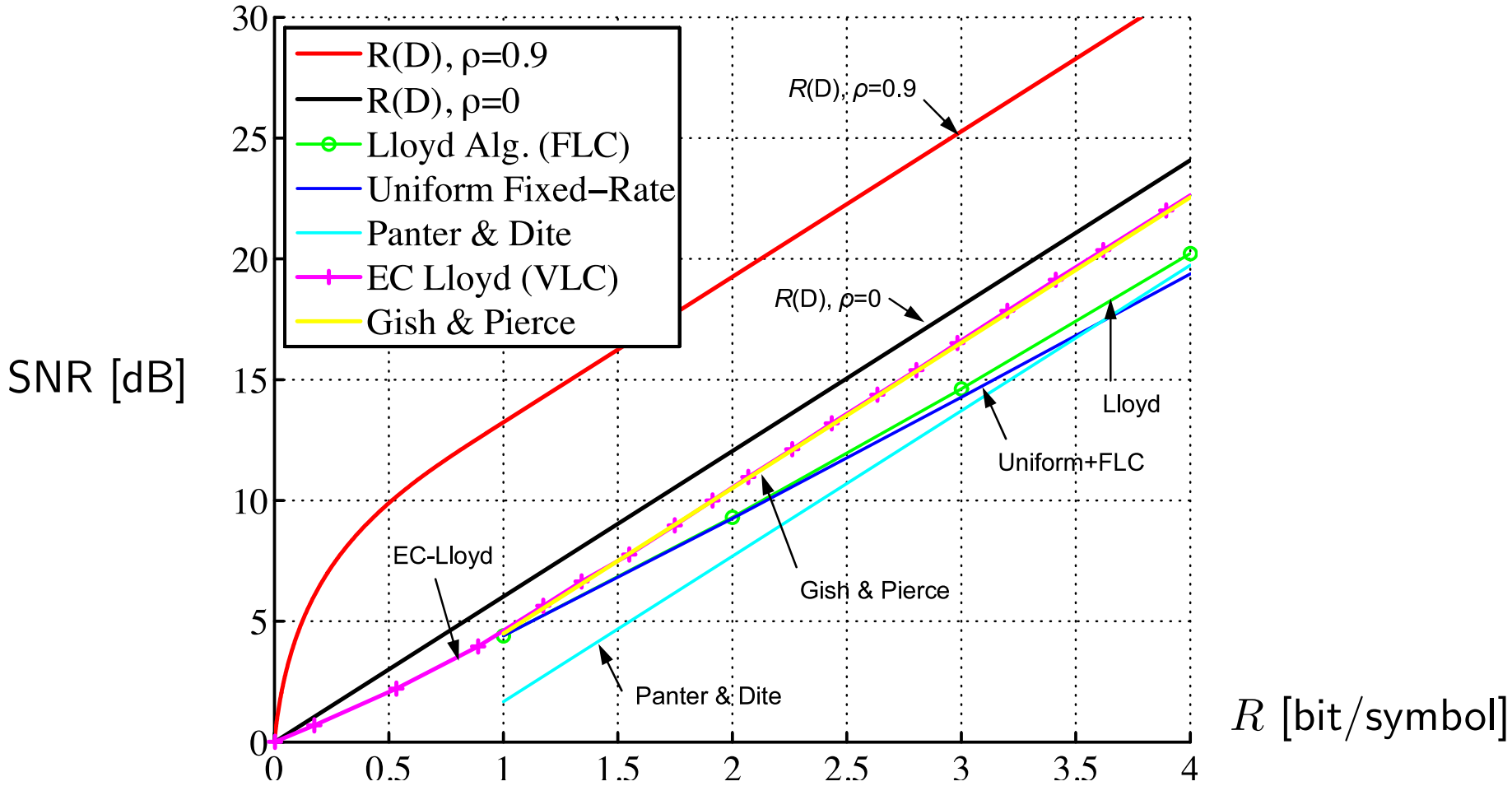
- Operational Distortion-rate function at high rates is given as

$$\boxed{D_X(R) = \varepsilon_X^2 \cdot \sigma^2 \cdot 2^{-2R}} \tag{433}$$

- Values of ε_X^2 for quantization method X

Method	*Shannon Lower Bound (SLB)*	*Panter & Dite* (Lloyd Quant. & FLC)	*Gish & Pierce* (ECSQ & VLC)
Uniform pdf	$\frac{6}{\pi e} \approx 0.7$	1 (1.53 dB to SLB)	1 (1.53 dB to SLB)
Laplacian pdf	$\frac{e}{\pi} \approx 0.86$	$\frac{9}{2} = 4.5$ (7.1 dB to SLB)	$\frac{e^2}{6} \approx 1.23$ (1.53 dB to SLB)
Gaussian pdf	1	$\frac{\sqrt{3}\pi}{2} \approx 2.72$ (4.34 dB to SLB)	$\frac{\pi e}{6} \approx 1.42$ (1.53 dB to SLB)

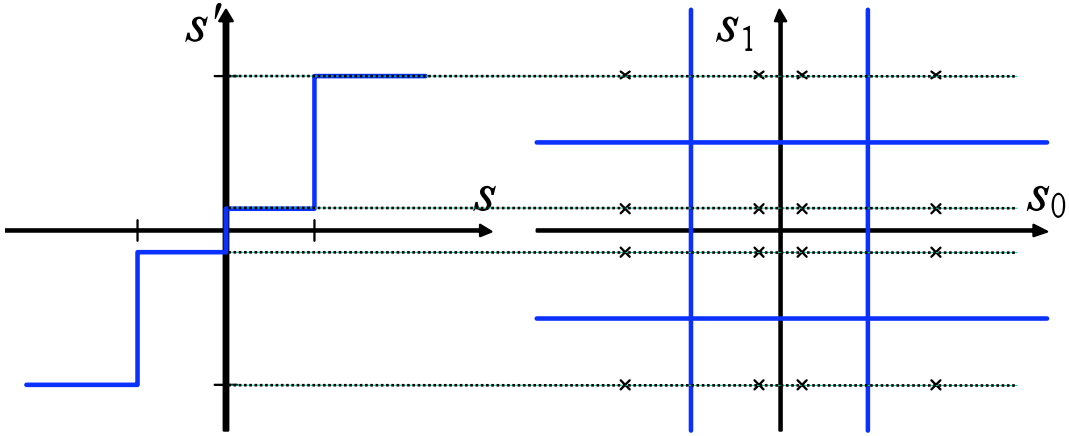
Performance of Scalar Quantizers for Gaussian Sources



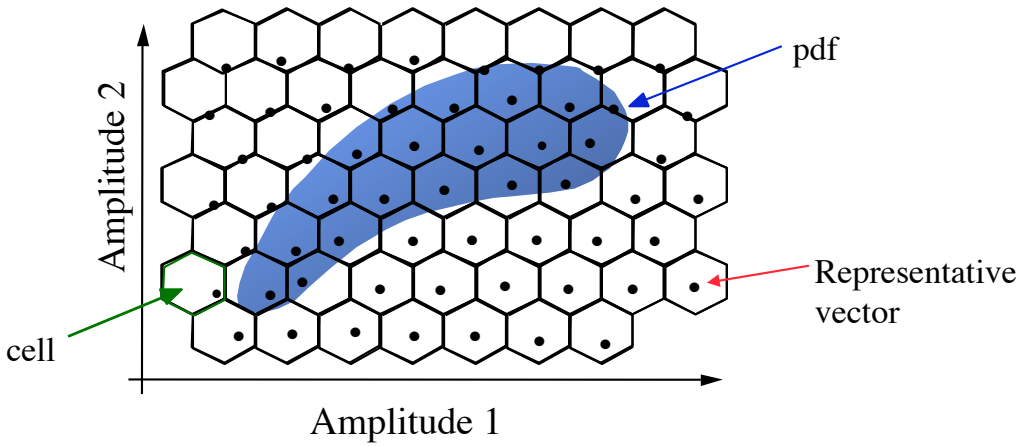
- Entropy-constrained scalar quantizer is 1.53 dB from distortion rate curve
- For sources with memory: Statistical dependencies cannot be exploited

Can We Further Improve Quantization?

- Scalar quantization: Special case of vector quantization (with $N = 1$)



- Vector quantization with $N > 1$ allows a number of new options



Vector Quantization

- Vector quantization:
  - Generalization of scalar quantization
  - Map vector of $N > 1$ samples to representative vectors
- Many models and design techniques used in vector quantization are natural generalizations of scalar quantization
- Vector quantizer Q of dimension N and size K is a mapping from a point in N-dimensional Euclidean space $\mathbb{R}^N$ into a finite set $\mathcal{C}$ containing K code vectors or code words

$$Q : \mathbb{R}^N \rightarrow \mathcal{C} \tag{434}$$

- Vector quantizer splits $\mathbb{R}^N$ into K quantization cells $\mathcal{C}_i$

$$C_i = \{\boldsymbol{s} \in \mathbb{R}^N : q(\boldsymbol{s}) = \boldsymbol{s'}\} \tag{435}$$

- The cells form a partition of $\mathbb{R}^N$

$$\bigcup_i C_i = \mathbb{R}^N \qquad \text{and} \qquad C_i \bigcap C_j = \emptyset \ \text{ for } \ i \neq j \tag{436}$$

Measuring Vector Quantizer Performance

- Average distortion for a N-dimensional vector quantizer

$$D = E\{d_N(\boldsymbol{S}, \boldsymbol{S'})\} = \int_{\boldsymbol{\mathcal{R}}^N} d_N(\boldsymbol{s}, \boldsymbol{s'}) f(\boldsymbol{s})\, d\boldsymbol{s} \tag{437}$$

- Using the partitioning of $\boldsymbol{\mathcal{R}}^N$ into cells C_i and the codebook $\boldsymbol{\mathcal{C}} = \{\boldsymbol{s}'_0, \boldsymbol{s}'_1, ...\}$ for a given quantizer Q

$$D = \sum_{i=0}^{K-1} \int_{C_i} d_N(\boldsymbol{s}, \boldsymbol{s}'_i) f(\boldsymbol{s})\, d\boldsymbol{s} \tag{438}$$

- For MSE distortion

$$d_N(\boldsymbol{s}, \boldsymbol{s}'_i) = \frac{1}{N} \|\boldsymbol{s} - \boldsymbol{s}'_i\| = \frac{1}{N} (\boldsymbol{s} - \boldsymbol{s}'_i)^T (\boldsymbol{s} - \boldsymbol{s}'_i) = \frac{1}{N} \sum_{n=0}^{N-1} (s_n - s'_{i,n})^2 \tag{439}$$

- Average rate (bit/scalar) for a N-dimensional vector quantizer of size K

$$R = \frac{1}{N} E\{-\log_2 p(\boldsymbol{S}'_i)\} = -\frac{1}{N} \sum_{i=0}^{K-1} p_i \log_2 p_i \tag{440}$$

The Linde-Buzo-Gray (LBG) Algorithm

Given is

- a sufficiently large realization $\{\boldsymbol{s}_n\}$ of considered source
- the number K of reconstruction vectors $\{\boldsymbol{s'}_i\}$

Iterative quantizer design (extension of Lloyd algorithm)

1. Choose an initial set of reconstruction vectors $\{\boldsymbol{s'}_i\}$
2. Associate all vectors of the training set $\{\boldsymbol{s}_n\}$ with one of the quantization cells $\mathcal{C}_i$ according to

$$\alpha(\boldsymbol{s_n}) = \arg\min_{\forall i} \; d_N(\boldsymbol{s}_n, \boldsymbol{s'}_i) \qquad \text{(nearest neighbor condition)}$$

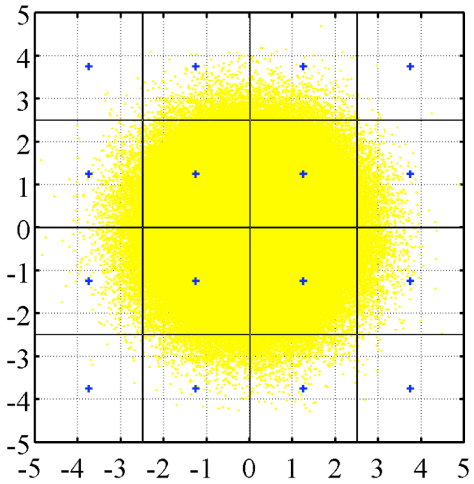
   and update the decision thresholds $\{u_i\}$ accordingly
3. Update the reconstruction vectors $\{\boldsymbol{s'}_i\}$ according to

$$\boldsymbol{s'}_i = \arg\min_{\boldsymbol{s'} \in \mathbb{R}} \; E\{d_N(\boldsymbol{S}, \boldsymbol{s'}) \,|\, \alpha(\boldsymbol{S}) = i\} \qquad \text{(centroid condition)}$$

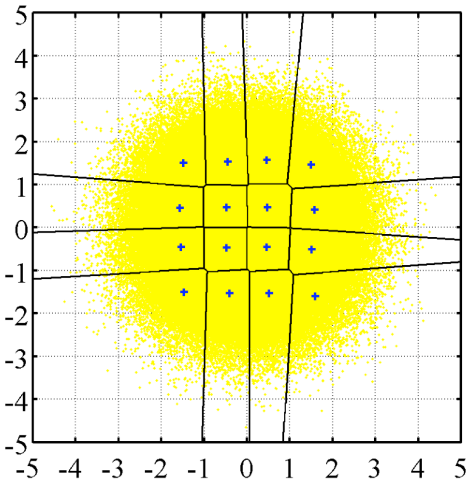
4. Repeat the previous two steps until convergence

LBG Algorithm Result for Gaussian IID

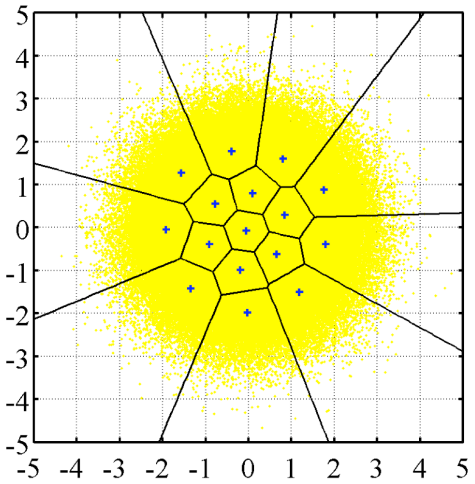
Result for dimension $N = 2$ and size $K = 16$ corresponding to $R = 2$ bit/sample



initialization



after iteration 8

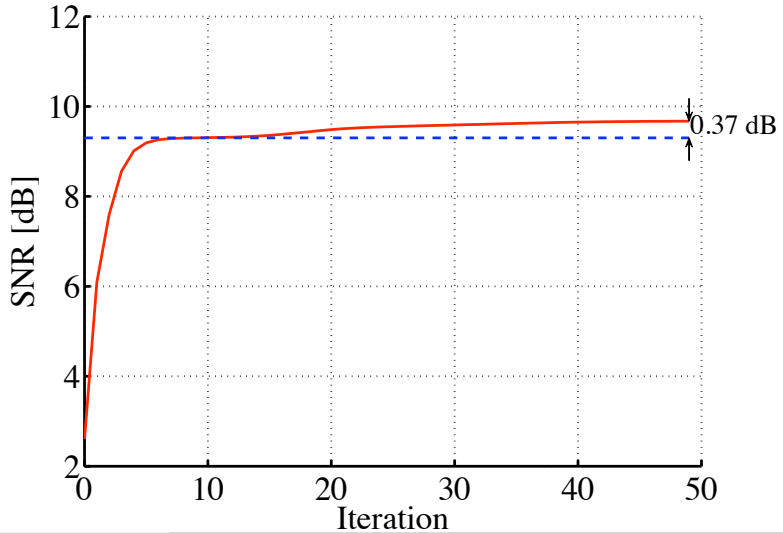


after iteration 49

- Initialization:

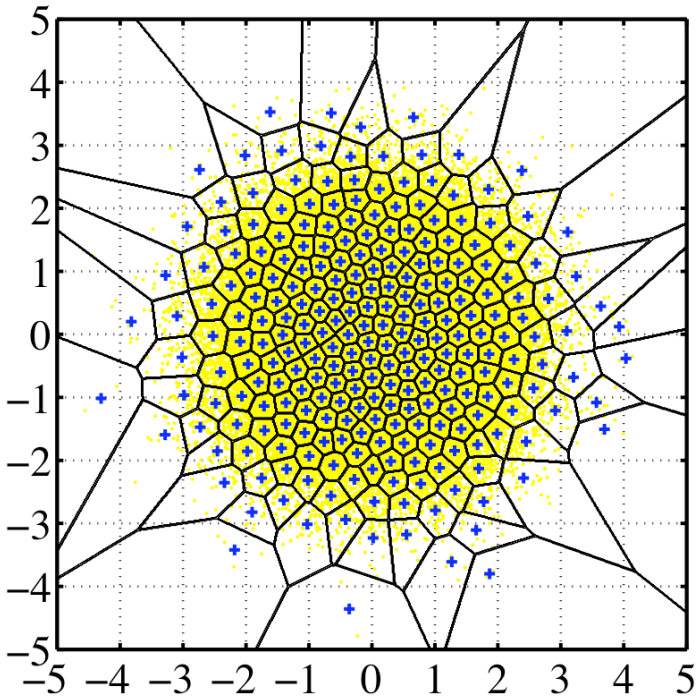
  $$\boldsymbol{s'}_{i+4k} = (-3.75 + 2.5i, -3.75 + 2.5k)^T$$

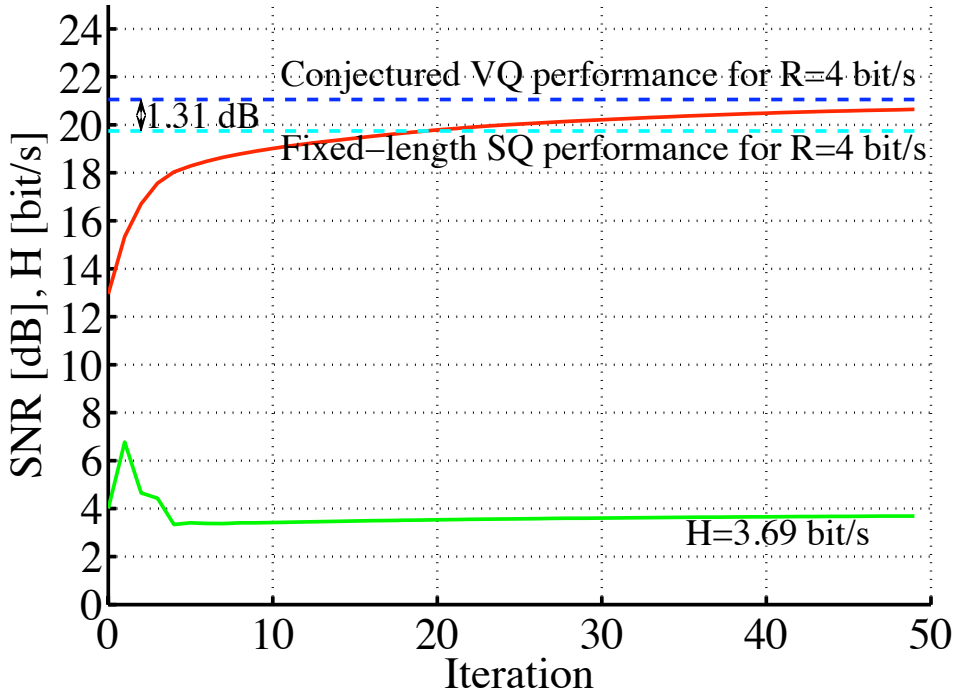
- After iteration 8: Same performance as in scalar case: 9.3 dB
- After iteration 49: Improvement to 9.67 dB



LBG Algorithm Result for Gaussian IID

Result for dimension $N = 2$ and size $K = 256$ corresponding to $R = 4$ bit/sample

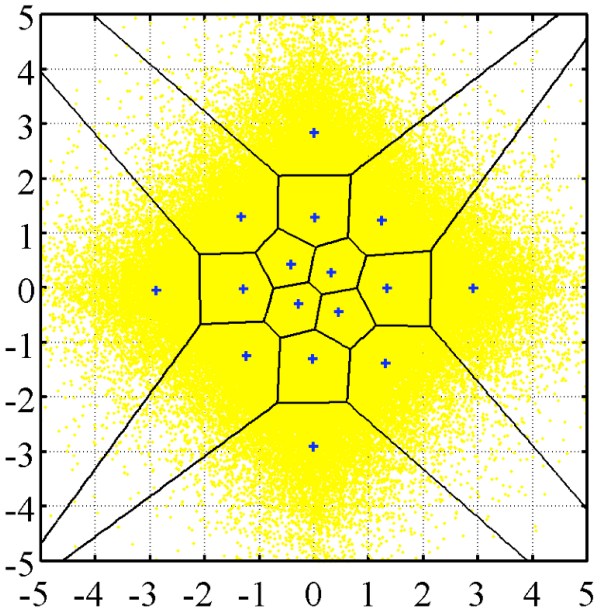


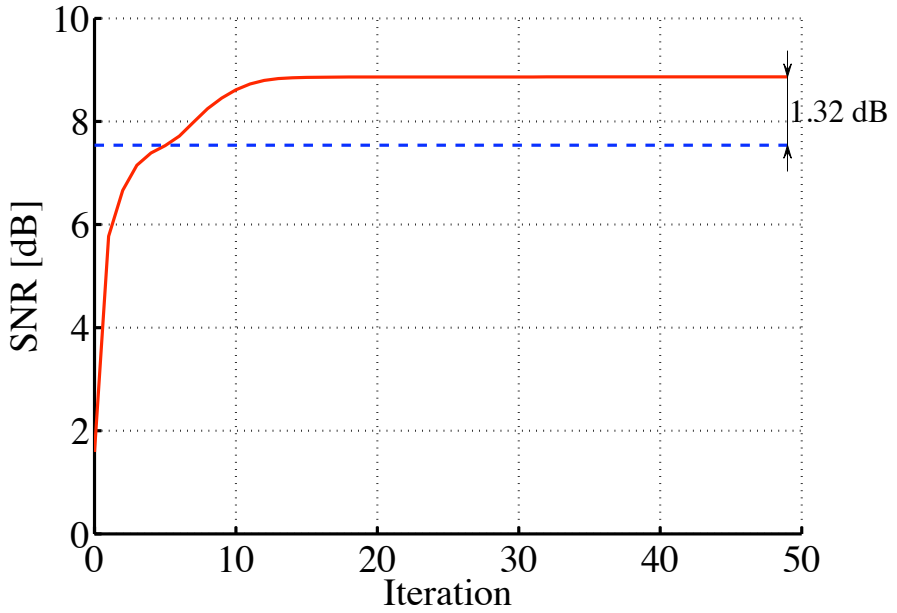


- Random initialization
- Gain around 0.9 dB for two-dimensional VQ compared to SQ with fixed-length codes resulting in 20.64 dB (of conjectured 21.05 dB)

LBG Algorithm Result for Laplacian IID

Result for dimension $N = 2$ and size $K = 16$ corresponding to $R = 2$ bit/sample





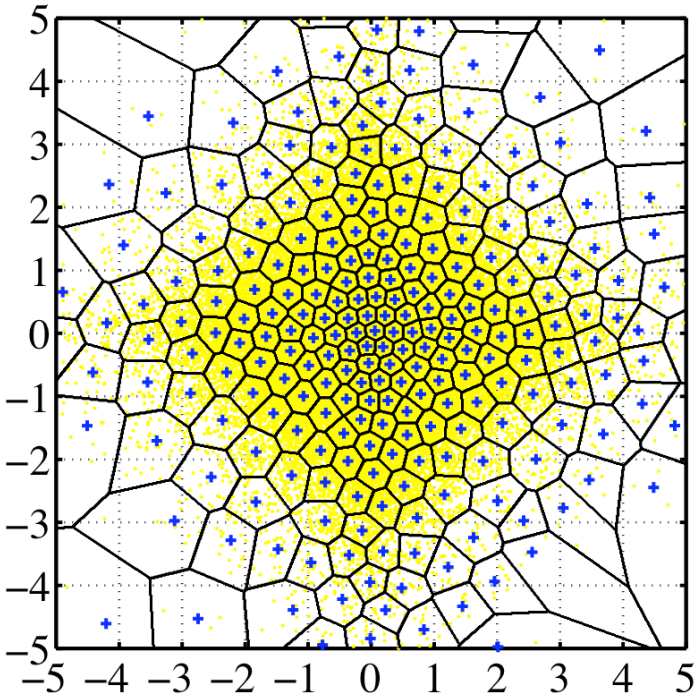
- Initialization (equal to experiment with Gaussian iid):

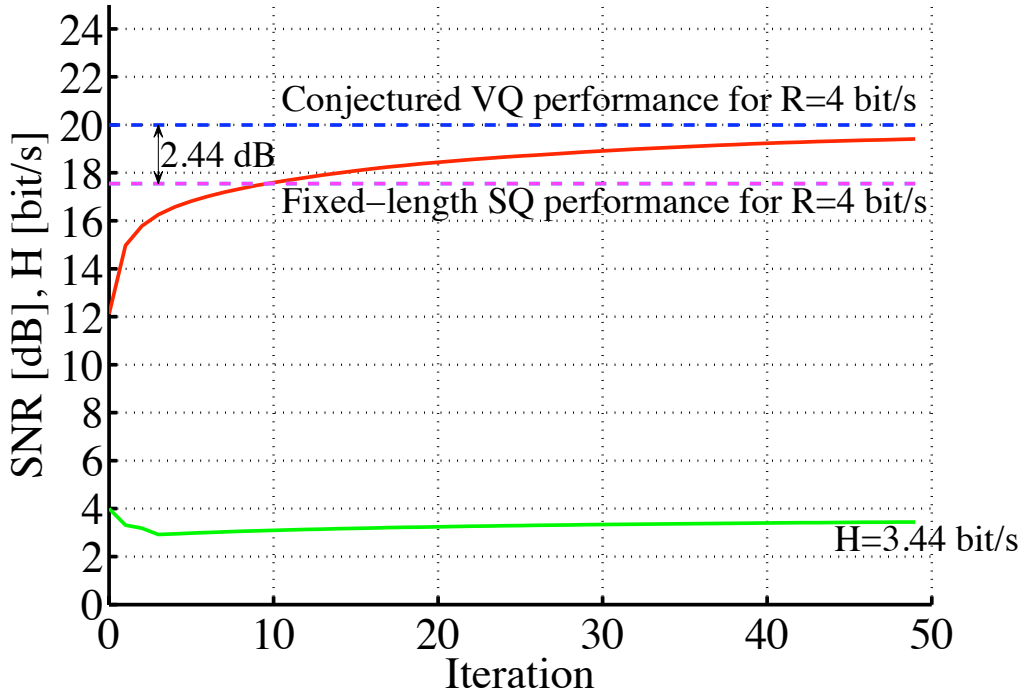
$$\boldsymbol{s'}_{i+4k} = (-3.75 + 2.5i, -3.75 + 2.5k)^T$$

- Large gain (1.32 dB) for two-dimensional VQ compared to SQ with fixed-length codes resulting in 8.87 dB

LBG Algorithm Result for Laplacian IID

Result for dimension $N = 2$ and size $K = 256$ corresponding to $R = 4$ bit/sample





- Random initialization
- Large gain (1.84 dB) for two-dimensional VQ compared to SQ with fixed-length codes resulting in 19.4 dB (of conjectured 19.99 dB)

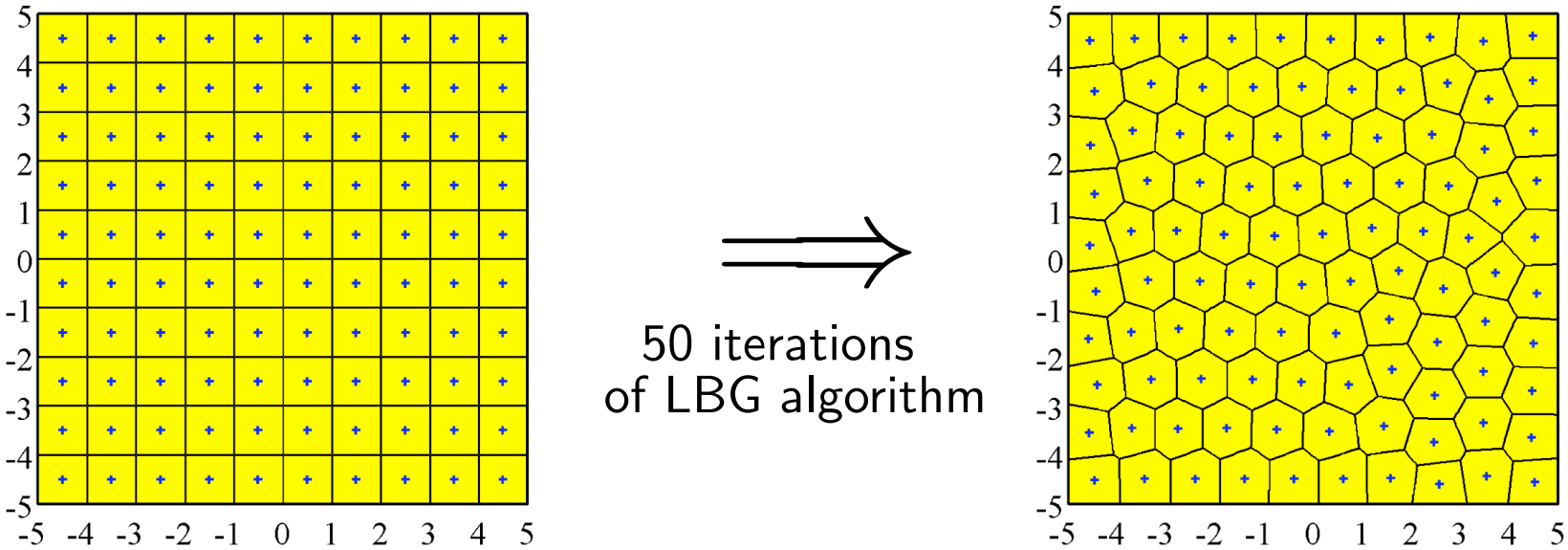
The Vector Quantizer Advantage

Gain over scalar quantization can be assigned to 3 effects

- **Space filling advantage**:
  - $\mathbb{Z}$ lattice is not most efficient sphere packing in N dimensions $(N > 1)$
  - Independent from source distribution or statistical dependencies
  - Maximum gain for $N \to \infty$: 1.53 dB
- **Shape advantage**:
  - Exploit shape of source pdf
  - Can also be exploited using entropy-constrained scalar quantization
- **Memory advantage**:
  - Exploit statistical dependencies of the source
  - Can also be exploited using predictive coding, transform coding, block entropy coding or conditional entropy coding

Space Filling Advantage

Consider uniform iid source with $f(s) = 1/A$ for $-A/2 \leq s \leq A/2$ and $A = 10$



- $D_U(R)$ for SQ of uniform distribution is given as $D_U(R) = \frac{A^2}{12} 2^{-2R}$; with $A = 10$ and $R = 3.32$ bit/scalar we have $D_U(R) = 19.98$ dB
- LBG algorithm converged towards 20.08 dB showing an approximate hexagonal lattice in 2D

Space-Filling Advantage: Densest Sphere Packings

Densest packings, highest kissing numbers, and approximate gain using VQ

Dim.	Densest Packing	Name	Highest Kissing Number	Approximate Gain [dB]
1	$\mathbb{Z}$	– Integer lattice	2	0
2	A_2	– Hexagonal lattice	6	0.17
3	$A_3 \simeq D_3$	– Cuboidal lattice	12	0.29
4	D_4		24	0.39
5	D_5		40	0.47
6	E_6		72	0.54
7	E_7		126	0.60
8	E_8	– Gosset lattice	240	0.66
9	Λ_9	– Laminated lattice	240	0.70
10	P_{10c}	– Non-lattice arrangement	336	0.74
12	K_{12}	– Coxeter-Todd lattice	756	0.81
16	$BW_{16} \simeq \Lambda_{16}$	– Barnes-Wall lattice	4320	0.91
24	Λ_{24}	– Leech lattice	196560	1.04
100				1.35
∞				1.53

Chou-Lookabaugh-Gray Algorithm: ECVQ

Given is

- a sufficiently large realization $\{\boldsymbol{s}_n\}$ of considered sources
- a Lagrange parameter λ

Iterative quantizer design (extension of EC Lloyd algorithm)

1. Choose initial set of reconstruction vectors $\{\boldsymbol{s}'_i\}$ and codeword lengths $\{\ell_i\}$
2. Associate all samples of the training set $\{\boldsymbol{s}_n\}$ with one of the quantization cells $\mathcal{C}_i$ according to

$$\alpha(\boldsymbol{s}_n) = \arg\min_{\forall \boldsymbol{s}'_i} \; d_N(\boldsymbol{s}_n, \boldsymbol{s}'_i) + \lambda \cdot \ell_i$$

3. Update the reconstruction vectors $\{\boldsymbol{s}'_i\}$ according to

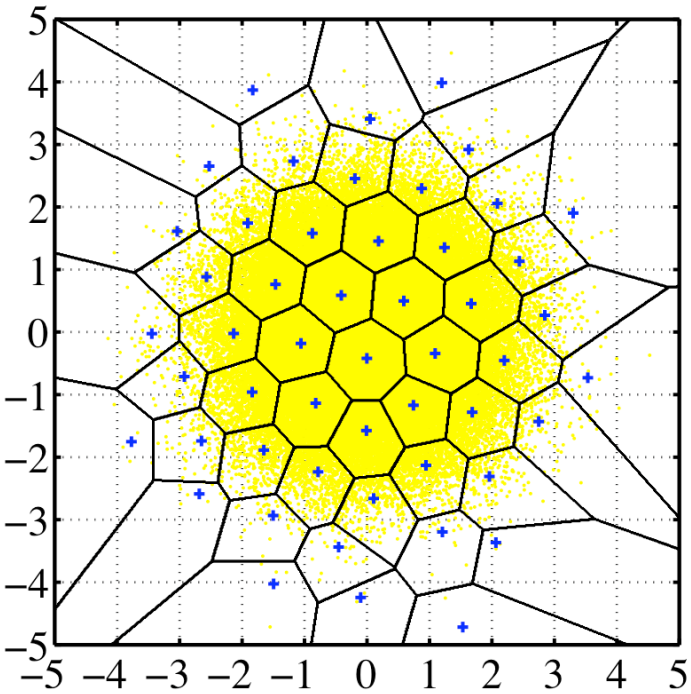
$$\boldsymbol{s}'_i = \arg\min_{\boldsymbol{s}' \in \mathcal{R}} \; E\{d_N(\boldsymbol{S}, \boldsymbol{s}') \,|\, \alpha(\boldsymbol{S}) = i\}$$

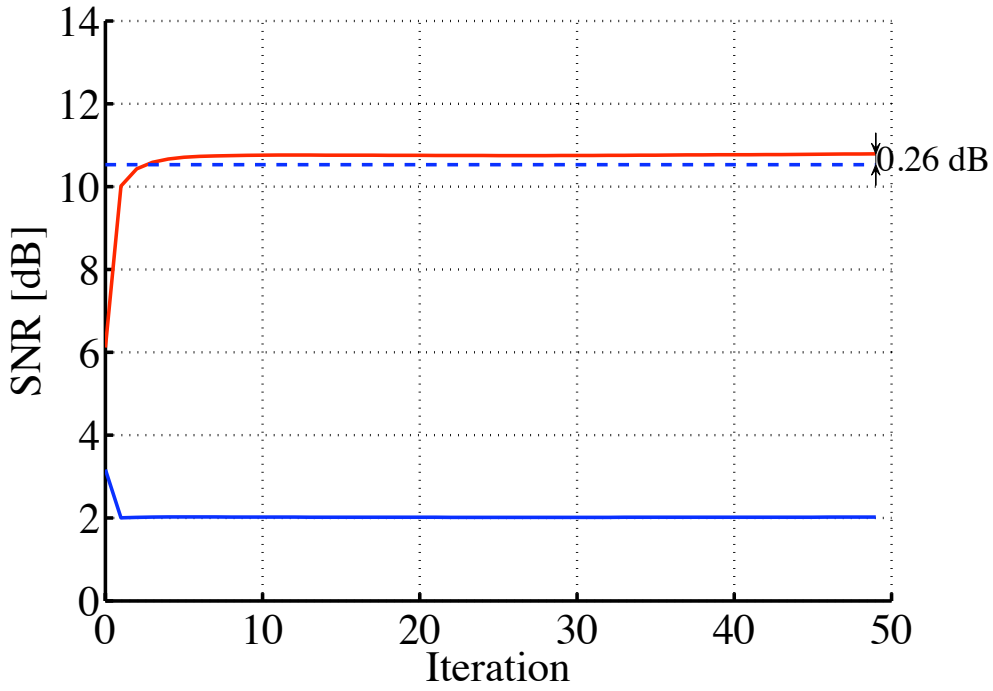
4. Update the codeword lengths ℓ_i according to

$$\ell_i = -\log_2 p_i$$

5. Repeat previous three steps until convergence

Shape Advantage: Results for Gaussian IID ($N = 2$, $K = 16$)

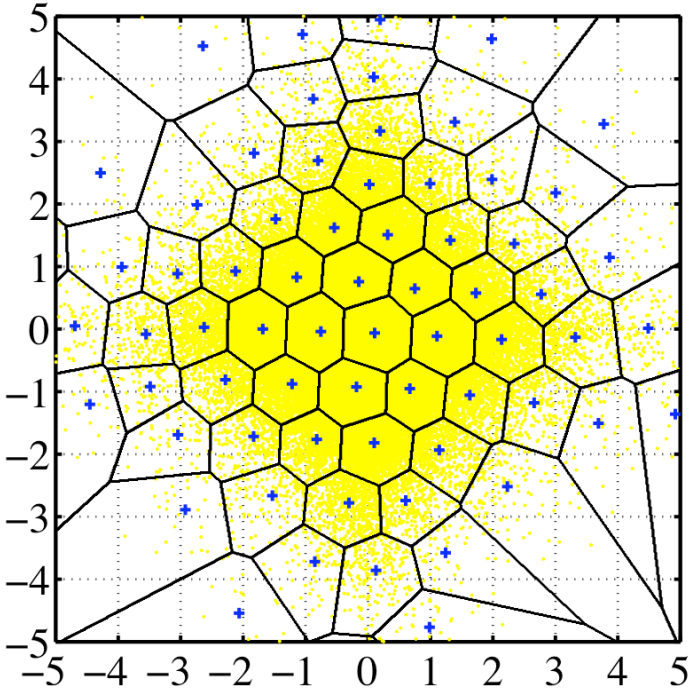


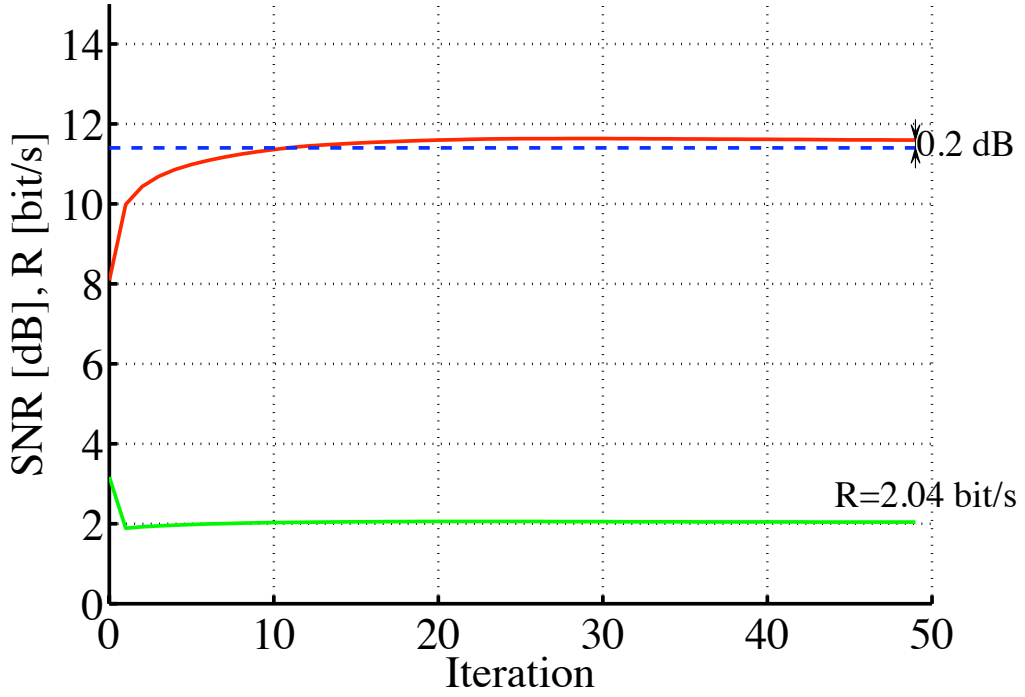


Result of CLG algorithm for Gaussian iid

- Gain of ECVQ compared to ECSQ is 0.26 dB
- Gain of VQ compared to SQ with fixed-length codes is 0.37 dB

Shape Advantage: Results for Laplace IID ($N = 2$, $K = 16$)

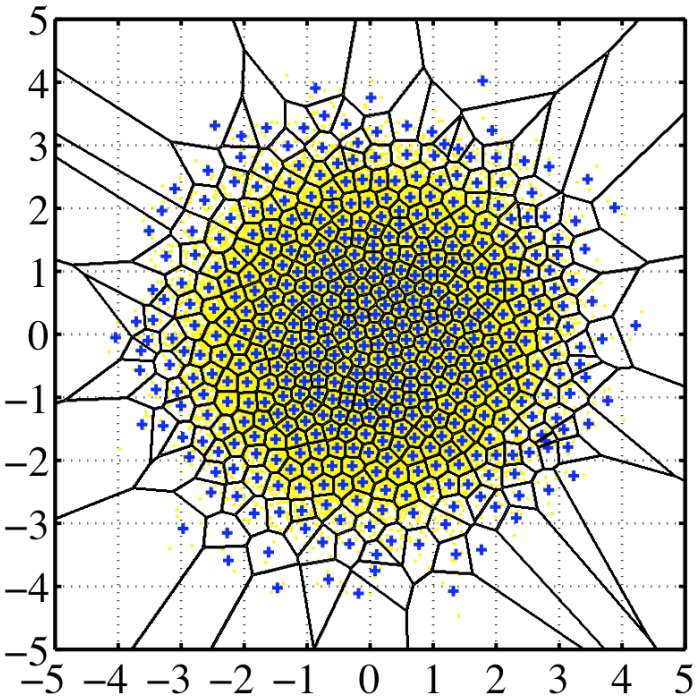


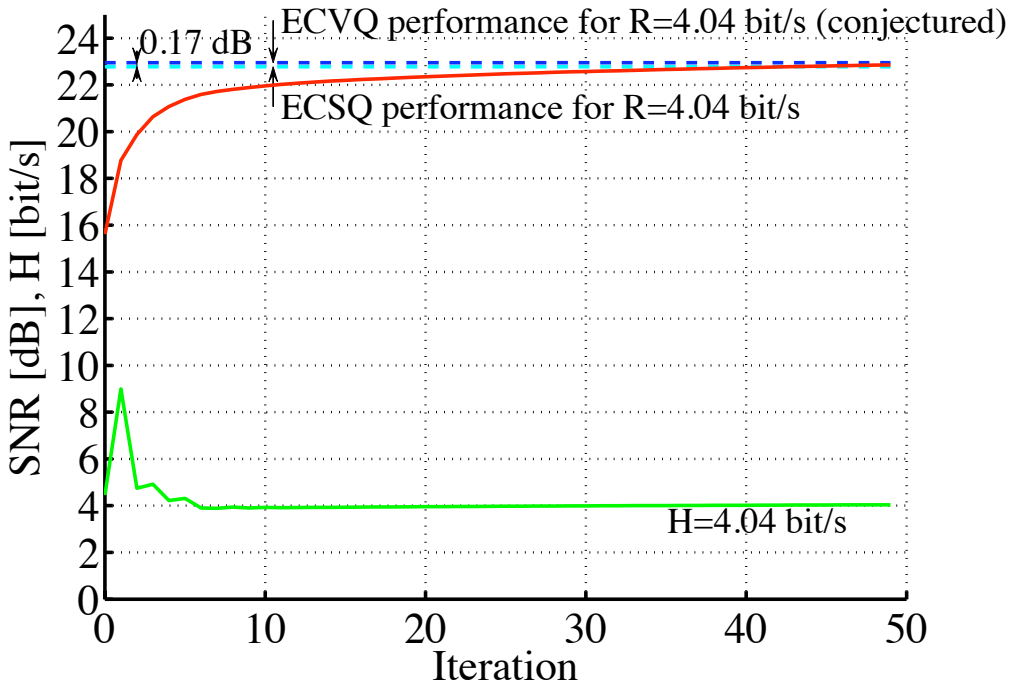


Result of CLG algorithm for Laplace iid

- Gain of ECVQ compared to ECSQ is 0.20 dB
- Gain of VQ compared to SQ with fixed-length codes is 1.32 dB

Shape Advantage: Results for Gaussian IID ($N = 2$, $K = 256$)

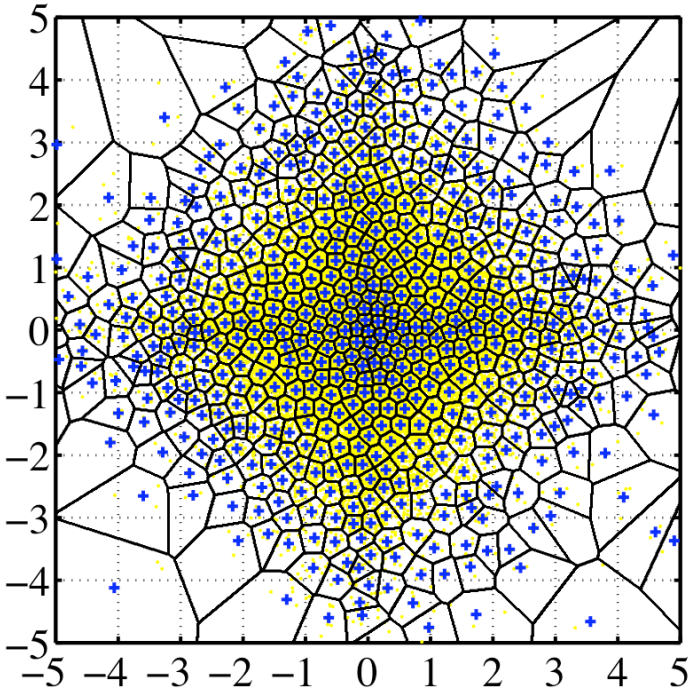


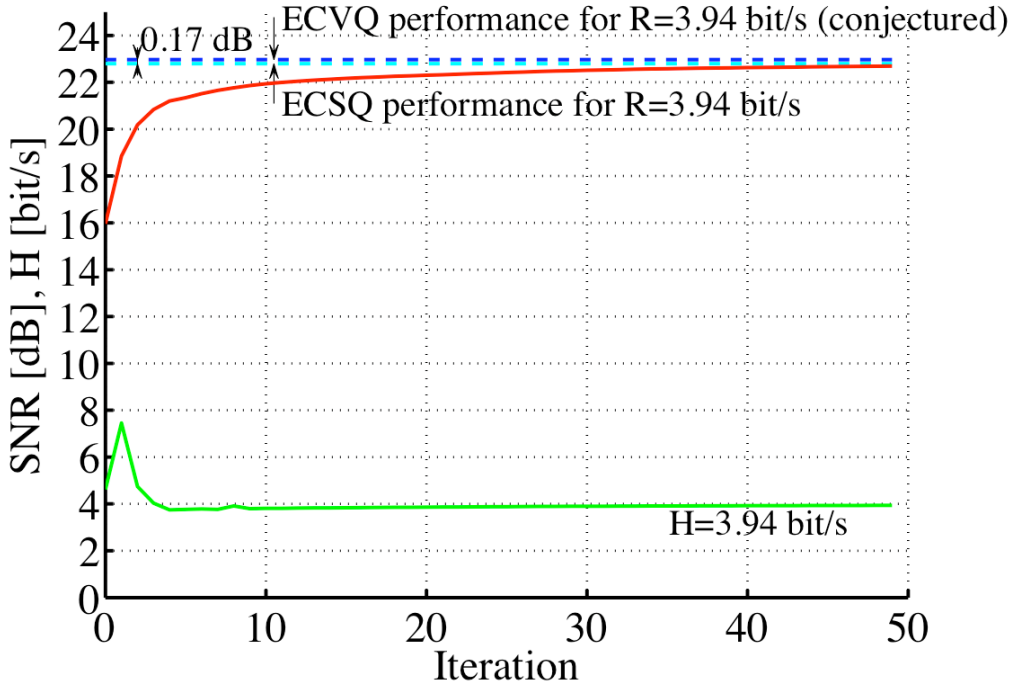


Result of CLG algorithm for Gaussian iid

- Gain of ECVQ compared to ECSQ is 0.17 dB
- Gain of VQ compared to SQ with fixed-length codes is 0.9 dB

Shape Advantage: Results for Laplace IID ($N = 2$; $K = 256$)





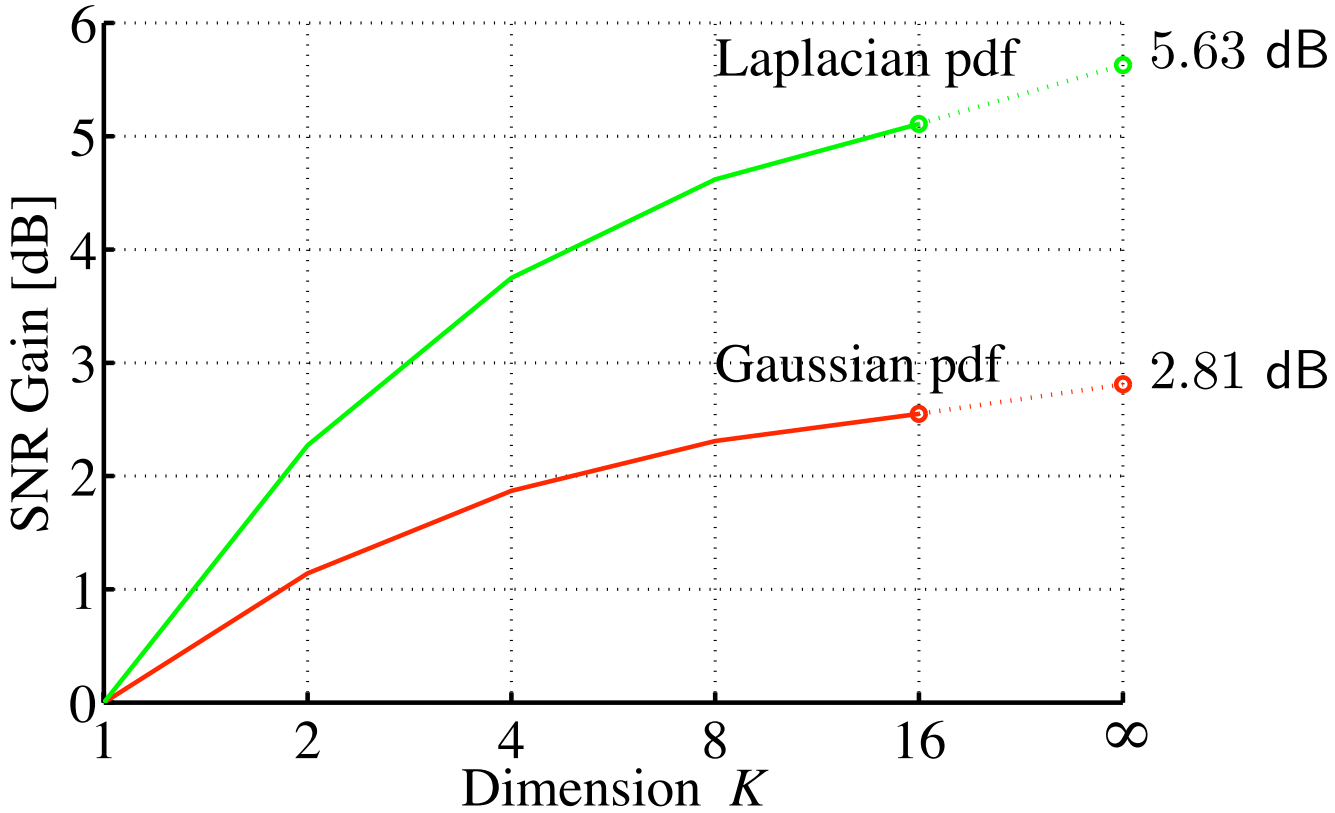
Result of CLG algorithm for 2D Laplace i.i.d.

- Gain of ECVQ compared to ECSQ is 0.17 dB
- Gain of VQ compared to SQ with fixed-length codes is 1.84 dB

$\Longrightarrow$ Entropy coding of quantization indices only leaves the space-filling gain, which is approximately 0.17 dB for $N = 2$

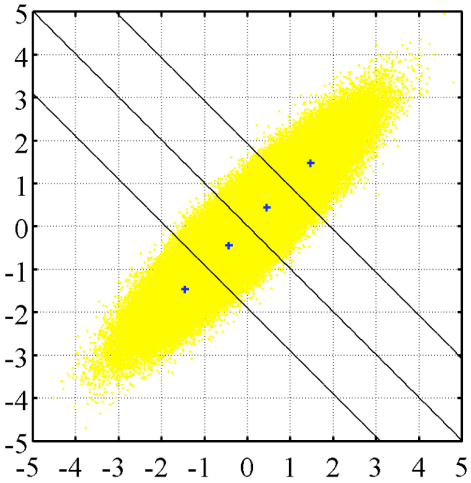
Summary on Shape Advantage

- When comparing ECSQ with ECVQ for iid sources, the gain due to $K > 1$ reduces to the space filling gain
- VQ with fixed-length codes can also exploit the gain that ECSQ shows compared to SQ with fixed-length codes



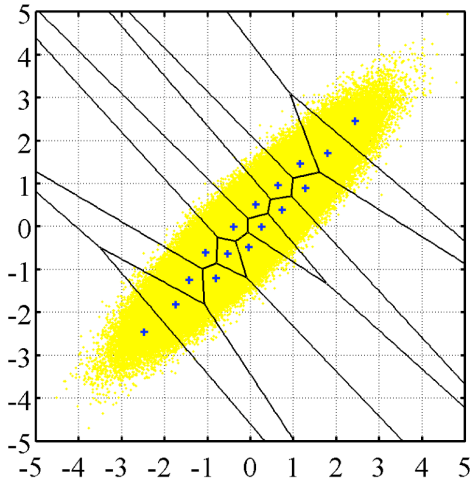
Memory Advantage: Results for Gauss-Markov with $\rho = 0.9$

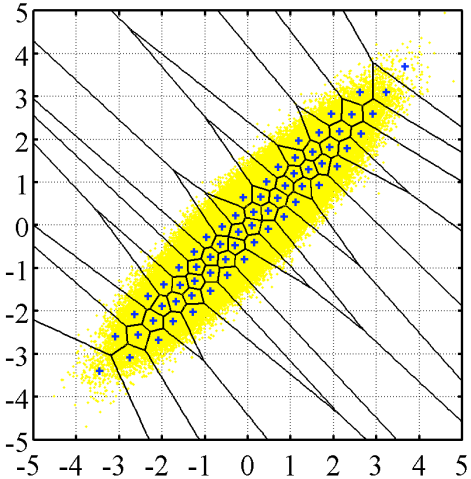
VQ results from LBG algorithm for Gauss-Markov source with correlation $\rho = 0.9$



$\Longleftarrow \quad R = 1$ bit/scalar

$R = 2$ bit/scalar $\Longrightarrow$

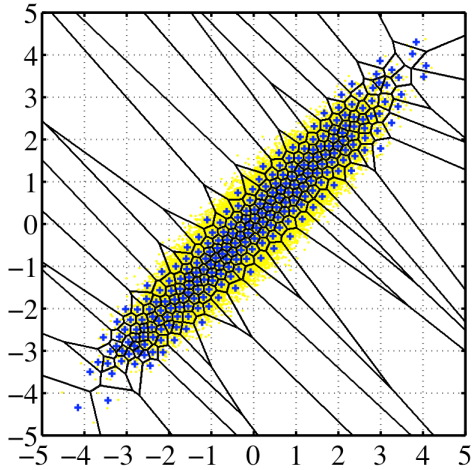




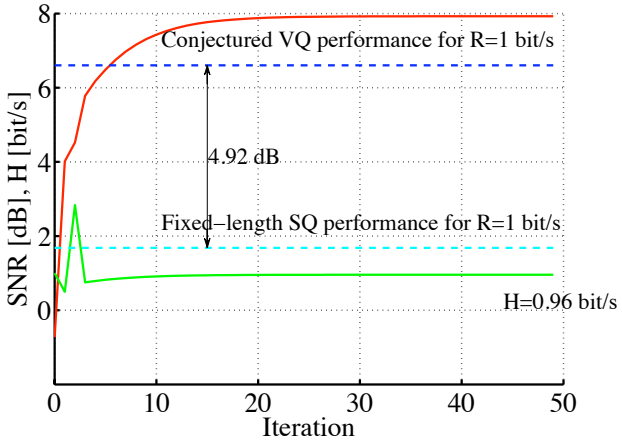
$\Longleftarrow \quad R = 3$ bit/scalar

$R = 4$ bit/scalar $\Longrightarrow$

LBG algorithm has been extended by re-inserting discarded symbols s'_i using random choices

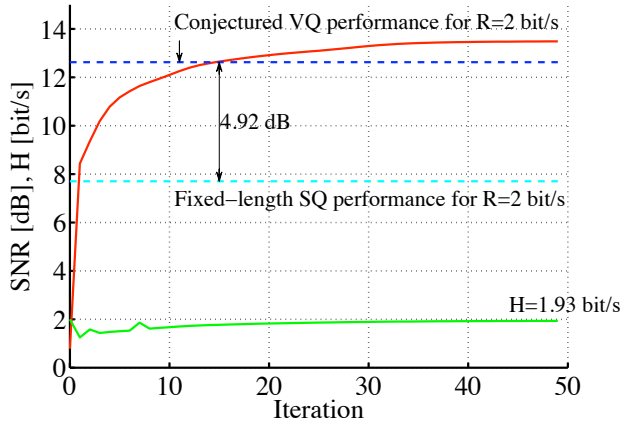


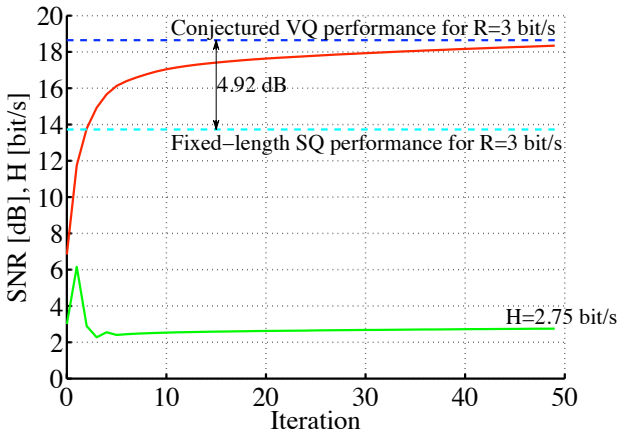
Memory Advantage: Results for Gauss-Markov with $\rho = 0.9$



$\Leftarrow$ 1 bit/scalar

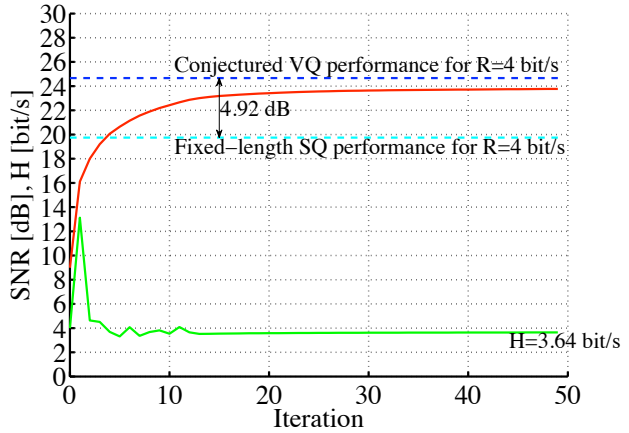
2 bit/scalar $\Rightarrow$





$\Leftarrow$ 3 bit/scalar

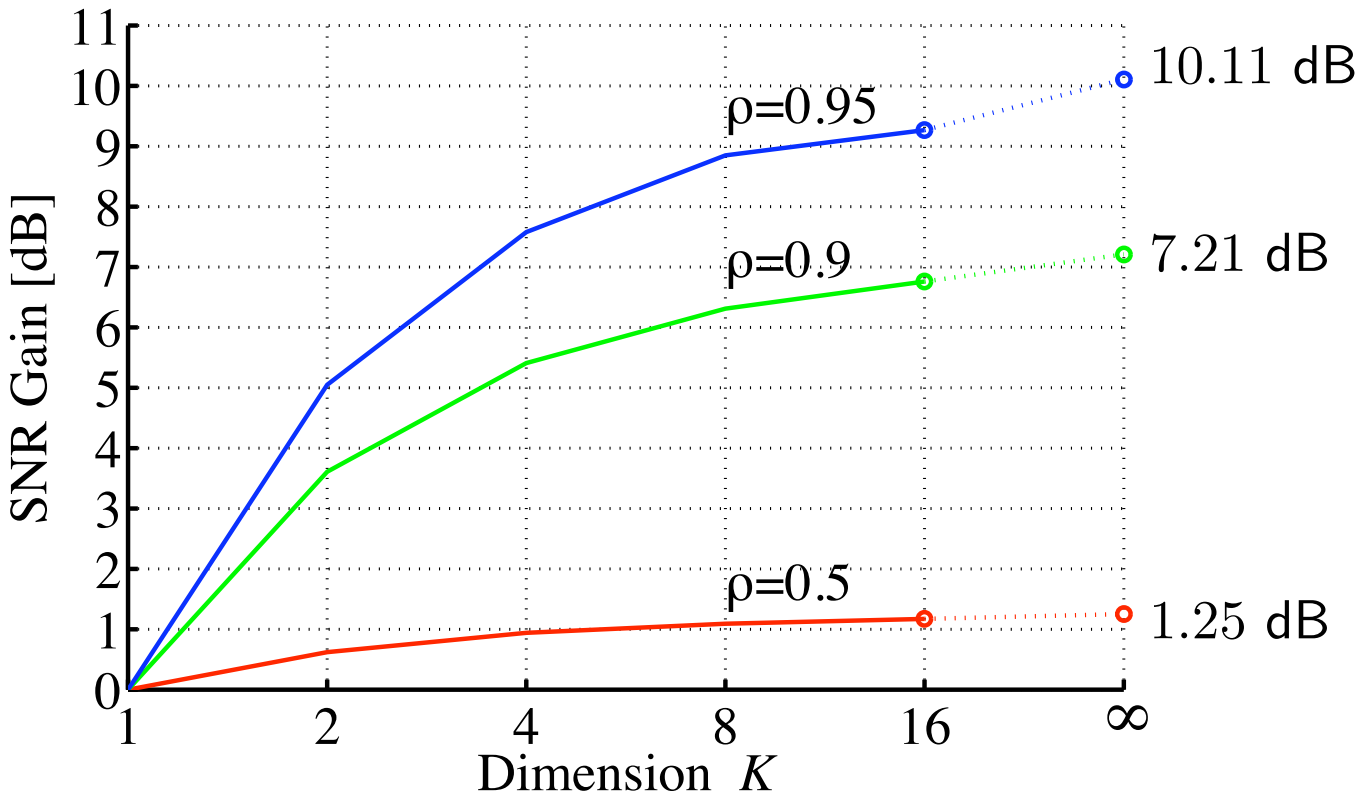
4 bit/scalar $\Rightarrow$



- Gains are additive from space-filling, shape and memory effects
- For high rates, conjectured VQ performance is approached

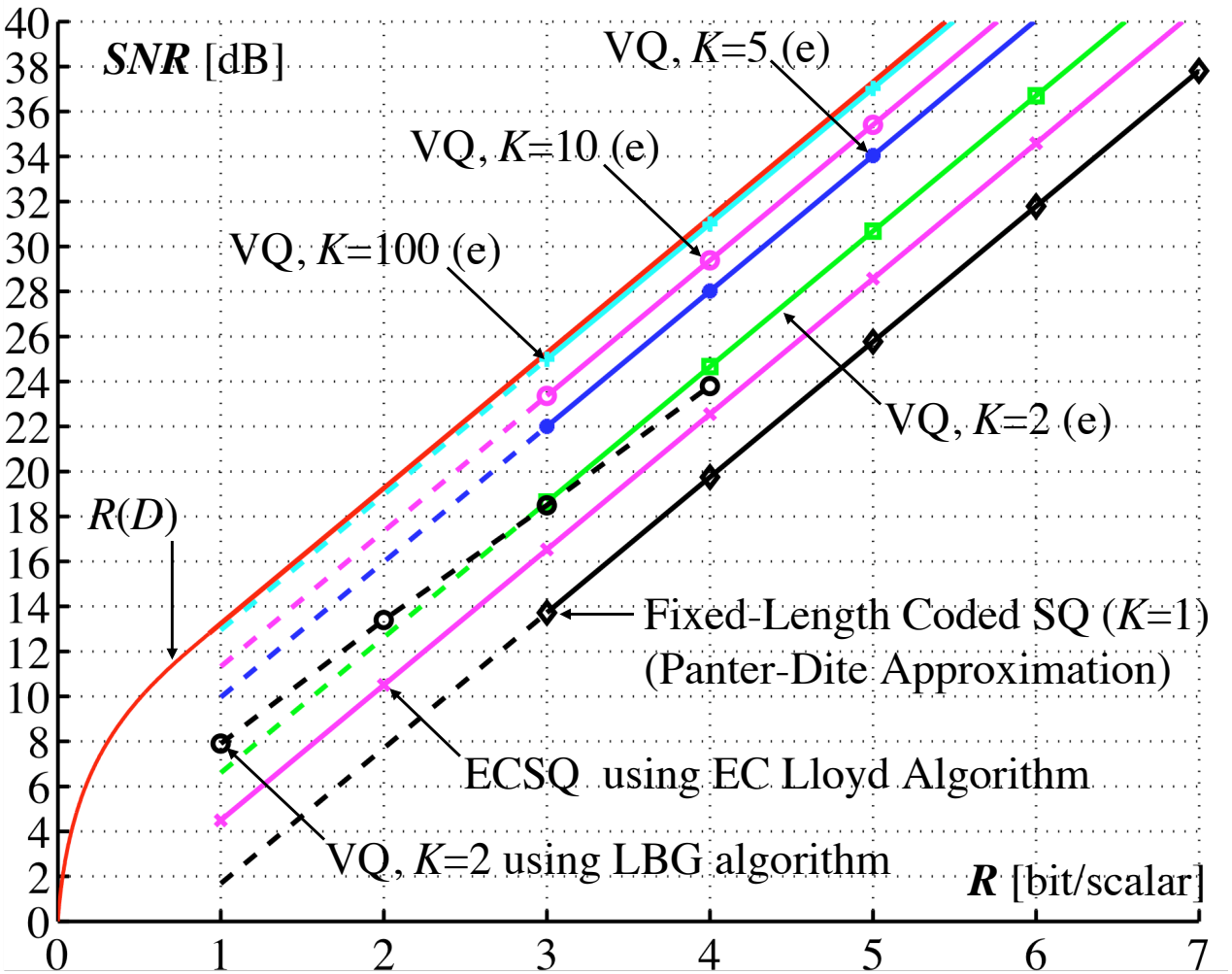
Summary on Memory Advantage

- Largest gain to be made if source contains statistical dependencies
- Exploiting the memory advantage is one of the most relevant aspects of source coding (shape advantage can be obtained using entropy coding)
- Remainder of source coding course will consider this issue



Vector Quantizer Advantage for a Gauss-Markov Source

- Gauss-Markov source with correlation factor $\rho = 0.9$
- Conjectured numbers are empirically verified for $K = 2$



Vector Quantization with Structural Constraints

- Vector quantizers can asymptotically achieve rate-distortion curve for $N \to \infty$
- Complexity requirements: Storage and computation
- Delay
- Impose structural constraints can reduce complexity
  - Tree-Structured VQ
  - Transform VQ
  - Multistage VQ
  - Shape-Gain VQ
  - Lattice Codebook VQ
  - Predictive VQ
- Predictive VQ can be seen as a generalization of very popular techniques: Motion compensation in video coding and various techniques in speech coding

Chapter Summary

Scalar quantization

- Lloyd quantizer: Minimum distortion for given number of representative levels
- Variable length coding: Additional gains by entropy-constrained quantization
- Optimal scalar quantizer for high rates: Uniform quantizer

Vector quantization

- Vector quantizers can achieve rate-distortion curve for $N \to \infty$
- Space filling gain: Only exploited by vector quantizers (1.53 dB for $N \to \infty$)
- Shape gain: Can also be exploited by entropy coding of quantization indices
- Memory gain: Can be exploited by predictive coding, transform coding, or entropy coding using joint or conditional probability mass functions

Vector quantization can achive rate-distortion bound. – Are we done?

$\Longrightarrow$ No! Complexity of vector quantizers is the issue

$\Longrightarrow$ **Design a coding system with optimum rate distortion performance, such that the delay, complexity, and storage requirements are met**

Exercise 15

Consider a symmetric scalar quantizer with 3 intervals and a quantizer input with a zero-mean Laplace pdf,

$$q(x) = \begin{cases} -b & : \quad x < -a \\ 0 & : \quad |x| \leq a \\ b & : \quad x > a \end{cases} \qquad\qquad f(x) = \frac{1}{2m} e^{-\frac{|x|}{m}}$$

(a) Derive the optimal reconstruction value b as a function of the decision threshold a for MSE distortion.

Express the resulting distortion as function of a and the variance $\sigma^2 = 2m^2$.

(b) Determine the decision threshold a in a way that a Lloyd quantizer for MSE distortion is obtained.

Determine the distortion and rate for the Lloyd quantizer by assuming fixed-length coding ($R = \log_2 N$) and compare the obtained R-D point with the Shannon lower bound.

(c) Can the derived optimal quantizer for fixed-length coding be improved by adding entropy coding (without changing the decision thresholds and reconstruction levels)?

Exercise 16

Given is a Centroidal quantizer (not necessarily a Lloyd quantizer) for MSE distortion and a source X. The quantizer has 5 reconstruction levels $\{-3, -1, 0, 1, 3\}$ which are chosen with probabilities $\{0.05, 0.1, 0.4, 0.3, 0.15\}$ and achieves an MSE of 1.05.

(a) Determine the mean μ and variance σ^2 of the source X.

(b) With $q(X)$ being the quantizer output and $e(X) = X - q(X)$ being the quantization error, determine the correlations $E\{X\, q(X)\}$, $E\{X\, e(X)\}$, and $E\{q(X)\, e(X)\}$.

Exercise 17

Consider a discrete Markov process $\mathbf{X} = \{X_n\}$ with the symbol alphabet $\mathcal{A}_X = \{0, 2, 4, 6\}$ and the conditional pmf

$$p_{X_n|X_{n-1}}(x_n|x_{n-1}) = \begin{cases} a & : \quad x_n = x_{n-1} \\ \frac{1}{3}(1-a) & : \quad x_n \neq x_{n-1} \end{cases},$$

for $x_n, x_{n-1} \in \mathcal{A}_X$. The parameter a, with $0 < a < 1$, is a variable that specifies the probability that the current symbol is equal to the previous symbol. For $a = 1/4$, our source $\mathbf{X}$ would be i.i.d.
Given is a quantizer of size 2 with the reconstruction levels $s'_0 = 1$ and $s'_1 = 5$ and the decision threshold $u_1 = 3$.

(a) Assume optimal entropy coding using the marginal probabilities of the quantization indices and determine the rate-distortion point of the quantizer.

(b) Can the overall quantizer performance be improved by applying conditional entropy coding (e.g., using arithmetic coding with conditional probabilities)? How does it depend on the parameter a?

Exercise 18

Calculate the gain of optimal 2-dimensional vector quantization relative to optimal scalar quantization for high rates on the example of a uniform pdf.

Hint:
For high rates, border effects can be neglected. It can be assumed that the signal space for which the pdf is non-zero is completely filled with regular quantization cells.

Exercise 19

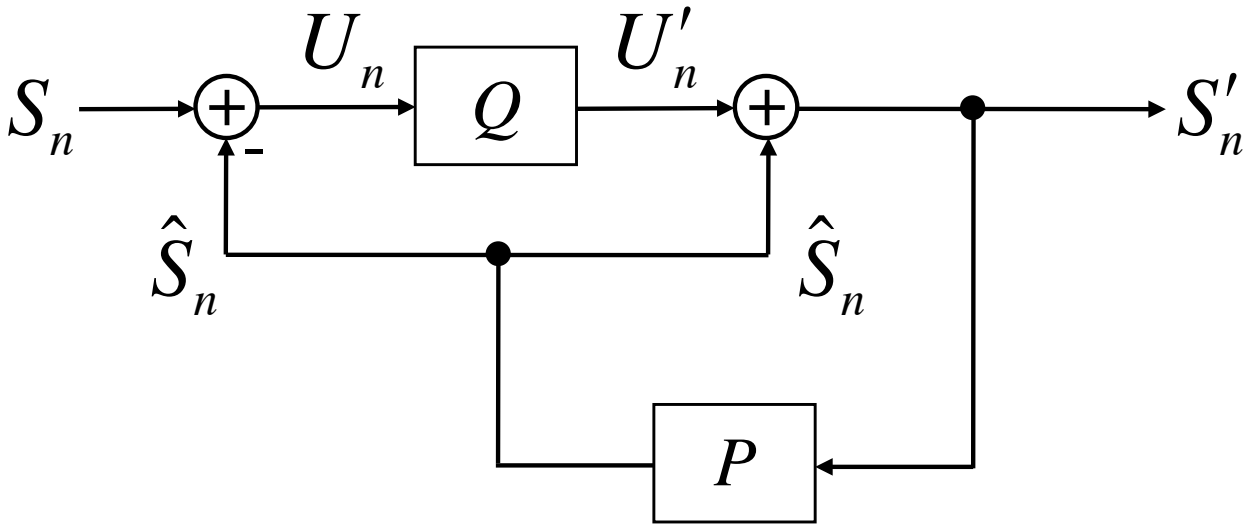
Consider scalar quantization of a Laplacian source at high rates:

$$f(x) = \frac{\lambda}{2} \cdot e^{-\lambda\,|x|} \qquad \text{with} \qquad \sigma_S^2 = \frac{2}{\lambda^2}$$

In a given system, the used quantizer is a Lloyd quantizer with fixed-length entropy coding (the number of quantization intervals represents a power of 2). How many bits per sample can be saved if the quantizer is replaced by an entropy-constrained quantizer with optimal entropy coding?

Note: The operation points of the quantizers can be accurately described by high rate approximations.

Predictive Coding



Outline

Part I: Source Coding Fundamentals

- Probability, Random Variables and Random Processes
- Lossless Source Coding
- Rate-Distortion Theory
- Quantization
- **Predictive Coding**
  - Optimal Prediction
  - Linear and Affine Prediction
  - Predictive Coding: DPCM
- Transform Coding

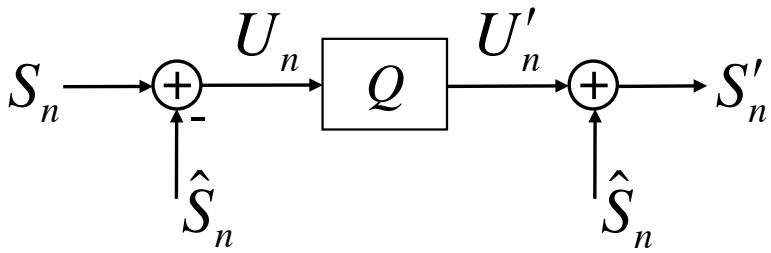
Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

Predictive Coding – Introduction

- The better the future of a random process is predicted from the past and the more redundancy the signal contains, the less new information is contributed by each successive observation of the process

- Predictive coding idea:
  1. Predict a sample using an estimate which is a function of past samples
  2. Quantize residual between signal and its prediction
  3. Add quantizer residual and prediction to obtain reconstructed sample



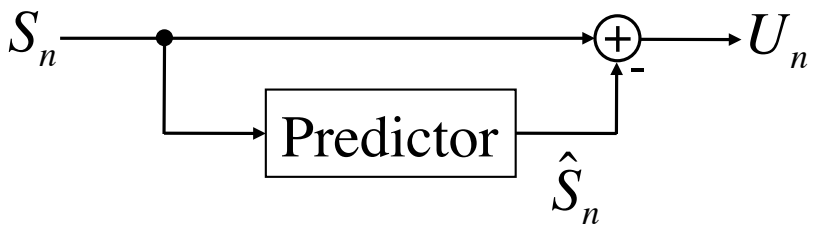
Problems:

- How to obtain the predictor $\hat{S}_n$?
- How to combine predictor and quantizer?

Prediction

- Statistical estimation procedure:

  Value of random variable S_n of random process $\{S_n\}$ is estimated using values of other random variables of the random process



- Select: Set of observed random variables $\mathcal{B}_n$

  $\Longrightarrow$ Typical example: N random variables that directly precede S_n

$$\mathcal{B}_n = \{S_{n-1}, S_{n-2}, \cdots, S_{n-N}\} \tag{441}$$

- Predictor for S_n: Deterministic function of observation set $\mathcal{B}_n$

$$\boxed{\hat{S}_n = A_n(\mathcal{B}_n)} \tag{442}$$

- Prediction error

$$\boxed{U_n = S_n - \hat{S}_n = S_n - A_n(\mathcal{B}_n)} \tag{443}$$

Prediction Performance

MSE distortion using $u_i = s_i - \hat{s}_i$ and $s'_i = u'_i + \hat{s}_i$

$$d_N(\boldsymbol{s}, \boldsymbol{s'}) = \frac{1}{N} \sum_{i=0}^{N-1} (s_i - s'_i)^2 = \frac{1}{N} \sum_{i=0}^{N-1} (u_i + \hat{s}_i - u'_i - \hat{s}_i)^2 = d_N(\boldsymbol{u}, \boldsymbol{u'}) \quad (444)$$

$\Longrightarrow$ Operational rate-distortion function of a predictive coding is equal to op. r-d function of (scalar) quantization of the prediction residuals

Operational distortion-rate function: $D(R) = \sigma_U^2 \cdot g(R)$

- σ_U^2: variance of the prediction residual
- $g(R)$: depends only on the type of the distribution of the residuals

$\Longrightarrow$ Assume stationary processes: $A_n(\cdot)$ becomes $A(\cdot)$

$\Longrightarrow$ Neglect the dependency on the distribution type

$\Longrightarrow$ Define: Predictor $A(\mathcal{B}_n)$ given an observation set $\mathcal{B}_n$ is optimal if it minimizes variance σ_U^2

Optimal Prediction

- Optimization criterion typically used in literature:

$$\boxed{\epsilon_U^2 = E\{U_n^2\} = E\Big\{\big(S_n - \hat{S}_n\big)^2\Big\} = E\Big\{\big(S_n - A(\mathcal{B}_n)\big)^2\Big\}} \tag{445}$$

- Minimization of second moment

$$\begin{aligned} \epsilon_U^2 &= E\{(U_n - \mu_U + \mu_U)^2\} \\ &= E\{(U_n - \mu_U)^2\} + 2E\{(U_n - \mu_U)\mu_U\} + E\{\mu_U^2\} \\ &= \sigma_U^2 + \mu_U^2 + 2\mu_U(E\{U_n\} - \mu_U) \\ &= \sigma_U^2 + \mu_U^2 \end{aligned} \tag{446}$$

  implies minimization of variance σ_U^2 and mean μ_U

- Solution: Conditional mean (see proof in [Wiegand and Schwarz])

$$\boxed{\hat{S}_n^* = A^*(\mathcal{B}_n) = E\{S_n \,|\, \mathcal{B}_n\}} \tag{447}$$

  $\Longrightarrow$ General case requires storage of large tables

Optimal Prediction for Autoregressive Processes

- Autoregressive process of order m (AR(m) process)

$$\begin{aligned} S_n &= Z_n + \mu_S + \sum_{i=1}^{m} a_i \cdot (S_{n-i} - \mu_S) \\ &= Z_n + \mu_S \cdot (1 - \boldsymbol{a}_m^T \boldsymbol{e}_m) + \boldsymbol{a}_m^T \boldsymbol{S}_{n-1}^{(m)} \end{aligned} \tag{448}$$

  where
  - $\{Z_n\}$ is a zero-mean iid process
  - μ_S is the mean of the AR(m) process
  - $\boldsymbol{a}_m = (a_1, \cdots, a_m)^T$ is a constant parameter vector
  - $\boldsymbol{e}_m = (1, \cdots, 1)^T$ is an m-dimensional unit vector

- Prediction of S_n given the vector $\boldsymbol{S}_{n-1} = (S_{n-1}, \cdots, S_{n-N})$ with $N \geq m$

$$\begin{aligned} E\{S_n \,|\, \boldsymbol{S}_{n-1}\} &= E\{Z_n + \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + \boldsymbol{a}_N^T \, \boldsymbol{S}_{n-1} \,|\, \boldsymbol{S}_{n-1}\} \\ &= \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + \boldsymbol{a}_N^T \, \boldsymbol{S}_{n-1} \end{aligned} \tag{449}$$

  where $\boldsymbol{a}_N = (a_1, \cdots, a_m, 0, \cdots, 0)^T$

Affine Prediction

- Suitable structural constraint: Affine predictor

$$\boxed{\hat{S}_n = A(\boldsymbol{S}_{n-k}) = h_0 + \boldsymbol{h}_N^T \boldsymbol{S}_{n-k}} \tag{450}$$

where $\boldsymbol{h}_N = (h_1, \cdots, h_N)^T$ is a constant vector and h_0 a constant offset

- Variance σ_U^2 of prediction residual only depends on $\boldsymbol{h}_N$

$$\begin{aligned}
\sigma_U^2(h_0, \boldsymbol{h}_N) &= E\Big\{ \big(U_n - E\{U_n\}\big)^2 \Big\} \\
&= E\Big\{ \Big(S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k} - E\Big\{ S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k} \Big\}\Big)^2 \Big\} \\
&= E\Big\{ \big(S_n - E\{S_n\} - \boldsymbol{h}_N^T \big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\big)^2 \Big\}
\end{aligned} \tag{451}$$

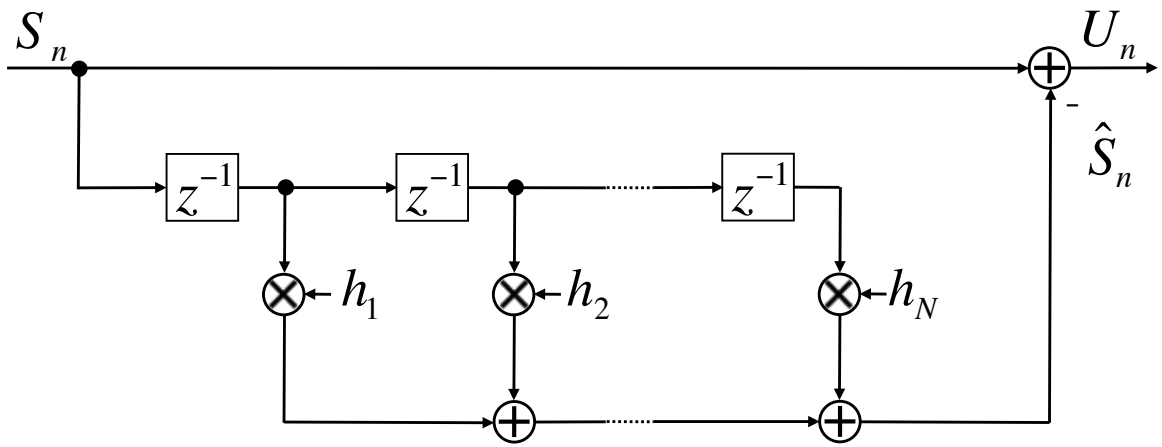
- Mean squared prediction error

$$\begin{aligned}
\epsilon_U^2(h_0, \boldsymbol{h}_N) &= \sigma_U^2(\boldsymbol{h}_N) + \mu_U^2(h_0, \boldsymbol{h}_N) = \sigma_U^2(\boldsymbol{h}_N) + E\Big\{ S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k} \Big\}^2 \\
&= \sigma_U^2(\boldsymbol{h}_N) + \big(\mu_S(1 - \boldsymbol{h}_N^T \boldsymbol{e}_N) - h_0\big)^2
\end{aligned} \tag{452}$$

- Minimize mean squared prediction error by setting

$$h_0^* = \mu_S \, (1 - \boldsymbol{h}_N^T \, \boldsymbol{e}_N) \tag{453}$$

Linear Prediction for Zero-Mean Processes



- The function used for prediction is linear, of the form

$$\hat{S}_n = h_1 \cdot S_{n-1} + h_2 \cdot S_{n-2} + \cdots + h_N \cdot S_{n-N} = \boldsymbol{h}_N^T \boldsymbol{S}_{n-1} \tag{454}$$

- Mean squared prediction error (same as variance for zero mean)

$$\begin{aligned} \sigma_U^2(\boldsymbol{h}_N) &= E\Big\{(S_n - \hat{S}_n)^2\Big\} = E\Big\{(S_n - \boldsymbol{h}_N^T \boldsymbol{S}_{n-1})(S_n - \boldsymbol{S}_{n-1}^T \boldsymbol{h}_N)\Big\} \\ &= E\big\{S_n^2\big\} - 2E\Big\{\boldsymbol{h}_N^T \boldsymbol{S}_{n-1} S_n\Big\} + E\Big\{\boldsymbol{h}_N^T \boldsymbol{S}_{n-1} \boldsymbol{S}_{n-1}^T \boldsymbol{h}_N\Big\} \\ &= E\big\{S_n^2\big\} - 2\boldsymbol{h}_N^T E\{S_n \boldsymbol{S}_{n-1}\} + \boldsymbol{h}_N^T E\Big\{\boldsymbol{S}_{n-1} \boldsymbol{S}_{n-1}^T\Big\} \boldsymbol{h}_N \end{aligned} \tag{455}$$

Autocovariance Matrix and Autocovariance Vector

- Variance $\sigma_S^2 = E\{S_n^2\}$
- Autocovariance vector (for zero mean: Autocorrelation vector)

$$\boldsymbol{c}_k = E\{S_n \boldsymbol{S}_{n-k}\} = \sigma_S^2 \cdot \begin{pmatrix} \rho_k \\ \vdots \\ \rho_i \\ \vdots \\ \rho_N + k - 1 \end{pmatrix} \quad \text{with} \quad \rho_i = E\{S_n \cdot S_{n-i}\} / \sigma_S^2 \tag{456}$$

- Autocovariance matrix (for zero mean: Autocorrelation matrix)

$$\boldsymbol{C}_N = E\Big\{\boldsymbol{S}_{n-1}\boldsymbol{S}_{n-1}^T\Big\} = \sigma_S^2 \cdot \begin{pmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{N-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{N-2} \\ \rho_2 & \rho_1 & 1 & \cdots & \rho_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{N-1} & \rho_{N-2} & \rho_{N-3} & \cdots & 1 \end{pmatrix} \tag{457}$$

Optimal Linear Prediction

- Prediction error variance

$$\boxed{\sigma_U^2(\boldsymbol{h}_N) = \sigma_S^2 - 2\boldsymbol{h}_N^T\boldsymbol{c}_k + \boldsymbol{h}_N^T\boldsymbol{C}_N\,\boldsymbol{h}_N} \tag{458}$$

- Minimization of $\sigma_U^2(\boldsymbol{h}_N)$ yields a system of linear equations

$$\boxed{\boldsymbol{C}_N \cdot \boldsymbol{h}_N = \boldsymbol{c}_k} \tag{459}$$

- When $\boldsymbol{C}_N$ is non-singular

$$\boxed{\boldsymbol{h}_N^* = \boldsymbol{C}_N^{-1} \cdot \boldsymbol{c}_k} \tag{460}$$

- Minimum of $\sigma_U^2(\boldsymbol{h}_N)$ is given as (with $(\boldsymbol{C}_N^{-1}\boldsymbol{c}_k)^T = \boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}$)

$$\begin{aligned}
\sigma_U^2(\boldsymbol{h}_N^*) &= \sigma_S^2 - 2\,(\boldsymbol{h}_N^*)^T\boldsymbol{c}_k + (\boldsymbol{h}_N^*)^T\boldsymbol{C}_N\,\boldsymbol{h}_N^* \\
&= \sigma_S^2 - 2\,\big(\boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}\big)\boldsymbol{c}_k + \big(\boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}\big)\boldsymbol{C}_N(\boldsymbol{C}_N^{-1}\boldsymbol{c}_k) \\
&= \sigma_S^2 - 2\,\boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}\boldsymbol{c}_k + \boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}\boldsymbol{c}_k \\
&= \sigma_S^2 - \boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}\boldsymbol{c}_k = \sigma_S^2 - \boldsymbol{c}_k^T\boldsymbol{h}_N^*
\end{aligned} \tag{461}$$

$\Longrightarrow$ Optimal prediction: Signal variance σ_S^2 is reduced by $\boldsymbol{c}_k^T\boldsymbol{C}_N^{-1}\boldsymbol{c}_k = \boldsymbol{c}_k^T\boldsymbol{h}_N^*$

Verification of Optimality

- The optimality of the solution can be verified by inserting $\boldsymbol{h}_N = \boldsymbol{h}_N^* + \boldsymbol{\delta}_N$ into

$$\sigma_U^2(\boldsymbol{h}_N) = \sigma_S^2 - 2\,\boldsymbol{h}_N^T \boldsymbol{c}_k + \boldsymbol{h}_N^T \boldsymbol{C}_N\,\boldsymbol{h}_N \tag{462}$$

yielding

$$\begin{aligned}
\sigma_U^2(\boldsymbol{h}_N) &= \sigma_S^2 - 2(\boldsymbol{h}_N^* + \boldsymbol{\delta}_N)^T \boldsymbol{c}_k + (\boldsymbol{h}_N^* + \boldsymbol{\delta}_N)^T \boldsymbol{C}_N\,(\boldsymbol{h}_N^* + \boldsymbol{\delta}_N) \\
&= \sigma_S^2 - 2\,(\boldsymbol{h}_N^*)^T \boldsymbol{c}_k - 2\,\boldsymbol{\delta}_N^T \boldsymbol{c}_k + \\
&\qquad (\boldsymbol{h}_N^*)^T \boldsymbol{C}_N\,\boldsymbol{h}_N^* + (\boldsymbol{h}_N^*)^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N + \boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{h}_N^* + \boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N \\
&= \sigma_U^2(\boldsymbol{h}_N^*) - 2\boldsymbol{\delta}_N^T \boldsymbol{c}_k + 2\boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{h}_N^* + \boldsymbol{\delta}_N^T \boldsymbol{C}_N \boldsymbol{\delta}_N \\
&= \sigma_U^2(\boldsymbol{h}_N^*) + \boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N
\end{aligned} \tag{463}$$

- The additional term is always non-negative

$$\boldsymbol{\delta}_N^T \boldsymbol{C}_N\,\boldsymbol{\delta}_N \geq 0 \tag{464}$$

- It is equal to 0 if and only if $\boldsymbol{h}_N = \boldsymbol{h}_N^*$

$\Longrightarrow$ $\boldsymbol{h}_N^*$ is the optimal choice for the prediction parameters

The Orthogonality Principle

- Important property for optimal affine predictors

$$\begin{aligned} E\{U_n\, \boldsymbol{S}_{n-k}\} &= E\Big\{ \big(S_n - h_0 - \boldsymbol{h}_N^T \boldsymbol{S}_{n-k}\big)\, \boldsymbol{S}_{n-k} \Big\} \\ &= E\{S_n\, \boldsymbol{S}_{n-k}\} - h_0\, E\{\boldsymbol{S}_{n-k}\} - E\Big\{ \boldsymbol{S}_{n-k} \boldsymbol{S}_{n-k}^T \Big\}\, \boldsymbol{h}_N \\ &= \boldsymbol{c}_k + \mu_S^2\, \boldsymbol{e}_N - h_0\, \mu_S\, \boldsymbol{e}_N - (\boldsymbol{C}_N + \mu_S^2\, \boldsymbol{e}_N\, \boldsymbol{e}_N^T)\, \boldsymbol{h}_N \\ &= \boldsymbol{c}_k - \boldsymbol{C}_N \boldsymbol{h}_N + \mu_S\, \boldsymbol{e}_N \big(\mu_S\, (1 - \boldsymbol{h}_N^T\, \boldsymbol{e}_N) - h_0 \big) \end{aligned} \tag{465}$$

- Inserting the optimal solutions

$$\boldsymbol{h}_N^* = \boldsymbol{C}_N^{-1} \cdot \boldsymbol{c}_k \qquad \text{and} \qquad h_0^* = \mu_S\, (1 - \boldsymbol{h}_N^T\, \boldsymbol{e}_N) \tag{466}$$

yields

$$\boxed{E\{U_n\, \boldsymbol{S}_{n-k}\} = \boldsymbol{0}} \tag{467}$$

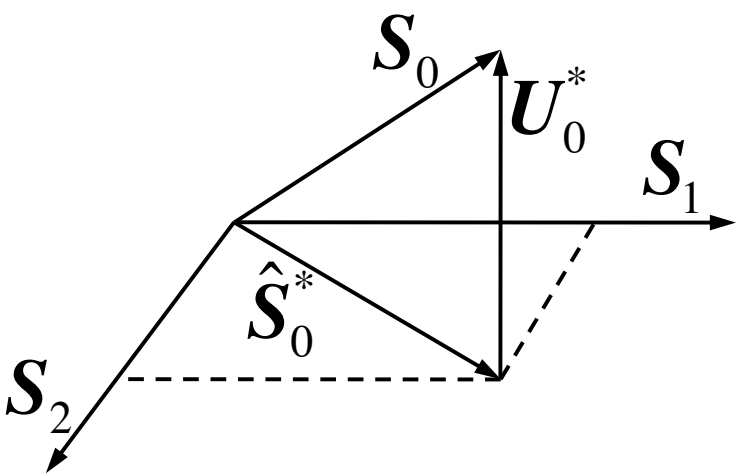
$\Longrightarrow$ For optimal affine prediction, the correlation between the observation vector and the prediction residual is zero

Geometric Interpretation of Orthogonality Principle

- For optimal affine prediction, the correlation between the prediction residual U_n and the observation vector $\boldsymbol{S}_{n-k}$ is zero

$$\boxed{E\{U_n\, \boldsymbol{S}_{n-k}\} = \boldsymbol{0}} \tag{468}$$

- For optimum affine filter design, prediction error should be orthogonal to input signal



- Approximate a vector $\boldsymbol{S}_0$ by a linear combination of $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$
- Best approximation $\hat{\boldsymbol{S}}_0^*$ is given by projection of $\boldsymbol{S}_0$ onto the plane spanned by $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$
- Error vector $\boldsymbol{U}_0^*$ has minimum length and is orthogonal to the projection

One-Step Prediction

- Random variable S_n is predicted using the N directly preceding random variables $\boldsymbol{S}_{n-1} = (S_{n-1}, \cdots, S_{n-N})^T$
- Using $\phi_k = E\{ \big(S_n - E\{S_n\}\big)\big(S_{n+k} - E\{S_{n+k}\}\big) \}$, the normal equations are given as

$$\begin{bmatrix} \phi_0 & \phi_1 & \cdots & \phi_{N-1} \\ \phi_1 & \phi_0 & \cdots & \phi_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix} \begin{bmatrix} h_1^N \\ h_2^N \\ \vdots \\ h_N^N \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix} \tag{469}$$

  where h_k^N represent elements of $\boldsymbol{h}_N^* = (h_1^N, \cdots, h_N^N)^T$

- Changing the equation to

$$\begin{bmatrix} \phi_1 & \phi_0 & \phi_1 & \cdots & \phi_{N-1} \\ \phi_2 & \phi_1 & \phi_0 & \cdots & \phi_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_N & \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix} \begin{bmatrix} 1 \\ -h_1^N \\ -h_2^N \\ \vdots \\ -h_N^N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{470}$$

One-Step Prediction

- Including the prediction error variance for optimal linear prediction using the N preceding samples

$$\begin{aligned} \sigma_N^2 &= \sigma_S^2 - \boldsymbol{c}_1^T \boldsymbol{C}_N^{-1} \boldsymbol{c}_1 = \sigma_S^2 - \boldsymbol{c}_1^T \boldsymbol{h}_N^* \\ &= \phi_0 - h_1^N \phi_1 - h_2^N \phi_2 - \cdots - h_N^N \phi_N \end{aligned} \tag{471}$$

  yields and additional row in the matrix

$$\underbrace{\begin{bmatrix} \phi_0 & \phi_1 & \phi_2 & \cdots & \phi_N \\ \phi_1 & \phi_0 & \phi_1 & \cdots & \phi_{N-1} \\ \phi_2 & \phi_1 & \phi_0 & \cdots & \phi_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_N & \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix}}_{\boldsymbol{C}_{N+1}} \underbrace{\begin{bmatrix} 1 \\ -h_1^N \\ -h_2^N \\ \vdots \\ -h_N^N \end{bmatrix}}_{\boldsymbol{a}_N} = \begin{bmatrix} \sigma_N^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{472}$$

- The resulting equation is called **augmented normal equation**

One-Step Prediction

- Multiplying both sides of the augmented normal equation with $\boldsymbol{a}_N^T$ yields

$$\sigma_N^2 = \boldsymbol{a}_N^T \, \boldsymbol{C}_{N+1} \, \boldsymbol{a}_N \tag{473}$$

- Combing the equations for 0 to N preceding samples into one matrix equation yields

$$\boldsymbol{C}_{N+1} \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -h_1^N & 1 & \ddots & 0 & 0 \\ -h_2^N & -h_1^{N-1} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & 1 & 0 \\ -h_N^N & -h_{N-1}^{N-1} & \cdots & -h_1^1 & 1 \end{bmatrix} = \begin{bmatrix} \sigma_N^2 & X & \cdots & X & X \\ 0 & \sigma_{N-1}^2 & \ddots & X & X \\ 0 & 0 & \ddots & X & X \\ \vdots & \vdots & \ddots & \sigma_1^2 & X \\ 0 & 0 & \cdots & 0 & \sigma_0^2 \end{bmatrix}$$

- Taking the determinant of both sides of the equation gives

$$|\boldsymbol{C}_{N+1}| = \sigma_N^2 \cdot \sigma_{N-1}^2 \cdot \ldots \cdot \sigma_0^2 \tag{474}$$

- Prediction error variance σ_N^2 for optimal linear prediction using the N preceding samples

$$\sigma_N^2 = \frac{|\boldsymbol{C}_{N+1}|}{|\boldsymbol{C}_N|} \tag{475}$$

One-Step Prediction for Autoregressive Processes

- Recall: AR(m) process with mean μ_S and $\boldsymbol{a}_m = (a_1, \cdots, a_m)^T$

$$S_n = Z_n + \mu_S(1 - \boldsymbol{a}_m^T \boldsymbol{e}_m) + \boldsymbol{a}_m^T \boldsymbol{S}_{n-1}^{(m)} \tag{476}$$

- Prediction using N preceding samples in $\boldsymbol{h}_N$ with $N \geq m$:
  Define $\boldsymbol{a}_N = (a_1, \cdots, a_m, 0, \cdots, 0)^T$
- Prediction error

$$U_n = S_n - \boldsymbol{h}_N^T \boldsymbol{S}_{n-1} = Z_n + \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + (\boldsymbol{a}_N - \boldsymbol{h}_N)^T \boldsymbol{S}_{n-1} \tag{477}$$

- Subtracting the mean $E\{U_n\} = \mu_S(1 - \boldsymbol{a}_N^T \boldsymbol{e}_N) + (\boldsymbol{a}_N - \boldsymbol{h}_N)^T E\{\boldsymbol{S}_{n-1}\}$

$$U_n - E\{U_n\} = Z_n + (\boldsymbol{a}_N - \boldsymbol{h}_N)^T \big(\boldsymbol{S}_{n-1} - E\{\boldsymbol{S}_{n-1}\}\big) \tag{478}$$

- Optimal prediction: covariances between U_n and $\boldsymbol{S}_{n-1}$ must be equal to 0

$$\begin{aligned} \boldsymbol{0} &= E\Big\{\big(U_n - E\{U_n\}\big)\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\Big\} \\ &= E\Big\{Z_n\big(\boldsymbol{S}_{n-k} - E\{\boldsymbol{S}_{n-k}\}\big)\Big\} + \boldsymbol{C}_N(\boldsymbol{a}_N - \boldsymbol{h}_N) \end{aligned} \tag{479}$$

  yields

$$\boxed{\boldsymbol{h}_N^* = \boldsymbol{a}_N} \tag{480}$$

One-Step Prediction in Gauss-Markov Processes

- Gauss-Markov process is a particular AR(1) process

$$S_n = Z_n + \mu_S(1-\rho) + \rho \cdot S_{n-1}, \tag{481}$$

  for which the iid process $\{Z_n\}$ has a Gaussian distribution
- Auto-covariance matrix and its inverse

$$\boldsymbol{C}_2 = \sigma_S^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \qquad \boldsymbol{C}_2^{-1} = \frac{1}{\sigma_S^2(1-\rho^2)} \begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix} \tag{482}$$

- Auto-covariance vector

$$\boldsymbol{c}_1 = \sigma_S^2 \begin{pmatrix} \rho \\ \rho^2 \end{pmatrix} \tag{483}$$

- Optimum predictor $\boldsymbol{h}_2^* = \boldsymbol{C}_2^{-1}\boldsymbol{c}_1$

$$\boldsymbol{h}_2^* = \frac{1}{1-\rho^2} \begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix} \begin{pmatrix} \rho \\ \rho^2 \end{pmatrix} = \frac{1}{1-\rho^2} \begin{pmatrix} \rho - \rho^3 \\ -\rho^2 + \rho^2 \end{pmatrix} = \begin{pmatrix} \rho \\ 0 \end{pmatrix}$$

- First element of $\boldsymbol{h}_N^*$ is equal to ρ, all other elements are equal to 0

One-Step Prediction in Gauss-Markov Processes

- Minimum prediction residual

$$\sigma_U^2 = \frac{|\boldsymbol{C}_2|}{|\boldsymbol{C}_1|} = \frac{\sigma_S^4 - \sigma_S^4\,\rho^2}{\sigma_S^2} = \sigma_S^2\,(1-\rho^2) \tag{484}$$

- Prediction residual for filter h_1

$$U_n = S_n - h_1 S_{n-1}$$

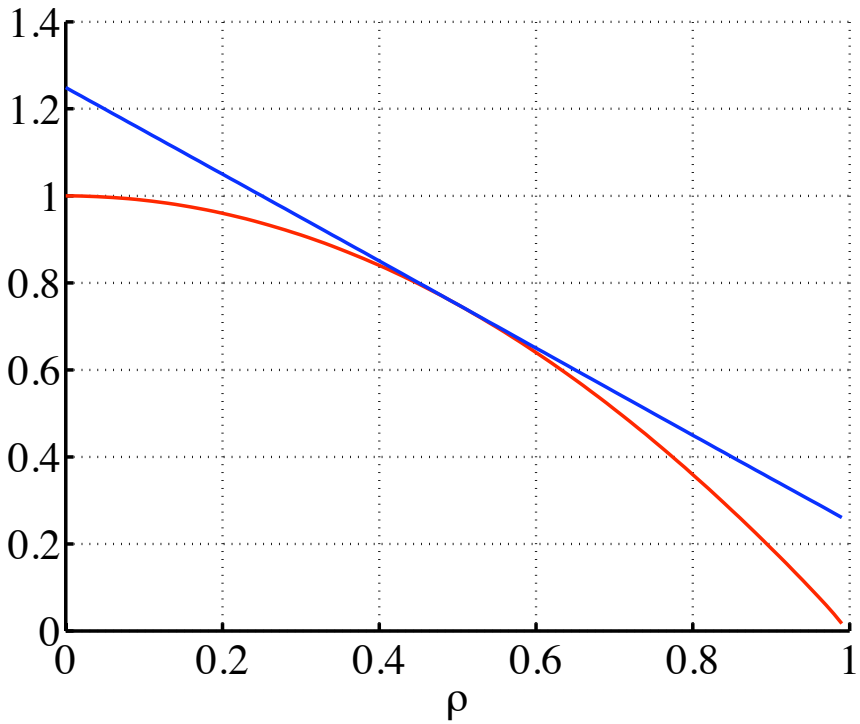
- Average squared error

$$\begin{aligned}\sigma_U^2(h_1) &= E\{U_n^2\} \\ &= \sigma_S^2(1 + h_1^2 - 2\rho h_1)\end{aligned}$$

- Note: Setting derivative to zero

$$\frac{\partial \sigma_U^2(h_1)}{\partial h_1} = \sigma_S^2(2h_1 - 2\rho) \stackrel{!}{=} 0$$

also yields the result $h_1 = \rho$



Prediction Gain

- Prediction gain using $\boldsymbol{\Phi}_N = \boldsymbol{C}_N/\sigma_S^2$ and $\boldsymbol{\phi}_1 = \boldsymbol{c}_1/\sigma_S^2$

$$G_P = \frac{E\{S_n^2\}}{E\{U_n^2\}} = \frac{\sigma_S^2}{\sigma_U^2} = \frac{\sigma_S^2}{\sigma_S^2 - \boldsymbol{c}_1^T\,\boldsymbol{C}_N^{-1}\,\boldsymbol{c}_1} = \frac{1}{1 - \boldsymbol{\phi}_1^T\,\boldsymbol{\Phi}_N^{-1}\,\boldsymbol{\phi}_1}, \tag{485}$$

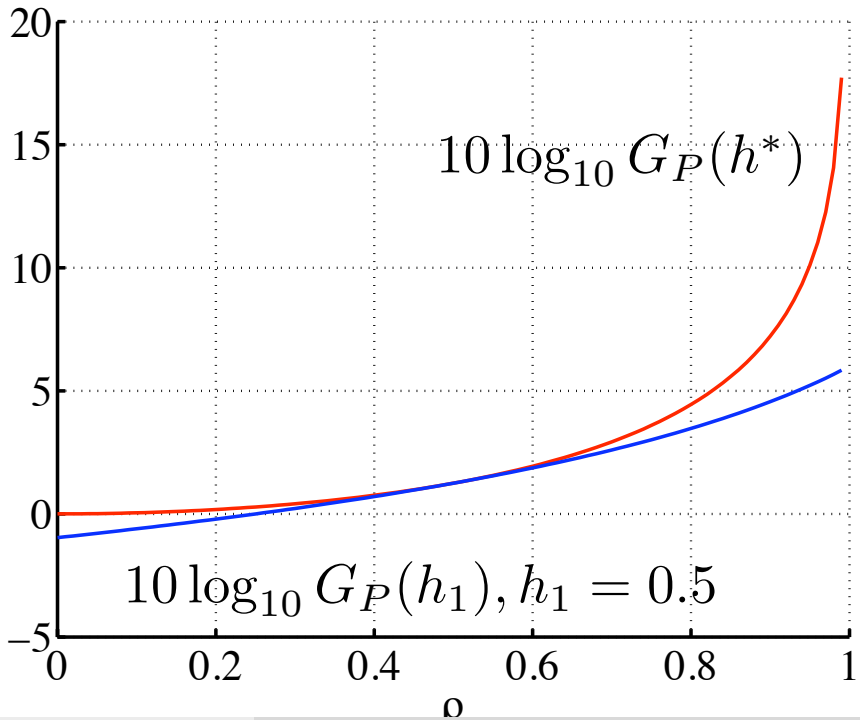
- Prediction gain for optimal prediction in first-order Gauss-Markov process

$$G_P(h^*) = \frac{1}{1-\rho^2} \tag{486}$$

- Prediction gain for filter h_1

$$\begin{aligned} G_P(h_1) &= \frac{\sigma_S^2}{\sigma_S^2(1 + h_1^2 - 2\rho h_1)} \\ &= \frac{1}{1 + h_1^2 - 2\rho h_1} \end{aligned}$$

- At high bit rates, $10\log_{10} G_P$: SNR improvement achieved by predictive coding



Optimum Linear One-Step Prediction for $K = 2$

- The normalized auto-correlation matrix and its inverse follow as

$$\boldsymbol{\Phi}_2 = \begin{pmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{pmatrix} \qquad \boldsymbol{\Phi}_2^{-1} = \frac{1}{1-\rho_1^2} \begin{pmatrix} 1 & -\rho_1 \\ -\rho_1 & 1 \end{pmatrix} \tag{487}$$

- With normalized correlation vector

$$\boldsymbol{\phi}_1 = \begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix} \tag{488}$$

we obtain the optimum predictor

$$\begin{aligned} \boldsymbol{h}_2^* &= \boldsymbol{\Phi}_2^{-1} \boldsymbol{\phi}_1 = \frac{1}{1-\rho_1^2} \begin{pmatrix} 1 & -\rho_1 \\ -\rho_1 & 1 \end{pmatrix} \begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix} = \frac{1}{1-\rho_1^2} \begin{pmatrix} \rho_1 - \rho_1\rho_2 \\ -\rho_1^2 + \rho_2 \end{pmatrix} \\ &= \frac{1}{1-\rho_1^2} \begin{pmatrix} \rho_1(1-\rho_2) \\ \rho_2 - \rho_1^2 \end{pmatrix} \end{aligned} \tag{489}$$

- For AR(1) sources, where we have $\rho_2 = \rho_1^2$, second coefficient does not improve prediction gain
- General: For AR(m) sources, only m coefficients are unequal to zero

Prediction in Images: Intra-Picture Prediction

- Random variables are samples within that image
- Derivations on linear prediction for zero-mean random variables (subtract μ_S or roughly 127 from 8-bit picture)
- Pictures are typically scanned line-by-line from upper left corner to lower right corner
- 1-d horizontal prediction:

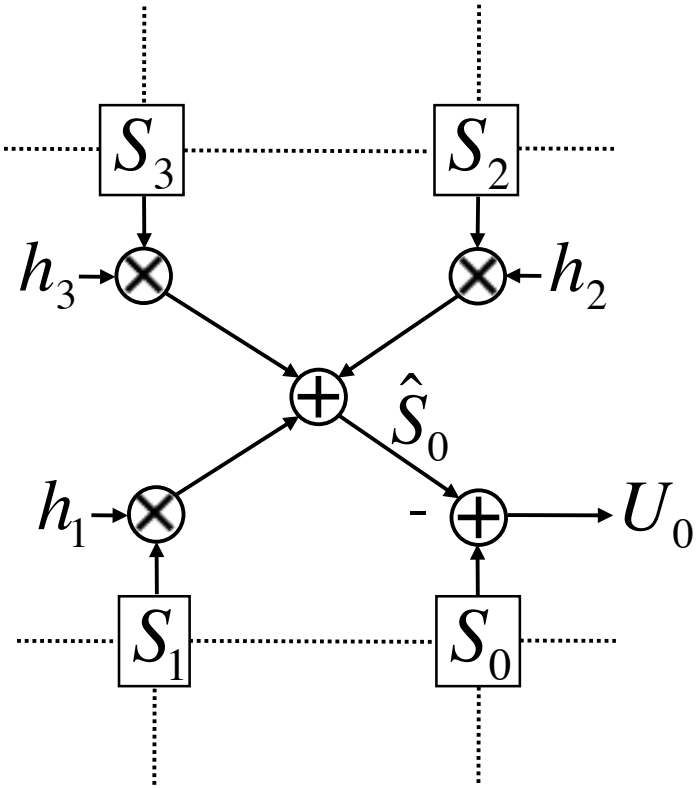
$$\hat{S}_0 = h_1 \cdot S_1$$

- 1-d vertical prediction:

$$\hat{S}_0 = h_2 \cdot S_2$$

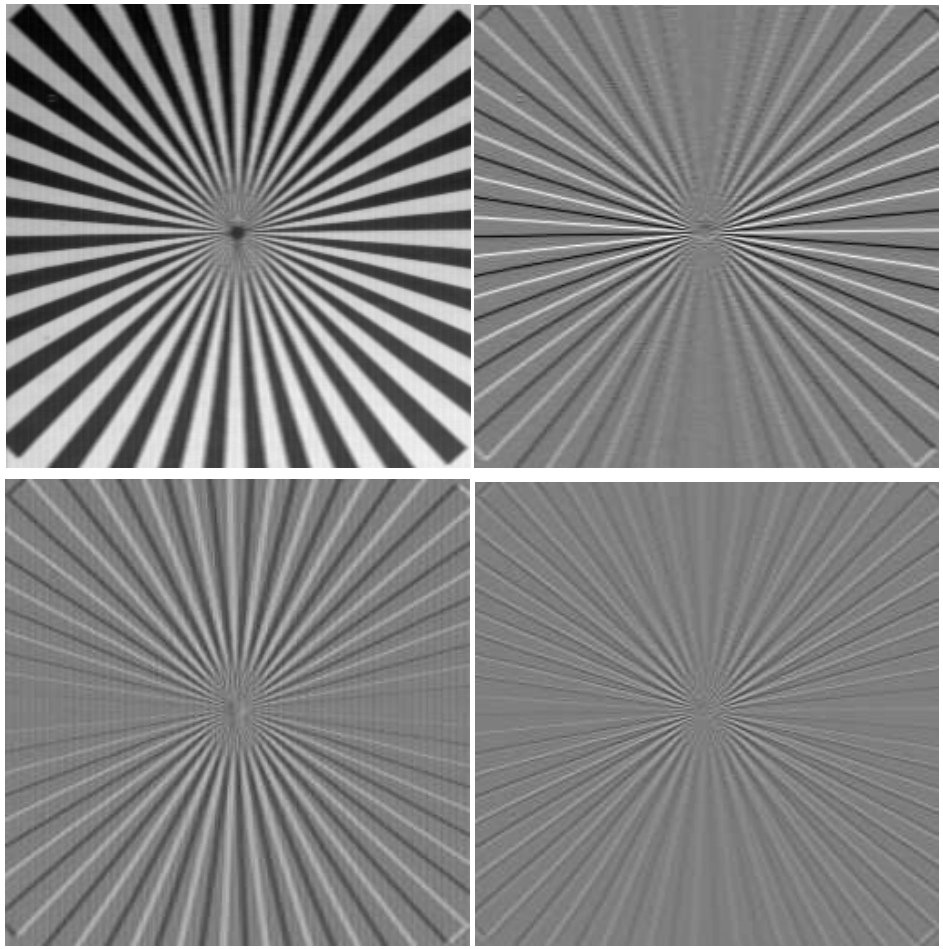
- 2-d prediction:

$$\hat{S}_0 = \sum_{i=1}^{3} h_i S_i$$



Prediction Example: Test Pattern

$\sigma_S^2 = 4925.81$
$(s - 127)$



vertical predictor
$h_1 = 0$
$h_2 = 0.932$
$h_3 = 0$
$\sigma_U^2(\boldsymbol{h}) = 646.67$
$G_P = 8.82$ dB

horizontal predictor
$h_1 = 0.953$
$h_2 = 0$
$h_3 = 0$
$\sigma_U^2(\boldsymbol{h}) = 456.17$
$G_P = 10.33$ dB

2-d predictor
$h_1 = 0.911$
$h_2 = 0.871$
$h_3 = -0.788$
$\sigma_U^2(\boldsymbol{h}) = 109.90$
$G_P = 16.51$ dB

Prediction Example: Picture "Lena"

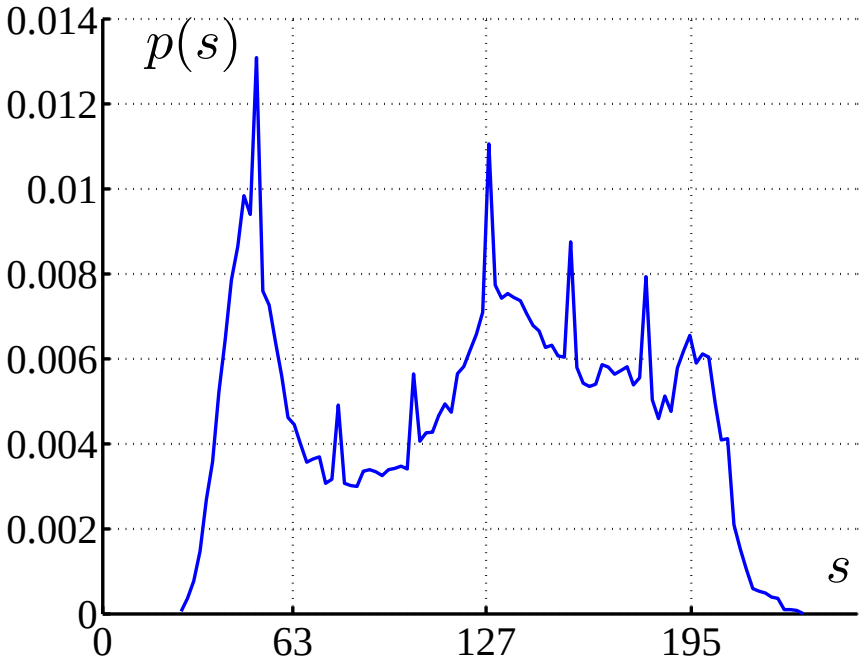
256×256 center cropped picture
$\sigma_S^2 = 2746.43$
$(s - 127)$

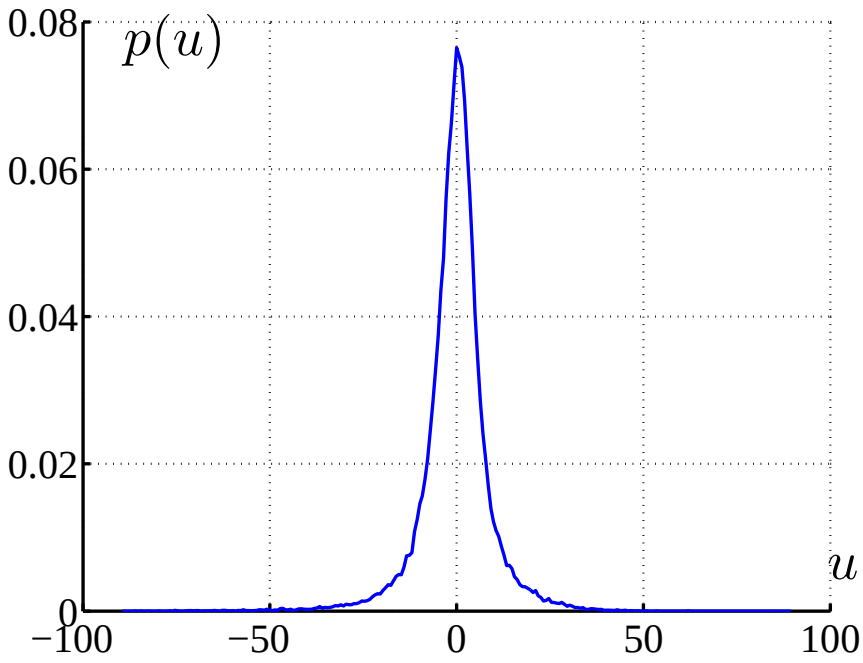
vertical predictor
$h_1 = 0$
$h_2 = 0.977$
$h_3 = 0$
$\sigma_U^2(\boldsymbol{h}) = 123.61$
$G_P = 13.47$ dB

horizontal predictor
$h_1 = 0.962$
$h_2 = 0$
$h_3 = 0$
$\sigma_U^2(\boldsymbol{h}) = 212.36$
$G_P = 11.12$ dB

2-d predictor
$h_1 = 0.623$
$h_2 = 0.835$
$h_3 = -0.48$
$\sigma_U^2(\boldsymbol{h}) = 80.35$
$G_P = 15.34$ dB

Prediction Example: PMFs for Picture Lena





- Pmfs $p(s)$ and $p(u)$ change significantly due to prediction operation
- Entropy changes significantly
  (rounding prediction signal towards integer: $E\{\lfloor U_n + 0.5 \rfloor^2\} = 80.47$)

$$H(S) = 7.44\,\text{bit/sample} \qquad H(U) = 4.97\,\text{bit/sample} \tag{490}$$

- Linear prediction can be used for lossless coding: JPEG-LS

Asymptotic Prediction Gain

Consider upper bound for prediction gain: $N \to \infty$

- One-step prediction of a random variable S_n given the countably infinite set of preceding random variables $\{S_{n-1}, S_{n-2}, \cdots\}$ and $\{h_0, h_1, \cdots\}$

$$U_n = S_n - h_0 - \sum_{i=1}^{\infty} h_i \, S_{n-i}, \tag{491}$$

- Orthogonality criterion: U_n is uncorrelated with all S_{n-k} for $k > 0$
- Furthermore, U_{n-k} for $k > 0$ is fully determined by a linear combination of past input values S_{n-k-i} for $i \geq 0$
- Hence, U_n is uncorrelated with U_{n-k} for $k > 0$

$$\phi_{UU}(k) = \sigma^2_{U,\infty} \cdot \delta(k) \qquad \Longleftrightarrow \qquad \Phi_{UU}(\omega) = \sigma^2_{U,\infty} \tag{492}$$

where $\sigma^2_{U,\infty}$ is the asymptotic one-step prediction error variance for $N \to \infty$

Asymptotic Prediction Error Variance

- For one-step prediction we showed

$$|\boldsymbol{C}_N| = \sigma_{N-1}^2 \cdot \sigma_{N-2}^2 \cdot \sigma_{N-3}^2 \cdots \sigma_0^2 \tag{493}$$

  which yields

$$\frac{1}{N} \ln |\boldsymbol{C}_N| = \ln |\boldsymbol{C}_N|^{\frac{1}{N}} = \frac{1}{N} \sum_{i=0}^{N-1} \ln \sigma_i^2 \tag{494}$$

- If a sequence of numbers $\alpha_0, \alpha_1, \alpha_2, \cdots$ approaches a limit α_∞, the average value approaches the same limit,

$$\lim_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N-1} \alpha_i = \alpha_\infty \tag{495}$$

- Hence, we can write

$$\lim_{N \to \infty} \ln |\boldsymbol{C}_N|^{\frac{1}{N}} = \lim_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N-1} \ln \sigma_i^2 = \ln \sigma_\infty^2 \tag{496}$$

  yielding

$$\sigma_\infty^2 = \exp\left(\lim_{N \to \infty} \ln |\boldsymbol{C}_N|^{\frac{1}{N}} \right) = \lim_{N \to \infty} |\boldsymbol{C}_N|^{\frac{1}{N}} \tag{497}$$

Asymptotic Prediction Error Variance

- Asymptotic One-Step Prediction Error Variance

$$\sigma_{U,\infty}^2 = \lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}} \tag{498}$$

- Determinant of $N \times N$ matrix: Product of its eigenvalues $\xi_i^{(N)}$

$$\lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}} = \lim_{N\to\infty} \left(\prod_{i=0}^{N-1} \xi_i^{(N)} \right)^{\frac{1}{N}} = 2^{\left(\lim_{N\to\infty} \sum_{i=0}^{N-1} \frac{1}{N} \log_2 \xi_i^{(N)} \right)} \tag{499}$$

- Apply Grenander and Szegö's theorem

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} G\left(\xi_i^{(N)}\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G\left(\Phi(\omega)\right) \, \mathrm{d}\omega \tag{500}$$

- Expression using power spectral density

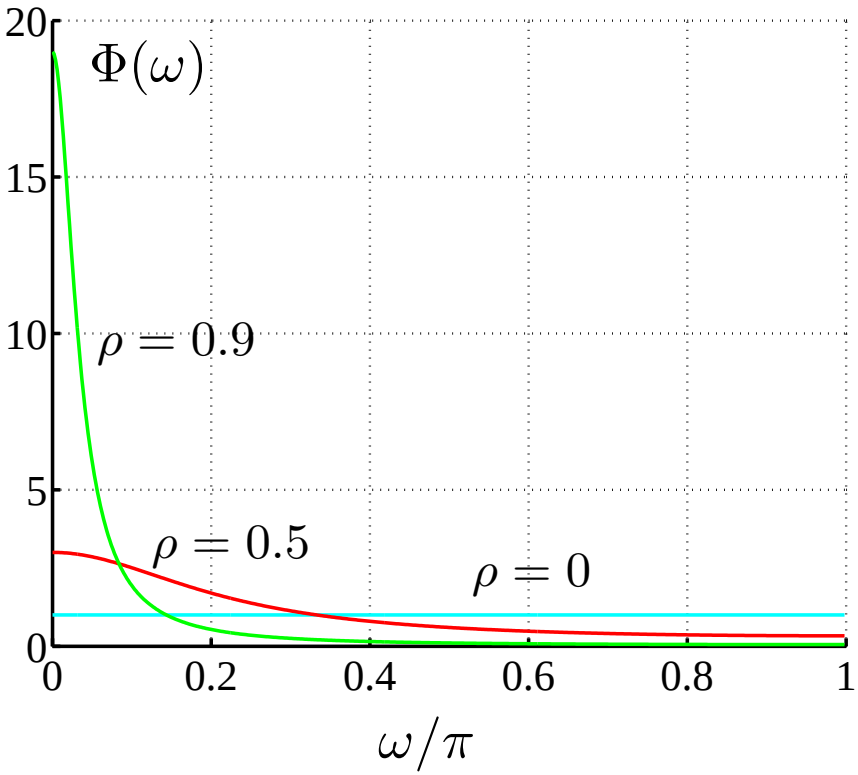
$$\sigma_{U,\infty}^2 = \lim_{N\to\infty} |\boldsymbol{C}_N|^{\frac{1}{N}} = 2^{\frac{1}{2\pi} \int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) \, \mathrm{d}\omega} \tag{501}$$

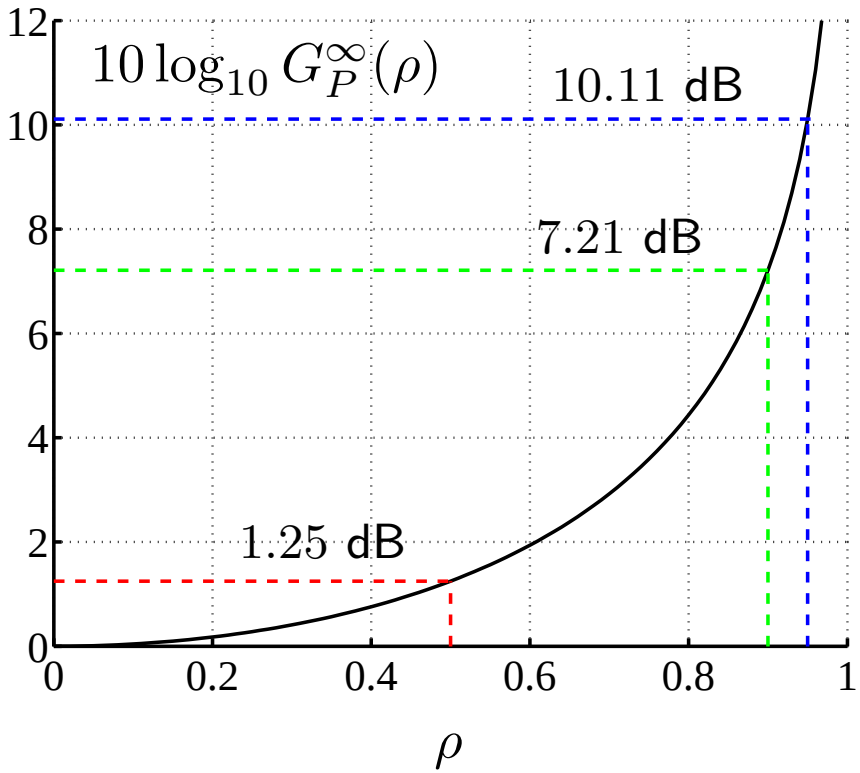
Asymptotic Prediction Gain

- Prediction gain G_P^{∞}

$$G_P^{\infty} = \frac{\sigma_S^2}{\sigma_{U,\infty}^2} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi(\omega)\, d\omega}{2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi(\omega)\, d\omega}} \quad \begin{matrix} \leftarrow \text{Arithmetic mean} \\ \leftarrow \text{Geometric mean} \end{matrix} \tag{502}$$

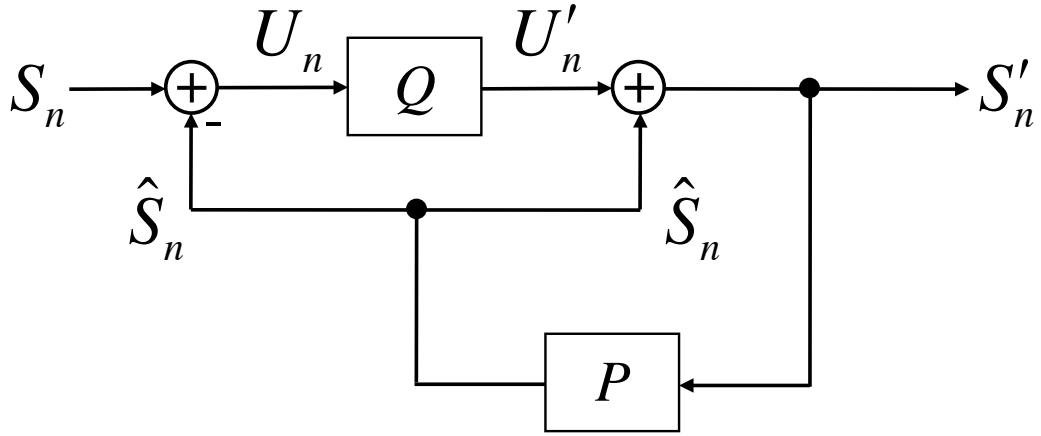
- Result for first-order Gauss-Markov source (can also be computed differently)



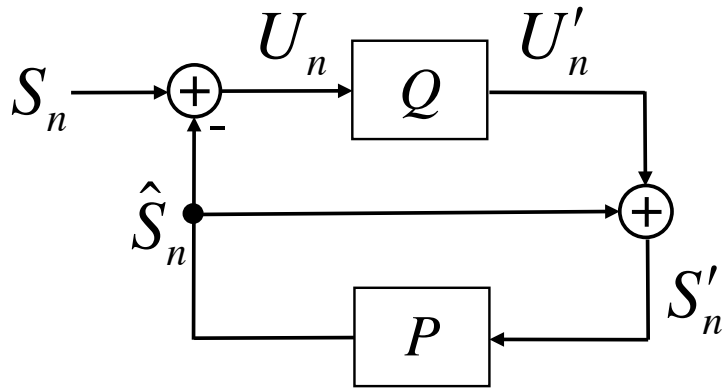


Differential Pulse Code Modulation (DPCM)

- Combining prediction with quantization requires simultaneous reconstruction of predictor at encoder and decoder
  $\Longrightarrow$ Use quantized samples for prediction

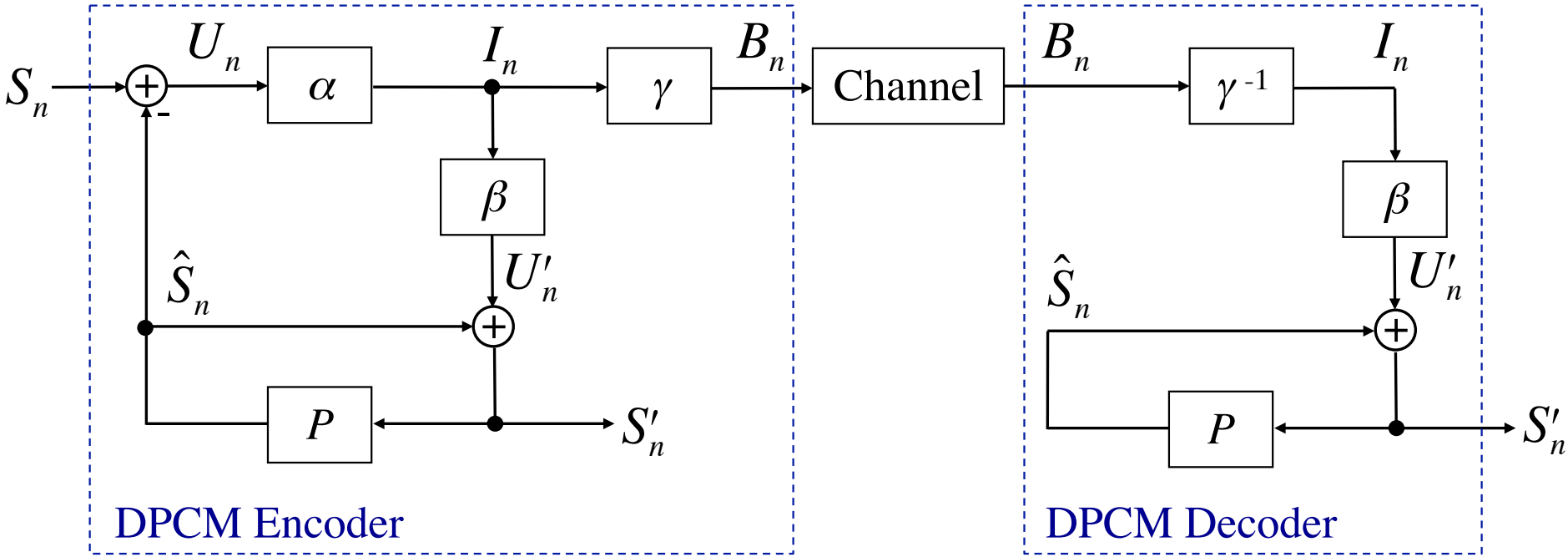


- Re-drawing yields block-diagram with typical DPCM structure



DPCM Codec

- Redrawing with encoder mapping α, lossless coding γ, and decoder mapping β yields DPCM encoder



- DPCM encoder contains DPCM decoder except for γ^{-1}

DPCM and Quantization

- Prediction $\hat{S}_n$ for a sample S_n is generated by linear filtering of reconstructed samples S'_n from the past

$$\hat{S}_n = \sum_{i=1}^{K} h_i \; S'_{n-i} = \sum_{i=1}^{K} h_i \; (S_{n-i} + Q_{n-i}) = \boldsymbol{h}^T \cdot (\boldsymbol{S}_{n-1} + \boldsymbol{Q}_{n-1}) \tag{503}$$

  with $Q_n = S'_n - S_n$ being the quantization error signal

- Prediction error variance (for zero-mean input) is given by

$$\begin{aligned} \sigma_U^2 &= E\{U_n^2\} = E\Big\{(S_n - \hat{S}_n)^2\Big\} = E\Big\{(S_n - \boldsymbol{h}^T \boldsymbol{S}_{n-1} - \boldsymbol{h}^T \boldsymbol{Q}_{n-1})^2\Big\} \\ &= E\{S_n^2\} + \boldsymbol{h}^T E\Big\{\boldsymbol{S}_{n-1}\boldsymbol{S}_{n-1}^T\Big\}\,\boldsymbol{h} + \boldsymbol{h}^T E\Big\{\boldsymbol{Q}_{n-1}\boldsymbol{Q}_{n-1}^T\Big\}\,\boldsymbol{h} \\ &\quad -2\boldsymbol{h}^T E\{S_n \boldsymbol{S}_{n-1}\} - 2\boldsymbol{h}^T E\{S_n \boldsymbol{Q}_{n-1}\} + 2\boldsymbol{h}^T E\Big\{\boldsymbol{S}_{n-1}\boldsymbol{Q}_{n-1}^T\Big\}\,\boldsymbol{h} \end{aligned} \tag{504}$$

- Defining $\boldsymbol{\Phi} = E\{\boldsymbol{S}_{n-1}\boldsymbol{S}_{n-1}^T\}/\sigma_S^2$ and $\boldsymbol{\phi} = E\{S_n \boldsymbol{S}_{n-1}\}/\sigma_S^2$ we get

$$\begin{aligned} \sigma_U^2 &= \sigma_S^2 \left(1 + \boldsymbol{h}^T \boldsymbol{\Phi}\, \boldsymbol{h} - 2\boldsymbol{h}^T \boldsymbol{\phi}\right) \\ &\quad + \boldsymbol{h}^T E\Big\{\boldsymbol{Q}_{n-1}\boldsymbol{Q}_{n-1}^T\Big\}\,\boldsymbol{h} - 2\boldsymbol{h}^T E\{S_n \boldsymbol{Q}_{n-1}\} + 2\boldsymbol{h}^T E\Big\{\boldsymbol{S}_{n-1}\boldsymbol{Q}_{n-1}^T\Big\}\,\boldsymbol{h} \end{aligned} \tag{505}$$

DPCM for a Gauss-Markov Source

- Calculate $R(D)$ for zero-mean Gauss-Markov process

$$S_n = Z_n + \rho \cdot S_{n-1} \tag{506}$$

- Consider a one-tap linear prediction filter $\boldsymbol{h} = [h]$
- Normalized auto-correlation matrix $\boldsymbol{\Phi} = [1]$ and cross-correlation $\boldsymbol{\phi} = [\rho]$
- Prediction error variance

$$\begin{aligned} \sigma_U^2 \;=\; & \sigma_S^2 \left(1 + h^2 - 2\,h\,\rho\right) + h^2 E\{Q_{n-1}^2\} \\ & -2hE\{S_n Q_{n-1}\} + 2h^2 E\{S_{n-1} Q_{n-1}\} \end{aligned} \tag{507}$$

- Using $S_n = Z_n + \rho \cdot S_{n-1}$, the second row in above equation becomes

$$\begin{aligned} & -2hE\{S_n Q_{n-1}\} + 2h^2 E\{S_{n-1} Q_{n-1}\} \\ & = \; -2hE\{Z_n Q_{n-1}\} - 2h\rho E\{S_{n-1} Q_{n-1}\} + 2h^2 E\{S_{n-1} Q_{n-1}\} \\ & = \; -2hE\{Z_n Q_{n-1}\} + 2h(h-\rho) E\{S_{n-1} Q_{n-1}\} \end{aligned} \tag{508}$$

- With setting $h = \rho$, we have

$$E\{Z_n Q_{n-1}\} = 0 \qquad 2h(h-\rho) E\{S_{n-1} Q_{n-1}\} = 0 \tag{509}$$

DPCM for a Gauss-Markov Source

- For $h = \rho$, expression for prediction error variance simplifies to

$$\sigma_U^2 = \sigma_S^2 \left(1 - \rho^2\right) + \rho^2 E\{Q_{n-1}^2\} \tag{510}$$

- Assume: Prediction error for Gaussian source has also Gaussian distribution
- Model expression for quantization error $D = E\{Q_{n-1}^2\}$ by an operational distortion rate function

$$D(R) = \sigma_U^2 \cdot g(R) \tag{511}$$

- Expression for prediction error variance becomes dependent on rate

$$\boxed{\sigma_U^2 = \sigma_S^2 \cdot \frac{1 - \rho^2}{1 - g(R)\,\rho^2}} \tag{512}$$

- Operational distortion-rate function for DPCM of Gauss-Markov

$$\boxed{D(R) = \sigma_U^2 \cdot g(R) = \sigma_S^2 \cdot \frac{1 - \rho^2}{1 - g(R)\,\rho^2} \cdot g(R)} \tag{513}$$

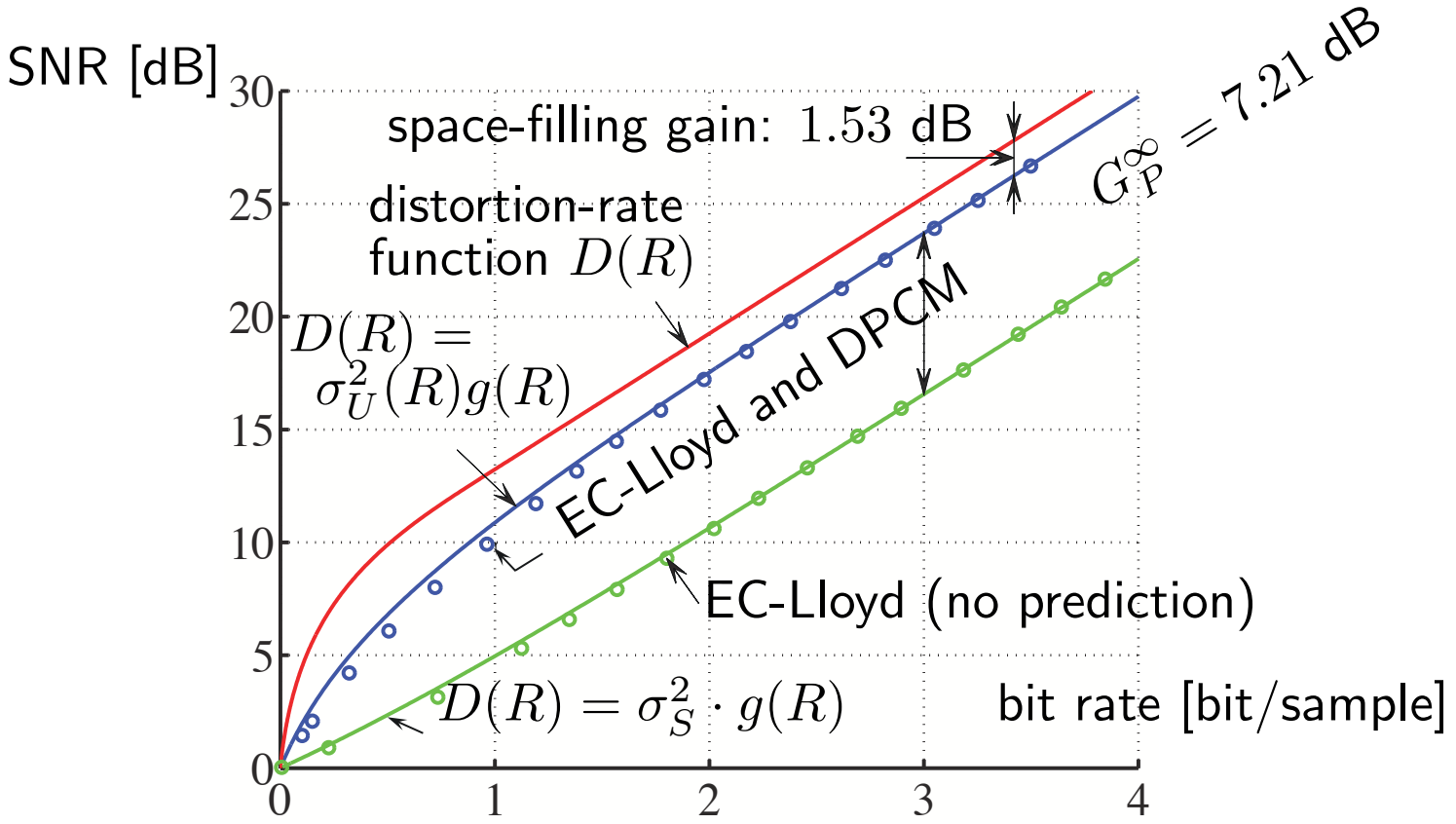
Computation of DPCM Distortion-Rate Function

- Operational distortion-rate function for DPCM and ECSQ for a Gauss-Markov source

$$\boxed{D(R) = \sigma_U^2 \cdot g(R) = \sigma_S^2 \cdot \frac{1-\rho^2}{1-g(R)\,\rho^2} \cdot g(R)} \tag{514}$$

- Algorithm for designing ECSQ inside DPCM loop
  1. Initialization with a small value of λ, set $s'_n = s_n$, $\forall n$ and $h = \rho$
  2. Create signal u_n using s'_n and DCPM
  3. Design ECSQ (α, β, γ) using signal u_n and the current value of λ by minimizing $D + \lambda R$
  4. Conduct DPCM encoding/decoding using ECSQ (α, β, γ)
  5. Measure $\sigma_U^2(R)$ as well as rate R and distortion D
  6. Increase λ and start again with step 2
- Algorithm shows problems at low bit rates: Instabilities

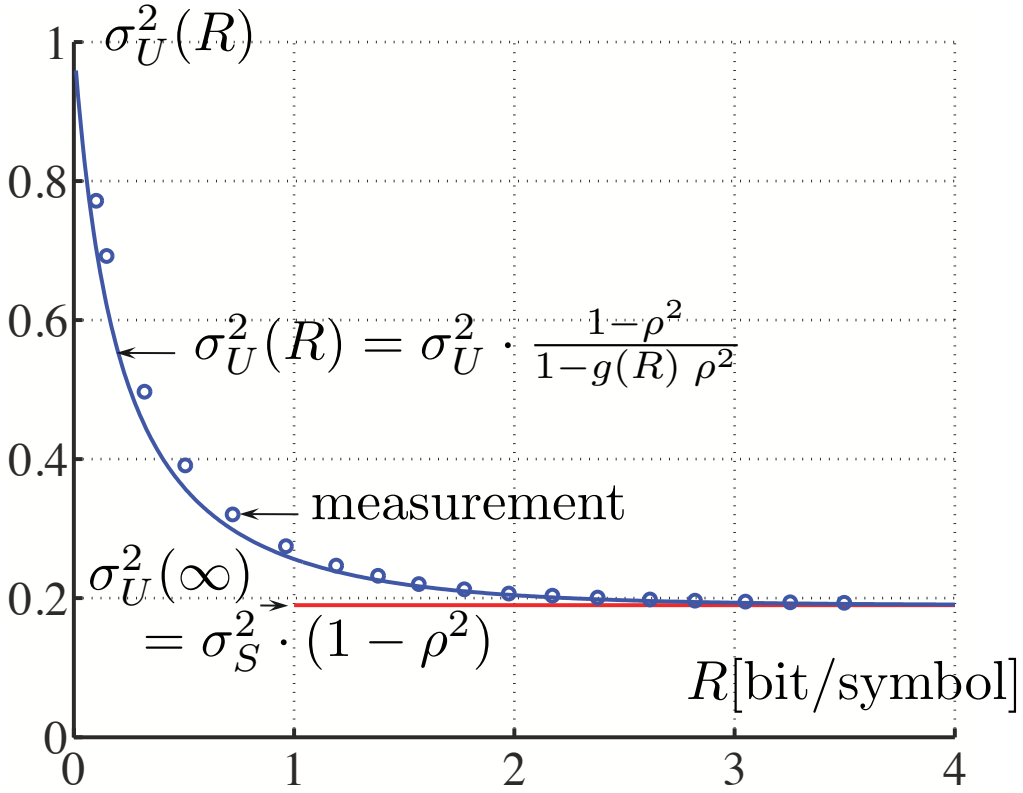
Comparison of Theoretical and Experimental Results



- For high rates and Gauss-Markov sources, shape and memory gain achievable
- Space-filling gain can only be achieved using vector quantization
- Theoretical model provides a useful description

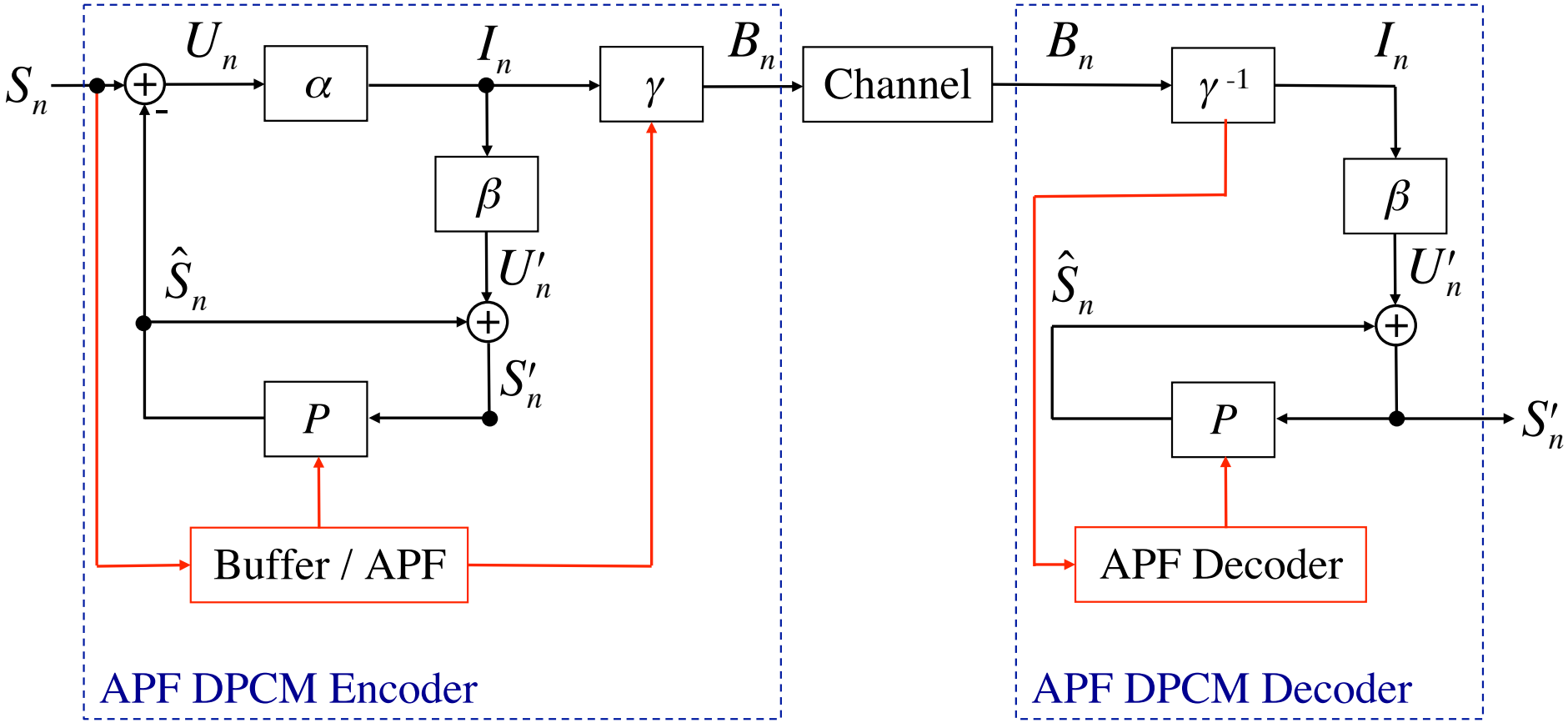
Comparison of Theoretical and Experimental Results

- Prediction error variance σ_U^2 depends on bit rate
- Theoretical model provides a useful description



Adaptive Differential Pulse Code Modulation (ADPCM)

- For quasi-stationary sources like speech, fixed predictor is not well suited
- ADPCM: Adapt the predictor based on the recent signal characteristics
- Forward adaptation: Send new predictor values (requires additional bit rate)



Forward-Adaptive Prediction: Motion Compensation

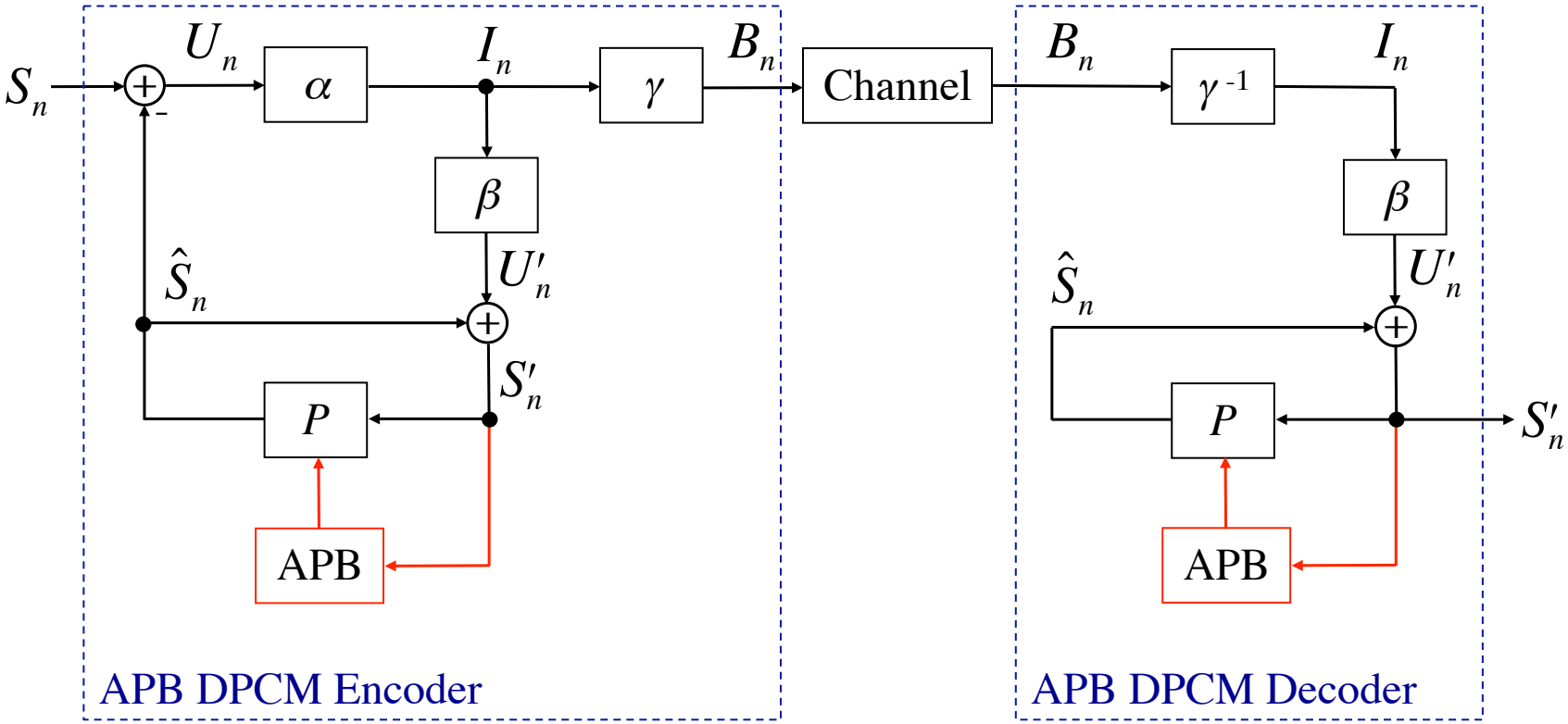
- Since predictor values are sent, extend prediction to vectors/blocks
- Use statistical dependencies between two pictures
- Prediction signal obtained by searching a region in a previously decoded picture that best matches the block to be coded
- Let $s[x, y]$ represent intensity at location (x, y)
- Let $s'[x, y]$ represent intensity in a previously decoded picture at (x, y)

$$J = \min_{(dx,dy)} \sum_{x,y} (s[x, y] - s'[x - dx, y - dy])^2 + \lambda \cdot R(dx, dy) \qquad (515)$$

- Predicted signal is specified through motion vector (dx, dy)
- $R(dx, dy)$ represents the number of bits required for coding the motion vector
- Prediction error $u[x, y]$ is quantized (often using transform coding)
- Bit rate is sum of motion vector and prediction residual bit rate

Backward Adaptive DPCM

- Backward adaptation: Use predictor computed from recently decoded signal
  - No additional bit rate
  - Error resilience issues
  - Accuracy of predictor
  - Decoder complexity

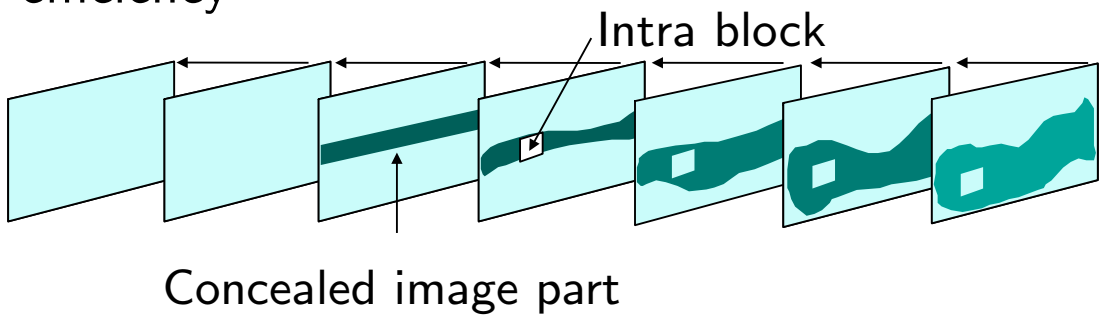


Transmission Errors in DPCM

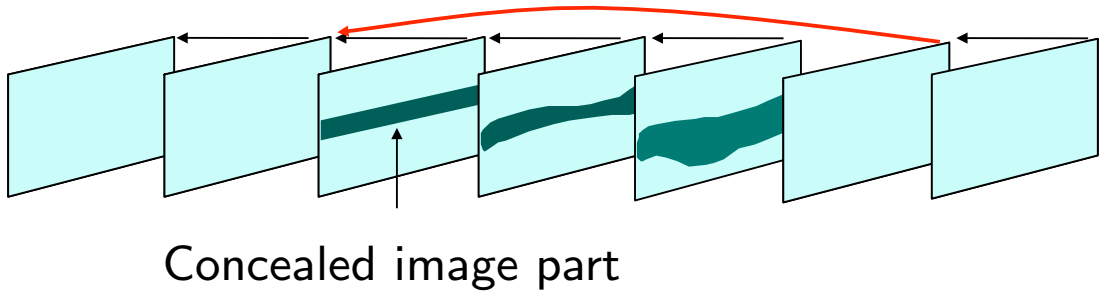
- When transmission error occurs, DPCM causes error propagation

Example: Motion compensation in video coding

- Try to conceal image parts that are in error
- Code lost image parts without referencing concealed image parts helps but reduces coding efficiency



- Use "clean" reference picture for motion compensation



Chapter Summary

Prediction

- Estimate random variable from already observed random variables
- Optimal predictor: Conditional mean

Linear and affine prediction

- Simple and efficient structure
- Optimal predictor given by Wiener-Hopf equation
- AR(m) processes: Optimal predictor has m coefficients
- Optimal prediction error is orthogonal to input signal
- Non-matched predictor can increase signal variance

Predictive quantization: DPCM

- Combination of affine prediction and ECSQ is simple and efficient
- Can exploit linear dependencies between samples
- Forward and backward adaptation
- Transmission errors cause error propagation

Exercise 20

Given is a stationary random process $\mathbf{S} = \{S_n\}$.

We consider affine prediction of a random variable S_n given the N preceding random variables $\mathbf{S}_{n-1} = [S_{n-1}\ S_{n-2}\ \cdots\ S_{n-N}]^T$.

Derive all formulas (as given below) as function of the mean μ_s, the variance σ_S^2, the N-th order autocovariance matrix $\mathbf{C}_N$ and the autocovariance vector $\mathbf{c}_1 = E\{(S_n - \mu_S)(\mathbf{S}_{n-1} - \mu_S \mathbf{e}_N)\}$, where $\mathbf{e}_N$ is a N-dimensional vector with all entries equal to 1.

(a) Derive the affine predictor that minimizes the mean squared prediction error.

(b) Derive expressions for the mean and the variance of the resulting prediction error as well as for the mean squared error.

(c) Derive the affine predictor and the resulting mean, variance and mean squared error for the special case $N = 1$, menaing that a random variable S_n is predicted using the random variable S_{n-1}. The correlation coefficient between successive random variables is ρ.

Exercise 21

In image and video coding, a sample S_n is often predicted by directly using a previous sample S_{n-1}, i.e., by $\hat{S}_n = S_{n-1}$.

Consider a zero-mean stationary process $\mathbf{S} = \{S_n\}$ with the first-order correlation factor ρ.

(a) For what correlation factors ρ do we observe a prediction gain (the mean squared prediction error is smaller than the second moment of the input)?

(b) For what correlation factors is the loss versus optimal linear prediction smaller than 0.1 dB?

Exercise 22 - Part I

Consider prediction in images. Assume that an image can be considered as a realization of a stationary 2-d process with mean μ_S and variance σ_S^2.
We want to linearily predict a current sample based on up to three (already coded) neigbouring samples: the sample left of the current sample, the sample above the current sample, and the sample to the top-left of the current sample.
The correlation factor between two horizontally adjacent samples is ρ_H, the correlation factor between two vertically adjacent samples is ρ_V, and the correlation factor between two diagonally adjacent samples is ρ_D (same in both directions).
The goal is to design linear predictors that minimize the mean squared prediction error. The mean μ_S is subtracted before doing the prediction.

(a) Assume that $\rho_H > \rho_V$.
Compare optimal linear prediction using only the horizontally adjacent sample and optimal linear prediction using both the horizontally and the vertically adjacent sample.
Under which cicumstances is the prediction using both samples better than the prediction using only the horizontally adjacent sample?

Exercise 22 - Part II

(b) Consider the special case $\rho_H = \rho_V = \rho$ and $\rho_D = \rho^2$.
Derive the prediction gain $g = \sigma_S^2/\varepsilon^2$ for the optimal vertical predictors using

- the sample to the left
- the sample to the left and the sample above
- the sample to the left, the sample above, and the sample to the top-left

What are the prediction gains in dB for $\rho = 0.95$?

Exercise 23 – Part I

Given is a stationary AR(2) process

$$S_n = Z_n + \alpha_1 \cdot S_{n-1} + \alpha_2 \cdot S_{n-2}$$

where $\{Z_n\}$ represents zero-mean white noise.
The AR parameters are $\alpha_1 = 0.7$ and $\alpha_2 = 0.2$.

(a) Determine the correlation factors ρ_1 and ρ_2, where ρ_1 is the correlation factor between adjacent samples S_n and S_{n-1}, and ρ_2 is the correlation factor between samples S_n and S_{n-2} that are two sampling intervals apart.

(b) Derive the optimal linear predictor (minimizing the MSE) using the 2 previous samples.
Determine the prediction gain in dB.

(c) Derive the optimal linear predictor (minimizing the MSE) using only the directly preceeding sample.
What is the prediction gain in dB?.
What is the loss relative to an optimal prediction using the last two samples?

Exercise 23 – Part II

(d) Can the linear predictor using the directly preceeding sample, given by

$$U_n = S_n - \rho_1 \cdot S_{n-1}.$$

be improved by adding a second prediction stage

$$V_n = U_n - h \cdot U_{n-1}?$$

What is the optimal linear predictor for the second prediction stage?
What is the prediction gain achieved by the second prediction stage?
How big is the loss versus optimal linear prediction using the last two samples?

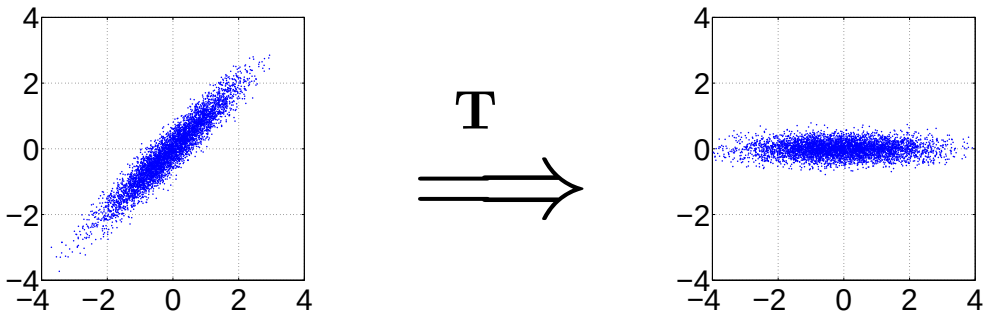
Exercise 24

Consider a zero-mean Gauss-Markov process with the correlation factor $\rho = 0.9$. The Gauss-Markov source is coded using DPCM at high rates. The quantizer is an entropy-contrained Lloyd quantizer with optimal entropy coding.

(a) Neglect the quantization and derive the optimal linear predictor (minimizing the MSE) using the previous sample.
Determine the prediction gain.

(b) Use the predictor derived in (a) inside the DPCM loop.
Assume that the prediction error has a Gaussian distribution.
What is the approximate coding gain compared to ECSQ without prediction at the rates $R_1 = 1$ bit per sample, $R_2 = 2$ bit per sample, $R_3 = 3$ bit per sample, $R_4 = 4$ bit per sample, and $R_5 = 8$ bit per sample?

Transform Coding



Outline

Part I: Source Coding Fundamentals

- Probability, Random Variables and Random Processes
- Lossless Source Coding
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- **Transform Coding**
  - Structure of Transform Coding Systems
  - Orthogonal Block Transforms
  - Bit Allocation for Transform Coefficients
  - Karhunen Loéve Transform (KLT)
  - Signal Independent Transforms (Hadamard, FFT, DCT)

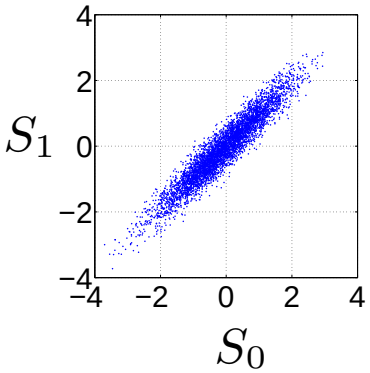
Part II: Application in Image and Video Coding

- Still Image Coding / Intra-Picture Coding
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

Transform Coding – Introduction

- Another concept for partially exploiting the memory gain
- Used in virtually all lossy image and video coding applications
- Samples of source s are grouped into vectors $\boldsymbol{s}$ of adjacent samples
- Transform coding consists of the following steps
  1. Linear analysis transform $\boldsymbol{A}$, converting source vectors $\boldsymbol{s}$ into transform coefficient vectors $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{s}$
  2. Scalar quantization of the transform coefficients $\boldsymbol{u} \mapsto \boldsymbol{u'}$
  3. Linear synthesis transform $\boldsymbol{B}$, converting quantized transform coefficient vectors $\boldsymbol{u'}$ into decoded source vectors $\boldsymbol{s'} = \boldsymbol{B}\boldsymbol{u'}$

Adjacent Samples

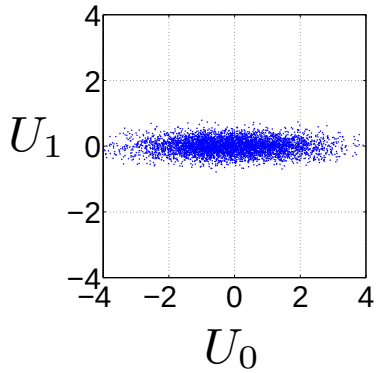


2D Transform:
Rotation by $\varphi = 45°$

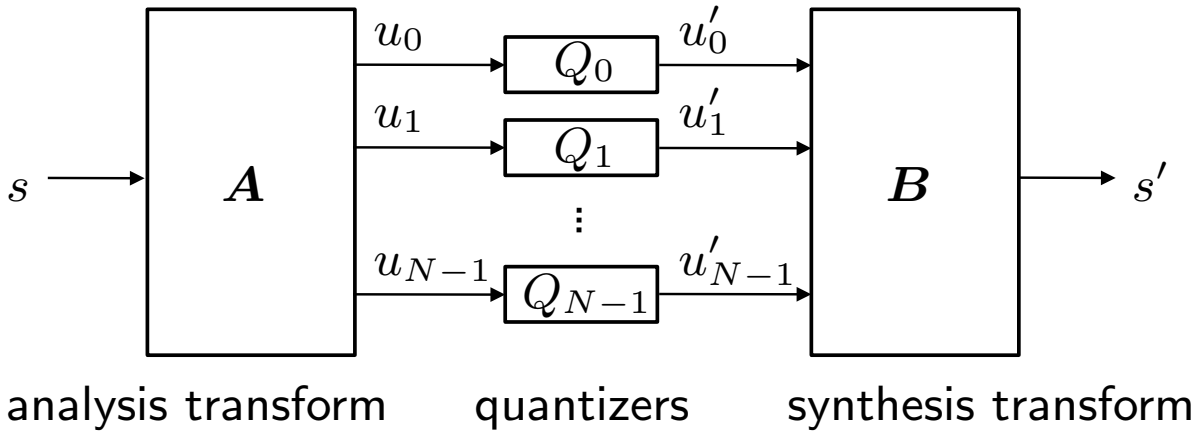
$$\boldsymbol{A} = \begin{bmatrix} \sin\varphi & \cos\varphi \\ \cos\varphi & -\sin\varphi \end{bmatrix}$$

$$\Longrightarrow$$

Transform Coefficients



Structure of Transform Coding Systems



- Synthesis transform is typically inverse of analysis transform
- Separate scalar quantizer Q_n for each transform coefficient u_n
- Vector quantization of all bands or some of them is also possible, but
  - Transforms are designed to have a decorrelating effect (memory gain)
  - Shape gain can be obtained by ECSQ
  - Space-filling gain is left as a possible additional gain for VQ

Combination of decorrelating transformation, scalar quantization and entropy coding is highly efficient – in terms of rate-distortion performance and complexity

Motivation of Transform Coding

Exploitation of statistical dependencies

- Transform are typically designed in a way that, for typical input signals, the signal energy is concentrated in a few transform coefficients
- Coding of a few coefficients and many zero-valued coefficients can be very efficient (e.g., using arithmetic coding, run-length coding)
- Scalar quantization is more effective in transform domain

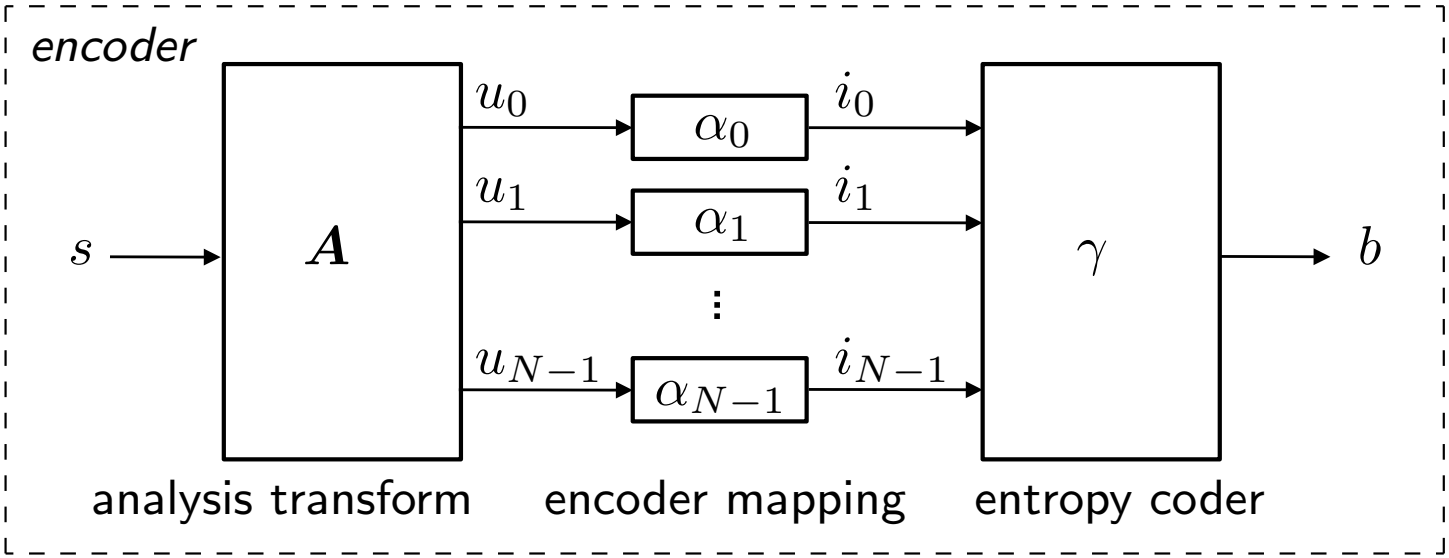
Efficient trade-off between coding efficiency & complexity

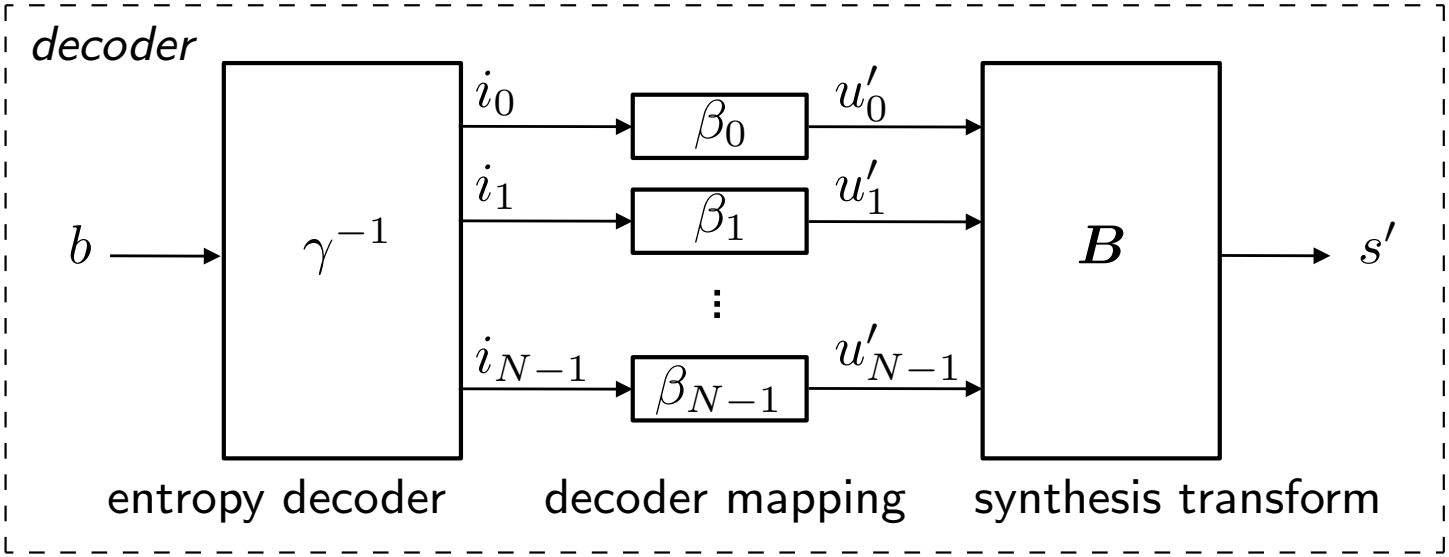
- Vector Quantization: Searching through codebook for best matching vector
- Combination of transform and scalar quantization typically results in a substantial reduction in computational complexity

Suitable for quantization using perceptual criteria

- In image & video coding, quantization in transform domain typically leads to an improvement in subjective quality
- In speech & audio coding, frequency bands might be used to simulate processing of human ear
- Reduce perceptually irrelevant content

Transform Encoder and Decoder





Linear Block Transforms

Linear Block Transform

- Each component of the N-dimensional output vector represents a linear combination of the N components of the N-dimensional input vector
- Can be written as matrix multiplication

- Analysis transform

$$\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s} \tag{516}$$

- Synthesis transform

$$\boldsymbol{s'} = \boldsymbol{B} \cdot \boldsymbol{u'} \tag{517}$$

- Vector interpretation: $\boldsymbol{s'}$ is represented as a linear combination of column vectors of $\boldsymbol{B}$

$$\boldsymbol{s'} = \sum_{n=0}^{N-1} u'_n \cdot \boldsymbol{b}_n = u'_0 \cdot \boldsymbol{b}_0 + u'_1 \cdot \boldsymbol{b}_1 + \cdots + u'_{N-1} \cdot \boldsymbol{b}_{N-1} \tag{518}$$

Linear Block Transforms

Perfect Reconstruction Property

- Consider case that no quantization is applied ($\boldsymbol{u}' = \boldsymbol{u}$)
- Optimal synthesis transform:

$$\boldsymbol{B} = \boldsymbol{A}^{-1} \tag{519}$$

- Reconstructed samples are equal to source samples

$$\boldsymbol{s}' = \boldsymbol{B}\,\boldsymbol{u} = \boldsymbol{B}\,\boldsymbol{A}\,\boldsymbol{s} = \boldsymbol{A}^{-1}\,\boldsymbol{A}\,\boldsymbol{s} = \boldsymbol{s} \tag{520}$$

Optimal Synthesis Transform (in presence of quantization)

- Optimality: Minimum MSE distortion among all synthesis transforms
- $\boldsymbol{B} = \boldsymbol{A}^{-1}$ is optimal if
  - $\boldsymbol{A}$ is invertible and produces independent transform coefficients
  - the component quantizers are centroidal quantizers
- If above conditions are not fulfilled, a synthesis transform $\boldsymbol{B} \neq \boldsymbol{A}^{-1}$ may reduce the distortion

Orthogonal Block Transforms

Orthonormal Basis

- An analysis transform $\boldsymbol{A}$ forms an **orthonormal basis** if
  - basis vectors (matrix rows) are orthogonal to each other
  - basis vectors have to length 1
- The corresponding transform is called an **orthogonal transform**
- The transform matrices are called **unitary matrices**
- Unitary matrices with real entries are called **orthogonal matrix**
- Inverse of unitary matrices: Conjugate transpose

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^{\dagger} \quad \text{(for orthogonal matrices: } \boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}\text{)} \tag{521}$$

Why are orthogonal transforms desirable?

- MSE distortion can be minimized by independent scalar quantization of the transform coefficients
- Orthogonality of the basis vectors sufficient: Vector norms can be taken into account in quantizer design

Properties of Orthogonal Block Transforms

- Transform coding with orthogonal transform and perfect reconstruction $\boldsymbol{B} = \boldsymbol{A}^{-1} = \boldsymbol{A}^{\dagger}$ preserves MSE distortion

$$\begin{aligned} d_N(\boldsymbol{s}, \boldsymbol{s'}) &= \frac{1}{N} \, (\boldsymbol{s} - \boldsymbol{s'})^{\dagger} \, (\boldsymbol{s} - \boldsymbol{s'}) \\ &= \frac{1}{N} \, \left(\boldsymbol{A}^{-1} \, \boldsymbol{u} - \boldsymbol{B} \, \boldsymbol{u'}\right)^{\dagger} \, \left(\boldsymbol{A}^{-1} \, \boldsymbol{u} - \boldsymbol{B} \, \boldsymbol{u'}\right) \\ &= \frac{1}{N} \, \left(\boldsymbol{A}^{\dagger} \, \boldsymbol{u} - \boldsymbol{A}^{\dagger} \, \boldsymbol{u'}\right)^{\dagger} \, \left(\boldsymbol{A}^{\dagger} \, \boldsymbol{u} - \boldsymbol{A}^{\dagger} \, \boldsymbol{u'}\right) \\ &= \frac{1}{N} \, (\boldsymbol{u} - \boldsymbol{u'})^{\dagger} \, \boldsymbol{A} \, \boldsymbol{A}^{-1} \, (\boldsymbol{u} - \boldsymbol{u'}) \\ &= \frac{1}{N} \, (\boldsymbol{u} - \boldsymbol{u'})^{\dagger} \, (\boldsymbol{u} - \boldsymbol{u'}) \\ &= d_N(\boldsymbol{u}, \boldsymbol{u'}) \end{aligned} \tag{522}$$

- Scalar quantization that minimizes MSE in transform domain also minimizes MSE in original signal space
- For the special case of orthogonal matrices: $(\cdots)^{\dagger} = (\cdots)^{\mathrm{T}}$

Properties of Orthogonal Block Transforms

- Covariance matrix of transform coefficients

$$\begin{aligned} \boldsymbol{C}_{UU} &= E\{(\boldsymbol{U} - E\{\boldsymbol{U}\})(\boldsymbol{U} - E\{\boldsymbol{U}\})^{\mathrm{T}}\} \\ &= E\Big\{\boldsymbol{A}\,(\boldsymbol{S} - E\{\boldsymbol{S}\})(\boldsymbol{S} - E\{\boldsymbol{S}\})^{\mathrm{T}}\,\boldsymbol{A}^{\mathrm{T}}\Big\} \\ &= \boldsymbol{A}\,\boldsymbol{C}_{SS}\,\boldsymbol{A}^{-1} \end{aligned} \tag{523}$$

- Since the trace of a matrix is similarity-invariant,

$$\mathrm{tr}(\boldsymbol{X}) = \mathrm{tr}(\boldsymbol{P}\,\boldsymbol{X}\,\boldsymbol{P}^{-1}), \tag{524}$$

and the trace of an autocovariance matrix is the sum of the variances of the vector components, we have

$$\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2 = \sigma_S^2. \tag{525}$$

- **The arithmetic mean of the variances of the transform coefficients is equal to the variances of the source**

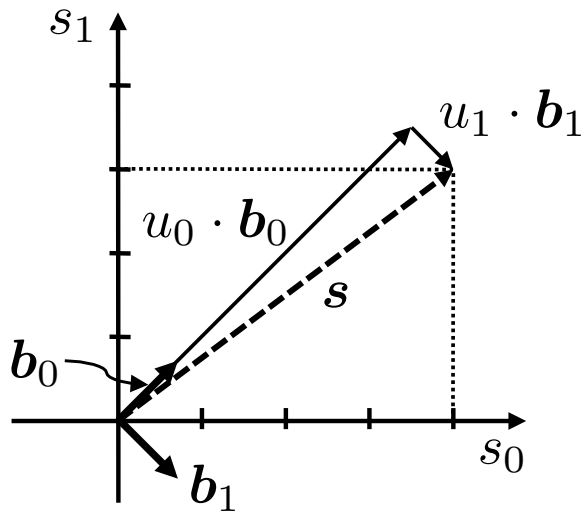
Geometrical Interpretation of Orthogonal Transforms

- Inverse 2-d transform matrix (= transpose of forward transform matrix)

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_0 & \boldsymbol{b}_1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \boldsymbol{A}^{\mathrm{T}}$$

- Vector interpretation for 2-d example

$$\begin{aligned} \boldsymbol{s} &= u_0 \cdot \boldsymbol{b}_0 + u_1 \cdot \boldsymbol{b}_1 \\ \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} &= u_0 \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + u_1 \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ \begin{bmatrix} 4 \\ 3 \end{bmatrix} &= 3.5 \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.5 \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{aligned}$$



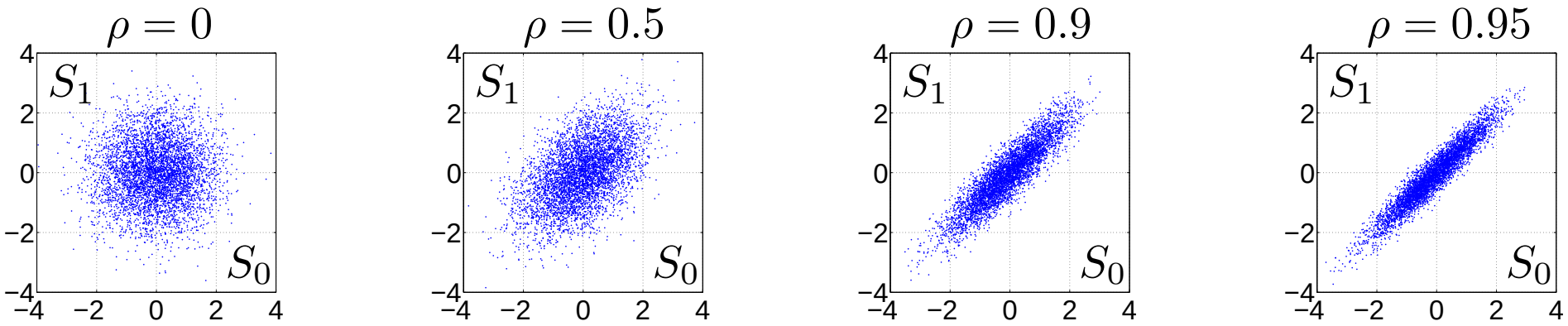
yielding transform coefficients

$$u_0 = \sqrt{2} \cdot 3.5 \qquad u_1 = \sqrt{2} \cdot 0.5$$

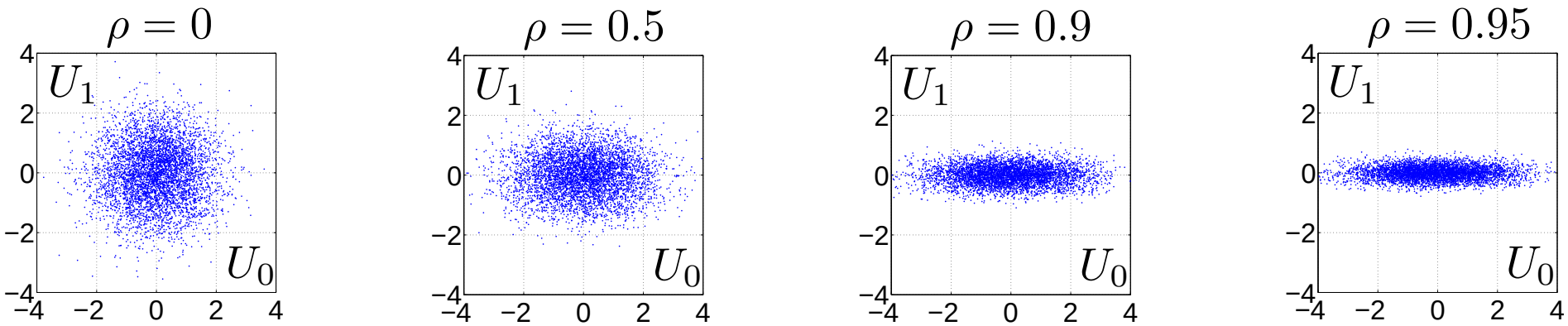
- An orthogonal transform is a rotation from the signal coordinate system into the coordinate system of the basis functions

Transform Example for $N = 2$

Adjacent samples of Gauss-Markov source with different correlation factors ρ

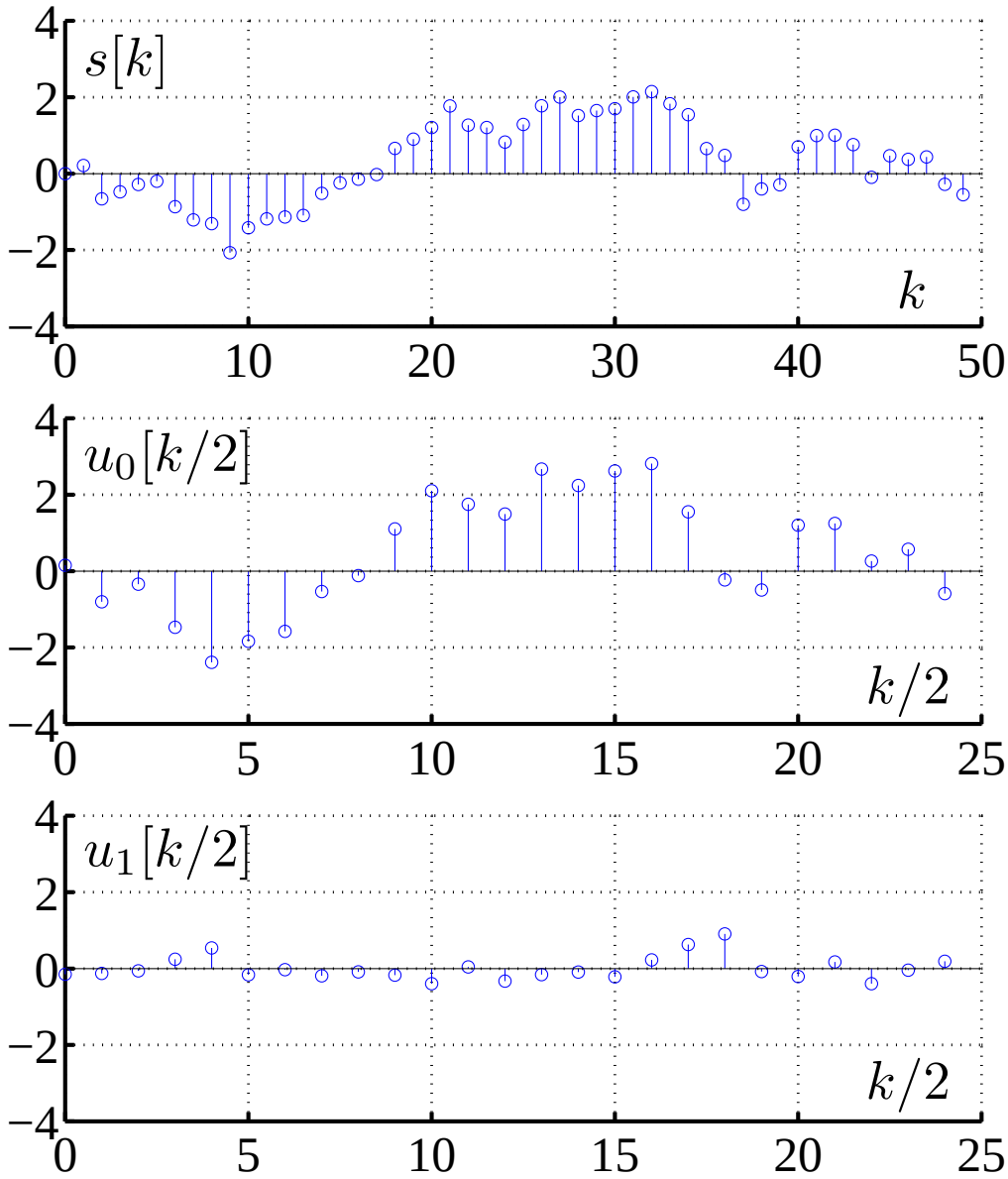


Transform coefficients for orthonormal 2D transform



Example for Waveforms (Gauss-Markov Source with $\rho = 0.95$)

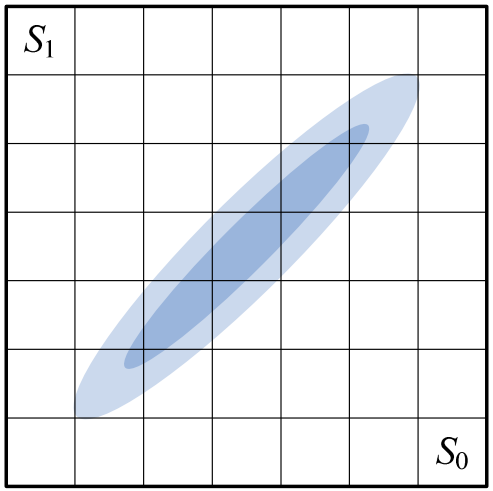
- Top: signal $s[k]$
- Middle:
  transform coefficient $u_0[k/2]$
  also called dc coefficient
- Bottom:
  transform coefficient $u_1[k/2]$
  also called ac coefficient
- Number of transform coefficients u_0 is half the number of samples s
- Number of transform coefficients u_1 is half the number of samples s



Scalar Quantization in Transform Domain

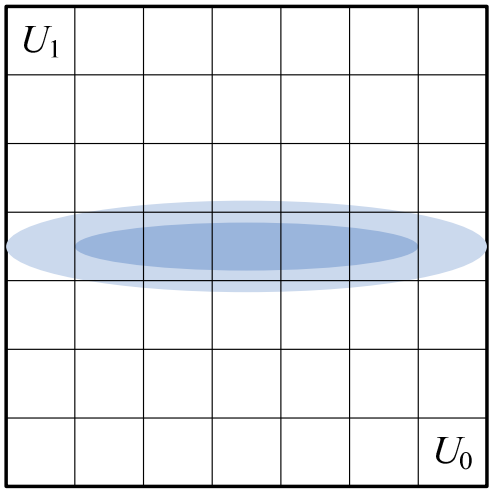
Consider Transform Coding with Orthogonal Transforms

direct coding



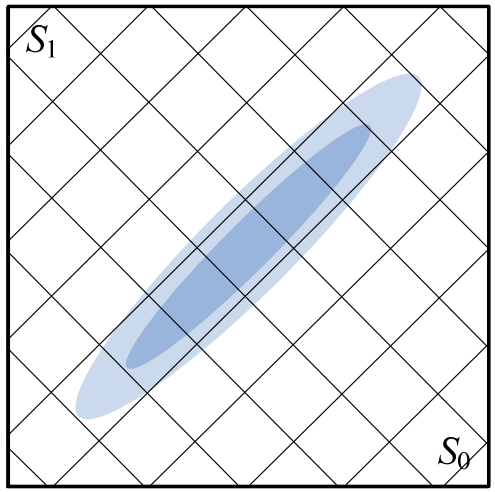
quantization cells

transform coding



quantization cells
in transform domain

transform coding



quantization cells
in signal space

- Quantization cells are
  - hyper-rectangles as in scalar quantization
  - but rotated and aligned with the transform basis vectors
- Number of quantization cells with appreciable probabilities is reduced
  $\Longrightarrow$ indicates improved coding efficiency for correlated sources

Bit Allocation for Transform Coefficients

- Problem: Distribute bit rate R among the N transform coefficients such that the resulting distortion D is minimized

$$\min \quad D(R) = \frac{1}{N} \sum_{i=1}^{N} D_i(R_i) \quad \text{subject to} \quad \frac{1}{N} \sum_{i=1}^{N} R_i \leq R \tag{526}$$

  with $D_i(R_i)$ being the oper. distortion-rate functions of the scalar quantizers

- Approach: Minimize Lagrangian cost function: $J = D + \lambda R$

$$\frac{\partial}{\partial R_i} \left(\sum_{i=1}^{N} D_i(R_i) + \lambda \sum_{i=1}^{N} R_i \right) = \frac{\partial D_i(R_i)}{\partial R_i} + \lambda \stackrel{!}{=} 0 \tag{527}$$

- Solution: Pareto condition

$$\frac{\partial D_i(R_i)}{\partial R_i} = -\lambda = \text{const} \tag{528}$$

- Move bits from coefficients with small distortion reduction per bit to coefficients with larger distortion reduction per bit

Bit Allocation for Transform Coefficients

- Operational distortion-rate function of scalar quantizers can be written as

$$D_i(R_i) = \sigma_i^2 \cdot g_i(R_i) \tag{529}$$

- Justified to assume that $g_i(R_i)$
  - is a continuous strictly convex function and
  - has a continuous strictly increasing derivative $g_i'(R_i)$ with $g_i'(\infty) = 0$
- Pareto condition becomes

$$-\sigma_i^2 \cdot g_i'(R_i) = \lambda \tag{530}$$

- If $\lambda \geq -\sigma_i^2 \, g_i'(0)$, the quantizer for u_i cannot be operated at the given slope $\Longrightarrow$ Set the corresponding component rate to $R_i = 0$
- Bit allocation rule

$$R_i = \begin{cases} 0 & : \quad -\sigma_i^2 g_i'(0) \leq \lambda \\ \eta_i\left(-\frac{\lambda}{\sigma_i^2}\right) & : \quad -\sigma_i^2 g_i'(0) > \lambda \end{cases} \tag{531}$$

  where $\eta_i(\cdot)$ denotes the inverse of the derivative $g_i'(\cdot)$
- Similar to reverse water-filling for Gaussian random variables

Approximation for Gaussian Sources

- Transform coefficients have also a Gaussian distribution
- Experimentally found approximation for entropy-constrained scalar quantization for Gaussian sources ($a \approx 0.952$)

$$g(R) = \frac{\pi e}{6a} \ln(a \cdot 2^{-2R} + 1) \tag{532}$$

- Use parameter

$$\theta = \lambda \frac{3\,(a+1)}{\pi e \ln 2} \qquad \text{with} \qquad 0 \leq \theta \leq \sigma^2_{\max} \tag{533}$$

- Bit allocation rule

$$R_i(\theta) = \begin{cases} 0 & : \quad \theta \geq \sigma_i^2 \\ \frac{1}{2} \log_2 \left(\frac{\sigma_i^2}{\theta}(a+1) - a \right) & : \quad \theta < \sigma_i^2 \end{cases} \tag{534}$$

- Resulting component distortions

$$D_i(\theta) = \begin{cases} \sigma_i^2 & : \quad \theta \geq \sigma_i^2 \\ -\frac{\varepsilon^2 \ln 2}{a} \cdot \sigma_i^2 \cdot \log_2 \left(1 - \frac{\theta}{\sigma_i^2} \frac{a}{a+1} \right) & : \quad \theta < \sigma_i^2 \end{cases} \tag{535}$$

High-Rate Approximation

- Assumption: High-rate approximation valid for all component quantizers
- High-rate approximation for distortion-rate function of component quantizers

$$D_i(R_i) = \varepsilon_i^2 \cdot \sigma_i^2 \cdot 2^{-2R_i} \tag{536}$$

  where ε_i^2 depends on transform coefficient distribution and quantizer
- Pareto condition

$$\frac{\partial}{\partial R_i} D_i(R_i) = -2 \ln 2 \, \varepsilon_i^2 \, \sigma_i^2 \, 2^{-2R_i} = -2 \ln 2 \, D_i(R_i) = -\lambda = \text{const} \tag{537}$$

  states that all quantizers are operated at the same distortion
- Bit allocation rule

$$R_i(D) = \frac{1}{2} \log_2 \left(\frac{\varepsilon_i^2 \, \sigma_i^2}{D} \right) \tag{538}$$

- Overall operational rate-distorion function

$$R(D) = \frac{1}{N} \sum_{i=0}^{N-1} R_i(D) = \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \left(\frac{\sigma_i^2 \varepsilon_i^2}{D} \right) \tag{539}$$

High-Rate Approximation

- Overall operational rate-distorion function

$$R(D) = \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \left(\frac{\sigma_i^2 \varepsilon_i^2}{D} \right) = \frac{1}{2} \log_2 \left(\frac{\tilde{\varepsilon}^2 \, \tilde{\sigma}^2}{D} \right) \tag{540}$$

  with geometric means

$$\tilde{\sigma}^2 = \left(\prod_{i=0}^{N-1} \sigma_i^2 \right)^{\frac{1}{N}} \qquad \text{and} \qquad \tilde{\varepsilon}^2 = \left(\prod_{i=0}^{N-1} \varepsilon_i^2 \right)^{\frac{1}{N}} \tag{541}$$

- Overall distortion-rate function

$$D(R) = \tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \tag{542}$$

- For Gaussian sources (transform coefficients are also Gaussian) and entropy-constrained scalar quantizers, we have $\varepsilon_i^2 = \varepsilon^2 = \frac{\pi e}{6}$, yielding

$$D_G(R) = \frac{\pi e}{6} \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \tag{543}$$

Transform Coding Gain at High Rates

- Transform coding gain is the ratio of the distortion for scalar quantization and the distortion for transform coding

$$G_T = \frac{\varepsilon_S^2 \cdot \sigma_S^2 \cdot 2^{-2R}}{\tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R}} = \frac{\varepsilon_S^2 \cdot \sigma_S^2}{\tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2} \tag{544}$$

with

σ_S^2 : variance of the input signal

ε_S^2 : factor of high-rate approximation for direct scalar quantization

- High-rate transform coding gain for Gaussian sources

$$G_T = \frac{\sigma_S^2}{\tilde{\sigma}^2} = \frac{\frac{1}{N}\sum_{i=0}^{N-1} \sigma_i^2}{\sqrt[N]{\prod_{i=0}^{N-1} \sigma_i^2}} \tag{545}$$

Ratio of arithmetic and geometric mean of the transform coefficient variances

- The high-rate transform coding gain for Gaussian sources is maximized if the geometric mean is minimized ($\Longrightarrow$ Karhunen Loève Transform)

Example: Orthogonal Transform with $N = 2$

- Input vector and transform matrix

$$\boldsymbol{s} = \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{546}$$

- Transformation

$$\boldsymbol{u} = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \boldsymbol{A} \cdot \boldsymbol{s} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} \tag{547}$$

- Coefficients

$$u_0 = \frac{1}{\sqrt{2}}(s_0 + s_1), \qquad u_0 = \frac{1}{\sqrt{2}}(s_0 - s_1) \tag{548}$$

- Inverse transformation

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}} = \boldsymbol{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{549}$$

Example: Orthogonal Transform with $N = 2$

- Variance of transform coefficients

$$\begin{aligned} \sigma_0^2 &= E\{U_0^2\} = E\left\{\frac{1}{2}(S_0 + S_1)^2\right\} = \frac{1}{2}\left(E\{S_0^2\} + E\{S_1^2\} + 2E\{S_0 S_1\}\right) \\ &= \frac{1}{2}\left(\sigma_S^2 + \sigma_S^2 + 2\sigma_S^2\rho\right) = \sigma_S^2(1+\rho) \qquad (550) \\ \sigma_1^2 &= E\{U_1^2\} = \sigma_S^2(1-\rho) \qquad (551) \end{aligned}$$

- Cross-correlation of transform coefficients

$$\begin{aligned} E\{U_0 U_1\} &= \frac{1}{2}E\{(S_0 + S_1)\cdot(S_0 - S_1)\} \\ &= \frac{1}{2}E\left\{\left(S_0^2 - S_1^2\right)\right\} = \sigma_S^2 - \sigma_S^2 = 0 \qquad (552) \end{aligned}$$

- Transform coding gain for Gaussian (assuming optimal bit allocation)

$$G_T = \frac{\sigma_S^2}{\sqrt{\sigma_0^2 + \sigma_1^2}} = \frac{1}{\sqrt{1-\rho^2}} \qquad (553)$$

Example: Analysis of Transform Coding for $N = 2$

- Rate-distortion cost before transform

$$J^{(0)} = 2(D + \lambda R) \quad \text{(for 2 samples)}$$

- Rate-distortion cost after transform

$$J^{(1)} = (D_0 + D_1) + \lambda(R_0 + R_1) \qquad \text{(for both transform coefficients)}$$

- Gain in r-d cost due to transform at same rate ($R_0 + R_1 = R$)

$$\Delta J = J^{(0)} - J^{(1)} = 2D - D_0 - D_1 \tag{554}$$

- For Gaussian sources, input and output of transform have Gaussian pdf
- With operational distortion-rate function for an entropy-constrained scalar quantizer at high rates ($D = \varepsilon^2 \cdot \sigma^2 \cdot 2^{-2R}$ with $\varepsilon^2 = \pi e/6$), we have

$$\Delta J = \varepsilon^2 \sigma_S^2 \big(2^{-2R+1} - (1+\rho)2^{-2R_0} - (1-\rho)2^{-2R_1}\big) \tag{555}$$

- By eliminating R_1 using $R_1 = 2R - R_0$, we get

$$\Delta J \;=\; \varepsilon^2 \sigma_S^2 \big(2^{-2R+1} - (1+\rho)2^{-2R_0} - (1-\rho)2^{-2(2R-R_0)}\big) \tag{556}$$

Example: Analysis of Transform Coding for $N = 2$

- Gain in rate-distortion cost due to transform

$$\Delta J \;=\; \varepsilon^2 \sigma_S^2 \big(2^{-2R+1} - (1+\rho) 2^{-2R_0} - (1-\rho) 2^{-2(2R-R_0)} \big) \tag{557}$$

- To maximize gain, we set

$$\frac{\partial}{\partial R_0} \Delta J = 2 \ln 2 \cdot (1+\rho) 2^{-2R_0} - 2 \ln 2 \cdot (1-\rho) 2^{-4R+2R_0} \stackrel{!}{=} 0 \tag{558}$$

  yielding the bit allocation rule

$$R_0 = R + \frac{1}{2} \log_2 \sqrt{\frac{1+\rho}{1-\rho}} \tag{559}$$

- Same expression is obtained by using the previously derived high rate bit allocation rule

$$R_i = \frac{1}{2} \log_2 \left(\frac{\varepsilon^2 \, \sigma_i^2}{D} \right) \tag{560}$$

- Operational high-rate distortion-rate function (Gaussian, ECSQ, $N = 2$)

$$D(R) = \frac{\pi e}{6} \cdot \sqrt{1-\rho^2} \cdot \sigma_S^2 \cdot 2^{-2R} \tag{561}$$

General Bit Allocation for Transform Coefficients

For Gaussian sources, the following points need to be considered:

- High-rate approximations are not valid for low bit rates; better approximations should be used for low rates
- For low rates, Pareto conditions cannot be fulfilled for all transform coefficients, since the component rates R_i must not be less then 0
- Solution:
  - Use generalized approximation of $D_i(R_i)$ for components quantizers
  - Set components rates R_i to zero for all transform coefficients, for which the Pareto condition $\frac{\partial}{\partial R_i} D(R_i) = -\lambda$ cannot be fullfilled for $R_i \geq 0$
  - Distribute rate among remaining coefficients

For non-Gaussian sources, the following needs to be considered in addition

- The transform coefficients have different (non-Gaussian) distributions (except for large transform sizes)
- Using the same quantizer design for all transform coefficients with $D_i(R_i) = \sigma_i^2 g(R_i)$ is suboptimal

Karhunen Loève Transform (KLT)

- Karhunen Loève Transform
  - Orthogonal transform that decorrelates the input vectors
  - Transform matrix depends on the source
- Autocorrelation matrix of input vectors $\boldsymbol{s}$

$$\boldsymbol{R}_{SS} = E\Big\{\boldsymbol{S}\boldsymbol{S}^{\mathrm{T}}\Big\} \tag{562}$$

- Autocorrelation matrix of transform coefficient vectors $\boldsymbol{u}$

$$\begin{aligned} \boldsymbol{R}_{UU} &= E\left\{\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\right\} = E\left\{(\boldsymbol{A}\boldsymbol{S})(\boldsymbol{A}\boldsymbol{S})^{\mathrm{T}}\right\} = \boldsymbol{A}\cdot E\left\{\boldsymbol{S}\boldsymbol{S}^{\mathrm{T}}\right\}\cdot\boldsymbol{A}^{\mathrm{T}} \\ &= \boldsymbol{A}\,\boldsymbol{R}_{SS}\,\boldsymbol{A}^{\mathrm{T}} \end{aligned} \tag{563}$$

- By multiplying with $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}$ from the front, we get

$$\boldsymbol{R}_{SS}\cdot\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{A}^{\mathrm{T}}\cdot\boldsymbol{R}_{UU} \tag{564}$$

- To get uncorrelated transform coefficients, we need to obtain a diagonal autocorrelation matrix $\boldsymbol{R}_{UU}$ for the transform coefficients

Karhunen Loève Transform (KLT)

- Expression for autocorrelation matrices

$$\boldsymbol{R}_{SS} \cdot \boldsymbol{A}^{\mathrm{T}} = \boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{R}_{UU} \tag{565}$$

- $\boldsymbol{R}_{UU}$ is a diagonal matrix if the eigenvector equation

$$\boldsymbol{R}_{SS} \cdot \boldsymbol{b}_i = \xi_i \cdot \boldsymbol{b}_i \tag{566}$$

  is fulfilled for all basis vectors $\boldsymbol{b}_i$ (column vectors of $\boldsymbol{A}^{\mathrm{T}}$, row vectors of $\boldsymbol{A}$)

- The transform matrix $\boldsymbol{A}$ decorrelates the input vectors if its rows are equal to the unit-norm eigenvectors $\boldsymbol{v}_i$ of $\boldsymbol{R}_{SS}$

$$\boldsymbol{A}_{KLT} = \begin{bmatrix} \boldsymbol{v}_0 \; \boldsymbol{v}_1 \; \cdots \; \boldsymbol{v}_{N-1} \end{bmatrix}^{\mathrm{T}} \tag{567}$$

- The resulting autocorrelation matrix $\boldsymbol{R}_{UU}$ is a diagonal matrix with the eigenvalues of $\boldsymbol{R}_{SS}$ on its main diagonal

$$\boldsymbol{R}_{UU} = \begin{bmatrix} \xi_0 & 0 & \cdots & 0 \\ 0 & \xi_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_{N-1} \end{bmatrix} \tag{568}$$

Optimality of KLT for Gaussian Sources

- Transform coding with orthogonal $N \times N$ transform matrix $\boldsymbol{A}$ and $\boldsymbol{B} = \boldsymbol{A}^T$
- Scalar quantization using scaled quantizers

$$D(R, \boldsymbol{A}_k) = \sum_{i=0}^{N-1} \sigma_i^2(\boldsymbol{A}_k) \cdot g(R_i) \tag{569}$$

  with $\sigma_i^2(\boldsymbol{A}_k)$ being variance of i-th transform coefficient and $\boldsymbol{A}_k$ being the transform matrix

- Consider an arbitrary orthogonal transform matrix $\boldsymbol{A}_0$ and an arbitrary bit allocation given by the vector $\boldsymbol{r} = [R_0, \cdots, R_{N-1}]^{\mathrm{T}}$ with $\sum_{i=0}^{N-1} R_i = R$
- Starting with arbitrary orthogonal matrix $\boldsymbol{A}_0$, apply iterative algorithm that generates a series of orthonormal transform matrices $\{\boldsymbol{A}_k\},\ k = 1, 2, ...$
- Iteration $\boldsymbol{A}_{k+1} = \boldsymbol{J}_k \boldsymbol{A}_k$ consists of Jacobi rotation and re-ordering
  $\Longrightarrow$ Transform matrix approaches a KLT matrix
- Can show that for all $\boldsymbol{A}_k$: $D(R, \boldsymbol{A}_{k+1}) \leq D(R, \boldsymbol{A}_{k+1})$
  $\Longrightarrow$ KLT is optimal transform for Gaussian sources (minimizes MSE)

Asymp. High-Rate Performance of KLT for Gaussian Sources

- Transform coefficient variances σ_i^2 are equal to the eigenvalues ξ_i of $\boldsymbol{R}_{SS}$
- High-rate approximation for Gaussian source and optimal ECSQ

$$\begin{aligned} D(R) &= \frac{\pi e}{6} \cdot \tilde{\sigma}^2 \cdot 2^{-2R} = \frac{\pi e}{6} \cdot \tilde{\xi} \cdot 2^{-2R} \\ &= \frac{\pi e}{6} \cdot 2^{\frac{1}{N} \sum_{i=0}^{N-1} \log_2 \xi_i} \cdot 2^{-2R} \end{aligned} \tag{570}$$

- For $N \to \infty$, we can apply the theorem of Szegö and Grenander for infinite Toeplitz matrices: If all eigenvalues ξ_i of an infinite autocorrelation matrix are finite and $G(\xi_i)$ is any continuous function over all eigenvalues,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} G(\xi_i) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G(\Phi(\omega)) d\omega \tag{571}$$

- Resulting distortion-rate function for KLT of infinite size for high rates

$$D_{\mathrm{KLT}}^{\infty}(R) = \frac{\pi e}{6} \cdot 2^{\frac{1}{2\pi} \int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) \cdot \mathrm{d}\omega} \cdot 2^{-2R} \tag{572}$$

Asymp. High-Rate Performance of KLT for Gaussian Sources

- Asymptotic distortion-rate function for KLT of infinite size for high rates

$$D_{\mathrm{KLT}}^{\infty}(R) = \frac{\pi e}{6} \cdot 2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega)\cdot \mathrm{d}\omega} \cdot 2^{-2R} \tag{573}$$

- Information distortion-rate function (fundamental bound) is by a factor $\varepsilon^2 = \pi e/6$ smaller

$$D(R) = 2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega)\cdot \mathrm{d}\omega} \cdot 2^{-2R} \tag{574}$$

- Asymptotic transform gain ($N \to \infty$) at high rates

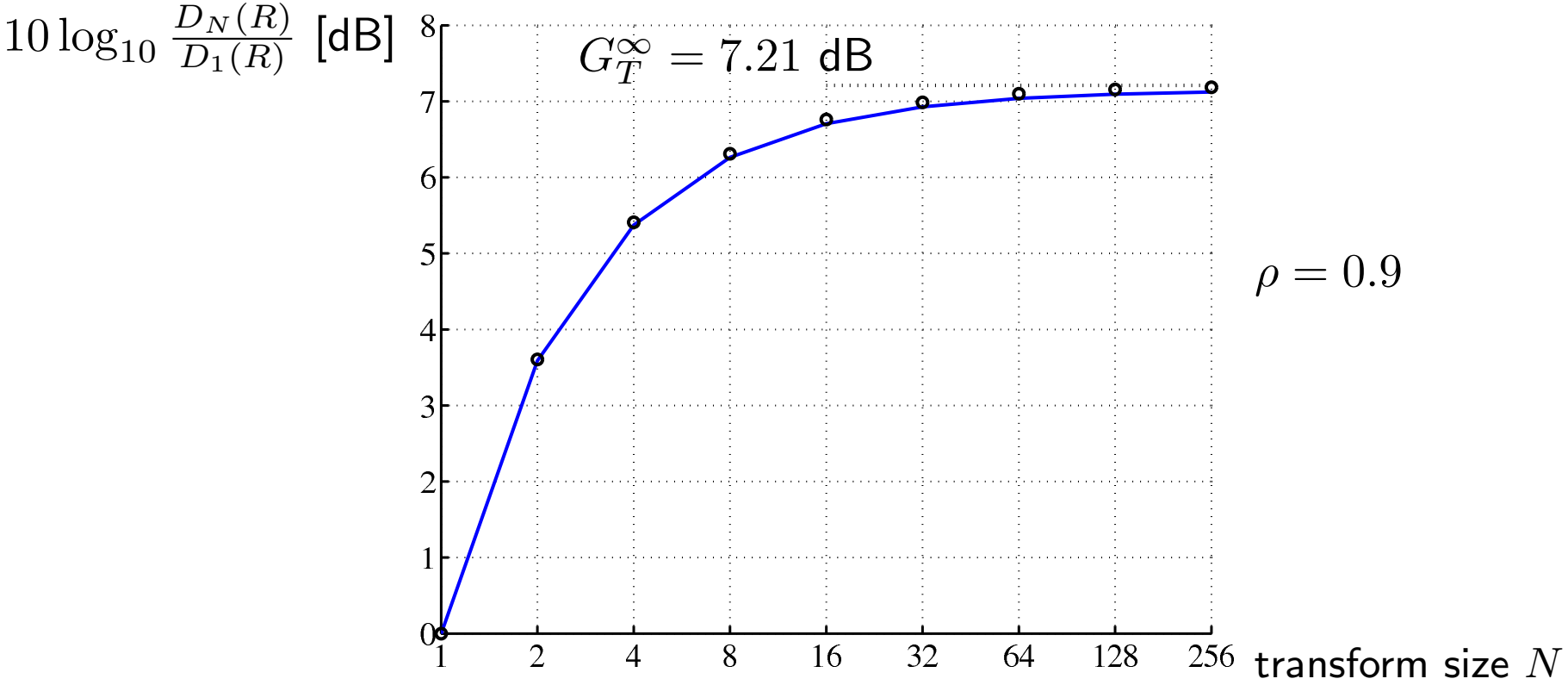
$$G_T^{\infty} = \frac{\varepsilon^2 \sigma_S^2 2^{-2R}}{D_{\mathrm{KLT}}^{\infty}(R)} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{SS}(\omega) d\omega}{2^{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2 \Phi_{SS}(\omega) d\omega}} \tag{575}$$

- Asymptotic transform gain ($N \to \infty$) at high rates is identical to the asymptotic prediction gain at high rates

High-Rate KLT Transform Gain for Gauss-Markov Sources

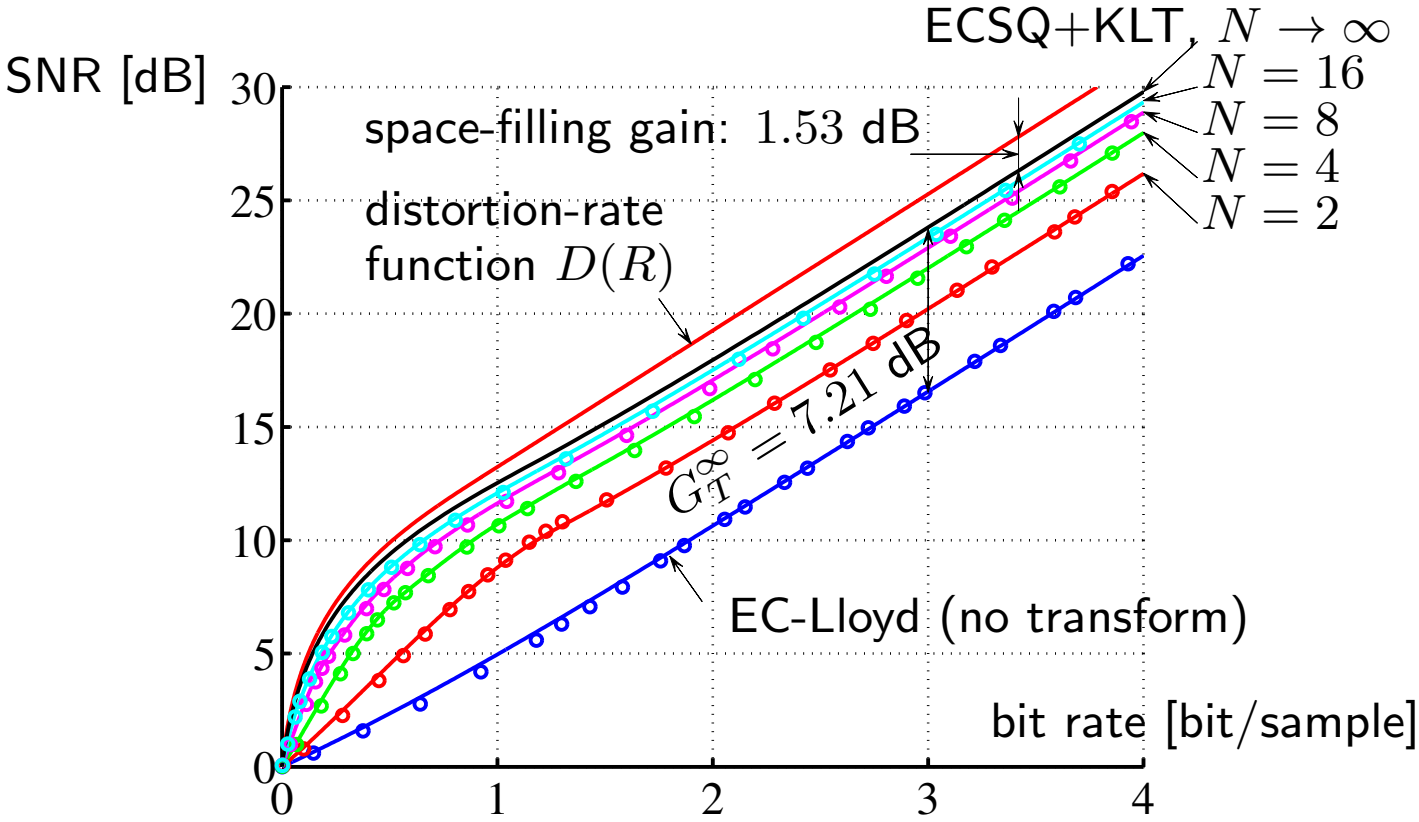
- Operational distortion-rate function for KLT of size N, ECSQ, and optimum bit allocation for Gauss-Markov sources with correlation factor ρ

$$D_N(R) = \frac{\pi e}{6} \cdot \sigma_S^2 \cdot (1-\rho^2)^{1-1/N} \cdot 2^{-2R} \tag{576}$$

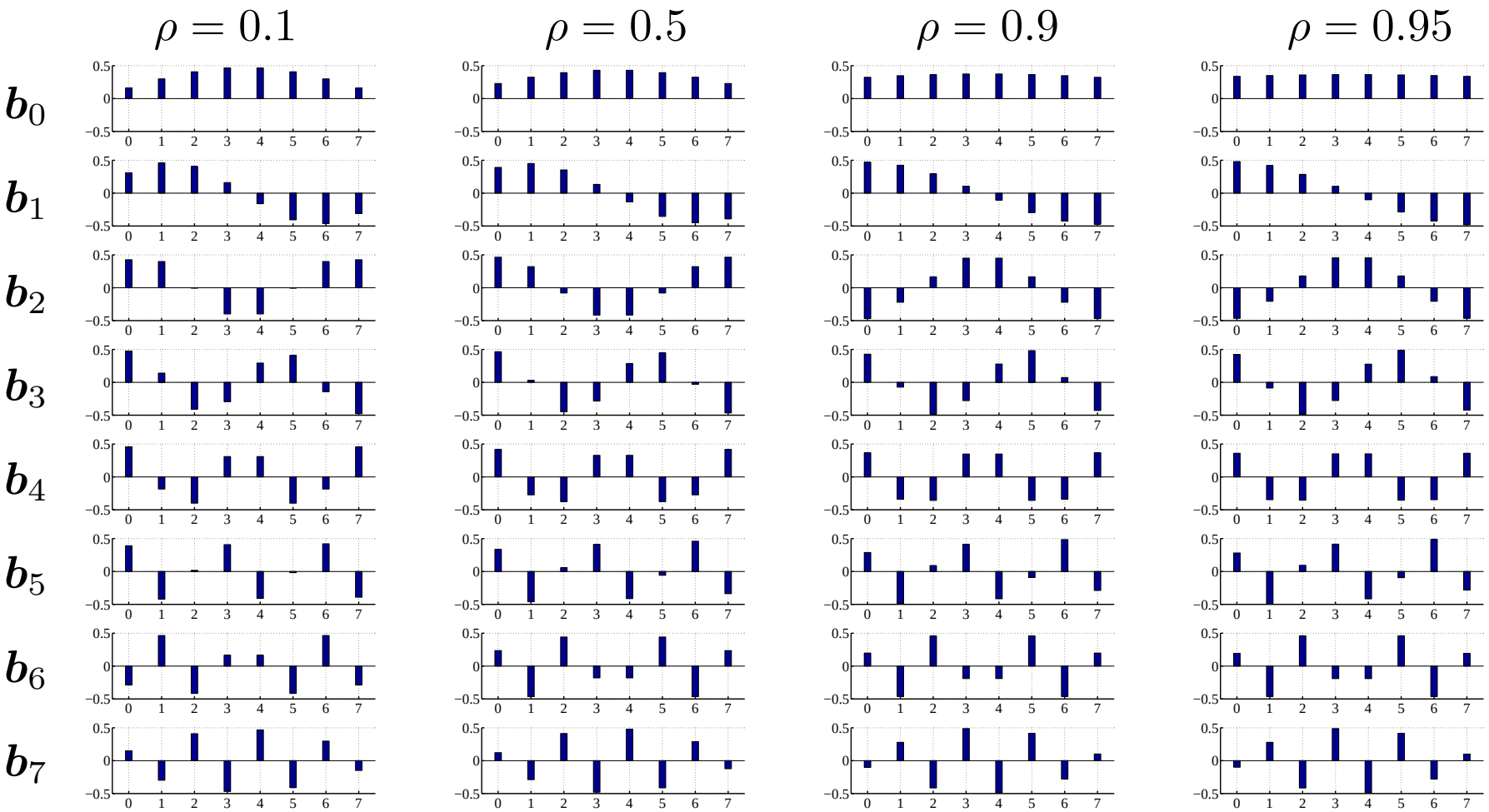


Operat. Distortion-Rate Functions for Gauss-Markov

- Distortion-rate curves for coding a first-order Gauss-Markov source with correlation factor $\rho = 0.9$ and different transform sizes N



KLT Basis Functions for Gauss-Markov Sources and Size $N = 8$



Walsh-Hadamard Transform

- Very simple orthogonal transform (only additions & final scaling)
- For transform sizes N that are positive integer power of 2

$$\boldsymbol{A}_N = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{A}_{N/2} & \boldsymbol{A}_{N/2} \\ \boldsymbol{A}_{N/2} & -\boldsymbol{A}_{N/2} \end{bmatrix} \quad \text{with} \quad \boldsymbol{A}_1 = [1]. \tag{577}$$

- Transform matrix for $N = 8$

$$\boldsymbol{A}_8 = \frac{1}{2\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{578}$$

- Piecewise-constant basis vectors
- Image & video coding: Produces subjectively disturbing artifacts when combined with strong quantization

Discrete Fourier Transform (DFT)

- Discrete version of the Fourier transform
- Forward Transform

$$u[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} s[n] \cdot e^{-j\frac{2\pi kn}{N}} \tag{579}$$

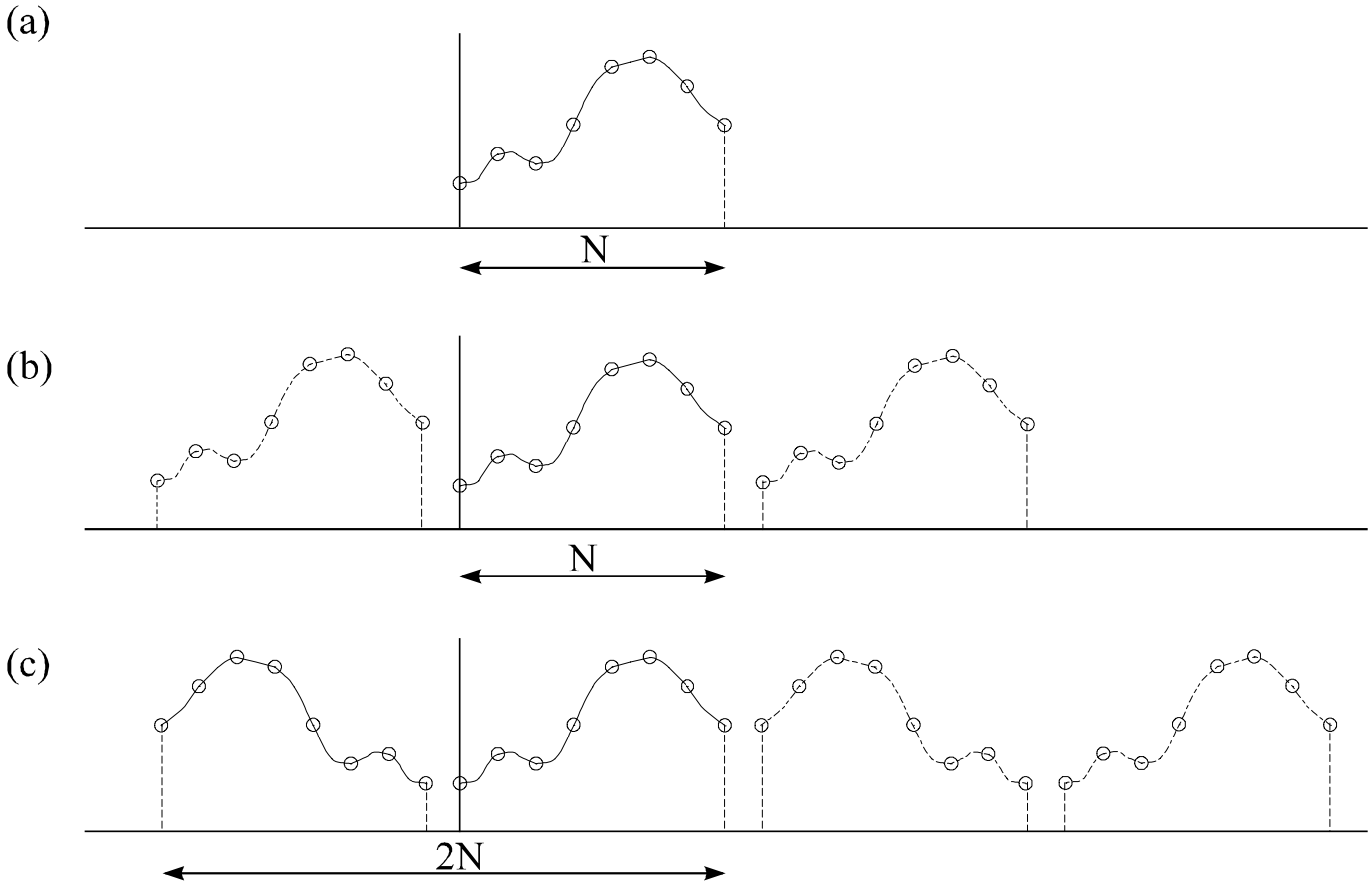
- Inverse Transform

$$s[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} u[k] \cdot e^{j\frac{2\pi kn}{N}} \tag{580}$$

- DFT is an orthonormal transform (specified by a unitary transform matrix)
- Produces complex transform coefficients
- For real inputs, it obeys the symmetry $u[k] = u^*[N-k]$, so that N real samples are mapped onto N real values
- FFT is a fast algorithm for DFT computation, uses sparse matrix factorization
- Implies periodic signal extension: Differences between left and right signal boundary reduces rate of convergence of Fourier series
- Strong quantization $\Longrightarrow$ Significant high-frequent artifacts

Discrete Fourier Transform vs. Discrete Cosine Transform

(a) Input time-domain signal
(b) Time-domain replica in case of DFT
(c) Time-domain replica in case of DCT-II



Derivation of DCT Type II

- Reduce quantization errors of DFT by introducing mirror symmetry and applying a DFT of approximately double size
- Signal with mirror symmetry

$$s^*[n] = \begin{cases} s[n-1/2] & : \quad 0 \le n < N \\ s[2N-n-3/2] & : \quad N \le n < 2N \end{cases} \tag{581}$$

- Transform coefficients (orthonormal: divide $u^*[0]$ by $\sqrt{2}$)

$$\begin{aligned} u^*[k] &= \frac{1}{\sqrt{2N}} \sum_{i=0}^{2N-1} s^*[i] e^{-j\frac{2\pi kn}{2N}} \\ &= \frac{1}{\sqrt{2N}} \sum_{n=0}^{N-1} s[n-1/2] \left(e^{-j\frac{\pi}{N}kn} + e^{-j\frac{\pi}{N}k(2N-n-1)} \right) \\ &= \frac{1}{\sqrt{2N}} \sum_{n=0}^{N-1} s[n] \left(e^{-j\frac{\pi}{N}k\left(n+\frac{1}{2}\right)} + e^{j\frac{\pi}{N}k\left(n+\frac{1}{2}\right)} \right) \\ &= \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} s[n] \cos\left(\frac{\pi}{N} k \left(n + \frac{1}{2} \right) \right) \end{aligned} \tag{582}$$

Discrete Cosine Transform (DCT)

- Implicit periodicity of DFT leads to loss in coding efficiency
- This can be reduced by introducing mirror symmetry at the boundaries and applying a DFT of approximately double size
- Due to mirror symmetry, imaginary sine terms get eliminated and only cosine terms remain
- Most common DCT is the so-called DCT-II (mirror symmetry with sample repetitions at both sides: $n = -\frac{1}{2}$)
- DCT and IDCT Type-II are given by

$$u[k] = \alpha_k \sum_{n=0}^{N-1} s[n] \cdot \cos\left[k \cdot \left(n + \frac{1}{2}\right) \cdot \frac{\pi}{N}\right] \tag{583}$$

$$s[n] = \sum_{k=0}^{N-1} \alpha_k \cdot u[k] \cdot \cos\left[k \cdot \left(n + \frac{1}{2}\right) \cdot \frac{\pi}{N}\right] \tag{584}$$

where $\alpha_0 = \sqrt{\frac{1}{N}}$ and $\alpha_n = \sqrt{\frac{2}{N}}$ for $n \neq 0$

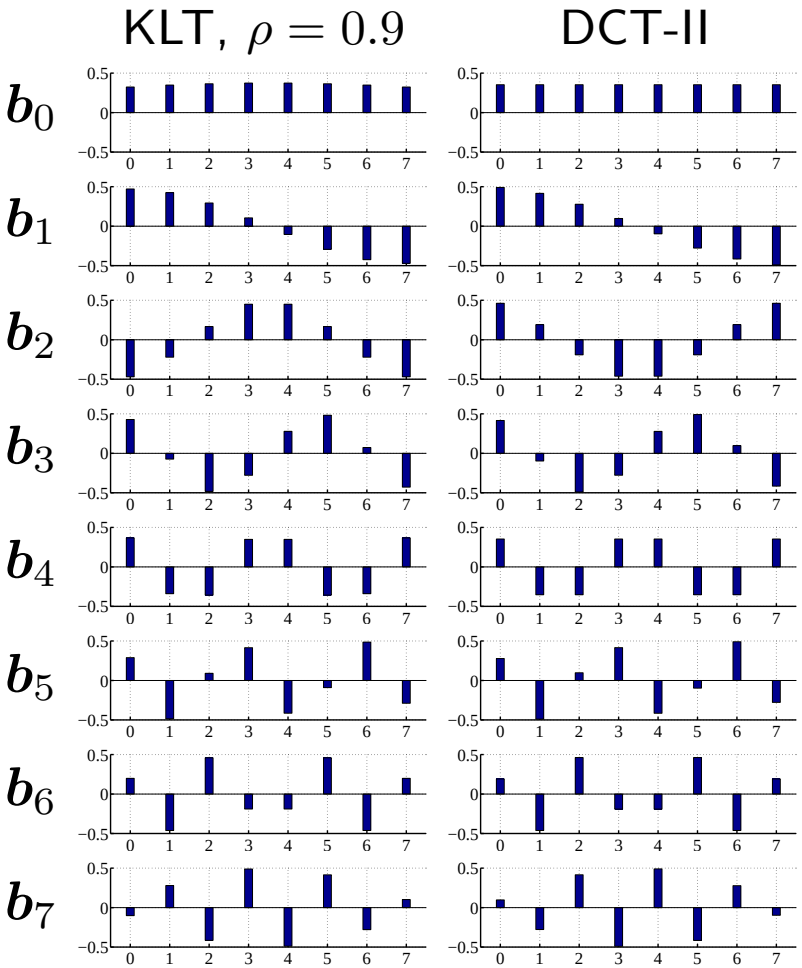
Comparison of DCT and KLT

- Correlation matrix of a first-order Markov processes can be written as

$$\boldsymbol{R}_{SS} = \sigma_S^2 \cdot \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \vdots & & & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix} \tag{585}$$

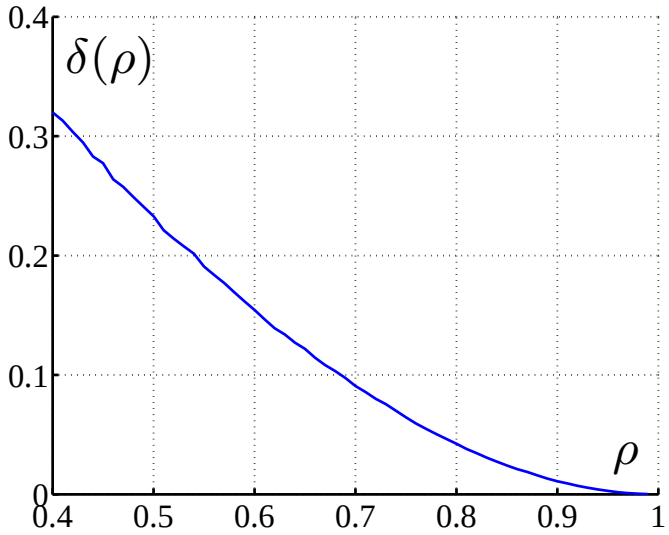
- DCT is a good approximation of the eigenvectors of $\boldsymbol{R}_{SS}$
- DCT basis vectors approach the basis functions of the KLT for first-order Markov processes with $\rho \to 1$
- DCT does not depend on input signal
- Fast algorithms for computing forward and inverse transform
- Justification for wide usage of DCT (or integer approximations thereof) in image and video coding:
  JPEG, H.261, H.262/MPEG-2, H.263, MPEG-4, H.264/AVC, H.265/HEVC

KLT Convergence Towards DCT for $\rho \to 1$



Difference between the transform matrices of KLT and DCT-II

$$\delta(\rho) = ||\boldsymbol{A}_{KLT}(\rho) - \boldsymbol{A}_{DCT}||_2^2$$



Two-dimensional Transforms

- 2-D linear transform:
  Input image is represented as a linear combination of basis images
- An orthonormal transform is separable and symmetric, if the transform of a signal block $\boldsymbol{s}$ of size $N \times N$ can be expressed as,

$$\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s} \cdot \boldsymbol{A}^{\mathrm{T}} \tag{586}$$

  where $\boldsymbol{A}$ is the transformation matrix and $\boldsymbol{u}$ is the matrix of transform coefficients, both of size $N \times N$.
- The inverse transform is

$$\boldsymbol{s} = \boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{s} \cdot \boldsymbol{A} \tag{587}$$

- Great practical importance:
  Transform requires 2 matrix multiplications of size $N \times N$ instead one multiplication of a vector of size $1 \times N^2$ with a matrix of size $N^2 \times N^2$
- Reduction of the complexity from $O(N^4)$ to $O(N^3)$

2-dimensional DCT Example

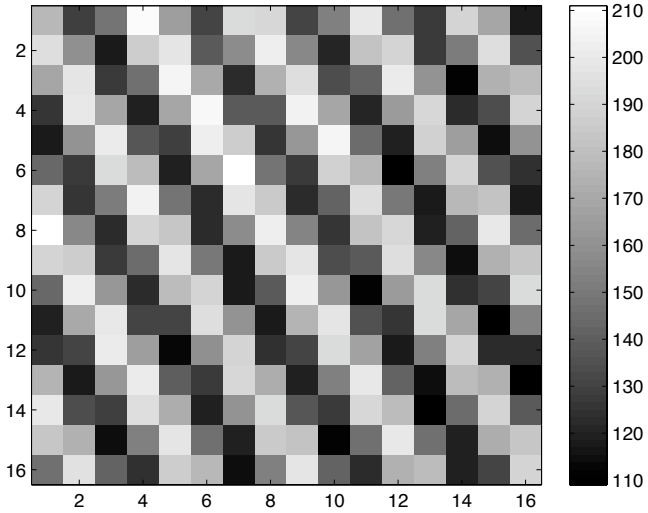
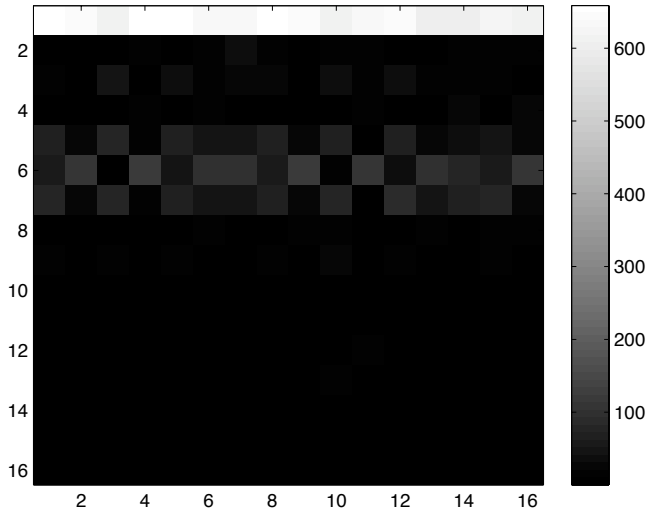


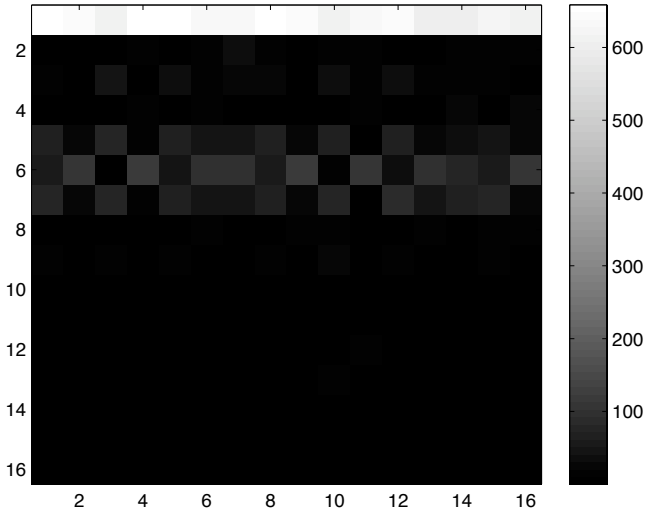
image block



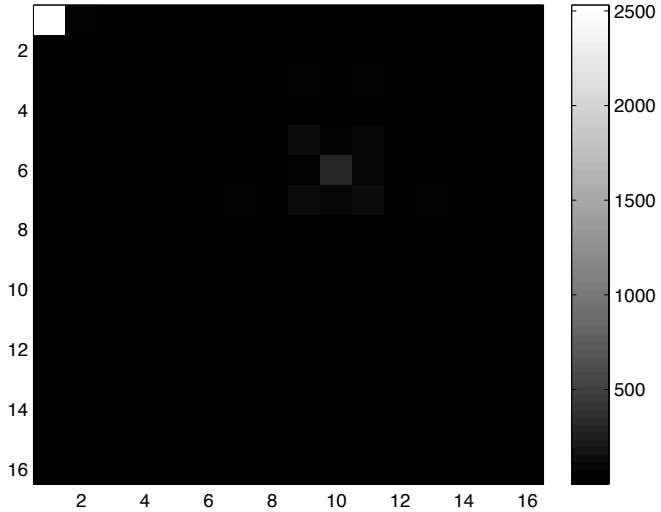
column-wise DCT

- 1-d DCT is applied to each column of an image block
- Notice the energy concentration in the first row (DC coefficients)

2-dimensional DCT Example



column-wise DCT

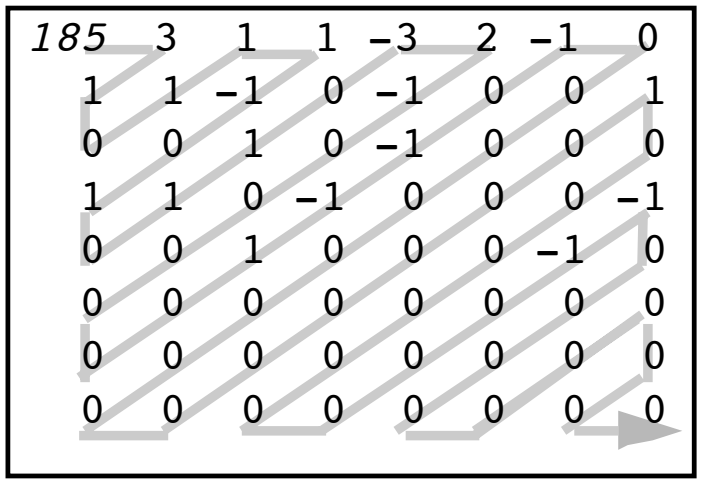


final result

- For convenience, column-wise DCT result is repeated on left side
- 1-d DCT is applied to each row of the intermediate result
- Notice the energy concentration in the first coefficient

Entropy Coding of Transform Coefficients

- AC coefficients are very likely equal to zero (for moderate quantization)
- For 2-d, ordering of the transform coefficients by zig-zag (or similar) scan
- Example for zig-zag scanning in case of a 2-d transform



- Huffman code for events {number of leading zeros, coefficient value} or events {end-of-block, number of leading zeros, coefficient value}
- Arithmetic coding: For example, use probabilities that particular coefficient is unequal to zero when quantizing with a particular step size

Chapter Summary

Orthogonal block transform

- Orthogonal transform: Rotation of coordinate system in signal space
- Purpose of transform: Decorrelation, energy concentration
  $\Longrightarrow$ Align quantization cells with primary axis of joint pdf
- KLT achieves optimum decorrelation, but is signal dependent
- DCT shows reduced blocking artifacts compared to DFT
- For Gauss-Markov and $\rho \to \infty$: DCT approaches KLT

Bit allocation and transform coding gain

- For Gaussian sources: Bit allocation proportional to logarithm of variances
- For high rates: Optimum bit allocation yields equal component distortion
- Larger transform size increases gain for Gauss-Markov source

Application of transform coding

- Widely used in image and video coding:
  DCT (or approximation) + quantization + (zig-zag) scan + entropy coding
  $\Longrightarrow$ JPEG, H.262/MPEG-2, H.263, MPEG-4, H.264/AVC, H.265/HEVC

Exercise 25

Consider a zero-mean Gauss-Markov process with variance σ_S^2 and correlation coefficient ρ. The source is coded using a transform coding system consisting of a N-dimensional KLT, optimal bit allocation and optimal entropy-constrained scalar quantizers with optimal entropy coding.

Show that the high-rate approximation of the operational distortion-rate function is given by

$$D(R) = \frac{\pi\, e}{6} \cdot \sigma_S^2 \cdot (1-\rho^2)^{\frac{N-1}{N}} \cdot 2^{-2R}$$

Exercise 26

In the video coding standard ITU-T Rec. H.264 the following forward transform is used (more accurately, only the inverse transform is specified in the standard, but the given transform is used in most actual encoder implementation),

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

How large is the high-rate transform coding gain (in dB) for a zero-mean Gauss-Markov process with the correlation factor $\rho = 0.9$?

By what amount (in dB) can the high-rate transform coding gain be increased if the transform is replaced by a KLT?

NOTE: The basis functions of the given transform are orthogonal to each other, but they don't have the same norm.

Exercise 27 – Part 1/2

Given is a zero-mean Gaussian process with the autocovariance matrix for $N = 4$

$$\mathbf{C}_{SS} = \sigma_S^2 \begin{bmatrix} 1.00 & 0.95 & 0.92 & 0.88 \\ 0.95 & 1.00 & 0.95 & 0.92 \\ 0.92 & 0.95 & 1.00 & 0.95 \\ 0.88 & 0.92 & 0.95 & 1.00 \end{bmatrix}$$

Consider transform coding with the Hadamard transform given by

$$\mathbf{A} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

The scalar quantizers for the transform coeff. have 5 operation points given by

$$\begin{aligned} R_i = 0 &\implies D_i = \sigma_i^2 \\ R_i = 1 &\implies D_i = 0.32\,\sigma_i^2 \\ R_i = 2 &\implies D_i = 0.09\,\sigma_i^2 \\ R_i = 3 &\implies D_i = 0.02\,\sigma_i^2 \\ R_i = 4 &\implies D_i = 0.01\,\sigma_i^2 \end{aligned}$$

Exercise 27 – Part 2/2

For each transform coefficients, any of the 5 operation points can be chosen.

Derive the optimal bit allocation (i.e., the component rates R_i for $i = 0, 1, 2, 3$) for the overall rate of $R = 1$ bit per sample.

What distortion D and SNR is achieved for this rate?

How big is the transform coding gain? Is it larger than, smaller than, or equal to the transform coding gain for high rates (the above given operation points are good approximations for optimal entropy-constrained quantizers for Gaussian sources and can be considered as valid for the comparison)?

Exercise 28

Consider transform coding with an orthogonal transform of a zero-mean Gaussian source with variance σ_S^2. The used scalar quantizers have the operational distortion rate function

$$D_i(R_i) = \sigma_i^2 \, g(R_i)$$

where $g(R)$ is some not further specified function.

We don't use an optimal bit allocation, but assign the same rate to all transform coefficients.

Does the transform coding still provide a gain in comparison to simple scalar quantization with the given quantizer, assuming that the Gaussian source is not iid?

Exercise 29

Consider a zero-mean Gauss-Markov process with variance $\sigma_S^2 = 1$ and correlation coefficient $\rho = 0.9$. As transform a KLT of size 3 is used, the resulting transform coefficient variances are

$$\sigma_0^2 = 2.7407, \qquad \sigma_1^2 = 0.1900, \qquad \sigma_2^2 = 0.0693$$

Consider high-rate quantization with optimal entropy-constrained scalar quantizers.

Derive the high-rate operational distortion rate function. What is the optimal high-rate bit allocation scheme for a given overall rate R?

Determine the component rates, the overall distortion and the SNR for a given overall bit rate R of 4 bit per sample.

Determine the high-rate transform coding gain.

PartII:

Application in Image and Video Coding

Outline

Part I: Source Coding Fundamentals

- Probability, Random Variables and Random Processes
- Lossless Source Coding
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

- **Still Image Coding / Intra-Picture Coding**
  - Representation of Images and Video
  - JPEG
  - Intra-Picture Coding in MPEG-2 Video
  - Intra-Picture Coding in H.263 and MPEG-2 Visual
  - Intra-Picture Coding in H.264/AVC
  - Intra-Picture Coding in H.265/HEVC
- Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)

Still Image Coding / Intra-Picture Coding – Overview

Still Image Coding

- Exchange, transmission and storage of images
- Used in virtually all digital cameras and picture editing applications
- JPEG: Most widely used image compression standard (based on DCT)
- JPEG-2000: Wavelet-based image compression (not discussed in lecture)
- JPEG-XR: Several improvements over JPEG (not discussed in lecture)

Intra-Picture Coding for Video

- Intra-picture coding: Some pictures of a video sequence need to be coded without referring to other picture inside the video sequence
- First picture of a video sequence has to be intra-picture coded
- Intra pictures in regular intervals (e.g., 1s) are required for enabling random access
- Typically, regularly inserted intra pictures consume large amount of bit rate
- H.262 | MPEG-2 Video / H.263 / MPEG-4 Visual: Conceptually similar to JPEG
- H.264 | MPEG-4 AVC: Additional coding tools yielding improved coding efficiency
- H.265 | MPEG-H HEVC: Increased flexibility and improved coding efficiency

Representation of Images and Video

Digital Images and Video

Image

- 2-d function $s(x, y)$ relating light intensity s to spatial coordinates (x, y)

Digital image

- Representation of a continuous image at discrete coordinates $[x, y]$
- Amplitudes $s[x, y]$ have finite alphabet, typically determined by the used bit depth
- Digital gray-level image of size $M \times N$ can be represented using a matrix notation

$$\begin{bmatrix} s[0,0] & s[1,0] & \cdots & s[M-1,0] \\ s[0,1] & s[1,1] & \cdots & s[M-1,1] \\ \vdots & \vdots & \ddots & \vdots \\ s[0,N-1] & s[1,N-1] & \cdots & s[M-1,N-1] \end{bmatrix}$$

- Typically characterized by image size $M \times N$ and bit depth
- Color images are typically composed of 3 sample arrays (for different color components)

Digital video

- Sequence of digital images captured at successive time instances
- Typically characterized by frame rate (in addition to image size and bit depth)

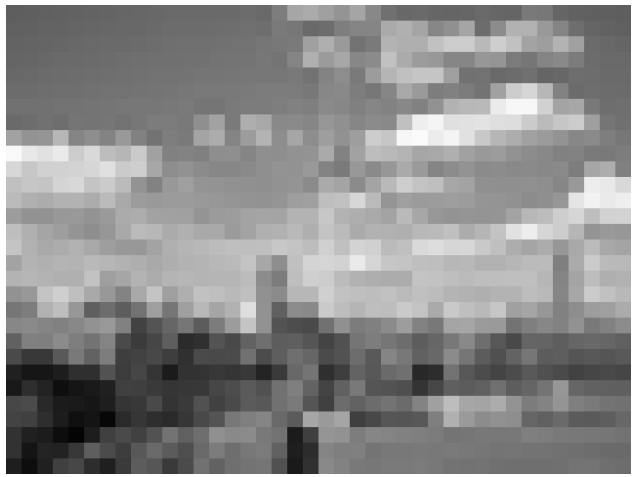
Spatial Resolution

Number of samples ($M \times N$) for discrete matrix representation

320×240 samples

160×120 samples

80×60 samples



40×30 samples

Gray-Level Resolution / Bit Depth

Number of gray levels for image representation (typically determined by bit depth)

256 gray levels (8 bit)

64 gray levels (6 bit)

16 gray levels (4 bit)

4 gray levels (2 bit)

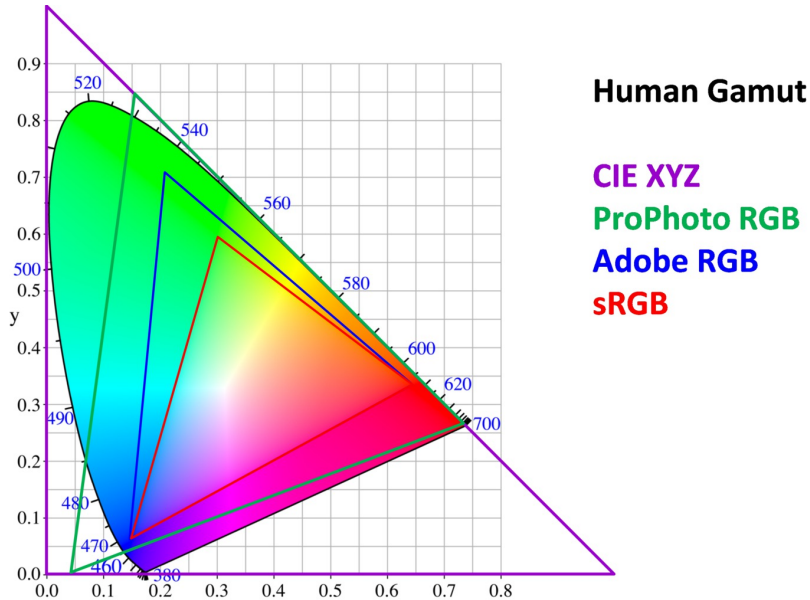
Representation of Color Images

Color components

- Require at least 3 color components (trichromatic vision)
- RGB typically used as reference color space
- Need to specify color of RGB primaries in CIE XYZ reference space

BT.709 RGB parameters

primary	x	y
red	0.6400	0.3300
green	0.3000	0.6000
blue	0.1500	0.0600
white D65	0.3127	0.3290



YCbCr Color Space

Definition of YCbCr color space

- More correct name is Y'CbCr, since Y' is a gamma-adjusted luminance component
- Not an absolute color space, but a different representation for RGB data
- Transform of gamma-adjusted and normalized RGB components r', g' and b' with a range of 0..1
- Typically used transform for 8-bit components Y, Cb and Cr

$$\begin{aligned} Y' &= \text{Round}(219 \cdot y' + 16) \\ Cb &= \text{Round}(224 \cdot pb' + 128) \\ Cr &= \text{Round}(224 \cdot pr' + 128) \end{aligned}$$

with

$$\begin{aligned} y' &= K_R \cdot r' + (1 - K_R - K_B) \cdot g' + K_B \cdot b' \\ pb' &= 0.5 \cdot (b' - y') / (1 - K_B) \\ pr' &= 0.5 \cdot (r' - y') / (1 - K_R) \end{aligned}$$

- The coefficients K_R and K_B are specified by application standards
- ITU-R Rec. BT.709 specifies $K_R = 0.2126$ and $K_B = 0.0722$
- Y' is called **luma** component, Cb and Cr are called **chroma** components

Advantages of YCbCr Color Representation

Properties of the YCbCr color representation

- Similar color decorrelation as in human visual system:
  - Y' component is related to brightness
  - Cb component represents a yellow-blue difference signal
  - Cr component represents a red-green difference signal
- Coding errors are introduced in perceptual meaningful way
- Effectiveness for coding of images and video experimentally verified

color image

Y' component

Cb component

Cr component

Subsampling of Chroma Components

Y channel

Cb channel

Cr channel

Visual importance of luma and chroma components

- YCbCr color space roughly approximates the color decorrelation in the human visual system
- Human visual system is more sensitive to details in luma (brightness) channel than to details in chroma channels
- Chroma channels can be downsampled for saving bit rate
- Chroma downsampling by a factor of 2 in horizontal or both spatial directions
- Location of chroma samples relative to luma samples has to be specified

Demonstration of Luma/Chroma Perception

Compare luma and chroma perception

- Selective low-pass filtering for luma or chroma components
- Use low-pass filter (1,4,6,4,1)/16

Order of presentation

1. Original luma and chroma components
2. Low-pass filtered luma component but original chroma components
3. Original luma and chroma components (repeated)
4. Original luma component but low-pass filtered chroma components

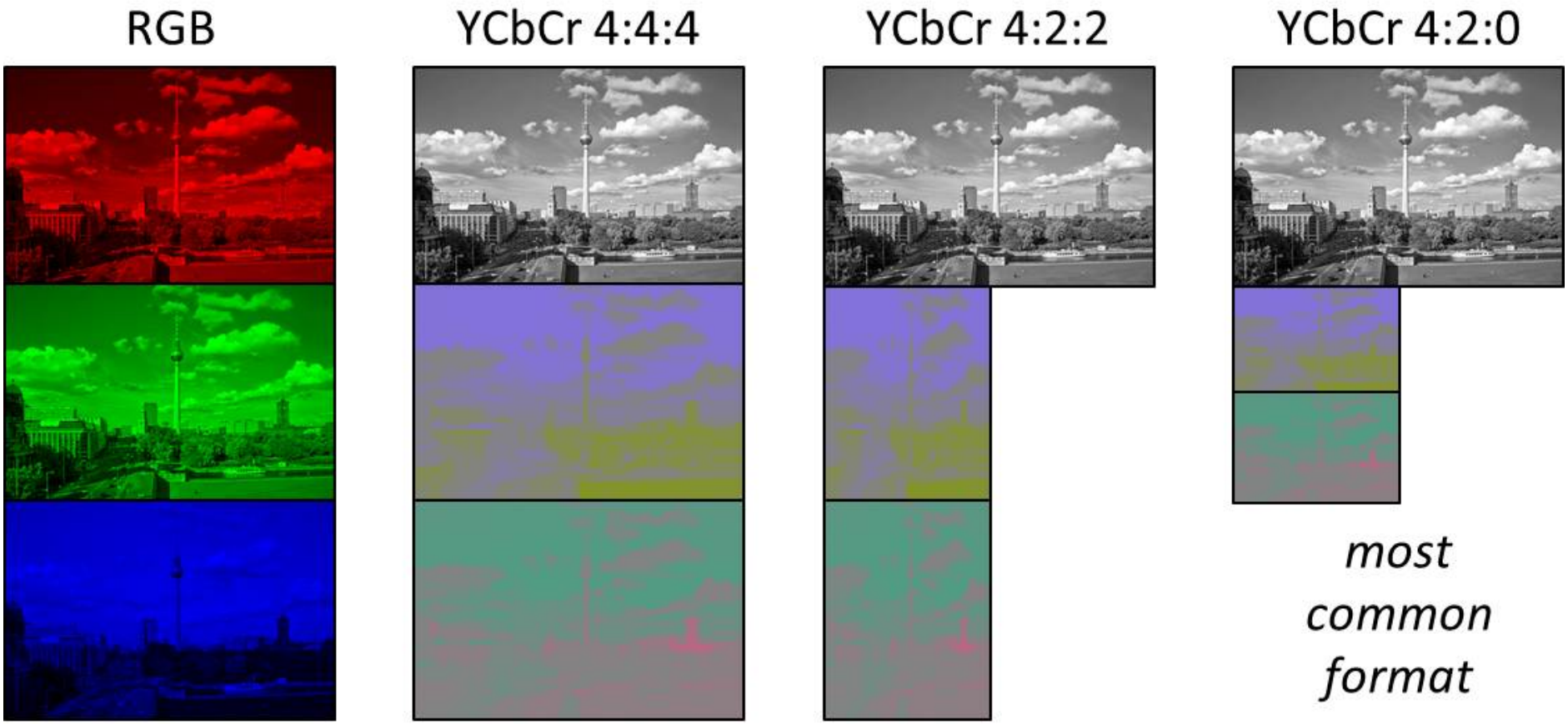
Demonstration: Original Picture

Demonstration: Filtered Luma Component

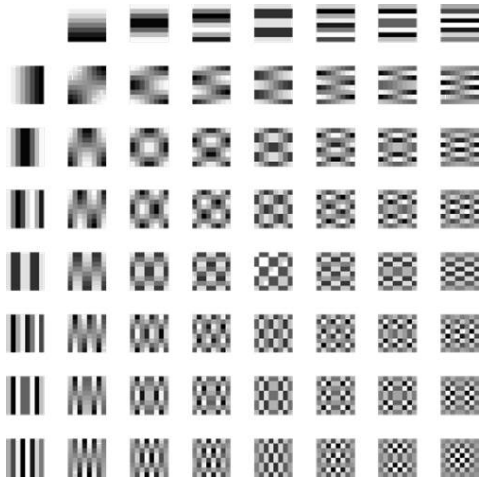
Demonstration: Original Picture (Repeated)

Demonstration: Filtered Chroma Components

Chroma Sampling Formats for Image and Video Coding



JPEG



JPEG – Overview

Joint Photographic Experts Group (JPEG)

- Standard is named after the group which created it
- Joint committee between ITU-T (formerly CCITT) and ISO/IEC JTC 1

Standard "*Digital Compression and Coding of Continuous-Tone Still Images*"

- Officially ITU-T Rec. T.81 and ISO/IEC 10918-1
- Commonly referred to as *JPEG*
- Specifies compression for gray-level and color images
- Work commenced in 1986, standard published in 1992

Applications of JPEG

- Storage format used in virtually all digital cameras (except for "raw" sensor data)
- Most pictures in the Internet are JPEG pictures
- Motion-JPEG is de facto standard for digital video editing

The Scope of Image and Video Coding Standardization

What is standardized?

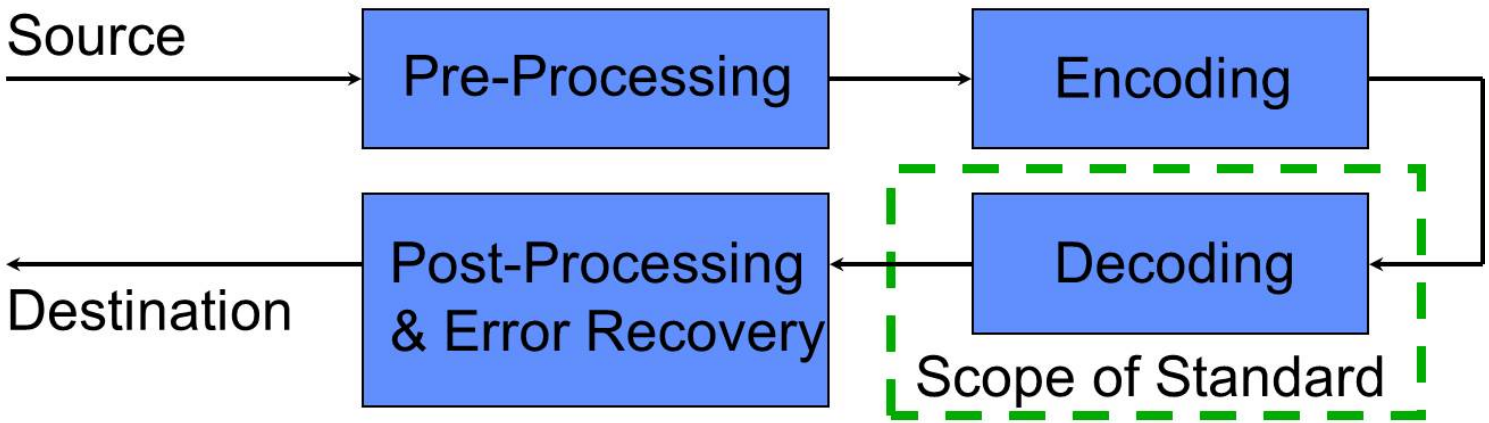
- **Data format** including constraints for the data
- **Decoding result** to be produced by a conforming decoder

⇒ Provides interoperability between different devices

⇒ Permits optimization beyond the obvious

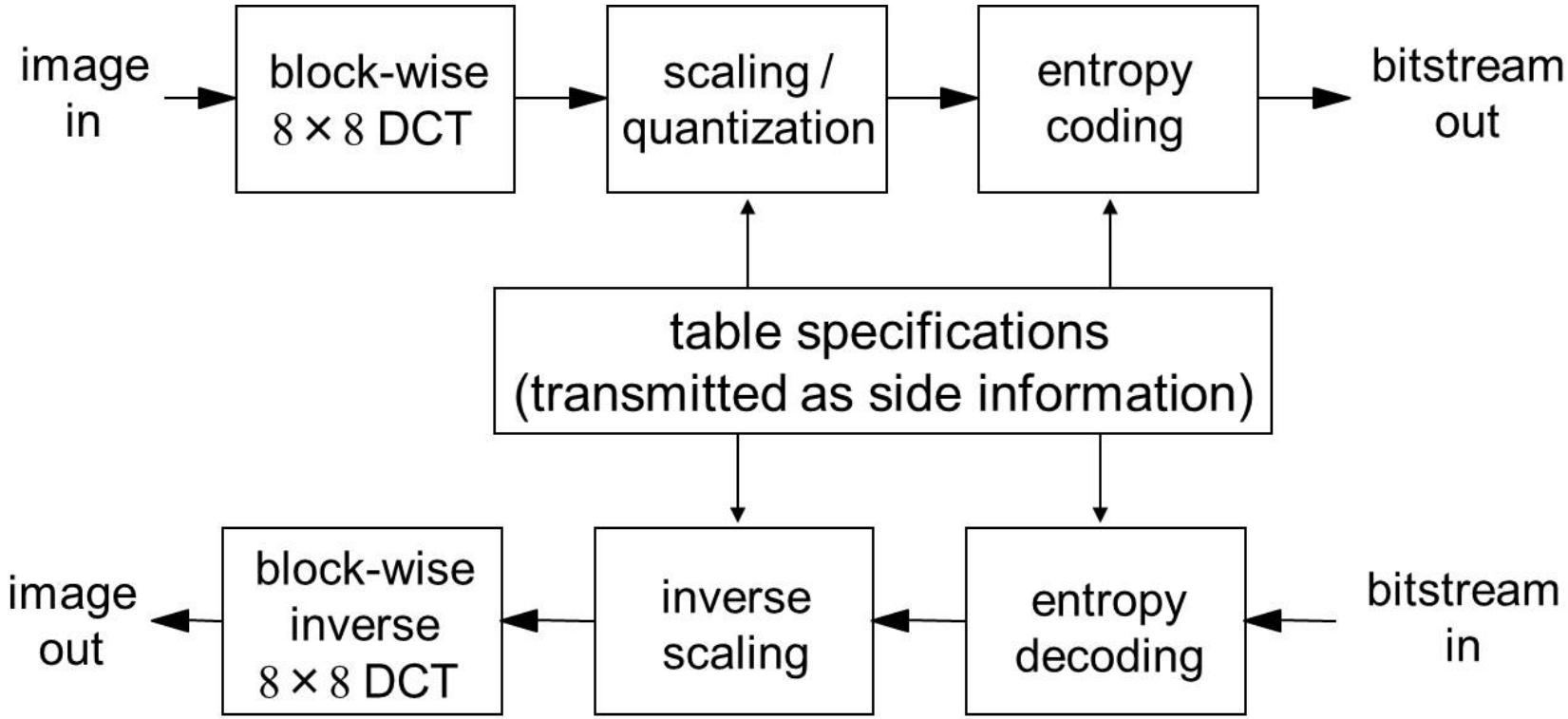
⇒ Permits complexity reduction for implementability

⇒ Provides no guarantee of quality



JPEG: Basic Codec Structure

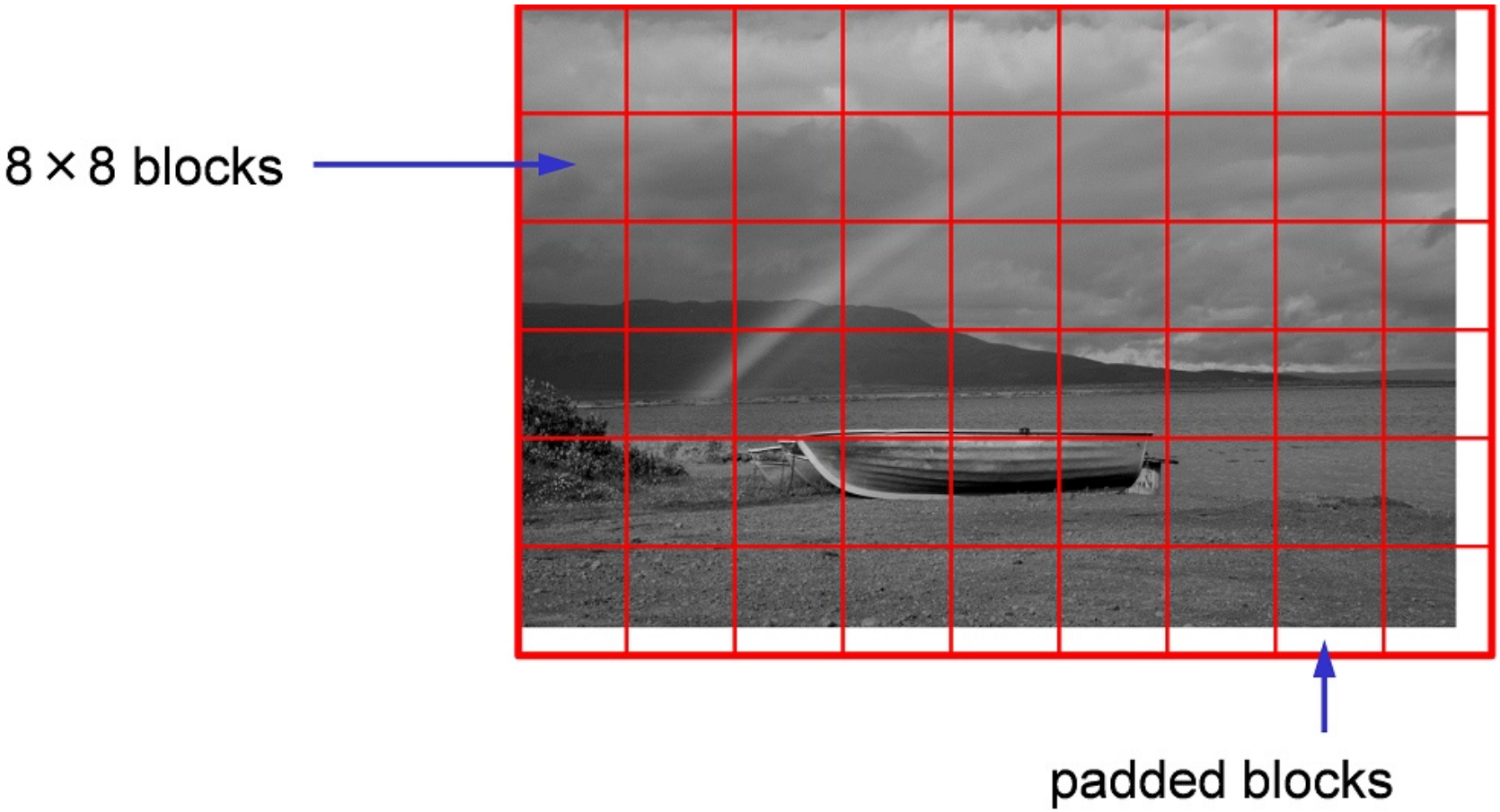
Encoder and decoder structure for each color component



JPEG: Partitioning of Color Components into 8×8 Blocks

Color components are coded independently of each other

- Color components are partitioned into 8×8 blocks (padding at borders)
- The 8×8 blocks are coded using transform coding



Two-dimensional Transform for Image Compression

Separable and symmetric 2-d orthogonal block transform

- 2-d linear transform: Each input block is represented as a linear combination of 2-d basis functions (or basis blocks)
- Separable and symmetric 2-d orthogonal block transform:
  Transform of an $N \times N$ block $\boldsymbol{s}$ can be written as

$$\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s} \cdot \boldsymbol{A}^{\mathrm{T}} \tag{588}$$

  where $\boldsymbol{A}$ is the $N \times N$ transform matrix and $\boldsymbol{u}$ is the $N \times N$ block of transform coefficients
- Inverse of separable and symmetric 2-d orthogonal block transform is given by

$$\boldsymbol{s}' = \boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{u}' \cdot \boldsymbol{A} \tag{589}$$

- Great practical importance:
  Separable transform requires 2 matrix multiplications of size $N \times N$ instead of one multiplication of a vector of size $1 \times N^2$ with a matrix of size $N^2 \times N^2$
  $\Longrightarrow$ Complexity reduction from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$

2-d Transform used in JPEG

2-d block transform in JPEG

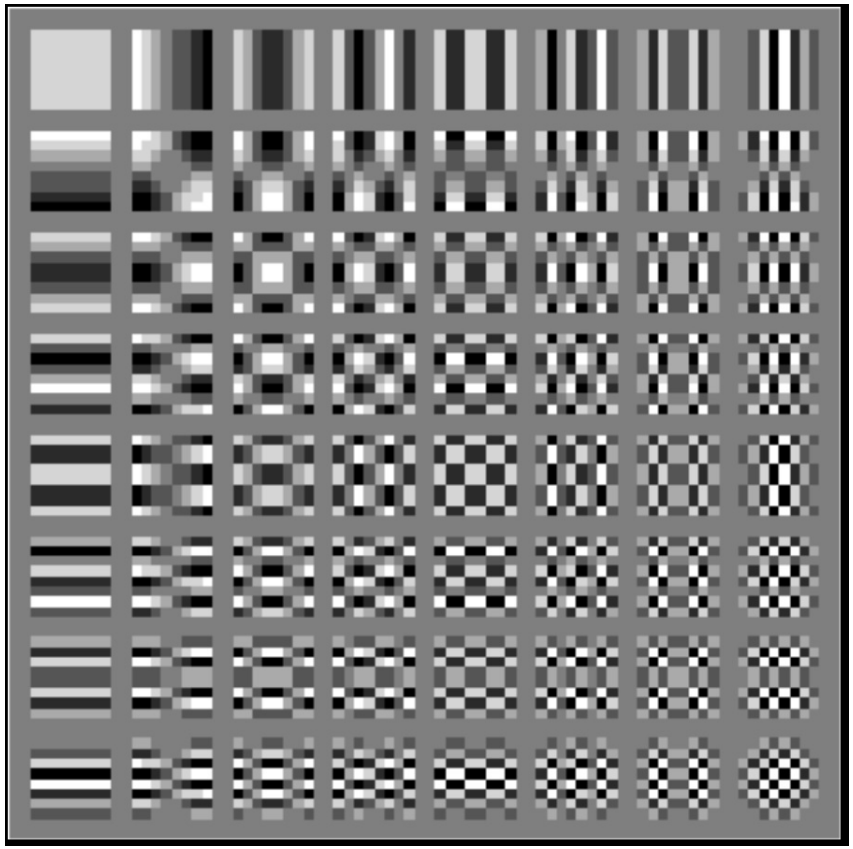
- Separable DCT of type II
- 8×8 transform matrix $\boldsymbol{A}$ consisting of elements a_{ik}, with $i, k = 0, \cdots, 7$, given by

$$a_{ik} = \alpha_i \cos \frac{\pi (2k+1) i}{16} \qquad (590)$$

with

$$\alpha_i = \frac{1}{4} \begin{cases} 1 & : \quad i = 0 \\ \sqrt{2} & : \quad i > 0 \end{cases} \qquad (591)$$

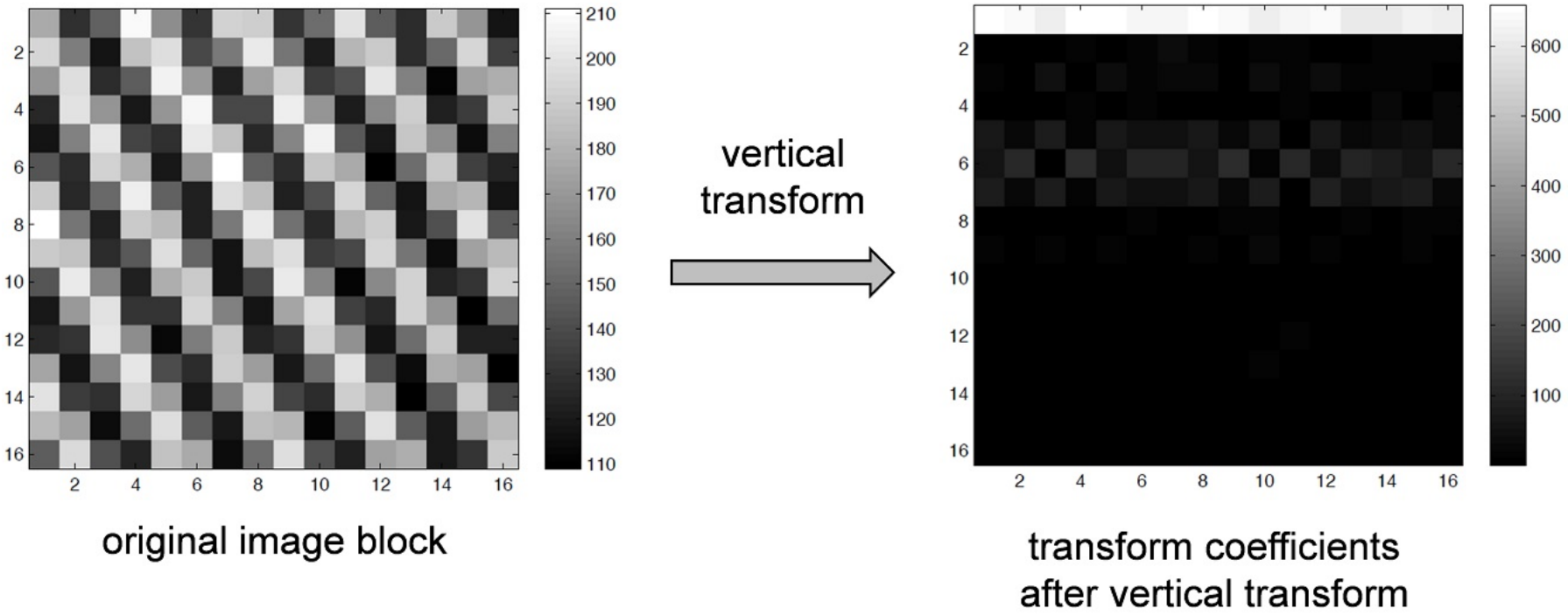
- Transform can be implemented using a fast butterfly algorithm



Transform specification in JPEG

- Ideal forward and backward transform are given in informative clause
- Specification contains normative accuracy requirements

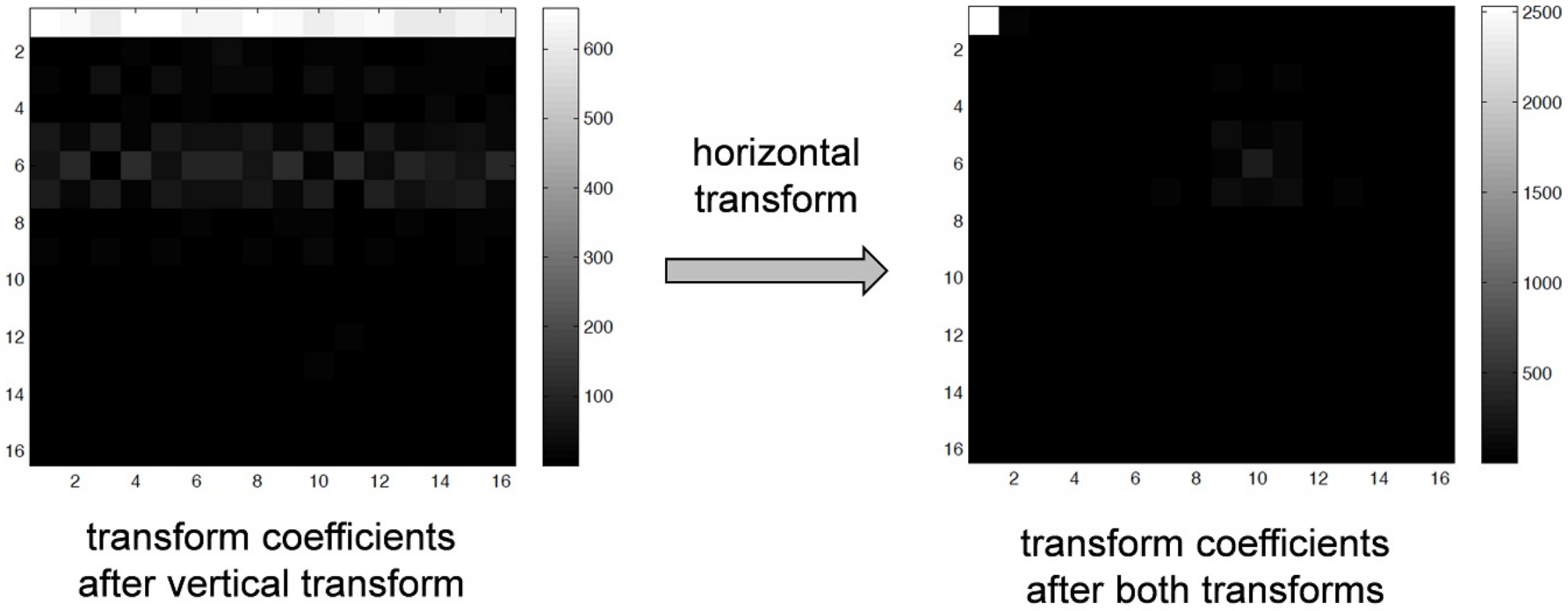
2-d DCT Example – Step 1: Vertical Transform



Example for a 16×16 DCT

- **Step 1**: Column-wise DCT on image block yielding intermediate block of transform coefficients
- Notice the energy concentration in the first row (DC coefficients)

2-d DCT Example – Step 2: Horizontal Transform



transform coefficients
after vertical transform

transform coefficients
after both transforms

Example for a 16×16 DCT

- **Step 2**: Row-wise DCT on intermediate block of transform coefficients yielding the final block of DCT coefficients
- Notice the energy concentration in the DC coefficient (top-left)

Quantization in JPEG

JPEG specifies uniform reconstruction quantizers for the transform coefficients

- Inverse quantization (scaling) in decoder is specified by

$$t'_{ik} = \Delta_{ik} \cdot q_{ik} \tag{592}$$

with
q_{i_k} : Quantization index for coefficient at location (i, k) inside block
$\Delta_{i,k}$: Quantization step size for coefficient at location (i, k)
$t'_{i,k}$: Reconstructed transform coefficient at location (i, k)

- No normative encoding procedure, but informative quantization rule

$$q_{ik} = \text{round}\left(\frac{t_{ik}}{\Delta_{ik}}\right) \qquad (t_{i,k}\text{: original transform coefficient}) \tag{593}$$

- Standard specifies accuracy requirements for combination of DCT and quantization
  $\Longrightarrow$ Leaves freedom for encoder designers
- Separate quantization step sizes can be selected for each coefficient location (i, k) and color component
  $\Longrightarrow$ Quantization tables have to be transmitted as side information (no defaults)
  $\Longrightarrow$ Additional freedom for encoder optimization

Quantization Tables in JPEG

Quantization tables

- Determine rate and distortion (among other parameters)
- Need to be transmitted (no default tables in JPEG)
- Example tables for YCbCr format are specified in Annex K of standard (empirically derived based on psychovisual threshold experiments)

luma blocks

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

chroma blocks

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

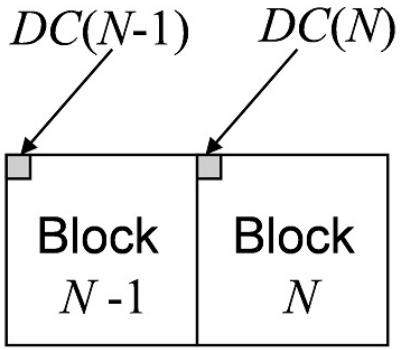
Entropy Coding in JPEG Baseline

Entropy coding of transform coefficient levels (quantization indices)

- Different concepts for DC and AC levels
- DC levels: Differential coding using codeword tables
- AC levels: Run-level coding of scanned coefficients

Coding of DC transform coefficient levels

- DC level is predicted by previous DC level as predictor
- Difference to predictor is coded using VLC and FLC
- Category C is coded using VLC
  - $\Longrightarrow$ Specifies range of values
  - $\Longrightarrow$ Specifies number of following bits (for FLC)
- FLC specifies actual value of *DIFF* inside category C
  - $\Longrightarrow$ If $DIFF > 0$, low-order bits of *DIFF*
  - $\Longrightarrow$ If $DIFF < 0$, low-order bits of $DIFF - 1$
- VLC table for category needs to be transmitted
  - $\Longrightarrow$ Increases side information (no default table)
  - $\Longrightarrow$ Allows adaptation to actual statistics



$DIFF = DC(N) - DC(N-1)$

Example VLC Table for Coding DC Difference Category

Example VLC table for coding category C

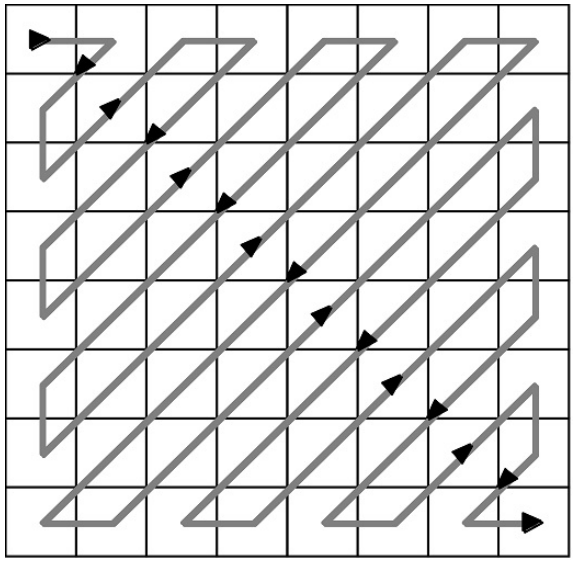
- Range of *DIFF* values is specified in standard
- Codeword assignment has to be transmitted
- Example shows recommended table for luma DC (Annex K of JPEG)

Category C	Range of *DIFF* value	Example codeword
0	0	00
1	-1, 1	010
2	-3, -2, 2, 3	011
3	-7..-4, 4..7	100
4	-15..-8, 8..15	101
5	-31..-16, 16..31	110
6	-63..-32, 32..63	1110
7	-127..-64, 64..127	11110
8	-255..-128, 128..255	111110
9	-511..-256, 256..511	1111110
10	-1023..-512, 512..1023	11111110
11	-2047..-1024, 1024..2047	111111110

Entropy Coding of AC Transform Coefficient Levels

Representation of AC levels

- Convert into sequence using a zig-zag scan
- AC coefficients are likely to be quantized to zero (in particular those at high-frequency locations)
- Successive runs of zeros are represented using a run (number of consecutive levels equal to zero)
- Non-zero AC levels are represented by a category and a value inside the category (same as for DC levels)



Coding of AC levels

- AC levels are coded using a combination of VLC and FLC
- Variable-length code table is used for coding events {run,category}
- VLC table includes a special symbol (EOB) for signaling the end-of-block (all remaining AC levels are equal to zero)
- Fixed-length code is used for coding the exact value inside a category (number of bits is given by category) – same as for DC difference levels
- VLC table has to be transmitted (no default table)

Example VLC Table for Run-Category Coding of AC Levels

Example for VLC table

- Standard defines ranges for categories (same as for DC, but no categories 0 and 11)
- Codeword assignment has to be transmitted
- Example shows first entries of recommended table for luma AC (Annex K of JPEG)

run/category	codeword
EOB	1010
0/1	00
0/2	01
0/3	100
0/4	1011
0/5	11010
0/6	1111000
0/7	11111000
0/8	1111110110
0/9	1111111110000010
0/10	1111111110000011
1/1	1100
1/2	11011
1/3	1111001
1/4	111110110
1/5	11111110110
1/6	1111111110000100
1/7	1111111110000101
1/8	1111111110000110
1/9	1111111110000111
1/10	1111111110001000

run/category	codeword
2/1	11100
2/2	11111001
2/3	1111110111
2/4	111111110100
2/5	1111111110001001
2/6	1111111110001010
2/7	1111111110001011
2/8	1111111110001100
2/9	1111111110001101
2/10	1111111110001110
3/1	111010
3/2	111110111
3/3	111111110101
3/4	1111111110001111
3/5	1111111110010000
3/6	1111111110010001
3/7	1111111110010010
3/8	1111111110010011
3/9	1111111110010100
3/10	1111111110010101
...	...

Example: JPEG Transform Coefficient Level Coding

Example for an 8×8 luma block:

- Last DC level: $DC(N-1) = 178$
- Use recommended luma tables

transform coefficient levels

185	3	1	0	-3	-1	0	0
1	0	-1	0	-2	0	0	0
0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Coding of DC transform coefficient level

- Prediction difference: $DIFF = 185 - 178 = 7$
- Category $C = 3$: Codeword "100"
- Fixed-length code (lowest 3 bits of "7"): "111"
- Final bit representation (6 bit): "100111"

Coding of AC transform coefficient levels

- Zig-zag scanning and conversion into (run,level) pairs yields

  $(0,3)\ (0,1)\ (2,1)\ (1,-1)\ (6,-3)\ (0,-1)\ (0,-2)\ (0,-1)\ (\mathrm{EOB})$

- Representation as (run,category) [FLC bits] sequence

  $(0,2)[11]\ (0,1)[1]\ (2,1)[1]\ (1,1)[0]\ (6,2)[00]\ (0,1)[0]\ (0,2)[01]\ (0,1)[0]\ (\mathrm{EOB})$

- Bit sequence: VLC bits [FLC bits] (in total: 46 bits for 63 AC levels)

  $01\ [11]\ 00\ [1]\ 11100\ [1]\ 1100\ [0]\ 111111110110\ [00]\ 00\ [0]\ 01\ [01]\ 00\ [0]\ 1010$

JPEG Compression Example – Original (YCbCr 4:2:0, 12 bpp)

JPEG Compression Example – 1:10 Compression (1.2 bpp)

JPEG Compression Example – 1:25 Compression (0.48 bpp)

JPEG Compression Example – 1:50 Compression (0.24 bpp)

JPEG Compression Example – 1:100 Compression (0.12 bpp)

JPEG Compression Example – 1:200 Compression (0.06 bpp)

Summary of JPEG

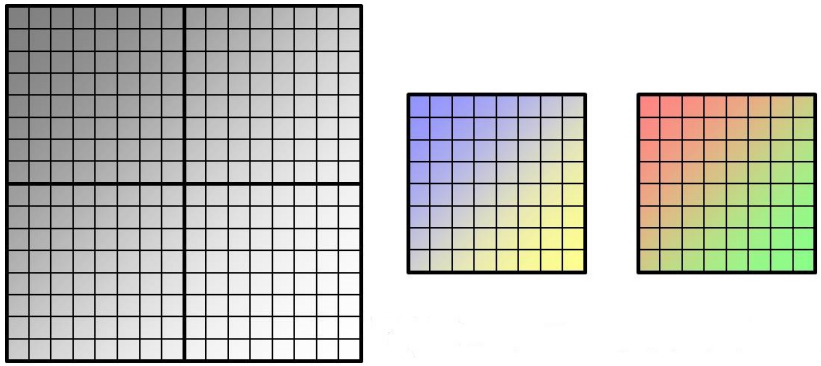
JPEG Baseline

- Minimum of capabilities (required for all DCT-based JPEG codecs)
- Source image: 1-4 color components with 8-bit per sample
- Sequential processing of 8×8 blocks
- Transform: Separable 8×8 discrete cosine transform (DCT) of type II
- Quantizer: Scalar uniform reconstruction quantizer (using quantization table)
- DC coding: Prediction and combination of VLC and FLC
- AC coding: Zig-zag scan and run-level coding (combination of VLC & FLC)
- VLC coding: 2 DC tables (category) & 2 AC tables (run/category)

Extended JPEG features

- Extended bit depth
- Adaptive binary arithmetic coding
- Progressive and hierarchical coding
- Lossless coding mode
- Extended file formats (e.g., EXIF)

Intra-Picture Coding in H.262 | MPEG-2 Video



Intra-Picture Coding in H.262 | MPEG-2 Video

H.262 | MPEG-2 Video

- Video coding standard jointly developed by ITU-T and ISO/IEC JTC 1
- Official name: ITU-T Rec. H.262 and ISO/IEC 13818-2
- Standard was finalized in 1994
- Still widely used in digital television and the DVD-Video optical disc format
- Three pictures types: I (intra), P (predictive) and B (bi-directional)
- Includes tools for interlaced video
- Most important conformance point: Main Profile
  - Color format: YCbCr 4:2:0
  - Bit depth: 8 bit per sample

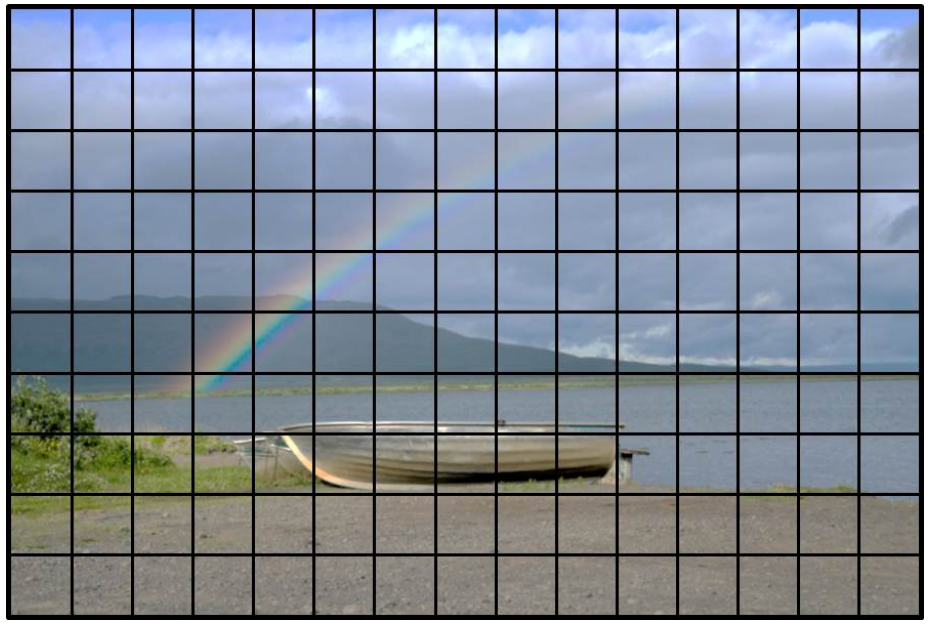
Intra-picture coding in H.262 | MPEG-2 Video

- Conceptually very similar to JPEG Baseline
- Details and actual syntax are different
- Fixed variable-length entropy coding tables

Macroblocks and Blocks in H.262 | MPEG-2 Video

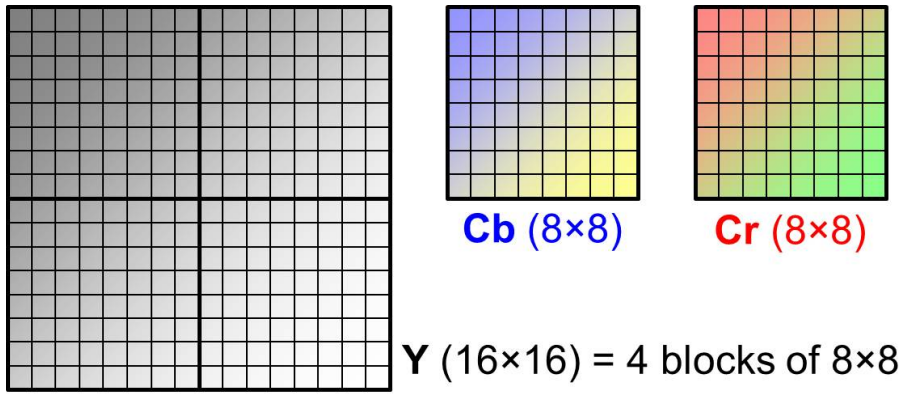
Picture partitioning into macroblocks

- Picture is partitioned into fixed-size **macroblocks**, which consist of 16×16 luma samples and the corresponding areas in the chroma components
- In 4:2:0 chroma sampling format, a macroblock corresponds to
  - one 16×16 luma block
  - two 8×8 chroma blocks

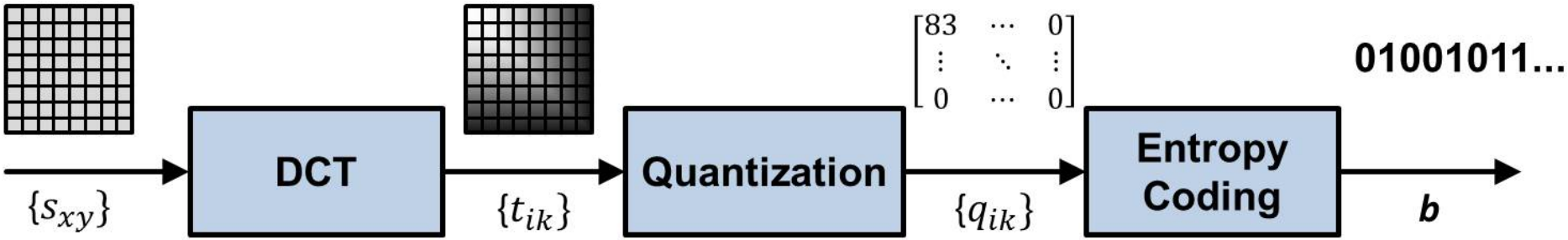


Coding of macroblocks

- Different **coding modes**, also referred to as **macroblock modes**
- Intra picture: 2 coding modes
  - Intra
  - Intra+Q (quantizer change)
- Intra mode: Transform coding for all six 8×8 **blocks** of a macroblock (4 luma and 2 chroma blocks)



Coding of 8×8 Intra Blocks in H.262 | MPEG-2 Video



Transform coding of 8×8 blocks

- Orthogonal block transform + scalar quantization + entropy coding
- Very similar to JPEG (but some differences in details)

Orthogonal block transform

- 2-d discrete coding transform (DCT) – same as in JPEG

Scalar quantization

- Quantization step size is specified by a quantization matrix and a quantization parameter (scaling for all coefficients, can be modified on macroblock basis)

Entropy coding of transform coefficient levels

- DC coefficient: Differential coding similar to JPEG
- AC coefficients: Zig-zag scan and run-level coding with EOB symbol

Quantization in H.262 | MPEG-2 Video

Standard specifies only construction of transform coefficients from levels

- Inverse quantization of intra DC coefficients (intra_dc_precision in range 8..11)

$$t'_{00} = q_{00} \cdot 2^{3-\text{intra\_dc\_precision}} \tag{594}$$

- Inverse quantization of AC coefficients in intra blocks

$$t'_{ik} = \text{sgn}(q_{ik}) \cdot \left\lfloor \frac{|q_{ik}| \cdot w_{ik} \cdot \text{QP}}{16} \right\rfloor \tag{595}$$

with
q_{i_k} : Quantization index for coefficient at location (i, k) inside block
$w_{i,k}$: Entry of quantization matrix for coefficient at location (i, k)
QP : Quantization parameter (also called "quantizer_scale")
$t'_{i,k}$: Reconstructed transform coefficient at location (i, k)

- Inverse quantization is followed by clipping to range $[-2048, 2047]$ and, thereafter, the so-called mismatch control operation

$$t'_{77} = \begin{cases} t'_{77} & : \ s \text{ is odd} \\ t'_{77} - 1 & : \ s \text{ is even and } t'_{77} \text{ is odd} \\ t'_{77} + 1 & : \ s \text{ is even and } t'_{77} \text{ is even} \end{cases} \quad \text{with} \quad s = \sum_{i=0}^{7} \sum_{k=0}^{7} t'_{ik} \tag{596}$$

Quantization Matrices in H.262 | MPEG-2 Video

Quantization matrices

- Quantization matrices define variation of quantizer step sizes among frequencies
  $\Longrightarrow$ Can be used for psychovisual optimization (to some extend)
- Quantization parameter QP is used for scaling the quantization matrices
  $\Longrightarrow$ Operation point can be modified on macroblock basis with a few bits
- For 4:2:0 data, two matrices are used: one for intra and one for non-intra
- Default quantization matrices can be replaced by user-defined matrices

default matrix for intra blocks

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

default matrix for non-intra blocks

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Coding of Transform Coefficients Levels for Intra 8×8 Blocks

DC transform coefficient level in intra blocks

- Very similar to JPEG
- Prediction using last coded DC level (reset at start of slice or non-intra MB)
- Difference is coded by a category (called dct_dc_size) a fixed-length code
- Entropy coding table is fixed in standard (cannot be modified)

AC transform coefficient levels

- Levels of a block are converted into vector using a zig-zag scan (same as in JPEG)
- Vector of levels is coded using **run-level code**
  - Entropy coding table for most frequent combinations of run and level (actual levels including sign, not categories as in JPEG)
  - Includes **end-of-block** (EOB) symbol
  - Includes **escape** symbol for less likely combinations, for which the actual run and level are transmitted with 6 and 12 bit, respectively
- Entropy coding tables are fixed in standard (cannot be modified)
- For intra, one of two defined tables can be selected on a picture basis

Encoder Control for Intra Pictures in H.262 | MPEG-2 Video

What parameters can be chosen in encoder?

- On sequence/picture level: Quantization matrix, intra vlc table, intra DC precision
- On macroblock level: Quantization parameter QP
- On block level: Transform coefficient levels

$\Longrightarrow$ **Transform coefficient levels have largest impact on coding efficiency**

Do the parameters of different blocks influence each other?

- Only last DC coefficient is used for prediction
- Typically very small impact on coding efficiency

$\Longrightarrow$ **Interdependencies between blocks can be neglected**

How can the selection of transform coefficient levels be optimized?

- Selection determines distortion and rate!
- Rounding to next level minimizes distortion, but typically produces a large rate
- For an operation point given by a Lagrange parameter λ, the combined cost measure $D + \lambda \cdot R$ should be minimized
- Not straightforward due to run-level coding

$\Longrightarrow$ **Need to consider dependencies in coding of successive levels**

$\Longrightarrow$ **Fixed decision levels cannot be optimal**

Example: Impact of Considering Rate in Quantization

Quantization example with $\Delta = 10$ and $\lambda = 10$

- Consider quantization of the following vector of transform coefficients

36	18	23	3	12	-4	-3	2	-6	0	···	0

- Rounding to nearest quantization level according to $q = \text{round}(c/\Delta)$ yields

4	2	2	0	1	0	0	0	-1	0	···	0

$\Longrightarrow$ Sequence of (run,level) values: (0,4)(0,2)(0,2)(1,1)(3,-1)(EOB)

$\Longrightarrow$ Bit sequence: (00001100)(01000)(01000)(0110)(001111)(10)

$\Longrightarrow$ Distortion $D = 87$, rate $R = 30 \quad \Longrightarrow \quad J = D + \lambda \cdot R = 387$

- Alternative quantization (considering rate by minimizing $J = D + \lambda \cdot R$)

4	2	2	0	1	0	0	0	0	0	···	0

$\Longrightarrow$ Sequence of (run,level) values: (0,4)(0,2)(0,2)(1,1)(EOB)

$\Longrightarrow$ Bit sequence: (00001100)(01000)(01000)(0110)(10)

$\Longrightarrow$ Distortion $D = 107$, rate $R = 24 \quad \Longrightarrow \quad J = D + \lambda \cdot R = 347 \; (< 387)$

Rate-Distortion Optimized Quantization (RDOQ)

General idea of rate-distortion optimized quantization for run-level coding

- Evaluate "all possible" vectors of transform coefficient levels
- Choose vector that minimizes $J = D + \lambda \cdot R$, with
  - D: Distortion for block (can be measured in transform domain)
  - R: Rate for transmitting levels using given entropy coding tables
  - λ: Lagrange parameter, e.g., given as function of quantization parameter QP

$\Longrightarrow$ Very complex: Require restriction for levels and suitable algorithm

Select reasonable set of potential levels for each coefficient

- Following observations can be made (considering absolute levels)
  - Levels greater than the level obtained by mathematically correct rounding don't need to be considered (larger distortion and larger rate)
  - Levels that are significantly smaller than the level obtained by mathematically correct rounding don't need to be considered (very large distortion)
- Good compromise is obtained by following set of 1-3 levels (absolute values)
  - Level q_0 obtained by mathematically correct rounding, $q_0 = \mathrm{round}(c/\Delta)$
  - Level q_1 obtained by rounding towards zero, $q_1 = \lfloor c/\Delta \rfloor$
  - If $q_1 > 1$, level $q_2 = q_1 - 1$

RDOQ Algorithm for Run-Level Coding

Start with first position $k = 0$ in scanning order

- For all potential levels $\{q_0\}$ determine
  - distortion $D_0 = (c_0 - q_0 \cdot \Delta)^2$
  - for non-zero levels q_0, rate R_0 using run-level coding tables
- Among all non-zero levels q_0, keep only the one that minimizes $J_0 = D_0 + \lambda \cdot R_0$
- At most two candidate vectors $\boldsymbol{q}_0 = [q_0]$ are considered for the further steps, one with $q_0 = 0$ and one with $q_0 \neq 0$

For each of the remaining positions k in scanning order

- Combine all potential levels $\{q_k\}$ for current position k with the selected candidate vectors $\boldsymbol{q}_{k-1}$ to new candidate vectors $\boldsymbol{q}_k$ and determine
  - distortion $D_k = D_{k-1} + (c_k - q_k \cdot \Delta)^2$
  - for non-zero levels q_k, rate R_k using run-level coding tables
- Among all vectors $\boldsymbol{q}_k$ with the last level q_k unequal to 0, discard all vectors except the one that minimizes $D_k + \lambda \cdot R_k$

After last scanning position k

- Determined the final cost $J = D + \lambda \cdot R$ for all remaining candidate vectors $\boldsymbol{q}$ (including EOB symbol) and choose the one that minimizes J

RDOQ Example

Simple RDOQ example using quantization step size $\Delta = 10$ and $\lambda = 10$

- Consider transform coefficient vector of 9 coefficients

36	18	23	3	12	-4	-3	2	-6

- Coefficient "36" at position $k = 0$ with potential levels $\{4, 3, 2\}$:
  - $\Rightarrow [\mathbf{4}]$: $D = 4^2 = 16$, $R = 8$ $\rightarrow$ $J = 96$
  - $\Rightarrow [\mathbf{3}]$: $D = 6^2 = 36$, $R = 6$ $\rightarrow$ $J = 96$ [discard]
  - $\Rightarrow [\mathbf{2}]$: $D = 16^2 = 256$, $R = 5$ $\rightarrow$ $J = 306$ [discard]
- Coefficient "18" at position $k = 1$ with potential levels $\{2, 1, 0\}$:
  - $\Rightarrow [4, \mathbf{2}]$: $D = 16 + 2^2 = 20$, $R = 8 + 5 = 13$ $\rightarrow$ $J = 150$
  - $\Rightarrow [4, \mathbf{1}]$: $D = 16 + 8^2 = 80$, $R = 8 + 3 = 11$ $\rightarrow$ $J = 190$ [discard]
  - $\Rightarrow [4, \mathbf{0}]$: $D = 16 + 18^2 = 340$, $R = 8$ (does not include last zero)
- Coefficient "23" at position $k = 2$ with potential levels $\{2, 1\}$:
  - $\Rightarrow [4, 2, \mathbf{2}]$: $D = 20 + 3^2 = 29$, $R = 13 + 5 = 18$ $\rightarrow$ $J = 209$
  - $\Rightarrow [4, 2, \mathbf{1}]$: $D = 20 + 13^2 = 189$, $R = 13 + 3 = 16$ $\rightarrow$ $J = 349$ [discard]
  - $\Rightarrow [4, 0, \mathbf{2}]$: $D = 340 + 3^2 = 349$, $R = 8 + 7 = 15$ $\rightarrow$ $J = 499$ [discard]
  - $\Rightarrow [4, 0, \mathbf{1}]$: $D = 340 + 13^2 = 509$, $R = 8 + 4 = 12$ $\rightarrow$ $J = 629$ [discard]
- Coefficient "3" at position $k = 3$ with potential levels $\{0\}$:
  - $\Rightarrow [4, 2, 2, \mathbf{0}]$: $D = 29 + 3^2 = 38$, $R = 18$ (without trailing zeros)

RDOQ Example (continued)

- Coefficient "12" at position $k = 4$ with potential levels $\{1, 0\}$:
  - $\Rightarrow [4, 2, 2, 0, \mathbf{1}]$: $D = 38 + 2^2 = 42$, $R = 18 + 4 = 22 \quad \rightarrow \quad J = 262$
  - $\Rightarrow [4, 2, 2, 0, \mathbf{0}]$: $D = 38 + 12^2 = 182$, $R = 18$ (without trailing zeros)
- Coefficient "-4" at position $k = 5$ with potential levels $\{0\}$:
  - $\Rightarrow [4, 2, 2, 0, 1, \mathbf{0}]$: $D = 42 + 4^2 = 58$, $R = 22$ (without trailing zeros)
  - $\Rightarrow [4, 2, 2, 0, 0, \mathbf{0}]$: $D = 182 + 4^2 = 198$, $R = 18$ (without trailing zeros)
- Coefficient "-3" at position $k = 6$ with potential levels $\{0\}$:
  - $\Rightarrow [4, 2, 2, 0, 1, 0, \mathbf{0}]$: $D = 58 + 3^2 = 67$, $R = 22$ (without trailing zeros)
  - $\Rightarrow [4, 2, 2, 0, 0, 0, \mathbf{0}]$: $D = 198 + 3^2 = 207$, $R = 18$ (without trailing zeros)
- Coefficient "2" at position $k = 7$ with potential levels $\{0\}$:
  - $\Rightarrow [4, 2, 2, 0, 1, 0, 0, \mathbf{0}]$: $D = 67 + 2^2 = 71$, $R = 22$ (without trailing zeros)
  - $\Rightarrow [4, 2, 2, 0, 0, 0, 0, \mathbf{0}]$: $D = 207 + 2^2 = 211$, $R = 18$ (without trailing zeros)
- Last coefficients "-6" with potential levels $\{-1, 0\}$ (including 2 bits for EOB):
  - $\Rightarrow [4, 2, 2, 0, 1, 0, 0, 0, \mathbf{-1}]$: $D = 71 + 4^2 = 87$, $R = 22 + 8 = 30 \quad \rightarrow \quad J = 387$
  - $\Rightarrow [4, 2, 2, 0, 1, 0, 0, 0, \mathbf{0}]$: $D = 71 + 6^2 = 107$, $R = 22 + 2 = 24 \quad \rightarrow \quad J = 347$
  - $\Rightarrow [4, 2, 2, 0, 0, 0, 0, 0, \mathbf{-1}]$: $D = 211 + 4^2 = 227$, $R = 18 + 9 = 27 \quad \rightarrow \quad J = 497$
  - $\Rightarrow [4, 2, 2, 0, 0, 0, 0, 0, \mathbf{0}]$: $D = 211 + 6^2 = 247$, $R = 18 + 2 = 20 \quad \rightarrow \quad J = 447$

$\Longrightarrow$ Selected transform coefficient levels: $[4, 2, 2, 0, 1, 0, 0, 0, 0]$ with $J = 347$

$\Longrightarrow$ For comparison, rounding would yield: $[4, 2, 2, 0, 1, 0, 0, 0, -1]$ with $J = 387$

Experimental Analysis of RDOQ for Intra-Picture Coding

Coding experiment with H.262 | MPEG-2 Video comparing

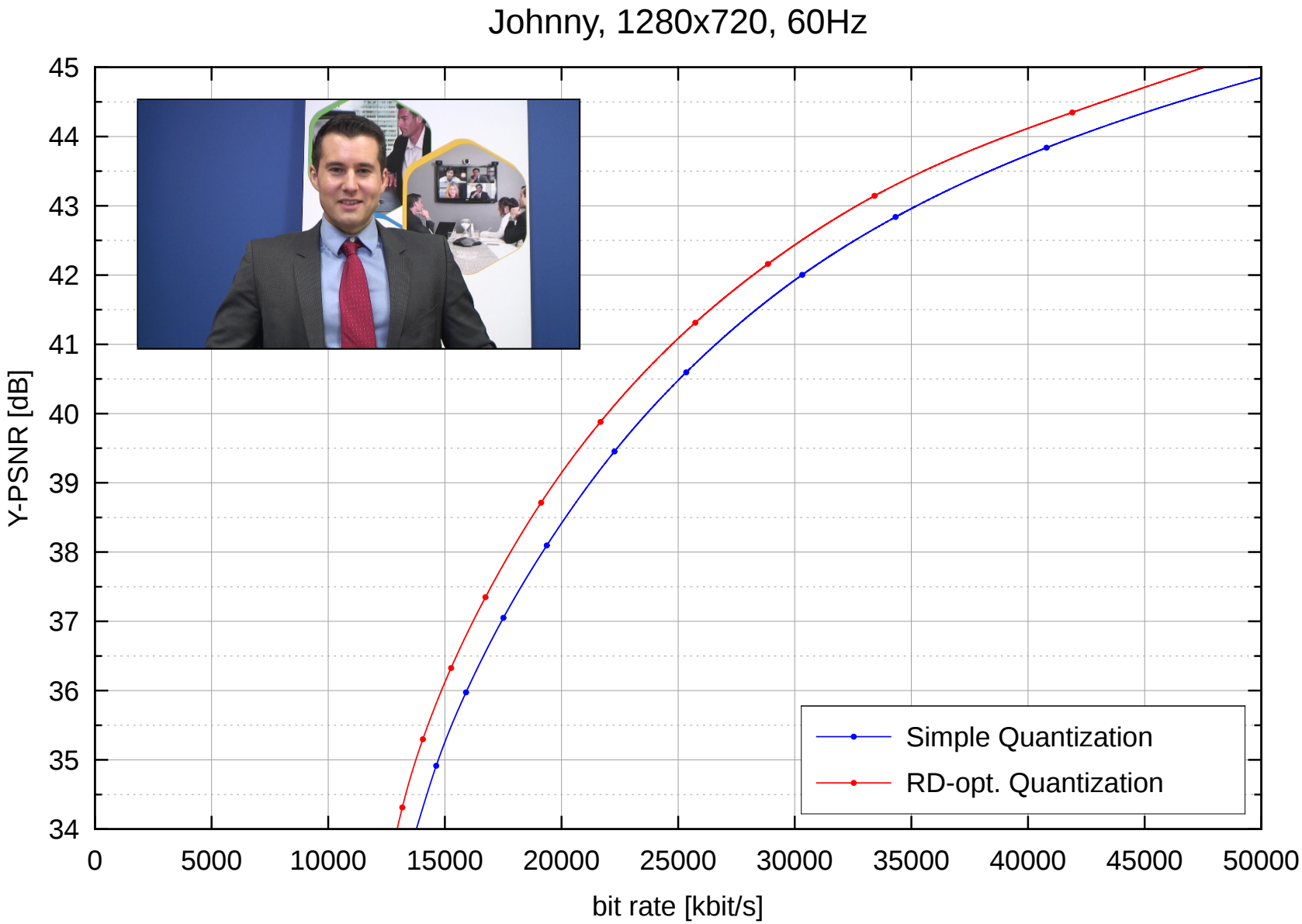
- Encoder with simple quantization using mathematically correct rounding
- Encoder using rate-distortion optimized quantization

Coding conditions

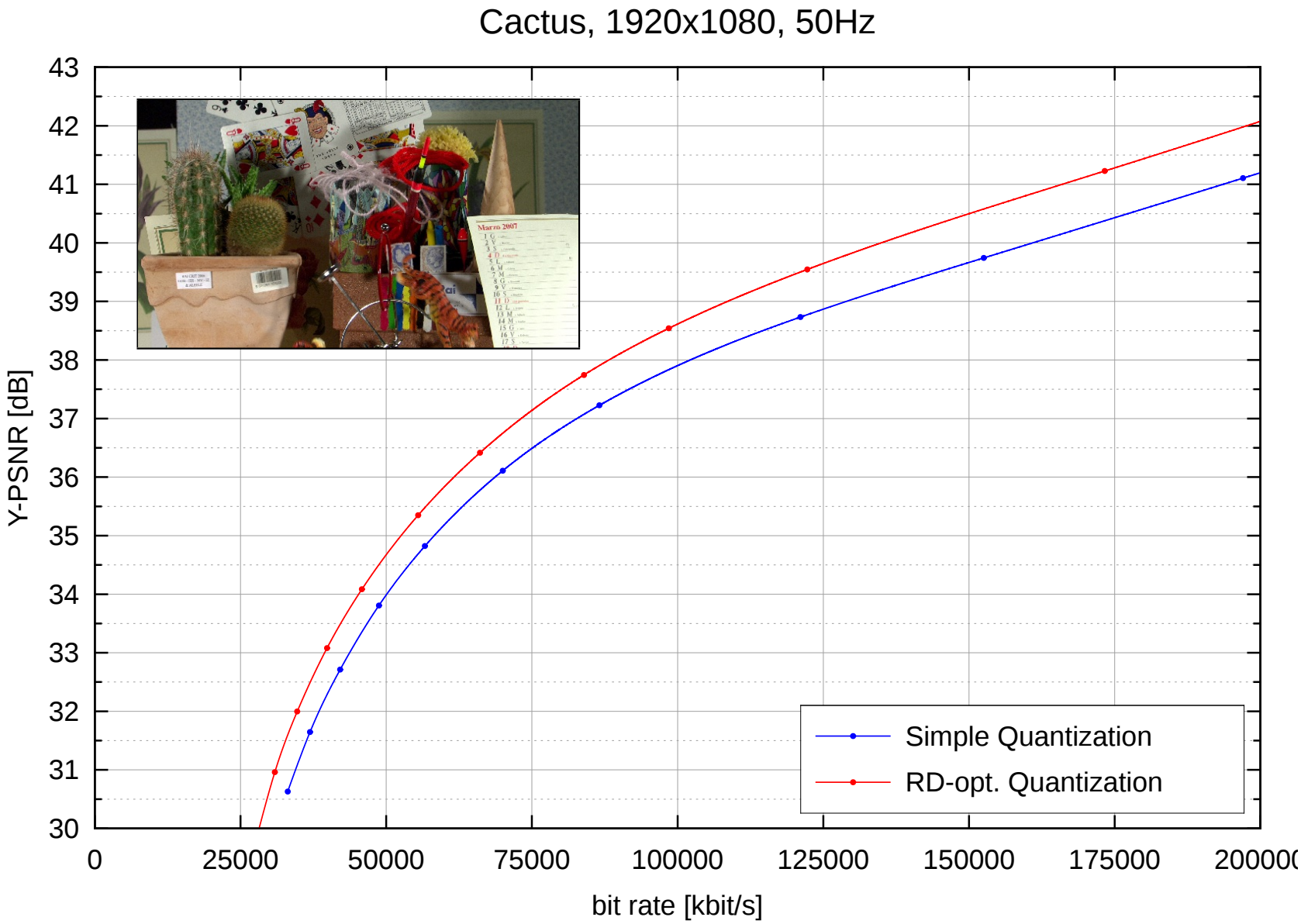
- 6 video conferencing sequences with a resolution of 1280×720
- 6 more complex video sequences with a resolution of 1920×1080
- 10 pictures of each sequence have been coded (intra only)
- Flat quantization matrices (since quality is measured using PSNR)
- Same quantization parameter for all macroblocks
- Bitstreams with different quantization parameters
- PSNR and rate have been measured
- Lagrange parameter λ has been coupled to quantization step size Δ using the experimentally determined relationship

$$\lambda = \text{const} \cdot \Delta^2$$

Quantization Comparison – Sequence "Johnny"



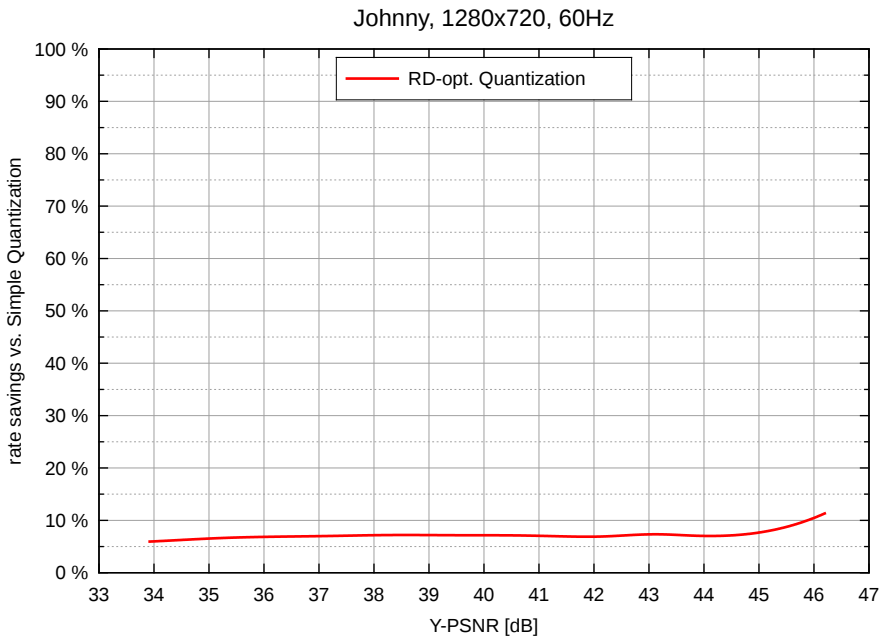
Quantization Comparison – Sequence "Cactus"

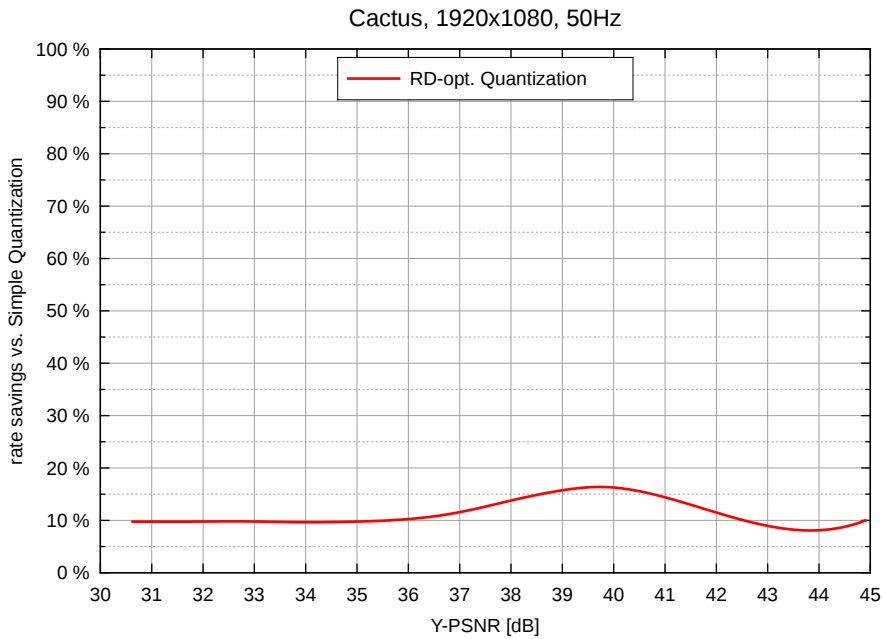


Quantization Comparison – Summary

Bit-rate savings of RDOQ versus simple quantization

- Bit-rate saving at a PSNR value is obtained by interpolating the r-d curves
- Average bit-rate savings are obtained by averaging the savings for 100 PSNR values
- Average bit-rate saving for all sequences: 9%





Summary of Intra-Picture Coding in H.262 | MPEG-2 Video

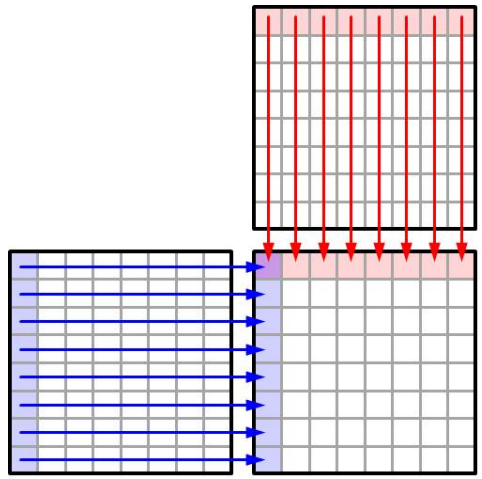
Intra-picture coding in H.262 | MPEG-2 Video

- Video coding standard of ITU-T and ISO/IEC JTC 1
- Still widely used in digital television and DVD-Video
- Intra-picture coding is very similar to JPEG Baseline
- Picture partitioning in macroblocks and blocks
- Scalar quantization of transform coefficients
- Quantization matrices are combined with quantization parameter
- Run-level coding for transform coefficients

Rate-distortion optimized quantization

- Quantization with fixed decision levels cannot be optimal due to dependencies in run-level entropy coding
- Actual rate for coding transform coefficient levels need to be considered
- Optimal quantization can be achieved by a trellis-like procedure
- Rate-distortion optimized quantization yields average bit-rate savings of about 10% relative to simple rounding (for intra-only coding)

Intra-Picture Coding in H.263 and MPEG-4 Visual



ITU-T Recommendation H.263 – Overview

ITU-T Recommendation H.263

- Video coding standard of ITU-T
- Developed by Visual Coding Experts Group (VCEG – ITU-T SG16/WP3/Q6)
- Primarily designed for low bit-rate video conferencing
- Example applications:
  - Video conferencing
  - Was used for Flash Video content
  - RealVideo codec (before RealVideo 8) was based on H.263
- First version (1995)
  - Very similar structure as H.262 | MPEG-2 Video
  - Several improvements relative to H.262 | MPEG-2 Video
  - For intra: Run-level-last coding and optimized coding tables
- Second version H.263+ (1998)
  - Improvements and new features
  - For intra: AC level prediction, adaptive scans, specialized quantization and entropy coding tables for intra blocks
- Third version H.263++ (2000)
  - Multiple reference pictures for motion-compensated coding

Intra-Picture Coding in H.263 Baseline

Basic design for intra-picture coding – similar to H.262 | MPEG-2 Video

- Partitioning into 16×16 macroblocks as in H.262 | MPEG-2 Video
- Transform coding of 8×8 blocks (DCT + scalar quantization + entropy coding)

Orthogonal block transform

- Same separable 8×8 DCT as in H.262 | MPEG-2 Video and JPEG

Scalar quantization of transform coefficients (only reconstruction is specified)

- Step size determined by quantization parameter QP (can be modified on MB basis)
- DC coefficient (uniform reconstruction quantizer)

$$t'_{00} = 8 \cdot q_{00} \tag{597}$$

- AC coefficients (uniform reconstruction quantizer with extra-wide deadzone)

$$t'_{ik} = \begin{cases} 0 & : \quad q_{ik} = 0 \\ \mathrm{sgn}(q_{ik}) \cdot \mathrm{QP} \cdot (2 \cdot |q_{ik}| + 1) & : \quad q_{ik} \neq 0 \text{ and QP is odd} \\ \mathrm{sgn}(q_{ik}) \cdot \mathrm{QP} \cdot (2 \cdot |q_{ik}| + 1) - 1 & : \quad q_{ik} \neq 0 \text{ and QP is even} \end{cases} \tag{598}$$

$\Longrightarrow$ Reconstructed levels are always odd-valued numbers (except zero)

$\Longrightarrow$ Has been found to prevent accumulation of IDCT mismatches (for inter)

Coding of Transform Coefficient Levels in H.263 Baseline

Coded block pattern (CBP)

- Signal which 8×8 blocks of a macroblock contain non-zero levels (AC levels)
- Concept was also used in H.262 | MPEG-2 Video, but only for inter macroblocks
- Required for run-level-last coding of AC levels
- In H.263, split into two components:
  - CBPC (two bits for the chroma blocks): Coded together with MB type
  - CBPY (four bits for the luma blocks): Coded as separate codeword

MB type	CBPC	codeword
Intra	00	1
Intra	01	001
Intra	10	010
Intra	11	011
Intra+Q	00	0001
Intra+Q	01	0000 01
Intra+Q	10	0000 10
Intra+Q	11	0000 11
stuffing	–	0000 0000 1

CBPY	codeword
0000	0011
0001	0010 1
0010	0010 0
0011	1001
0100	0001 1
0101	0111
0110	0000 10
0111	1011

CBPY	codeword
1000	0001 0
1001	0000 11
1010	0101
1011	1010
1100	0100
1101	1000
1110	0110
1111	11

Coding of DC transform coefficient level

- No prediction of DC coefficient (in contrast to JPEG and H.262 | MPEG-2 Video)
- Fixed-length code: 8 bit per DC level

Coding of AC Transform Coefficient Levels in H.263 Baseline

Coding of AC transform coefficient levels

- Convert matrix of AC levels to vector using the zig-zag scan
- Vector of AC levels is coded using **run-level-last code**
- Entropy coding table for most common combinations of
  - Run: Number of preceding levels equal to zero
  - Level: Value of the next non-zero level
  - Last: Flag indicating if the non-zero level is the last non-zero level in block
- Entropy coding table includes an **escape** symbol for less likely combinations, for which the "run", "level" and "last" are coded using fixed-length codes (6+8+1 bit)
- Entropy coding tables have been optimized for low rates

last	run	level	codeword (s = sign)
0	0	± 1	10s
0	0	± 2	1111 s
0	0	± 3	0101 01s
0	0	± 4	0010 111s
0	0	± 5	0001 1111 s
0	0	± 6	0001 0010 1s
0	0	± 7	0001 0010 0s
0	0	± 8	0000 1000 01s
0	0	± 9	0000 1000 00s
0	0	± 10	0000 0000 111s
0	0	± 11	0000 0000 110s
0	0	± 12	0000 0100 000s

last	run	level	codeword (s = sign)
...	...	...	...
1	31	± 1	0000 0100 110s
1	32	± 1	0000 0100 111s
1	33	± 1	0000 0101 1000 s
1	34	± 1	0000 0101 1001 s
1	35	± 1	0000 0101 1010 s
1	36	± 1	0000 0101 1011 s
1	37	± 1	0000 0101 1100 s
1	38	± 1	0000 0101 1101 s
1	39	± 1	0000 0101 1110 s
1	40	± 1	0000 0101 1111 s
	escape		0000 011

Advanced Intra Coding Mode in Annex I of H.263+

Advanced intra-picture coding mode (in optional Annex I) specifies

- Adaptive prediction of transform coefficients in intra blocks
- Adaptive scanning of transform coefficient levels (depending on prediction)
- Modified inverse quantization for intra
- Separate entropy coding table for intra blocks (optimized for intra)

Adaptive prediction and scanning

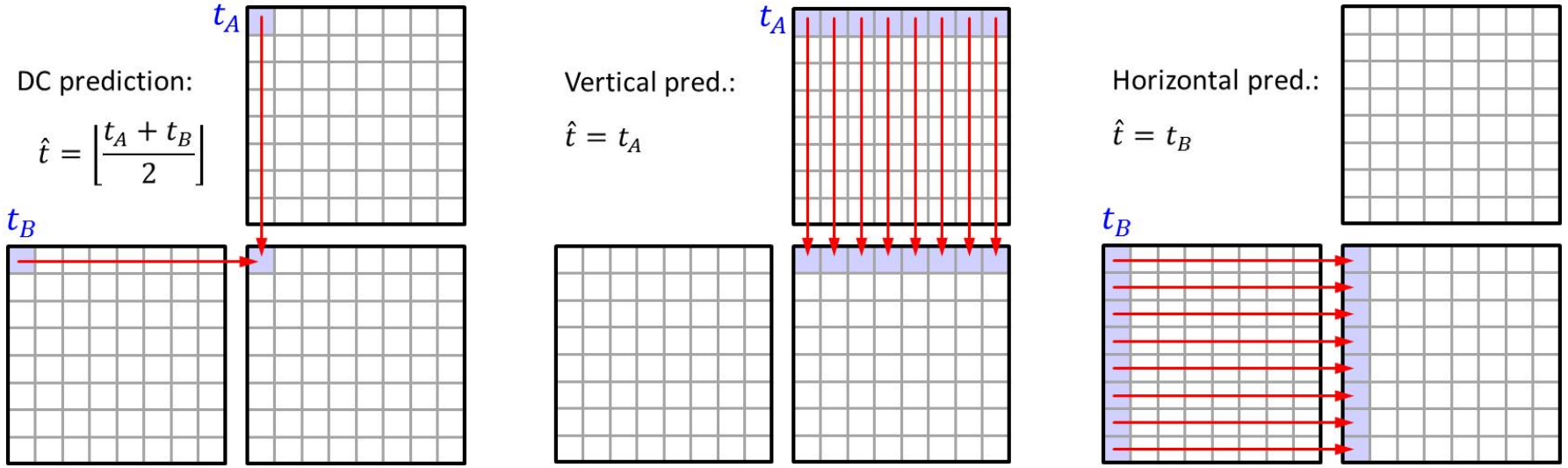
- Predict part of the transform coefficients using reconstructed transform coefficients of neighboring blocks of same color component
- Quantization and entropy coding of prediction residuals $t_{ik} - \hat{t}_{ik}$

$$t'_{ik} = \hat{t}_{ik} + Q^{-1}(q_{ik}) \qquad (Q^{-1}\text{: inverse quantization}) \qquad (599)$$

- 3 prediction modes (signaled at macroblock level)
  - DC prediction: Predict DC using left and above neighboring block
  - Vertical prediction: Predict first row of coefficients using above block
    $\Longrightarrow$ Particularly suitable for vertical structures
  - Horizontal prediction: Predict first column of coefficients using left block
    $\Longrightarrow$ Particularly suitable for horizontal structures

Prediction Modes for Intra Blocks in Advanced Intra Coding

Prediction of transform coefficients: 3 modes signaled at macroblock level



Modified quantization for intra macroblocks

- Use same quantization for all coefficients (including DC coefficient)
- Uniform reconstruction quantizer without extra-wide deadzone and without mismatch control (only important for inter macroblocks)

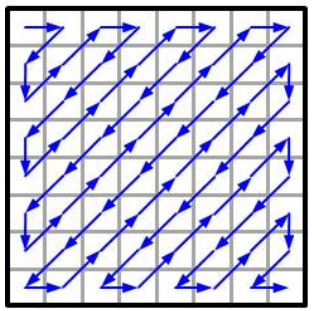
$$t'_{ik} = 2 \cdot \mathrm{QP} \cdot q_{ik}, \qquad \forall i, k = 0..7 \tag{600}$$

Entropy Coding in Advanced Intra Coding Mode

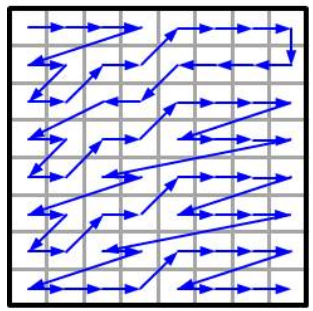
Scanning pattern is chosen based on prediction mode

- Goal: Concentrate zero levels at end of scanning pattern
- Non-zero coefficient distribution is depending on prediction mode
  - DC prediction: Conventional zig-zag scan
  - Vertical prediction: Horizontal scan (suitable for vertical structures)
  - Horizontal prediction: Vertical scan (suitable for horizontal structures)

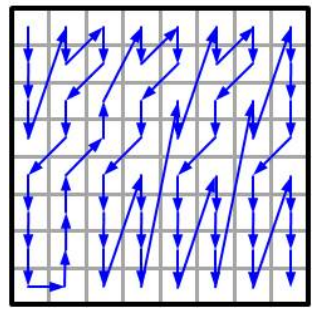
zig-zag scan
(DC pred.)



horizontal scan
(vertical pred.)



vertical scan
(horizontal pred.)



Coding of vector of transform coefficient levels

- No separate coding of DC coefficient $\Longrightarrow$ included in run-level-last code
- Entropy coding table for run-level-last code optimized for intra statistics

Further Extensions in H.263 for Improving Intra Coding

Annex E: Syntax-based arithmetic coding

- Specifies non-adaptive arithmetic coding for syntax elements
- Rarely used in practice

Annex J: Deblocking filter mode

- Specifies deblocking filter for reducing block-edge artifacts
- Strength of smoothing filter is controlled by quantization parameter
- Most useful for coding of following inter pictures, but also improves quality of intra pictures at low rates

Annex T: Modified quantization mode

- Improves ability to control bit rate (finer steps for modifying QP)
- Extends range of representable transform coefficient values
- Improves chroma fidelity by choosing smaller quantization step size for chroma than for luma (particularly for low rates)

Intra-Picture Coding in MPEG-4 Visual

International standard ISO/IEC 14496-2 (MPEG-4 Visual)

- Video coding standard of Moving Pictures Experts Group (MPEG)
- Includes H.263 Baseline decoder, contains several extensions
- Was used in digital cameras, DivX, ...

Coding of intra macroblocks

- Prediction of transform coefficients (similar to H.263 Annex I)
  - DC level is always predicted from left or above block (direction is determined by differences of neighboring DC levels)
  - First row/column of ACs can be optionally predicted using same prediction direction as for DC (usage is signaled at macroblock level)
- Two methods for quantization
  - MPEG-style: Quantization as in MPEG-2 Video (including weighting matrix)
  - H263-style: Quantization as in H.263 Baseline
- Scanning of transform coefficients
  - 3 scans: Zig-zag, horizontal, vertical (chosen based on prediction mode)
- Coding of vector of transform coefficient levels
  - Coded block pattern and run-level-last coding

Encoder Control for Intra Coding in H.263 and MPEG-4 Visual

Increased degree of freedom relative to H.262 | MPEG-2 Video

- In addition to transform coefficient levels, the method for transform coefficient selection can be selected on macroblock level
- H.263 (Annex I): DC, vertical or horizontal prediction mode
- MPEG-4 Visual: DC or DC and AC prediction

Rate-distortion optimized encoder control

- Determine bitstream $\boldsymbol{b}$ so that the distortion $D(\boldsymbol{s}, \boldsymbol{s'})$ between original picture $\boldsymbol{s}$ and reconstructed picture $\boldsymbol{s'}$ is minimized given a particular target rate $R \leq R_{\text{target}}$
- With $\mathcal{B}_c$ being the set of *conforming* bitstreams with $R \leq R_{\text{target}}$, we can write

$$\boldsymbol{b}^* = \arg\min_{\boldsymbol{b} \in \mathcal{B}_c} D\big(\boldsymbol{s}, \boldsymbol{s'}(\boldsymbol{b})\big) \tag{601}$$

  $\Longrightarrow$ Not feasible due to huge parameter space

  $\Longrightarrow$ Split into smaller optimization problems by partially ignoring dependencies

- Consider block of samples $\boldsymbol{s}_k$ (e.g., picture or macroblock) and optimize with respect to coding parameters $\boldsymbol{p}_k$ (e.g., modes and transform coefficient levels)

$$\min_{\boldsymbol{p}_k} D\big(\boldsymbol{s}_k, \boldsymbol{s'}_k(\boldsymbol{p}_k)\big) \qquad \text{subject to} \qquad R(\boldsymbol{p}_k) \leq R_c \tag{602}$$

Rate-distortion optimized encoder control

Lagrangian encoder control

- Constrained optimization problem for a block of samples $\boldsymbol{s}_k$

$$\min_{\boldsymbol{p}_k} \; D\big(\boldsymbol{s}_k, \boldsymbol{s}'_k(\boldsymbol{p}_k)\big) \qquad \text{subject to} \qquad R(\boldsymbol{p}_k) \leq R_c \tag{603}$$

  can be reformulated as unconstrained optimization problem

$$\min_{\boldsymbol{p}_k} \; D\big(\boldsymbol{s}_k, \boldsymbol{s}'_k(\boldsymbol{p}_k)\big) + \lambda \cdot R(\boldsymbol{p}_k) \tag{604}$$

- Consider partition of $\boldsymbol{s}_k$ into a number of subsets $\boldsymbol{s}_{k,i}$ (e.g. macroblocks)
- If coding parameters $\boldsymbol{p}_{k,i}$ are independent of each other and an additive distortion measure is used, we can write the optimization problem as

$$\sum_i \min_{\boldsymbol{p}_{k,i}} \; D\big(\boldsymbol{s}_{k,i}, \boldsymbol{s}'_{k,i}(\boldsymbol{p}_{k,i})\big) + \lambda \cdot R(\boldsymbol{p}_{k,i}) \tag{605}$$

  $\Longrightarrow$ Independent selection of coding parameters $\boldsymbol{p}_{k,i}$

- For coding decisions in image and video coding
  - Coding decisions are typically not independent (e.g., due to prediction)
  - For practical applicability: Consider past decisions, but ignore impact on future

Lagrangian Encoder Control in Image and Video Coding

Application of Lagrangian encoder control

- Can be applied to basically all decisions in an encoder
- **Quantization**: Select vector $\boldsymbol{q}$ of transform coefficient levels according to

$$\boldsymbol{q}^* = \arg\min_{\boldsymbol{q}} \; D(\boldsymbol{q}) + \lambda \cdot R(\boldsymbol{q}) \tag{606}$$

with $D(\boldsymbol{q})$: SSD distortion for choosing transform coefficient level vector $\boldsymbol{q}$
$R(\boldsymbol{q})$: Number of bits required for representing $\boldsymbol{q}$

$\Longrightarrow$ Rate-distortion optimized quantization (as considered for run-level coding)

- **Mode decision**: Select coding mode c for a macroblock or block

$$c^* = \arg\min_{c} \; D(c) + \lambda \cdot R(c) \tag{607}$$

with $D(c)$: SSD distortion for choosing coding mode c for the block
$R(c)$: Number of bits for block when coded with mode c

$\Longrightarrow$ Can be applied for selecting intra prediction mode

- **Motion search**: Will be considered later in lecture

Comparison of H.263 and MPEG-4 Visual with MPEG-2 Video

Comparison of coding efficiency for intra-picture coding

- Selection of all features that contribute to coding efficiency
  - H.262 | MPEG-2 Video conforming to Main profile
  - H.263+ with advanced intra coding, deblocking filter, modified quantization
  - MPEG-4 Visual with MPEG-style quantization

- Apply same level of encoder optimization for fair comparison
  - Best possible coding efficiency for given syntax
  - Ignore constraints such as real-time operation

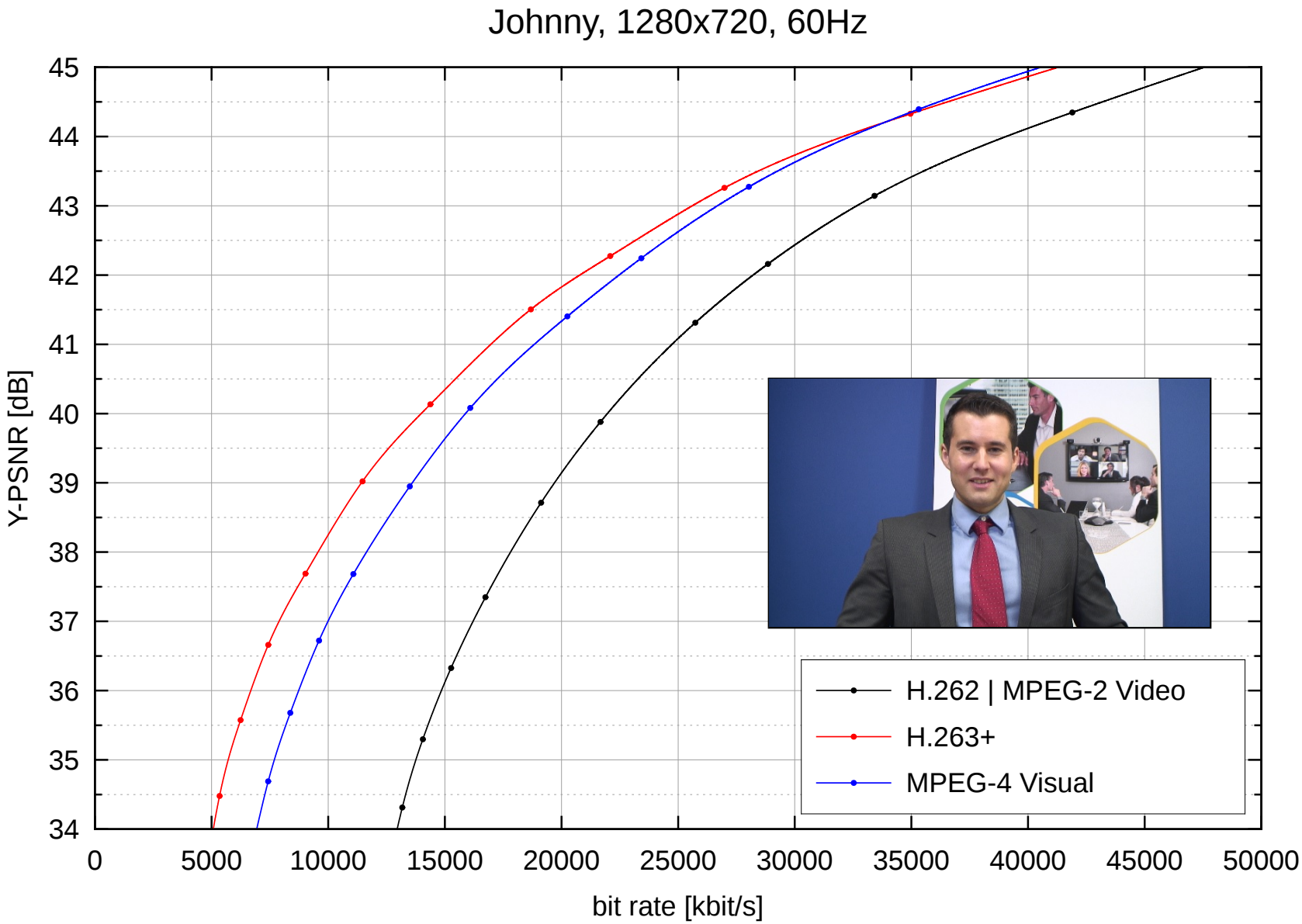
  $\Longrightarrow$ Use rate-distortion optimized quantization for all standards

  $\Longrightarrow$ Apply rate-distortion optimized mode decision where applicable

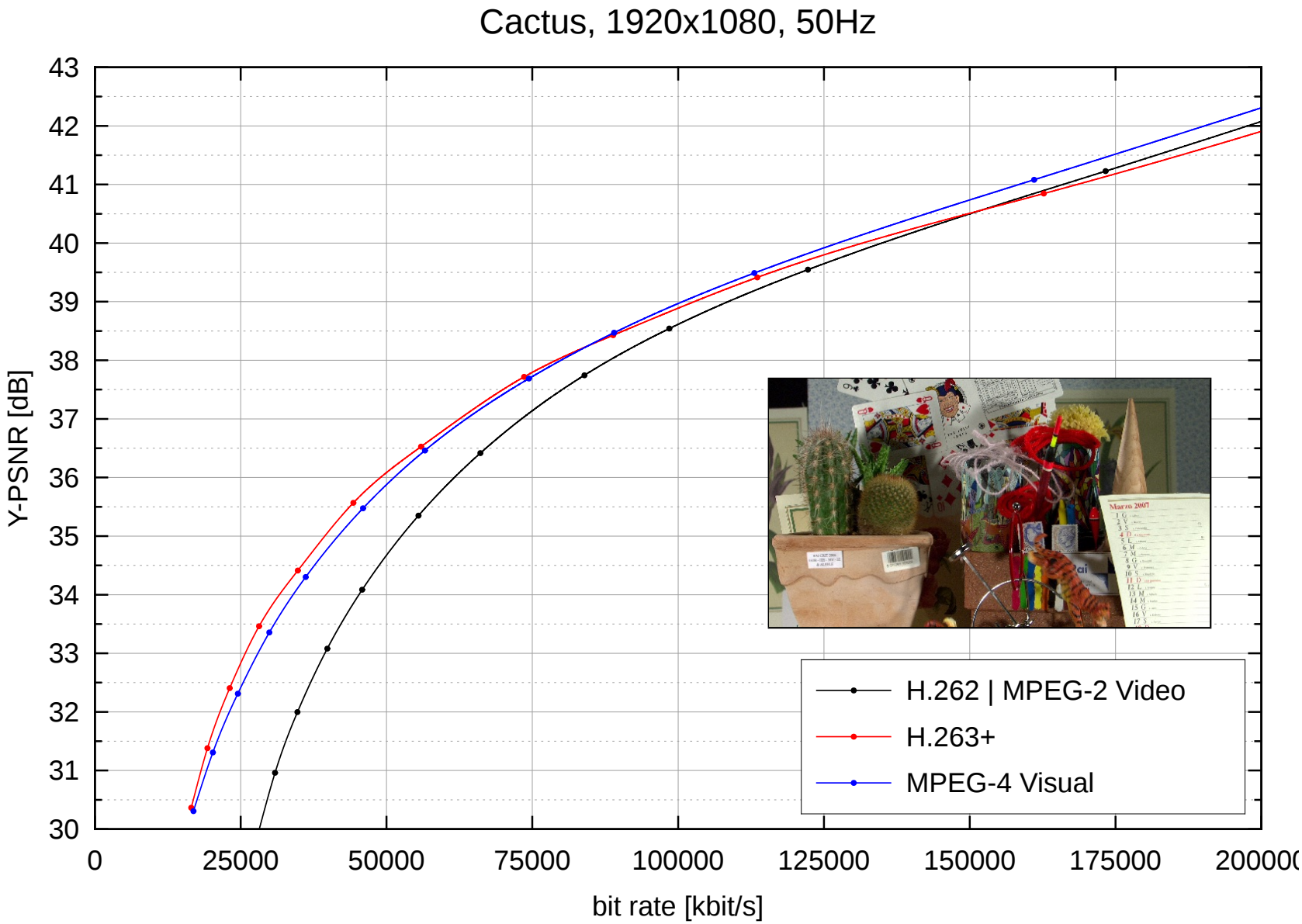
- General coding conditions
  - Encode 10 pictures of 12 video sequences (6 in 720p, 6 in 1080p)
  - Flat quantization matrices (quality is measured using PSNR)
  - Same quantization parameter for all macroblocks
  - Select Lagrangian parameter according to

$$\lambda = \text{const} \cdot \Delta^2 \qquad \text{(with experimentally determined factor)} \qquad (608)$$

Intra Coding Comparison – Sequence "Johnny"



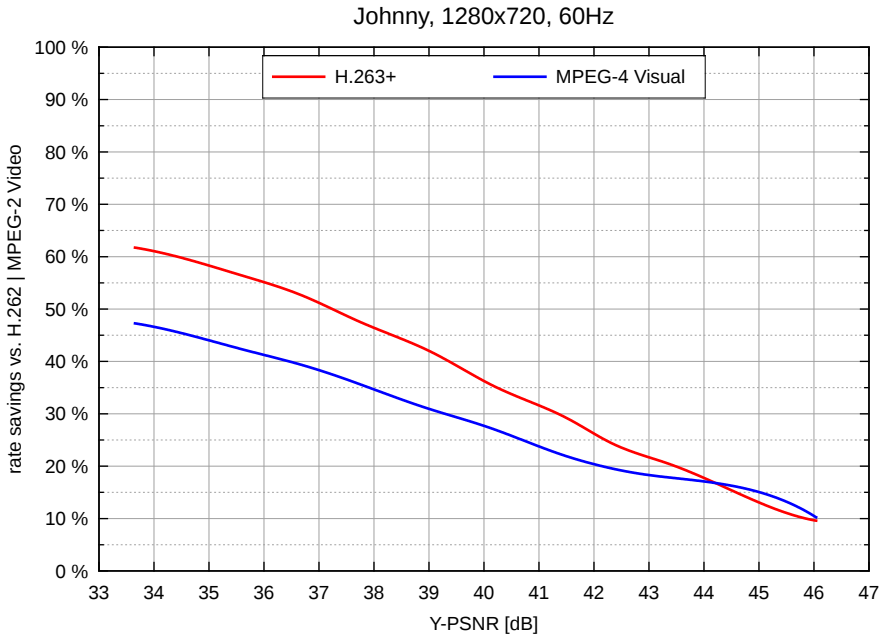
Intra Coding Comparison – Sequence "Cactus"

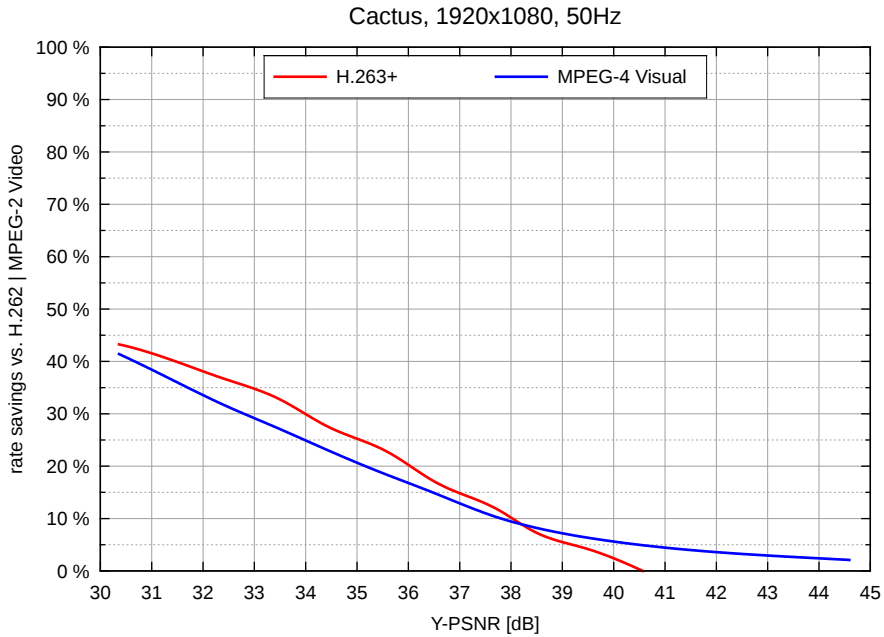


Intra Coding Comparison – Summary

Bit-rate savings of H.263 and MPEG-4 Visual versus H.262 | MPEG-2 Video

- Bit-rate saving at a PSNR value is obtained by interpolating the r-d curves
- Average bit-rate savings are obtained by averaging the savings for 100 PSNR values
- Average bit-rate saving for all sequences
  - H.263+ versus H.262 | MPEG-2 Video: 29%
  - MPEG-4 Visual versus H.262 | MPEG-2 Video: 25%
- Highest savings are obtained for low bit rates





Summary of Intra-Picture Coding in H.263 and MPEG-2 Visual

Intra-picture coding in H.263+

- Transform coding using 8×8 DCT
- Prediction of DC and, partly, AC coefficients
- Coded block pattern
- Adaptive scanning of transform coefficient levels
- Scalar quantization
- Run-level-last coding of transform coefficient levels
- Optional deblocking filter

Intra-picture coding in MPEG-4 Visual

- Similar tools as in advanced intra coding mode of H.263+ (Annex I)
- Additionally includes quantization weighting matrices

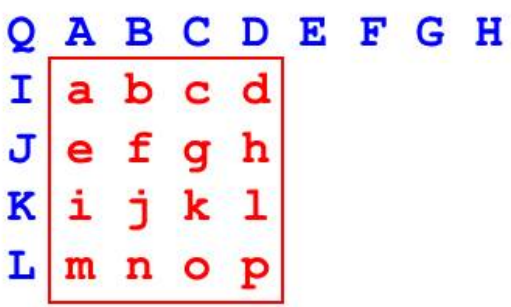
Rate-distortion optimized encoder control

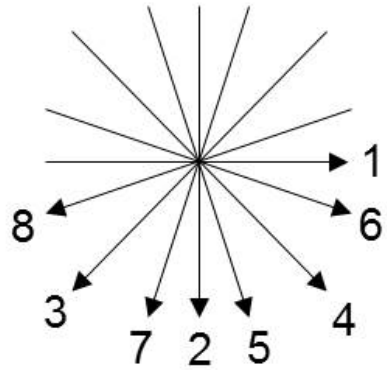
- Split overall optimization problem into smaller problems
- Encoder decision by minimizing $D + \lambda \cdot R$

$\Rightarrow$ Rate-distortion optimized quantization (RDOQ)

$\Rightarrow$ Rate-distortion optimized mode decision

Intra-Picture Coding in H.264 | MPEG-4 AVC





H.264 | MPEG-4 AVC – Overview

ITU-T Rec. H.264 | ISO/IEC 14496-10 (MPEG-4 Advanced Video Coding)

- Video coding standard jointly developed by ITU-T VCEG and ISO/IEC MPEG
- Widely used today in many application spaces
  - Digital television, blu-ray optical disc, digital cameras, mobile phones, video streaming, video conferencing
  - Supported in more than 1 billion devices
  - Every second bit in the Internet is part of a H.264 | MPEG-4 AVC bitstream
- Version 1 (2003):
  - Three profiles: Baseline, Main, Extended (4:2:0, 8 bit)
- Fidelity range extensions (2005)
  - Improvements for large picture sizes, other chroma formats, higher bit depth
  - High, High 10, High 4:2:2, High 4:4:4 profiles (removed later)
  - Later: Addition of High 4:4:4 Predictive and intra-only profiles
- Scalable video coding extension (2007)
  - Extension for scalable video coding (SVC)
- Multiview video coding extension (2009)
  - Extension for multiview video (MVC)
  - Used for 3d-blu-ray discs

Intra-Picture Coding in H.264 | MPEG-4 AVC

Main features of intra-picture coding

- Spatial intra prediction with multiple prediction modes
- Transform coding with 4×4 integer transform
- Optional 8×8 integer transform (High profile)
- Scalar quantization
- Two entropy coding methods:
  - Context-adaptive variable length coding (CAVLC)
  - Context-adaptive binary arithmetic coding (CABAC) – Main/High profile
- Deblocking filter (conceptually similar to Annex J of H.263)

Picture partitioning and intra coding modes

- Pictures are partitioned into 16×16 macroblocks
- 4 intra coding modes are supported (selection on MB level)
  - Intra-4×4
  - Intra-8×8 (High profile)
  - Intra-16×16
  - Intra-PCM (direct coding of samples)

Coding of Intra-4×4 macroblocks

Spatial prediction of blocks

- DC, horizontal, vertical prediction of transform coefficients (as in H.263, MPEG-4 Visual) can similarly also be realized in spatial domain
- Prediction in spatial domain offers more possibilities

Coding of luma component

- 16×16 luma block is partitioned into 16 4×4 blocks
- 4×4 blocks are spatially predicted
- 9 intra prediction modes are supported
- Prediction error of 4×4 blocks is transform-coded

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Coding of chroma components

- Both 8×8 chroma blocks are spatially predicted
- 4 intra prediction modes are supported
- Prediction error is coded using transform coding
- 4×4 transform and second level transform of DCs

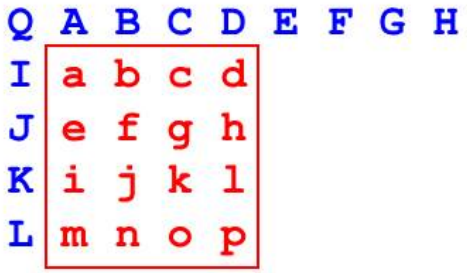
18	19
20	21

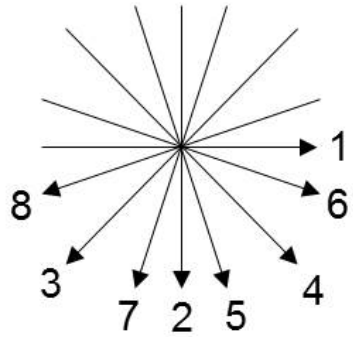
22	23
24	25

Spatial Intra Prediction for 4×4 Blocks

Spatial intra prediction

- Predict block using already coded neighboring samples
- DC prediction (mean value) and 8 directional prediction modes





Coding order and boundary conditions

- Blocks are coded in z-scan order
- Samples used for prediction have to be "available" (already reconstructed)
- Not all modes are available for all blocks
- Samples "E", "F", "G" and "H" can be replaced with sample "D"

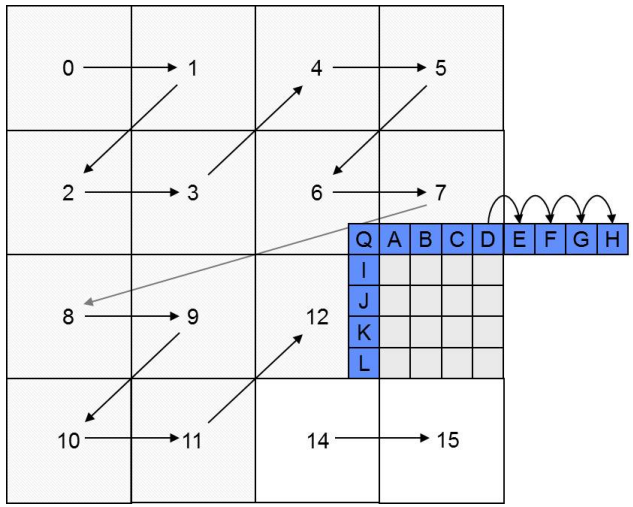
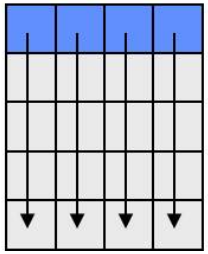
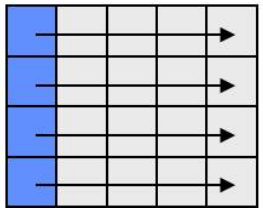


Illustration of Intra Prediction Modes

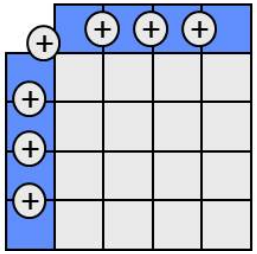
Mode 0 - Vertical



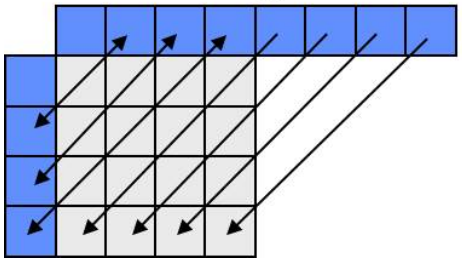
Mode 1 - Horizontal



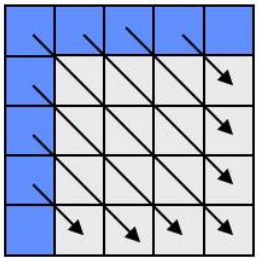
Mode 2 - DC



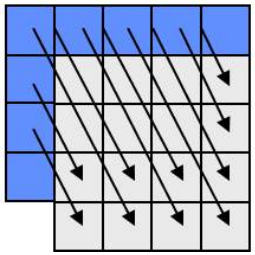
Mode 3 – Diagonal Down/Left



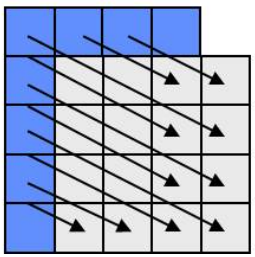
Mode 4 – Diagonal Down/Right



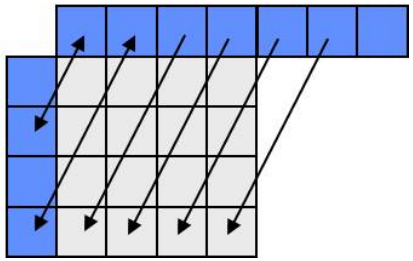
Mode 5 – Vertical-Right



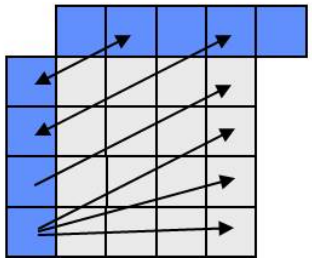
Mode 6 – Horizontal-Down



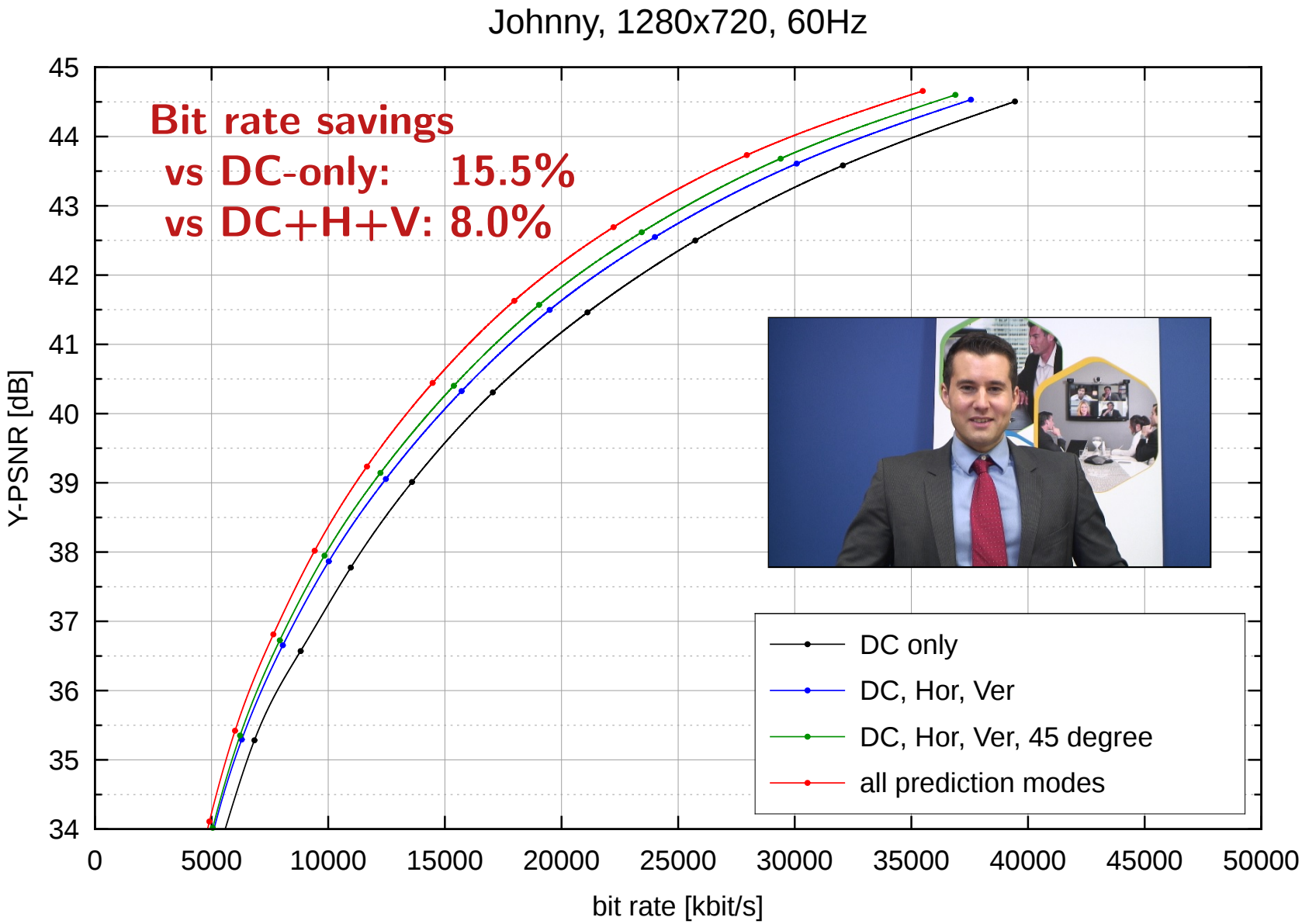
Mode 7 – Vertical-Left



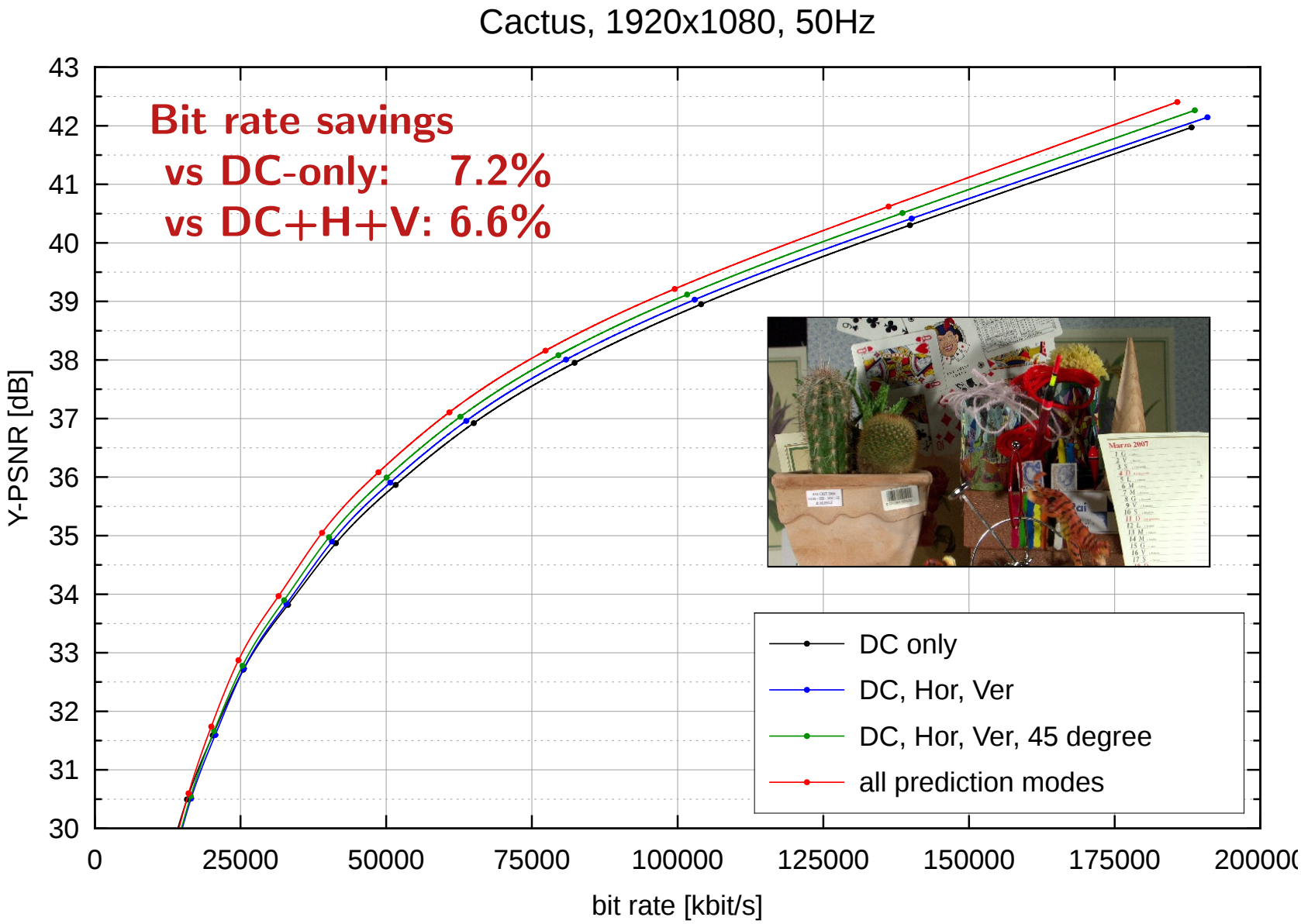
Mode 8 – Horizontal-Up



Efficiency of Intra Prediction – Sequence "Johnny"



Efficiency of Intra Prediction – Sequence "Cactus"



Transform Coding of 4×4 Blocks

Separable 4×4 integer transform

- Orthogonal block transform with integer approximation of DCT

$$\boldsymbol{u}_{4\times4} = \boldsymbol{A}_{4\times4} \cdot \boldsymbol{s}_{4\times4} \cdot \boldsymbol{A}_{4\times4}^{\mathrm{T}} \qquad \text{with} \qquad \boldsymbol{A}_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{609}$$

- Inverse is specified by exact integer operations
  $\Longrightarrow$ No accumulation of transform mismatches
- Easy implementation (only additions and bit shift operations)
- Basis vectors have different norms
  $\Longrightarrow$ Compensated by modifying the quantizer step size accordingly

Scalar quantization of transform coefficients

- Uniformly distributed reconstruction levels
- Logarithmic quantization step size control ($\Delta \approx \alpha \cdot 2^{\mathrm{QP}/6}$)
- Smaller quantization step sizes for chroma (as in Annex T of H.263)
- Support for quantization weighting matrices (High profile)
- Quantization parameter can change at macroblock level

Coding of Transform Coefficient Levels

Context-adaptive variable length coding (CAVLC)

- Coded block pattern (for all six 8×8 block) using VLC table
- Zig-zag scan for mapping matrix into vector
- Syntax element "coeff_token" for 4×4 blocks
  - Specifies number of non-zero coefficients and number of trailing ones
  - Chosen VLC table depends on number of non-zero coefficients in already coded neighboring blocks
- Additionally code "runs" and "levels" as well as signs for trailing ones

Context-adaptive binary arithmetic coding (CABAC) [Main, High profile]

- Binary arithmetic coding of all low-level syntax elements
- Coded block pattern (flag for each of the six 8×8 blocks)
- Flag for 4×4 blocks indicating whether non-zero levels are present
- Coding of a significance map
  - "significance_flag" indicating whether level is non-zero
  - if non-zero, "last_flag" indicating whether last non-zero level
- Coding of absolute levels (minus 1) and signs
- Probability models are adapted to statistics during encoding and decoding

Intra-16×16 and Intra-8×8 Macroblocks

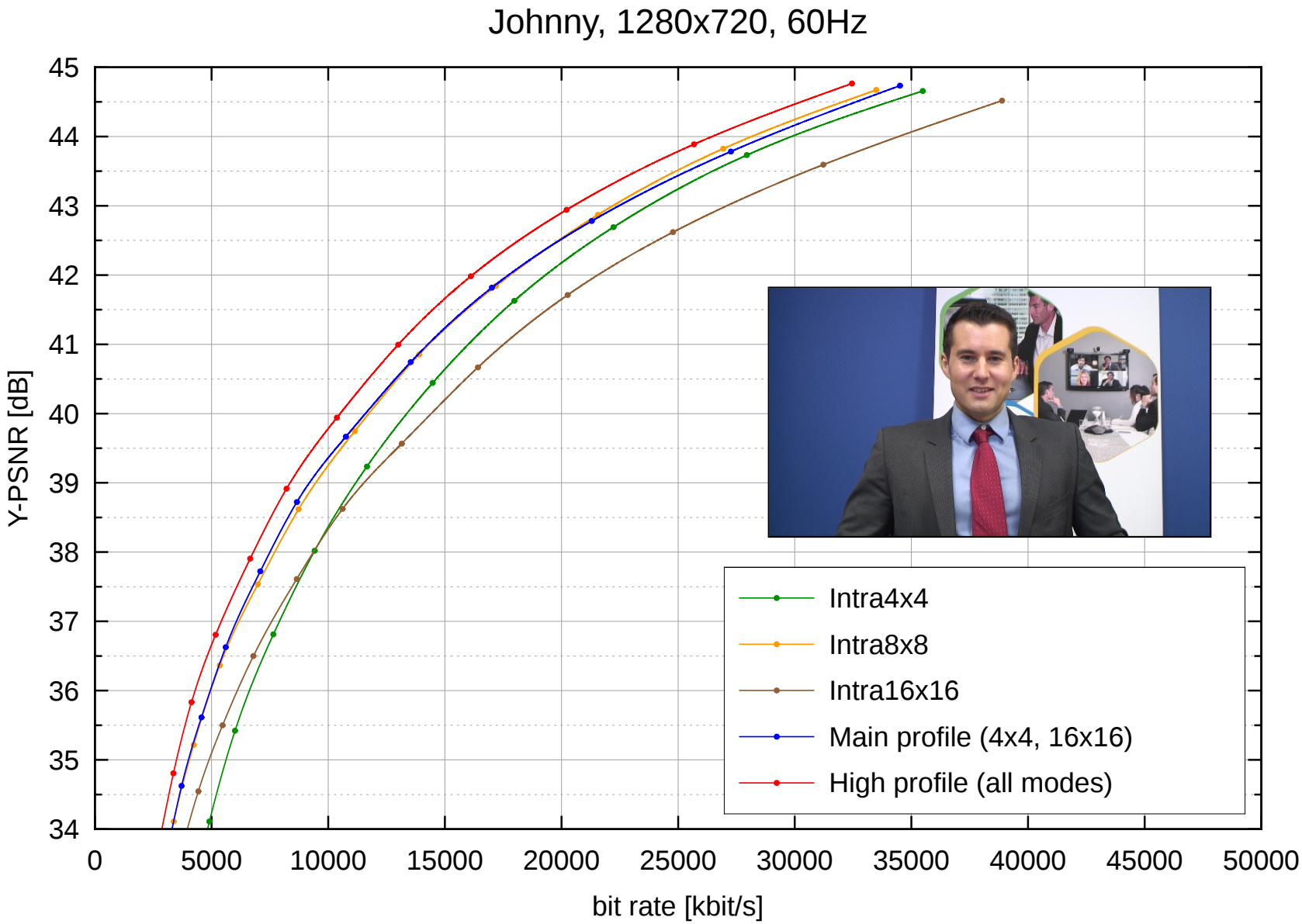
Intra-16×16 macroblock mode

- Prediction of 16×16 luma block
- Four prediction modes: DC, horizontal, vertical, planar
- 4×4 transform of all sixteen 4×4 blocks
- Additional 4×4 Hadamard transform of DC coefficients
- DC block is treated separately in entropy coding
- Similar concept with 2×2 Hadamard transform of DC coefficients is also used for chroma blocks (for all intra coding modes)

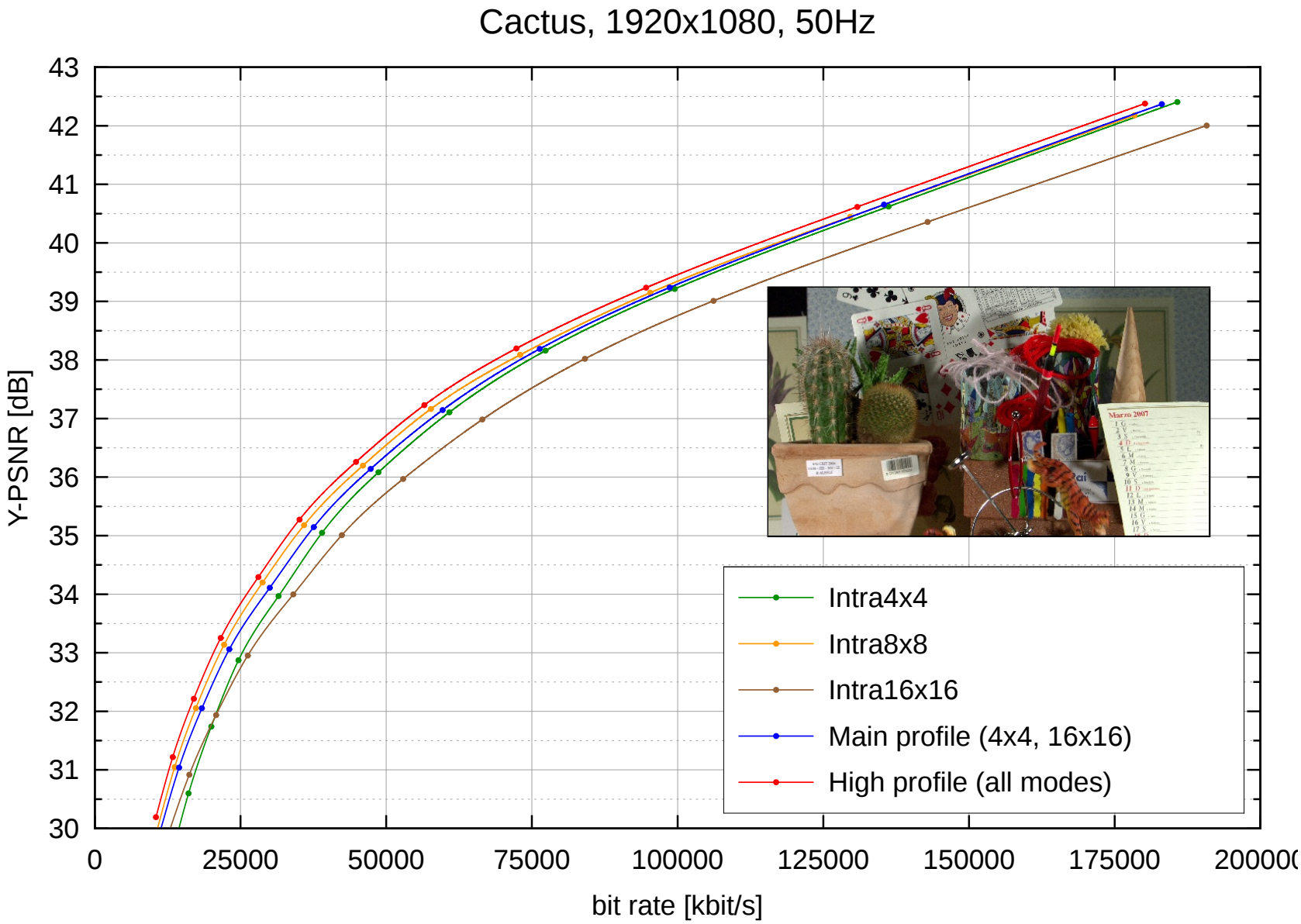
Intra-8×8 macroblock mode (High profile)

- Prediction of 8×8 blocks of luma component
- Same prediction modes as for Intra-4×4 (but extended to larger block size)
- Reference samples are low-pass filtered before they are used for prediction
- 8×8 integer transform for the four luma sub-blocks
- Entropy coding is extended to 8×8 transform blocks

Coding Efficiency of Intra Modes – Sequence "Johnny"



Coding Efficiency of Intra Modes – Sequence "Cactus"



Coding Efficiency Comparison with Older Standards

Comparison of coding efficiency for intra-picture coding

- Selection of all features that contribute to coding efficiency
  - H.262 | MPEG-2 Video conforming to Main profile
  - H.263+ with advanced intra coding, deblocking filter, modified quantization
  - MPEG-4 Visual with MPEG-style quantization
  - **H.264 | MPEG-4 AVC High profile with CABAC**

- Apply same level of encoder optimization for fair comparison
  - Best possible coding efficiency for given syntax
  - Ignore constraints such as real-time operation

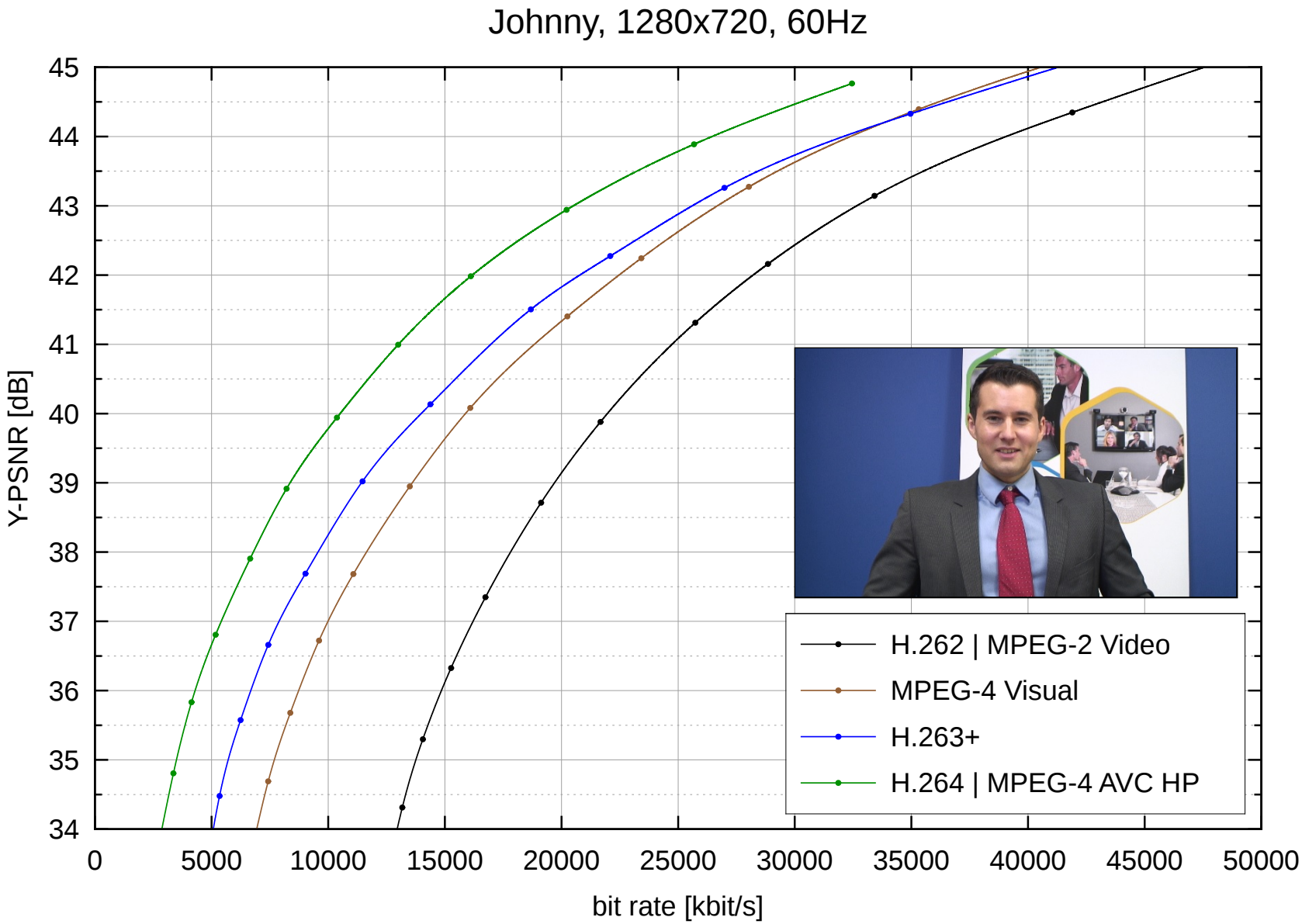
  $\Longrightarrow$ Use rate-distortion optimized quantization for all standards

  $\Longrightarrow$ Apply rate-distortion optimized mode decision where applicable

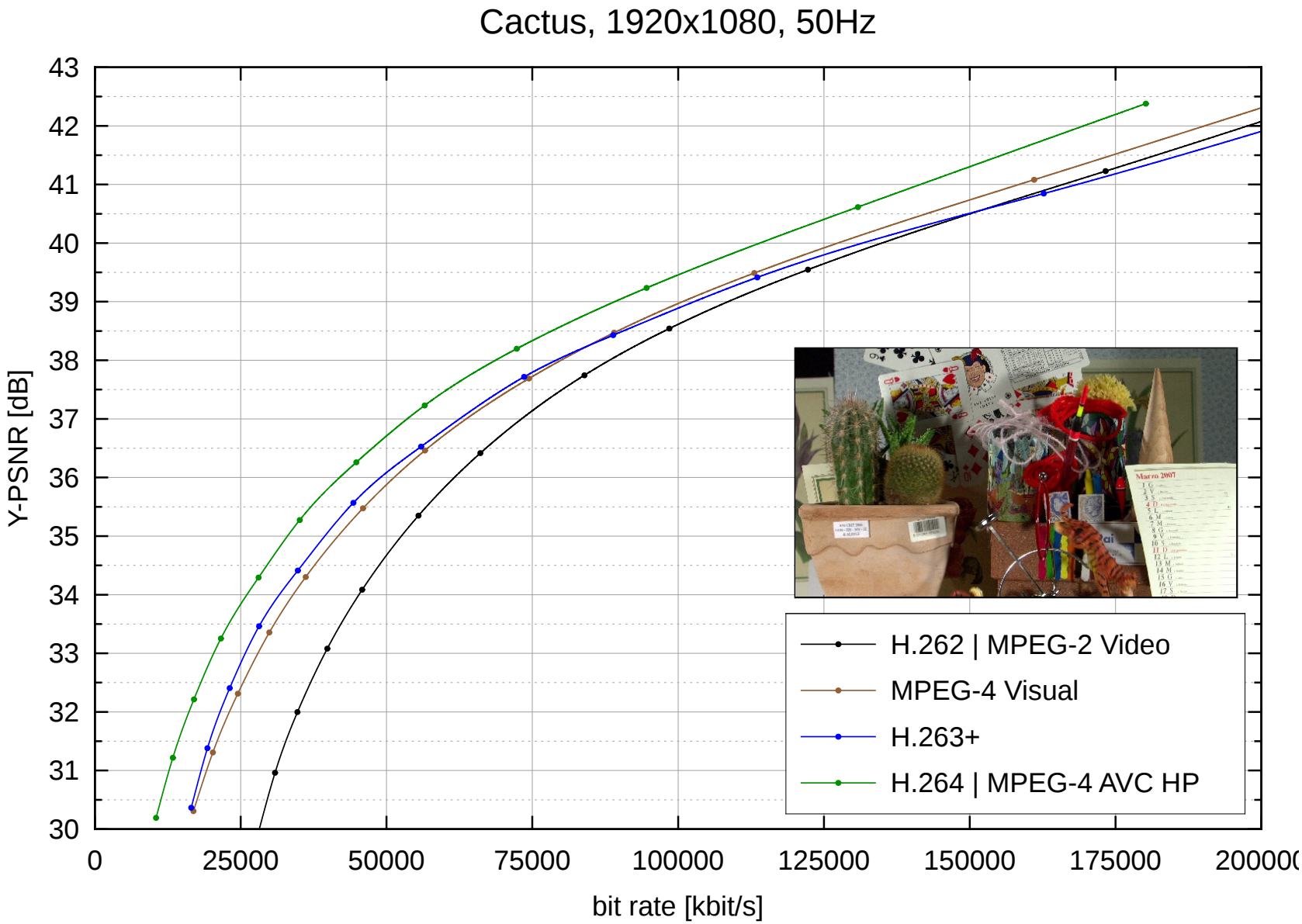
- General coding conditions
  - Encode 10 pictures of 12 video sequences (6 in 720p, 6 in 1080p)
  - Flat quantization matrices (quality is measured using PSNR)
  - Same quantization parameter for all macroblocks
  - Select Lagrangian parameter according to

$$\lambda = \text{const} \cdot \Delta^2 \qquad \text{(with experimentally determined factor)} \qquad (610)$$

Intra Coding Comparison – Sequence "Johnny"



Intra Coding Comparison – Sequence "Cactus"



Intra Coding Comparison – Summary

Bit-rate savings of H.264 | MPEG-4 AVC versus older video coding standards

- Bit-rate saving at a PSNR value is obtained by interpolating the r-d curves
- Average bit-rate savings are obtained by averaging the savings for 100 PSNR values
- Highest savings are obtained for low bit rates
- Average bit-rate saving for all sequences are summarized below

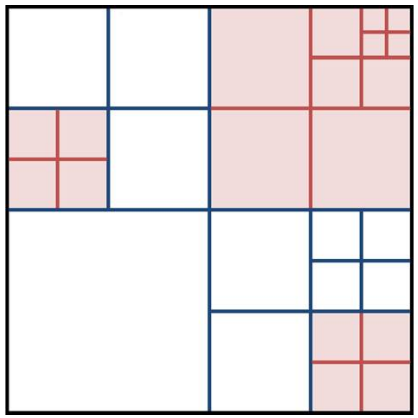
codec	average bit rate savings relative to ...		
	H.263+	MPEG-4	MPEG-2
H.264 / AVC	21.1 %	28.2 %	45.6 %
H.263+		9.7 %	32.5 %
MPEG-4			26.3 %

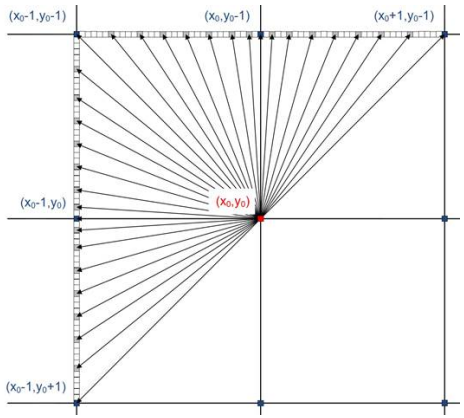
Summary of Intra-Picture Coding in H.264 | MPEG-4 AVC

Intra-picture coding in H.264 | MPEG-4 AVC

- Four intra macroblock modes
  - Intra-4×4, Intra-8×8, Intra-16×16: Prediction & transform coding
  - Intra-PCM: Direct coding of samples
- Intra prediction in spatial domain using neighboring samples
  - Intra-4×4 and Intra-8×8: Eight directional modes & DC prediction
  - Intra-16×16 and chroma: Four intra prediction modes
- Transform: Integer approximation of DCT
  - Intra-4×4: Transform of 4×4 blocks
  - Intra-8×8: Transform of 8×8 blocks
  - Intra-16×16 and chroma: Transform of 4×4 blocks + DC transform
- Scalar quantization
  - Uniform reconstruction quantizer (with optional weighting matrices)
  - Norms of basis vectors are taken into account in quantization
- Two methods for entropy coding
  - Context-adaptive variable length coding (CAVLC)
  - Context-adaptive binary arithmetic coding (CABAC)
- Deblocking filter

Intra-Picture Coding in H.265 | MPEG-H HEVC





H.265 | MPEG-H HEVC – Overview

ITU-T Rec. H.265 | ISO/IEC 23008-2 (MPEG-H High Efficiency Video Coding)

- Jointly developed by ITU-T VCEG and ISO/IEC MPEG
- Last video coding standard with focus on coding of high-resolution video
- Significantly increased coding efficiency, particularly for high-resolution video
- First version was finalized in January 2013
- First version specifies three profiles
  - Main profile (4:2:0 chroma format, 8 bit per sample)
  - Main 10 profile (4:2:0 chroma format, 10 bit per sample)
  - Main Still Picture profile (intra-only coding subset of Main profile)
- Extensions are under development
  - Fidelity range extension (other chroma samplings, higher bit depth)
  - Scalable video coding extension
  - Multiview video coding extension
  - Multiview video plus depth coding extension

Main Features of H.265 | MPEG-H HEVC

Main improvements relative to H.264 | MPEG-4 AVC

- Larger block sizes for transform coding and motion compensation
- Increased flexibility for partitioning a picture into blocks
- Improved interpolation filters and motion vector coding
- Increased number of intra prediction modes
- Improved coding of transform coefficient levels
- Additional in-loop filter: Sample-adaptive offset filter

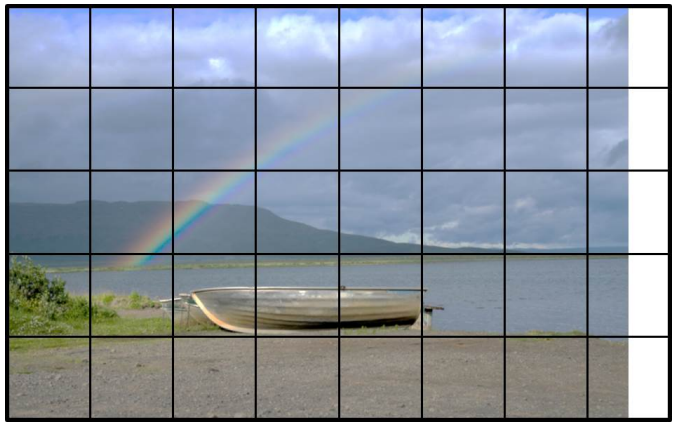
Intra-Picture Coding in H.265 | MPEG-H HEVC

- Spatial intra prediction and transform coding of prediction residual
- Increased number of intra prediction modes compared to H.264/AVC
- Larger transform sizes
- More flexible partitioning of a picture
- Improved coding of transform coefficient levels (for larger blocks)
- Deblocking filter and additional sample-adaptive offset filter

Picture Partitioning in H.265 | MPEG-H HEVC

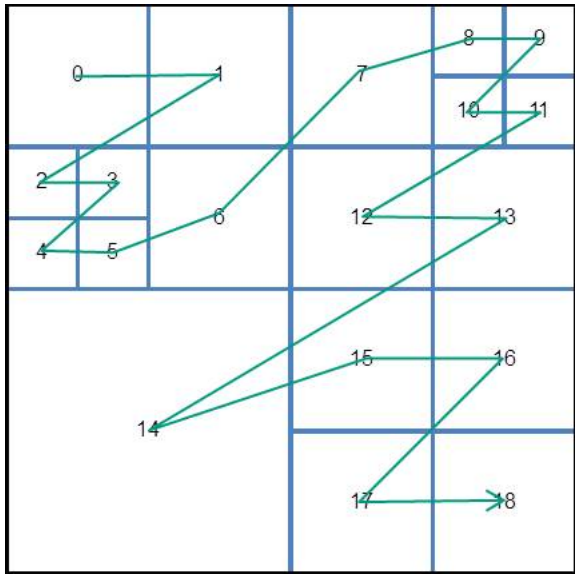
Picture partitioning into **coding tree blocks**

- Coding tree blocks (CTBs): Fixed size of 16×16, 32×32 or 64×64 luma samples
- Size of CTBs chosen by encoder
- Luma and chroma CTBs together with syntax are called **coding tree unit** (CTU)



Partitioning of coding tree blocks

- Quad-tree partitioning into **coding blocks** (CBs)
- Luma and chroma CBs together with syntax are called **coding unit** (CU)
- Maximum CU size: Size of the CTB
- Minimum CU size: Selected by encoder, but equal to or larger than 8×8 luma samples
- Coding mode (intra or inter) is chosen for CU
- CU is similar to macroblock in older standards
- Coding order: Z-scan



Example: Picture Partitioning into Coding Units

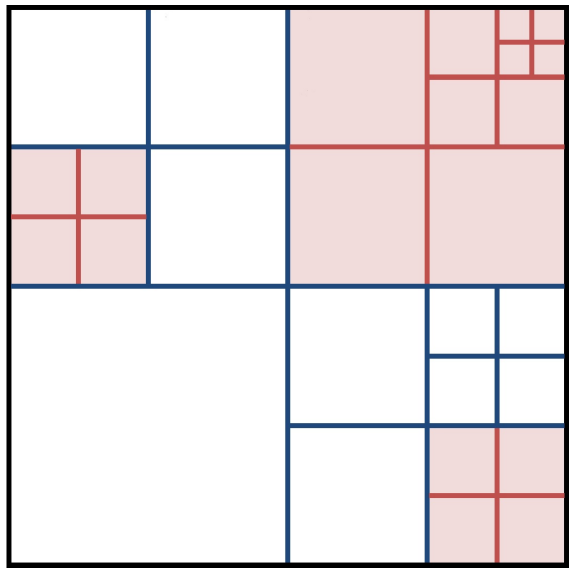
Example for picture partitioning into coding units

- Picture with 2560×1600 luma samples of HEVC test sequence "Traffic"
- Quadtree-based partitioning into coding unit represents a simple scheme for locally adapting the block sizes to the image structure

Intra Coding of Coding Units in H.265 | MPEG-H HEVC

Partitioning of a CB into **transform blocks** (TBs)

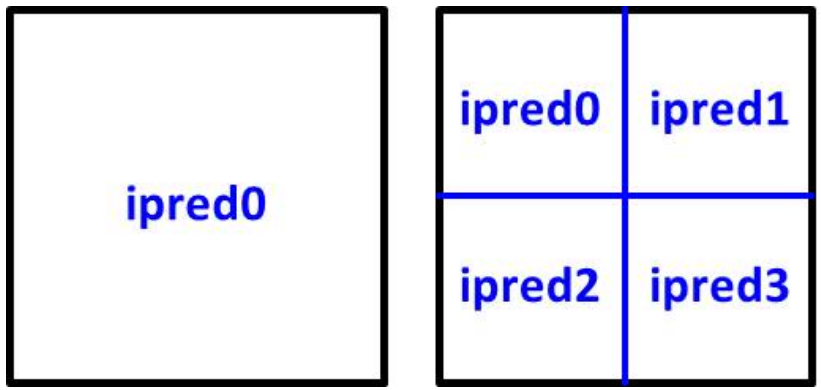
- Nested quad-tree partitioning
- TB corresponds to a single block transform
- Min. and max. TB size are selected by encoder
- Supported transforms: 4×4, 8×8, 16×16, 32×32
- Luma and chroma TBs together with syntax form a **transform unit** (TU)
- Special case: Chroma 4×4 blocks are not split



Intra prediction and mode signaling

- One or four luma intra prediction modes per coding unit
- One chroma prediction mode per CU
- Actual intra prediction is performed transform block by transform block

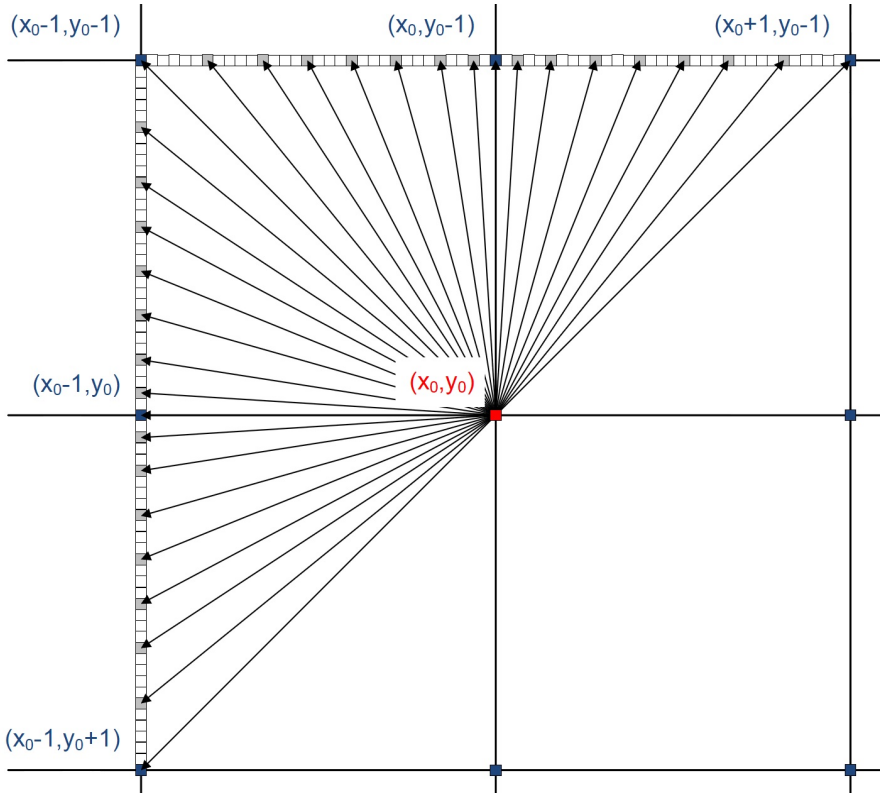
  $\Longrightarrow$ Improved prediction accuracy



Spatial Intra Prediction of Transform Blocks

Prediction of luma transform blocks

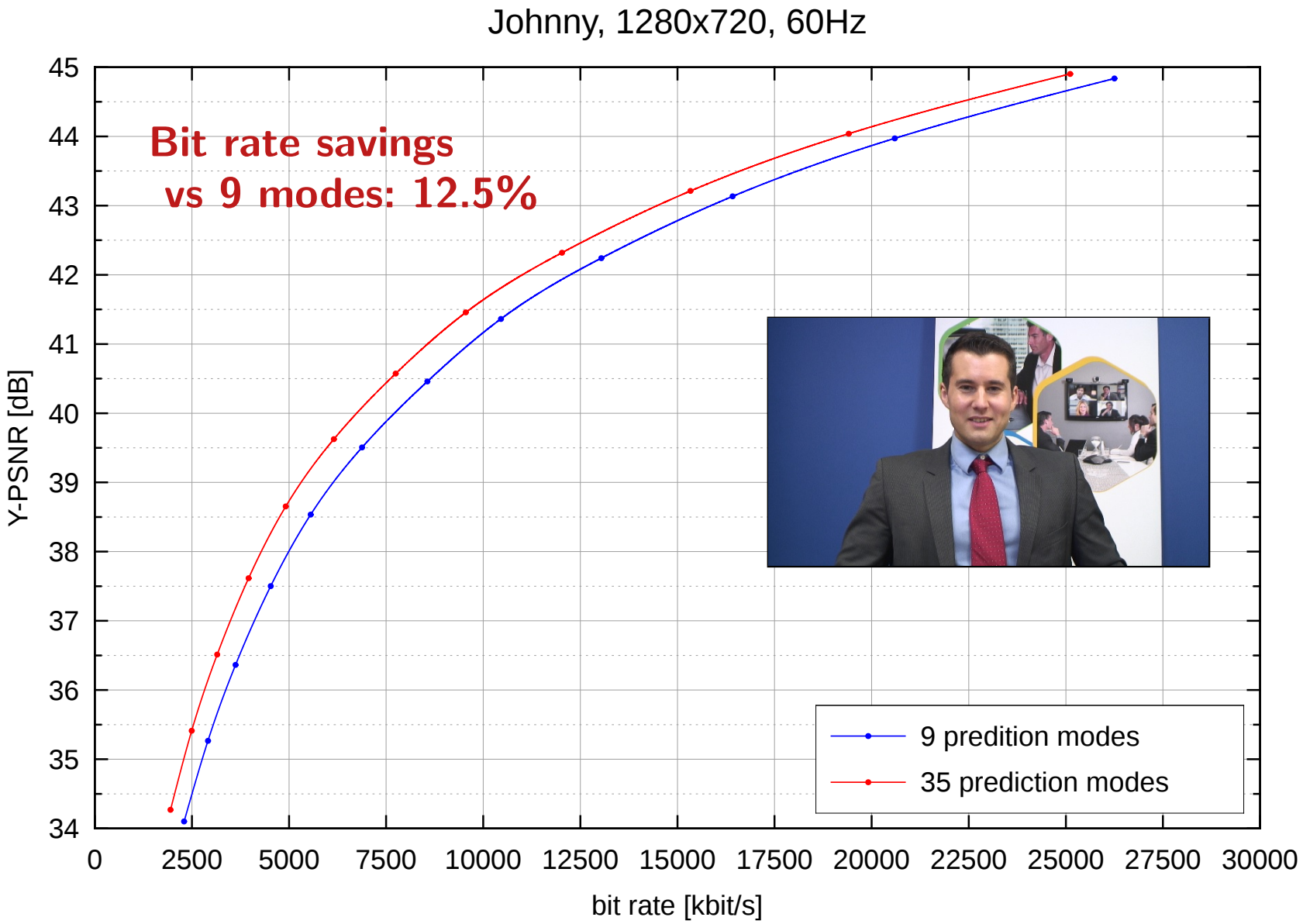
- Spatial intra prediction using neighboring samples of already coded blocks
- 33 directional prediction modes
- Additionally:
  DC and planar prediction
- Reference sample smoothing depending on block size and prediction direction



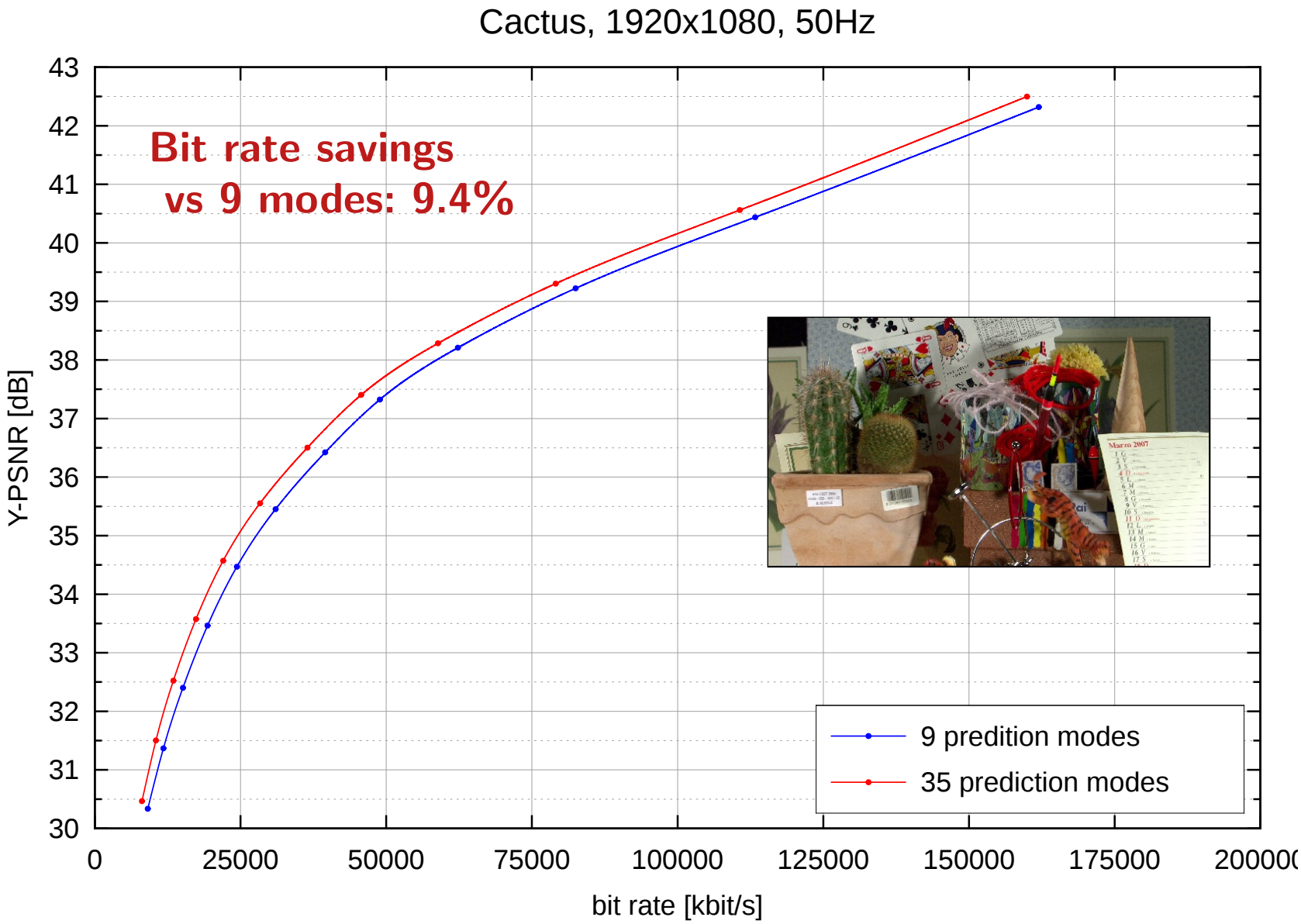
Prediction of chroma transform blocks

- Same prediction mechanism as for luma
- Prediction mode can be selected among the following:
  - Planar, DC, horizontal, vertical prediction mode
  - Same prediction modes as for associated luma block

Number of Intra Prediction Modes – Sequence "Johnny"



Number of Intra Prediction Modes – Sequence "Cactus"



Transform and Quantization in H.265 | MPEG-H HEVC

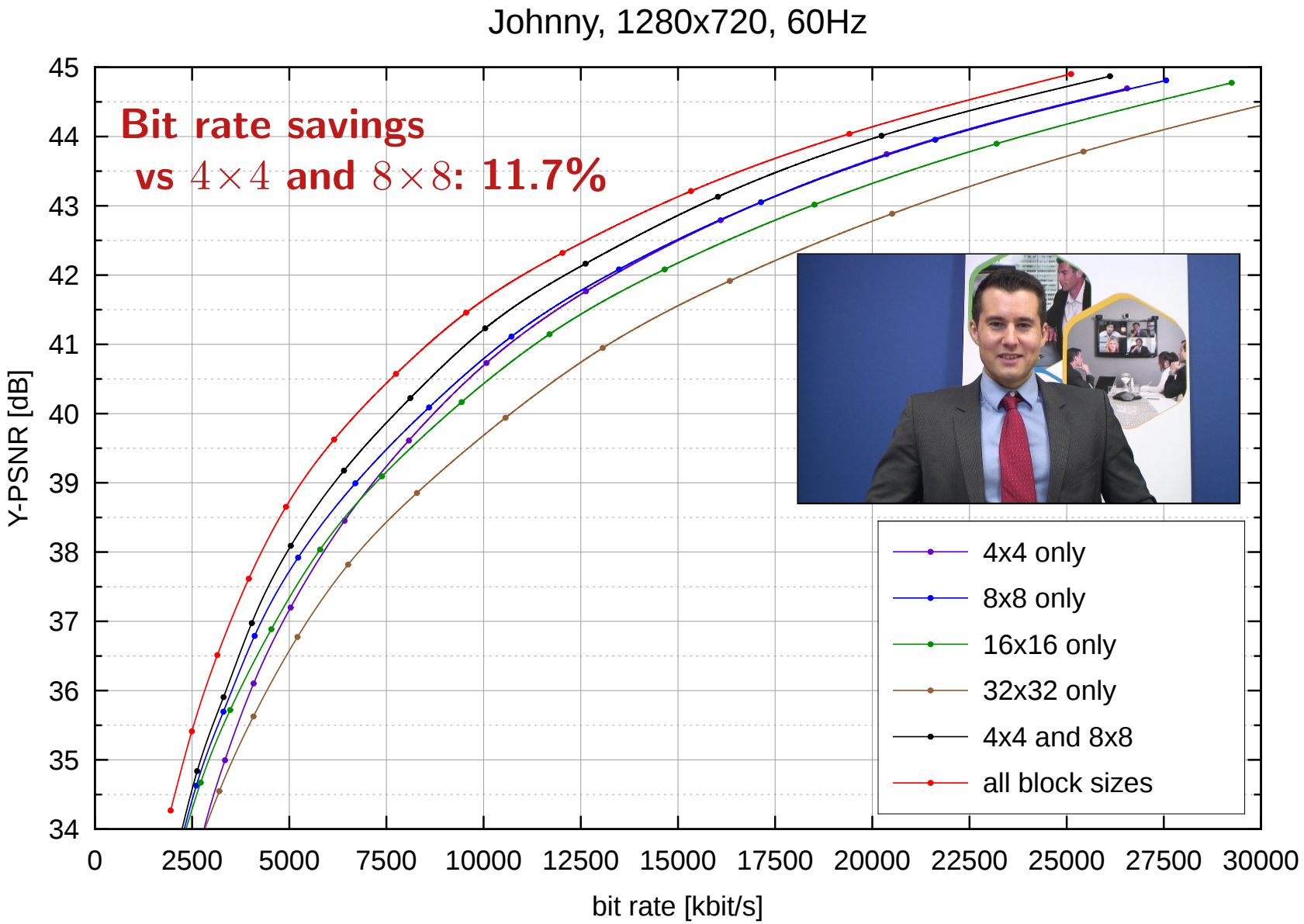
Separable 2-d transform

- 2-d block transform of the size of the transform block
- Separable integer approximation of discrete cosine transform (DCT)
- Basis function have approximately the same norm
- Supported transform sizes: 4×4, 8×8, 16×16, 32×32
- All transform sizes are specified by single 32×32 integer matrix
- Exception: 4×4 luma TBs of intra-picture predicted CUs
  - Separable integer approximation of discrete sine transform (DST)
  - Better fits the statistical properties that the residual amplitudes tend to increase with increasing distance from reference samples

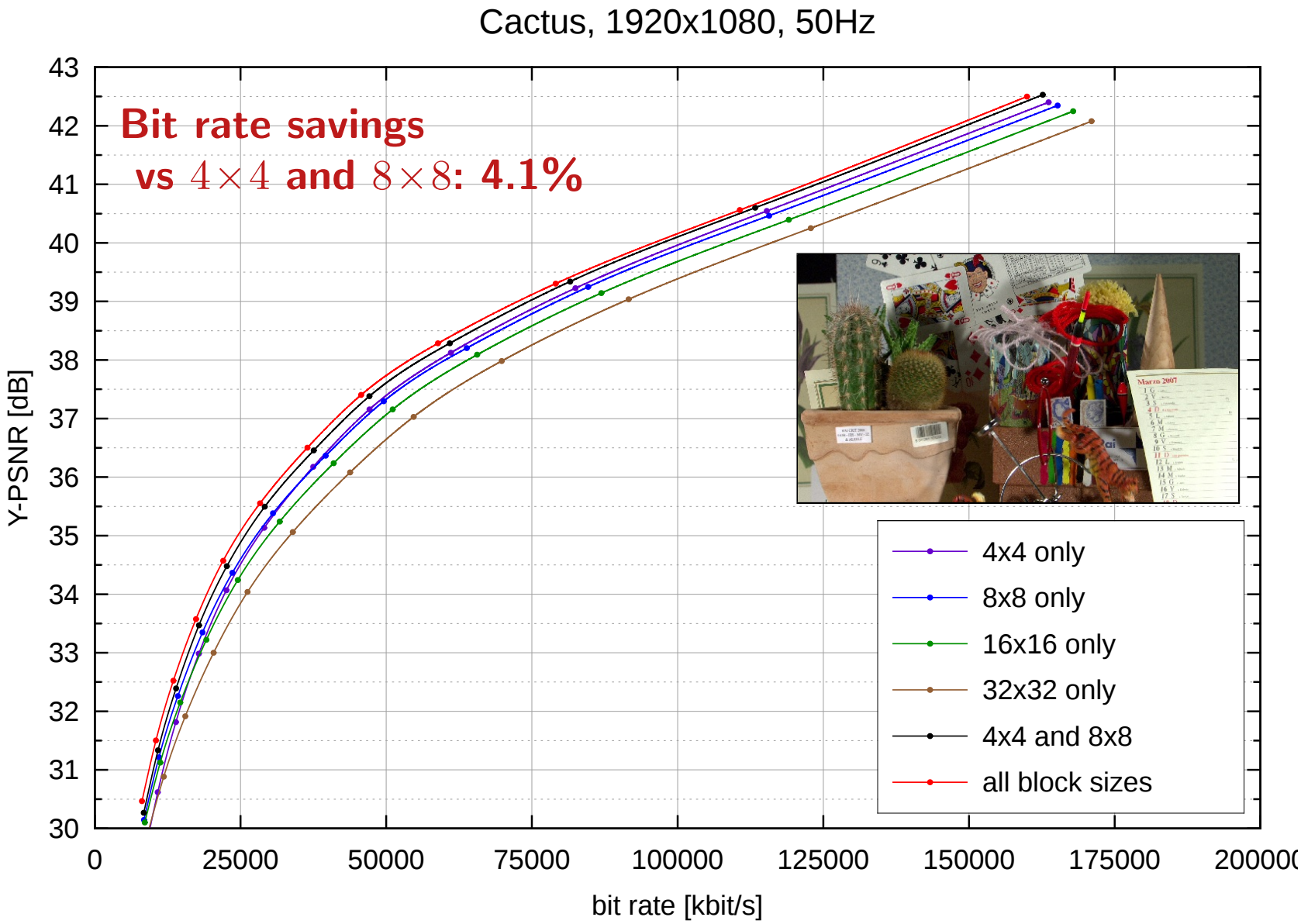
Quantization

- Same uniform reconstruction level quantizer as in H.264 | MPEG-4 AVC
- Quantization step size controlled by quantization parameter (QP)
- Approximately logarithmic mapping between QP and step size
- Quantization scaling matrices are supported

Block Sizes for Prediction and Transform – Sequence "Johnny"



Block Sizes for Prediction and Transform – Sequence "Cactus"



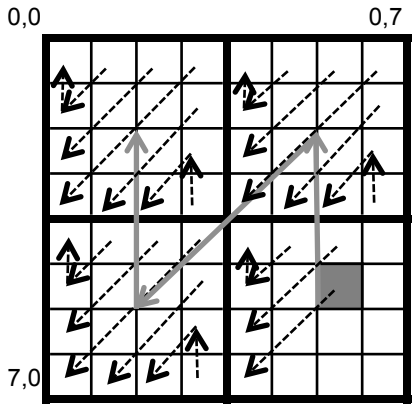
Entropy Coding of Transform Coefficient Levels

Only a single entropy coding method

- Context-adaptive binary arithmetic coding (CABAC)
- Core algorithm of CABAC is unchanged relative to H.264 | MPEG-4 AVC

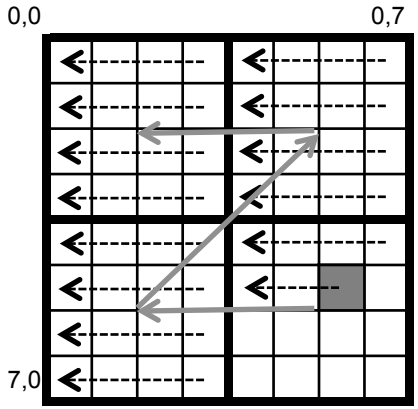
Coding of transform coefficient levels

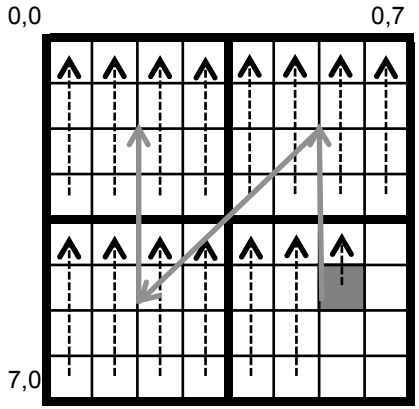
- Coded block flag for transform block
- x and y coordinate of last significant level in scanning order
- Significance flags for 4×4 sub-blocks
- Significance map for 4×4 sub-blocks
- Absolute levels and signs



Adaptive coefficient scanning

- For 4×4 and 8×8 TB in intra CUs, scan depends on intra prediction mode (horizontal, vertical, diagonal)
- For all other blocks: Diagonal scan





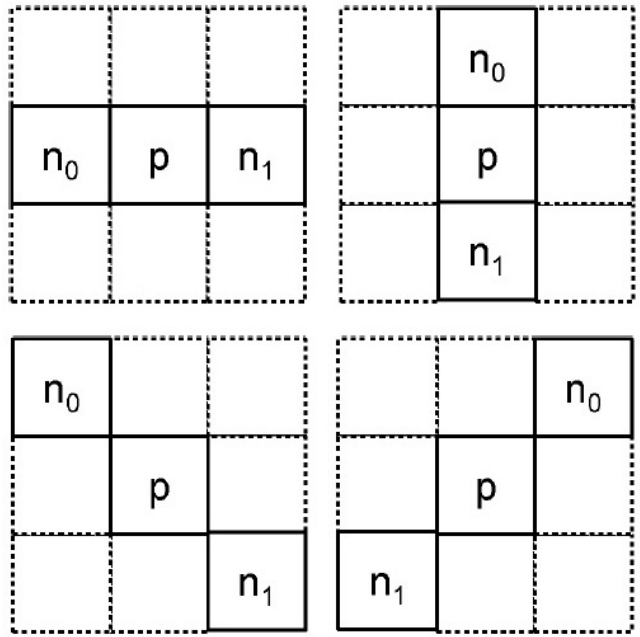
In-Loop Filtering

Deblocking filter

- Adaptive filter for reducing block edge boundaries
- Similar adaptation of boundary filter strength as in H.264 | MPEG-4 AVC
- Applied on 8×8 sample grid

Sample adaptive offset (SAO)

- Conditional adding of an offset to samples (offset are transmitted)
- Selected on region basis: Either not used or applied in one of two modes
- Mode 1: Band offset filtering
  - Offset depends on sample value
  - Split amplitude range into 32 bands
  - Offset values are transmitted for four consecutive bands
- Mode 2: Edge offset filtering
  - Choose one of 4 gradient directions
  - Classify sample into one of 5 categories based on neighbouring samples
  - Offset values are transmitted for 4 of these sample classes



Coding Efficiency Comparison with Prior Standards

Comparison of coding efficiency for intra-picture coding

- Selection of all features that contribute to coding efficiency
  - H.262 | MPEG-2 Video conforming to Main profile
  - H.263+ with advanced intra coding, deblocking filter, modified quantization
  - MPEG-4 Visual with MPEG-style quantization
  - H.264 | MPEG-4 AVC High profile with CABAC
  - **H.265 | MPEG-H HEVC Main profile**

- Apply same level of encoder optimization for fair comparison
  - Best possible coding efficiency for given syntax
  - Ignore constraints such as real-time operation

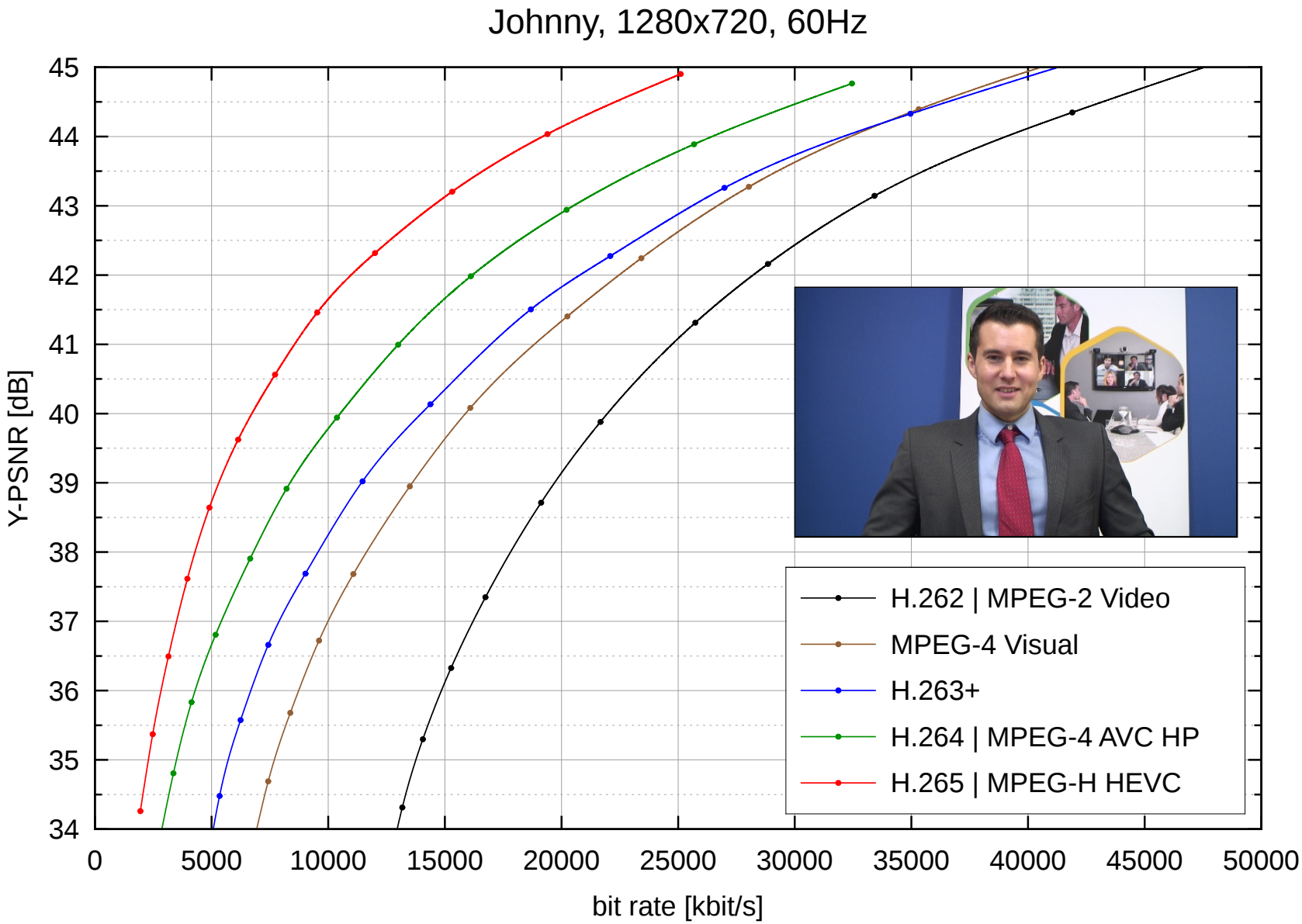
  $\Longrightarrow$ Use rate-distortion optimized quantization for all standards

  $\Longrightarrow$ Apply rate-distortion optimized mode decision where applicable

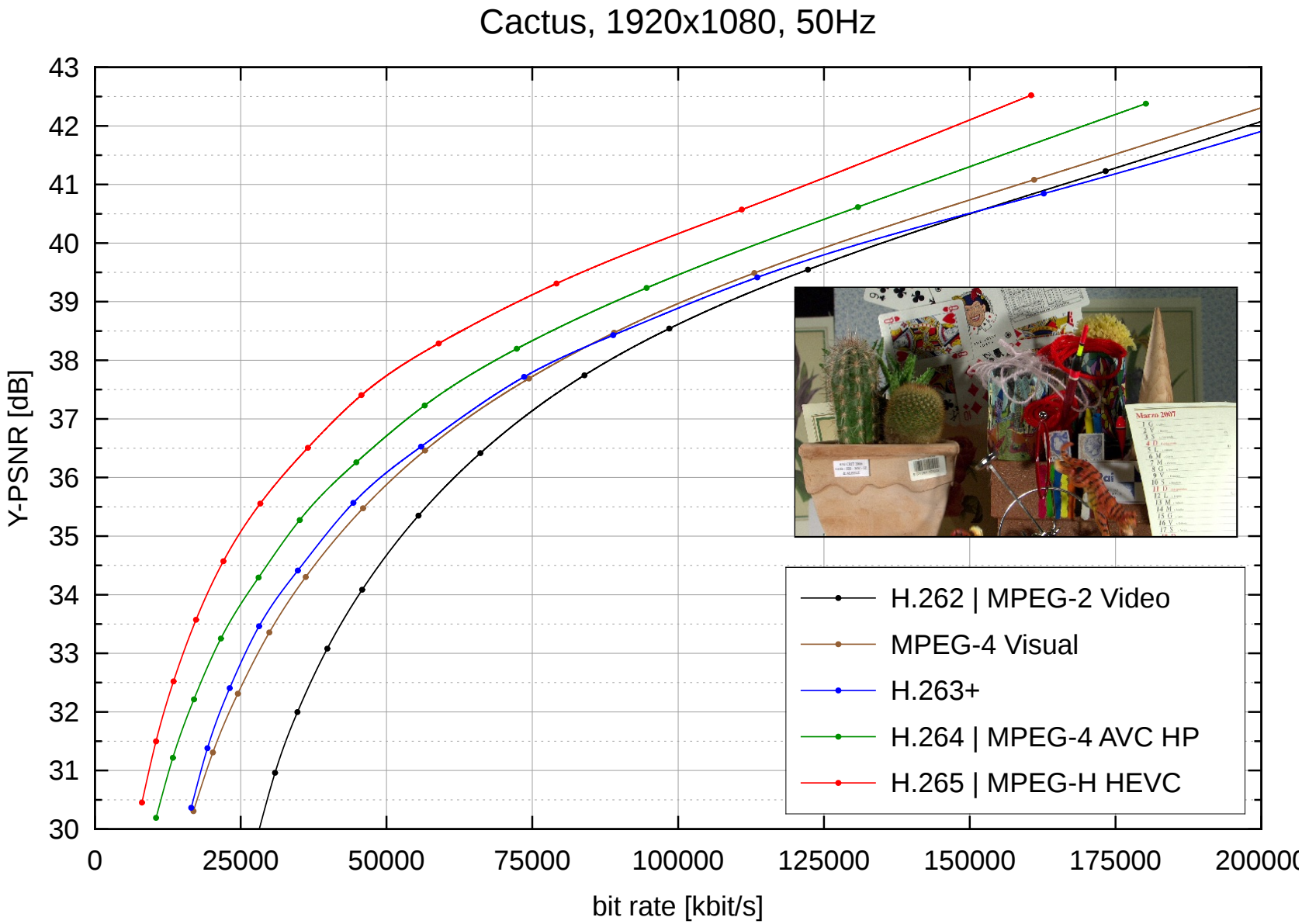
- General coding conditions
  - Encode 10 pictures of 12 video sequences (6 in 720p, 6 in 1080p)
  - Flat quantization matrices (quality is measured using PSNR)
  - Same quantization parameter for all macroblocks
  - Select Lagrangian parameter according to

$$\lambda = \text{const} \cdot \Delta^2 \qquad \text{(with experimentally determined factor)} \qquad (611)$$

Intra Coding Comparison – Sequence "Johnny"



Intra Coding Comparison – Sequence "Cactus"



Intra Coding Comparison – Summary

Bit-rate savings of H.265 | MPEG-H HEVC versus older video coding standards

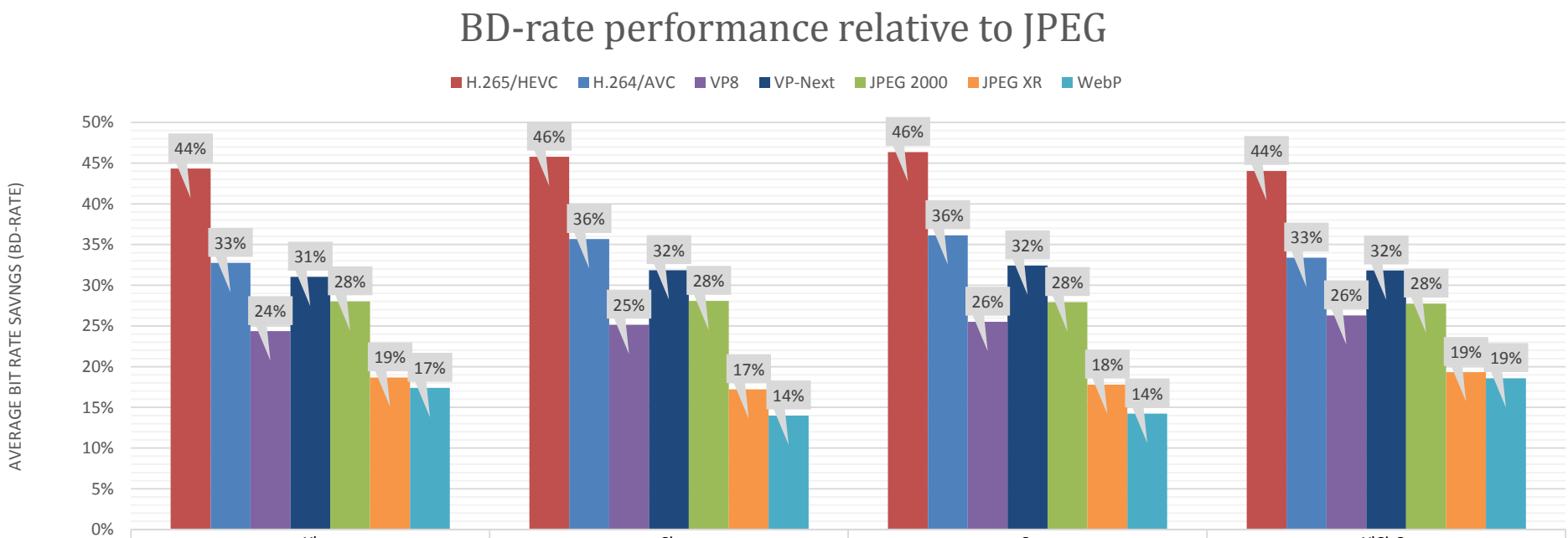
- Bit-rate saving at a PSNR value is obtained by interpolating the r-d curves
- Average bit-rate savings are obtained by averaging the savings for 100 PSNR values
- Highest savings are obtained for low bit rates
- Average bit-rate saving for all sequences are summarized below

codec	average bit rate savings relative to ...			
	H.264 / AVC	H.263+	MPEG-4	MPEG-2
H.265 / HEVC	25.1 %	40.3 %	45.2 %	57.9 %
H.264 / AVC		20.7 %	27.7 %	45.0 %
H.263+			9.5 %	31.9 %
MPEG-4				25.9 %

Intra Coding Efficiency versus JPEG Baseline

Coding efficiency comparison using available reference software versions

- Bit-rate savings relative to JPEG Baseline for JPEG test images



	Y'	Cb	Cr	Y'CbCr
H.265/HEVC	44%	46%	46%	44%
H.264/AVC	33%	36%	36%	33%
VP8	24%	25%	26%	26%
VP-Next	31%	32%	32%	32%
JPEG 2000	28%	28%	28%	28%
JPEG XR	19%	17%	18%	19%
WebP	17%	14%	14%	19%

[Nguyen, et. al., 2012]

Subjective Quality: Original (1024×704 samples, 4:2:0)

Subjective Quality (1:50 Compression): H.262 | MPEG-2 Video

Subjective Quality (1:50 Compression): H.263+

Subjective Quality (1:50 Compression): H.264/AVC

Subjective Quality (1:50 Compression): H.265/HEVC

Summary of Intra-Picture Coding in H.265 | MPEG-H HEVC

Intra-picture coding in H.265 | MPEG-H HEVC

- Partitioning of a picture
  - Partitioning into fixed-size coding tree blocks (typically 64×64 luma samples)
  - Quadtree-based partitioning into coding blocks and transform blocks
- Spatial intra prediction
  - 35 spatial intra prediction modes (35 directional modes)
  - Intra prediction is applied on transform blocks basis
- Transform and quantization
  - Integer approximation of DCT (special case: DST)
  - Supported transform sizes: 4×4, 8×8, 16×16, 32×32
  - Scalar quantizer with uniformly distributed reconstruction levels
- Entropy coding of transform coefficient levels
  - Coded block flags
  - Significance map: Last significant coefficient, significance flag for 4×4 sub-blocks, significance flags for levels
  - Absolute levels and signs
- In-loop filter: Deblocking and SAO filter

Outline

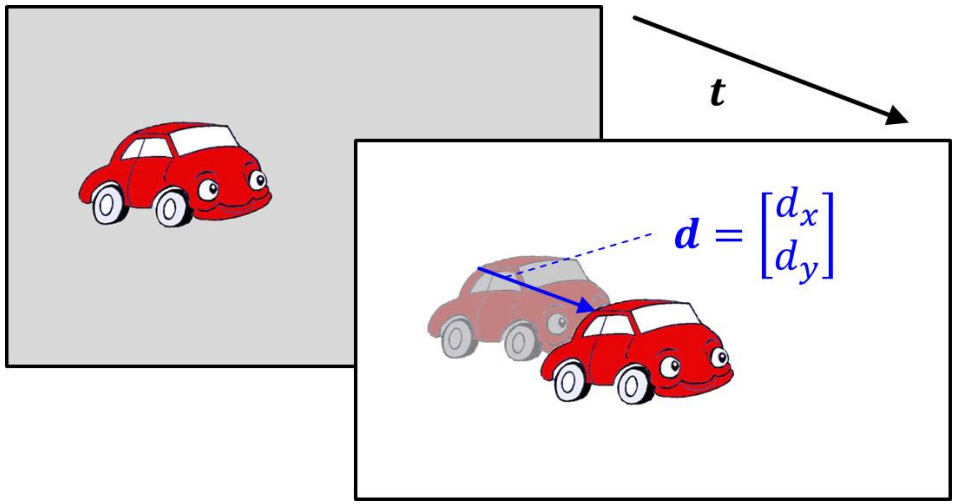
Part I: Source Coding Fundamentals

- Probability, Random Variables and Random Processes
- Lossless Source Coding
- Rate-Distortion Theory
- Quantization
- Predictive Coding
- Transform Coding

Part II: Application in Image and Video Coding

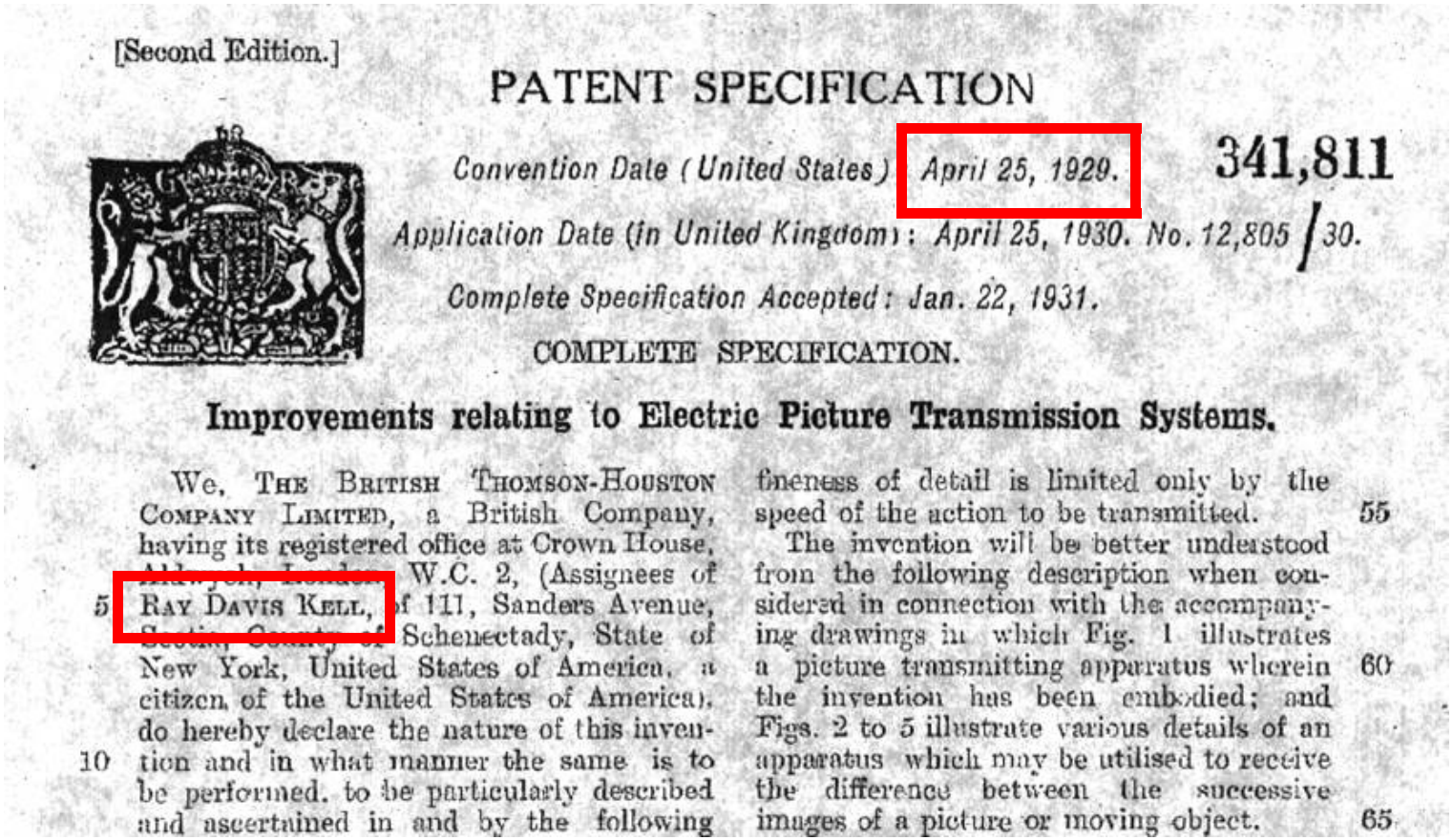
- Still Image Coding / Intra-Picture Coding
- **Hybrid Video Coding (From MPEG-2 Video to H.265/HEVC)**
  - Motion-compensated Prediction & Hybrid Video Coding
  - Encoder Control
  - Video Coding Standards

Motion-Compensated Prediction and Hybrid Video Coding



Similarities between Successive Pictures in a Video Sequence

General Idea of Hybrid Video Coding



[Second Edition.]

PATENT SPECIFICATION

Convention Date (United States) April 25, 1929. 341,811

Application Date (in United Kingdom): April 25, 1930. No. 12,805 / 30.

Complete Specification Accepted: Jan. 22, 1931.

COMPLETE SPECIFICATION.

Improvements relating to Electric Picture Transmission Systems.

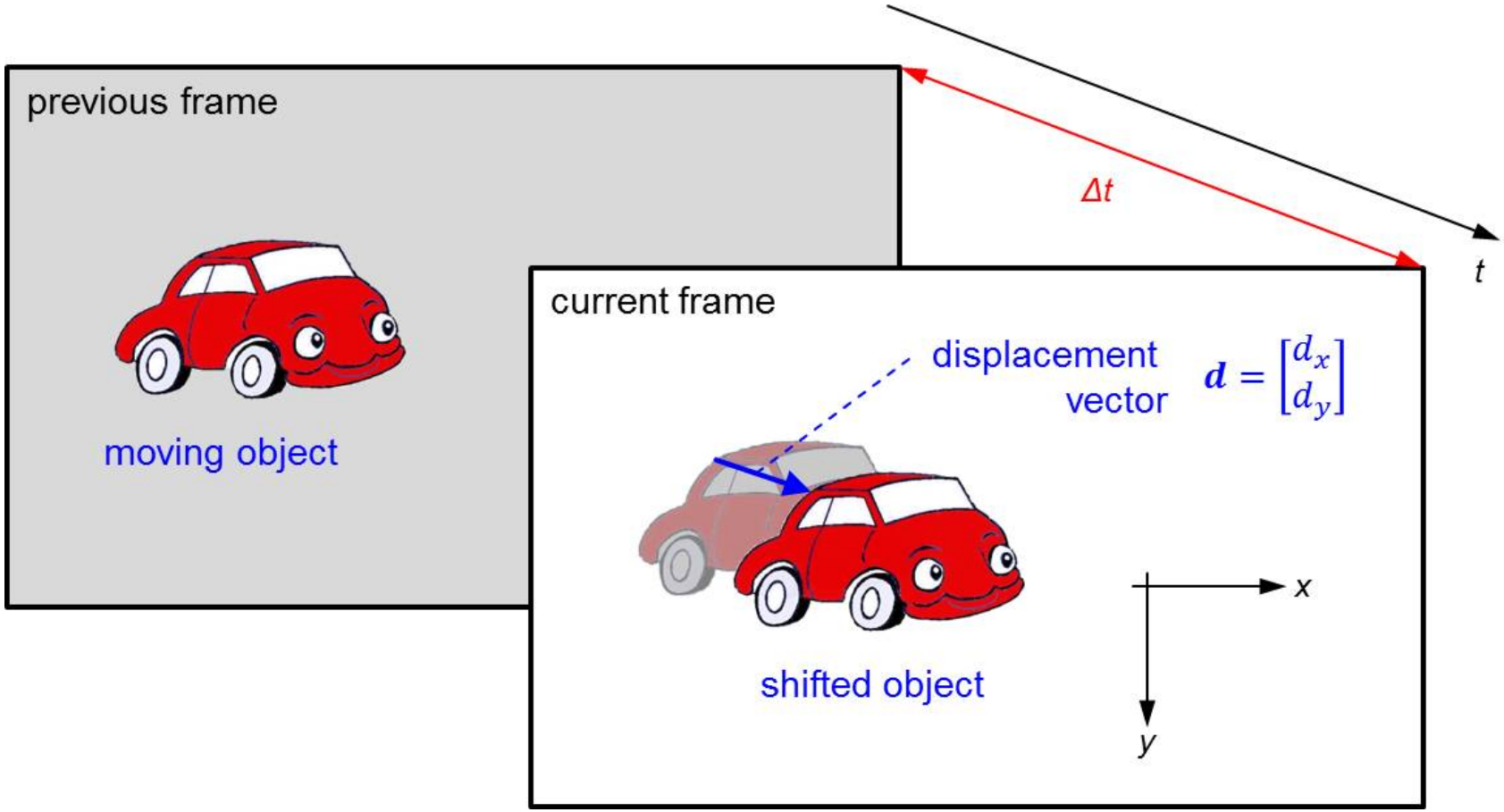
We, THE BRITISH THOMSON-HOUSTON COMPANY LIMITED, a British Company, having its registered office at Crown House, [illegible] W.C. 2, (Assignees of RAY DAVIS KELL, of 111, Sanders Avenue, [illegible] Schenectady, State of New York, United States of America, a citizen of the United States of America), do hereby declare the nature of this invention and in what manner the same is to be performed, to be particularly described and ascertained in and by the following

fineness of detail is limited only by the speed of the action to be transmitted.

The invention will be better understood from the following description when considered in connection with the accompanying drawings in which Fig. 1 illustrates a picture transmitting apparatus wherein the invention has been embodied; and Figs. 2 to 5 illustrate various details of an apparatus which may be utilised to receive the difference between the successive images of a picture or moving object.

"It has been customary in the past to transmit successive complete images of the transmitted picture. [· · ·]
In accordance with this invention, this difficulty is avoided by transmitting only the difference between successive images of the object."

Principle of Motion-Compensated Prediction



Prediction for luma signal $s[x, y, t]$ within the moving object:

$$\hat{s}[x, y, t] = s'(x - d_x, y - d_y, t - \Delta t) \tag{612}$$

Motion-Compensated Hybrid Video Coding

Hybrid video coding

- Combination of two techniques:
  - Motion-compensated prediction
  - Transform coding of prediction error
- All ITU-T and ISO/IEC video coding standards follow that principle

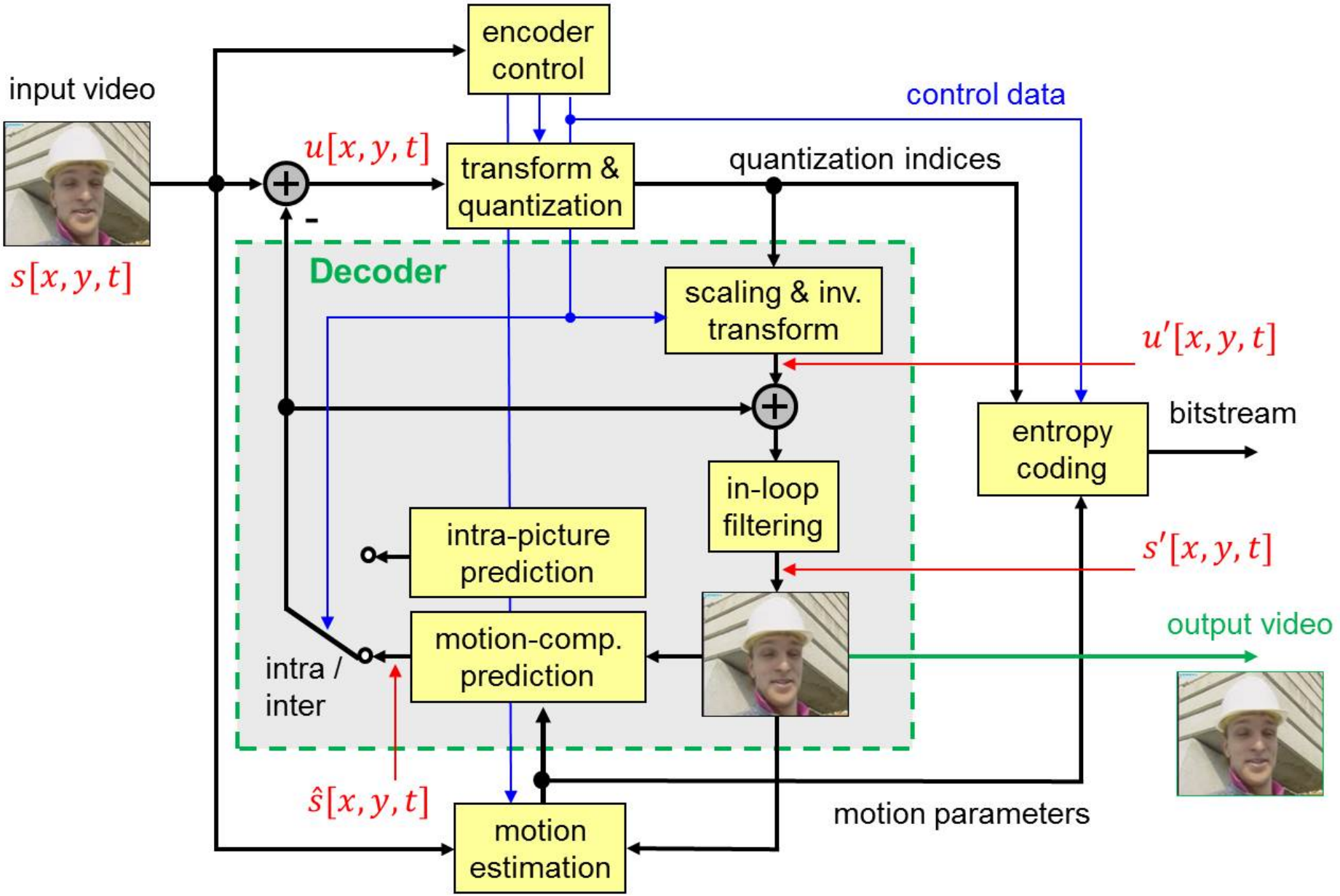
Motion-compensated prediction

- Key technique for video coding
- Substantial bit rate reduction compared to intra-picture coding

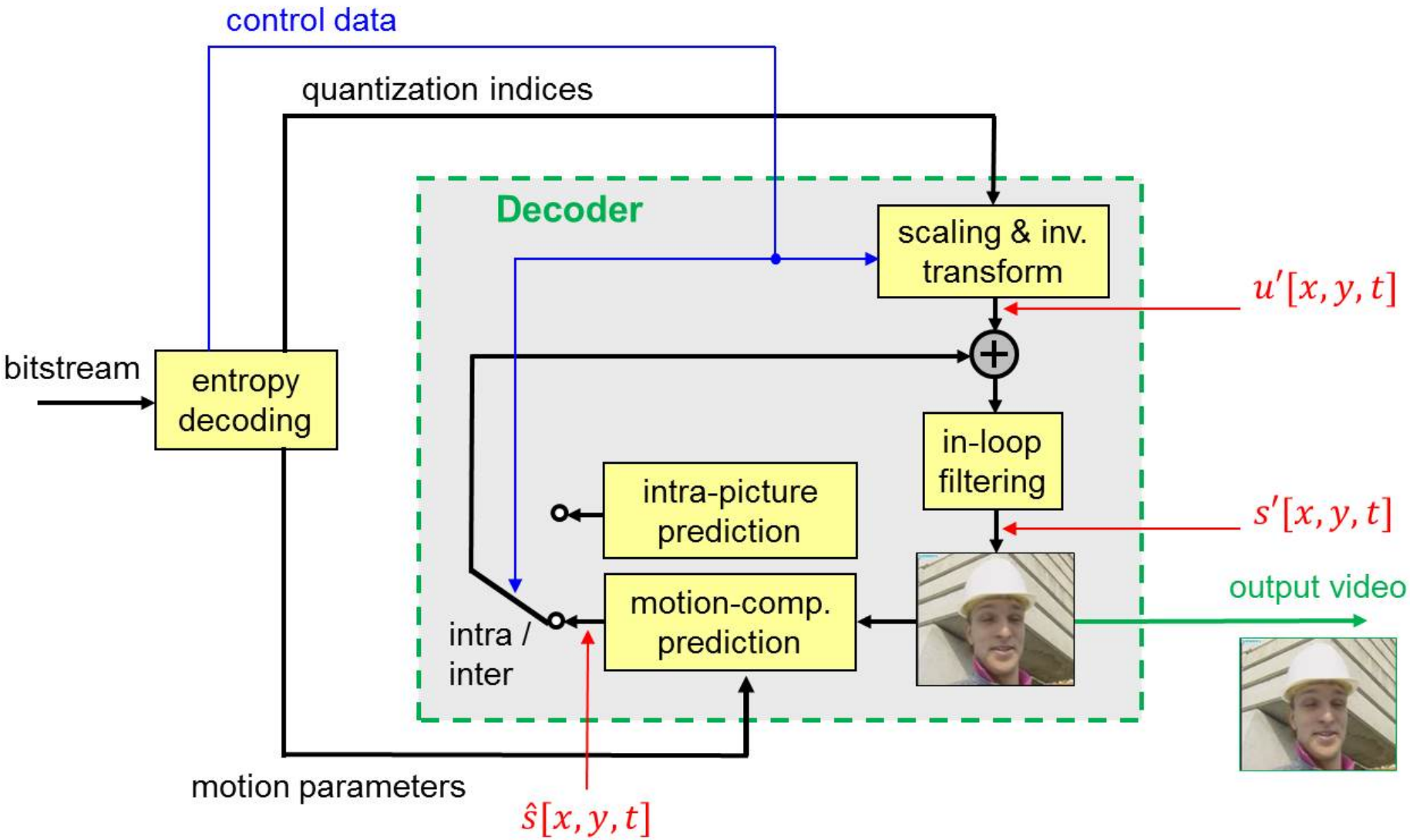
Practical hybrid video coding

- Not all parts of a picture can be efficiently predicted from a reference picture
  - Not all changes between pictures are caused by motion
  - Some parts can be occluded in reference picture
  - Complicated motion cannot be well compensated by used motion model
- For some parts, motion-compensated prediction could reduce coding efficiency
- Practical hybrid video coders allow to switch between motion-compensated prediction and intra-picture prediction

Structure of a Motion-Compensated Hybrid Video Encoder



Structure of a Motion-Compensated Hybrid Video Decoder

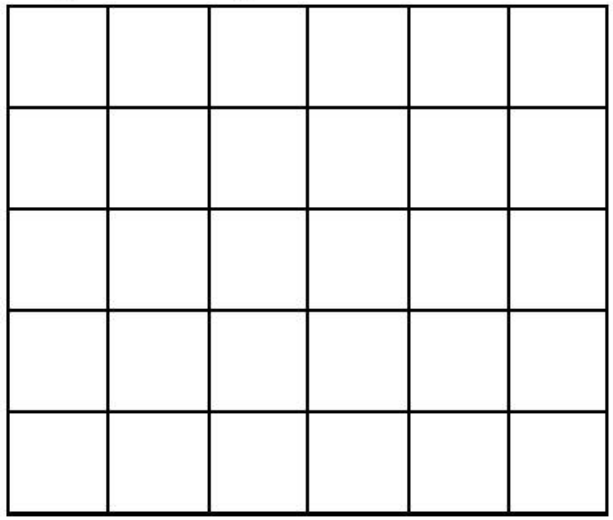


Example for Motion-Compensated Prediction

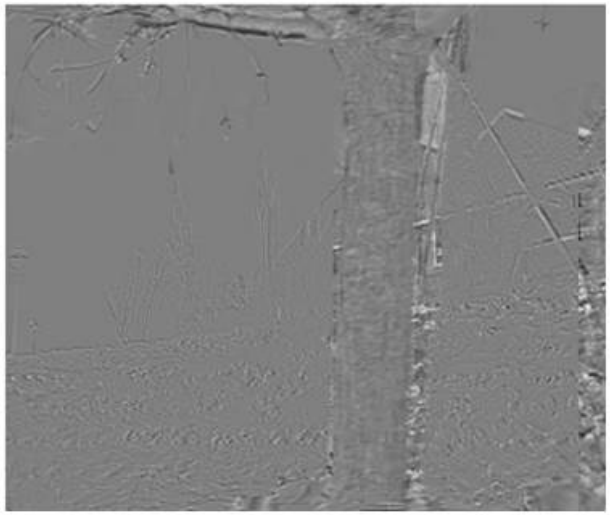
previous rec. frame $s'[x, y, t-1]$

current frame $s[x, y, t]$

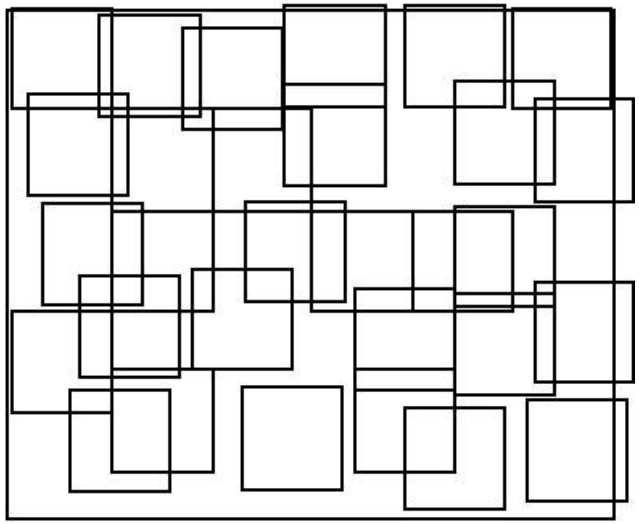
partitioning of current frame



prediction signal $\hat{s}[x, y, t]$
with motion vectors



prediction error signal
$u[x, y, t] = s[x, y, t] - \hat{s}[x, y, t]$



location of reference blocks
in previous frame

Design of Motion-Compensated Hybrid Codecs

Accuracy of motion parameters

- Full-sample or sub-sample accurate motion vectors (or motion parameters)
- For sub-sample accuracy, an interpolation filter is required

Motion models for describing the motion inside a region

- Simplest model: Translational motion $\Longrightarrow$ Used in all video coding standards
- Higher-order parametric motion models (e.g., affine motion model)

Selection of regions with constant motion (using same motion model)

- In principle, regions can have arbitrary shape $\Longrightarrow$ Need to transmit partitioning
- In today's coding standards: Square or rectangular blocks (fixed or variable size)

Selection of reference picture

- Always use previously coded/decoded picture
- Select one picture out of a set of previously coded/decoded pictures

Number of motion hypotheses

- Predict a region in a current frame using a single motion hypothesis, i.e., one reference picture with one motion vector (or motion parameter set)
- Weighted prediction of multiple motion hypotheses

Theoretical Performance Analysis of Hybrid Video Coding

Goal of analysis

- Approximate analysis of efficiency of motion-compensated video coding
- Approximate analysis of impact of displacement vector accuracy

Models for performance analysis

- Very simple signal model
- Consider coding at high bit rates
- Assume r-d optimal intra-picture coding using Gaussian model
- No consideration of bit rate for motion parameters

Further reading (papers include extended models)

[1] B. Girod, "The Efficiency of Motion-Compensating Prediction for Hybrid Coding of Video Sequences," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, no. 7, pp. 1140-1154, Aug. 1987.

[2] B. Girod, "Motion-Compensating Prediction with Fractional-Pel Accuracy," *IEEE Trans. on Communications*, vol. 41, no. 4, pp. 604-612, Apr. 1993.

[3] B. Girod, "Efficiency Analysis of Multihypothesis Motion-Compensated Prediction for Video Coding," *IEEE Trans. on Image Processing*, vol. 9, no. 2, pp. 173-183, Feb. 2000.

Model for Performance Analysis of Hybrid Video Coding

input video

$s[x,y]$

$u[x,y]$

r-d optimal intra-picture encoder

bitstream

intra-picture decoder

$u'[x,y]$

$\hat{s}[x,y] = s'[x - m_x, y - m_y]$

displacement estimate

$$\boldsymbol{m} = \boldsymbol{d} + \boldsymbol{\Delta}$$

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}$$

motion-compensated prediction

$s'[x,y]$

output video

true displacement $\boldsymbol{d}$

displacement error $\boldsymbol{\Delta}$

Model for Temporal Dependencies in Video Sequences

Continuous signal model

- Displaced signal with additive white noise

$$s_t(x,y) = s_{t-1}(x - d_x, y - d_y) + n^*(x,y)$$

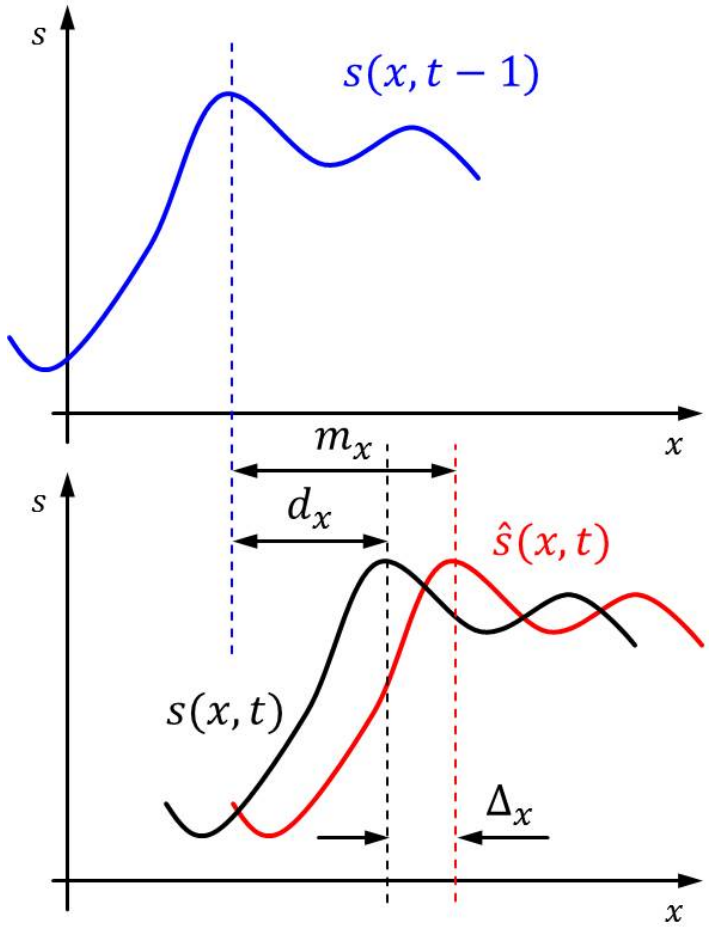
- Motion-compensated prediction

$$\hat{s}_t(x,y) = s'_{t-1}(x - m_x, y - m_y)$$

- High-rate approx.: $s'_{t-1}(x,y) = s_{t-1}(x,y)$

$$\begin{aligned} \hat{s}_t(x,y) &= s_{t-1}(x - d_x - \Delta_x, y - d_y - \Delta_y) \\ &= s_t(x - \Delta_x, y - \Delta_y) - n(x,y) \end{aligned}$$

with $n(x,y) = n^*(x - \Delta_x, y - \Delta_y)$ being the shifted noise term (same statistical properties)



- Resulting prediction error signal for current frame (omitting time index t)

$$\begin{aligned} u(x,y) &= s(x,y) - \hat{s}(x,y) = s(x,y) - s(x - \Delta_x, y - \Delta_y) + n(x,y) \\ &= s(x,y) * \big(\delta(x,y) - \delta(x - \Delta_x, y - \Delta_y)\big) + n(x,y) \end{aligned} \qquad (613)$$

Approximation of Rate-Distortion Functions

Gaussian model for signal $s(x,y)$ and prediction error $u(x,y)$

- Signal $s(x,y)$: Gaussian model provides reasonable approximation
- Prediction error $u(x,y)$: Gaussian model provides upper bound for r-d function
- Remember: Rate-distortion function $R(D)$ for 1-d Gaussian process $s(x)$ with power spectral density $\Phi_{ss}(\omega)$ is given by parametric formulation

$$D(\theta) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \min\left(\theta, \Phi_{ss}(\omega)\right) \, \mathrm{d}\omega \tag{614}$$

$$R(\theta) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \max\left(0, \frac{1}{2} \log_2 \frac{\Phi_{ss}(\omega)}{\theta}\right) \, \mathrm{d}\omega \tag{615}$$

- Extension to 2-d signal $s(x,y)$ with power spectral density $\Phi_{ss}(\omega_x, \omega_y)$

$$D(\theta) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \min\left(\theta, \Phi_{ss}(\omega_x, \omega_y)\right) \, \mathrm{d}\omega_x \, \mathrm{d}\omega_y \tag{616}$$

$$R(\theta) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \max\left(0, \frac{1}{2} \log_2 \frac{\Phi_{ss}(\omega_x, \omega_y)}{\theta}\right) \, \mathrm{d}\omega_x \, \mathrm{d}\omega_y \tag{617}$$

Conditional Power Spectral Density of Prediction Error

Power spectral density for given displacement error $\boldsymbol{\Delta} = [\Delta_x, \Delta_y]^{\mathrm{T}}$

- Prediction error signal

$$\begin{aligned} u(x,y) &= s(x,y) * \big(\delta(x,y) - \delta(x-\Delta_x, y-\Delta_y)\big) + n(x,y) \\ &= s(x,y) * f(x,y) + n(x,y) = v(x,y) + n(x,y) \end{aligned} \tag{618}$$

- Use vector notation $\boldsymbol{\omega} = [\omega_x, \omega_y]^{\mathrm{T}}$ and $\boldsymbol{\Delta} = [\Delta_x, \Delta_y]^{\mathrm{T}}$
- Power spectral density $\Phi_{uu}(\boldsymbol{\omega} \,|\, \boldsymbol{\Delta})$ for given displacement error $\boldsymbol{\Delta}$

$$\begin{aligned} \Phi_{uu}(\boldsymbol{\omega} \,|\, \boldsymbol{\Delta}) &= \Phi_{vv}(\boldsymbol{\omega} \,|\, \boldsymbol{\Delta}) + \Phi_{nn}(\boldsymbol{\omega}) \\ &= \Phi_{ss}(\boldsymbol{\omega}) \cdot F(\boldsymbol{\omega}) \cdot F^*(\boldsymbol{\omega}) + \Phi_{nn}(\boldsymbol{\omega}) \\ &= \Phi_{ss}(\boldsymbol{\omega}) \cdot \left(1 - e^{-j\boldsymbol{\omega}\boldsymbol{\Delta}^{\mathrm{T}}}\right) \left(1 - e^{j\boldsymbol{\omega}\boldsymbol{\Delta}^{\mathrm{T}}}\right) + \Phi_{nn}(\boldsymbol{\omega}) \\ &= 2 \cdot \Phi_{ss}(\boldsymbol{\omega}) \cdot \left(1 - \frac{e^{-j\boldsymbol{\omega}\boldsymbol{\Delta}^{\mathrm{T}}} + e^{j\boldsymbol{\omega}\boldsymbol{\Delta}^{\mathrm{T}}}}{2}\right) + \Phi_{nn}(\boldsymbol{\omega}) \\ &= 2 \cdot \Phi_{ss}(\boldsymbol{\omega}) \cdot \big(1 - \cos(\boldsymbol{\omega}\boldsymbol{\Delta})\big) + \Phi_{nn}(\boldsymbol{\omega}) \\ &= 2 \cdot \Phi_{ss}(\boldsymbol{\omega}) \cdot \Big(1 - \Re\Big\{e^{-j\boldsymbol{\omega}\boldsymbol{\Delta}}\Big\}\Big) + \Phi_{nn}(\boldsymbol{\omega}) \end{aligned} \tag{619}$$

Power Spectral Density of Prediction Error

Power spectral density $\Phi_{uu}(\boldsymbol{\omega})$ depends on displacement error pdf

- Power spectral density $\Phi_{uu}(\boldsymbol{\omega})$ of prediction error

$$\begin{aligned}\Phi_{uu}(\boldsymbol{\omega}) &= E\{\Phi_{uu}(\boldsymbol{\omega}\,|\,\boldsymbol{\Delta})\} \\ &= 2\cdot\Phi_{ss}(\boldsymbol{\omega})\cdot\Big(1-\Re\Big\{E\Big\{e^{-j\boldsymbol{\omega}\boldsymbol{\Delta}}\Big\}\Big\}\Big)+\Phi_{nn}(\boldsymbol{\omega}) \\ &= 2\cdot\Phi_{ss}(\boldsymbol{\omega})\cdot(1-\Re\{P(\boldsymbol{\omega})\})+\Phi_{nn}(\boldsymbol{\omega}) \qquad (620)\end{aligned}$$

- The spectrum $P(\boldsymbol{\omega})$ is the Fourier transform of the probability density function $f_{\boldsymbol{\Delta}}(\boldsymbol{\Delta})$ for the displacement error

$$\begin{aligned}P(\boldsymbol{\omega}) &= E\Big\{e^{-j\boldsymbol{\omega}\boldsymbol{\Delta}}\Big\} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{\boldsymbol{\Delta}}(\boldsymbol{\Delta})\,e^{-j\boldsymbol{\omega}\boldsymbol{\Delta}}\,\mathrm{d}\boldsymbol{\Delta} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{\boldsymbol{\Delta}}(\Delta_x,\Delta_y)\,e^{-j(\omega_x\Delta_x+\omega_y\Delta_y)}\,\mathrm{d}\Delta_x\,\mathrm{d}\Delta_y \qquad (621)\end{aligned}$$

Model for Distribution of Displacement Error

Motion estimate with given maximum accuracy

- Maximum displacement error is given by motion vector accuracy

$$\Delta_{\max} = 2^{-(1+\beta)} \tag{622}$$

with $\beta = 0$: Integer-sample accurate motion vectors
$\beta = 1$: Half-sample accurate motion vectors
$\beta = 2$: Quarter-sample accurate motion vectors

- Displacement error components are uniformly distributed inside $[-\Delta_{\max}, \Delta_{\max}]$

$$f_{\boldsymbol{\Delta}}(\Delta_x, \Delta_y) = \begin{cases} \frac{1}{4}\,\Delta_{\max}^{-2} & : \ |\Delta_x| \le \Delta_{\max} \text{ and } |\Delta_y| \le \Delta_{\max} \\ 0 & : \ \text{otherwise} \end{cases} \tag{623}$$

- Resulting spectrum $P(\omega_x, \omega_y)$

$$\begin{aligned} P(\omega_x, \omega_y) &= \frac{1}{4 \cdot \Delta_{\max}^2} \int_{-\Delta_{\max}}^{\Delta_{\max}} \int_{-\Delta_{\max}}^{\Delta_{\max}} e^{-j\omega_x \Delta_x} \cdot e^{-j\omega_y \Delta_y} \, \mathrm{d}\Delta_x \, \mathrm{d}\Delta_y \\ &= \operatorname{sinc}\big(\Delta_{\max} \cdot \omega_x\big) \cdot \operatorname{sinc}\big(\Delta_{\max} \cdot \omega_y\big) \\ &= \operatorname{sinc}\big(2^{-(1+\beta)} \cdot \omega_x\big) \cdot \operatorname{sinc}\big(2^{-(1+\beta)} \cdot \omega_y\big) \end{aligned} \tag{624}$$

Model for Image Signal

Model based on assumption of autocorrelation function $R_{ss}(\Delta_x, \Delta_y)$

- Isotropic autocorrelation function (for typical image signals: $\varrho \approx 0.9$)

$$R_{ss}(\Delta_x, \Delta_y) = E\{s(x,y) \cdot s(x-\Delta_x, y-\Delta_y)\} = \sigma_s^2 \cdot \varrho^{\sqrt{\Delta_x^2+\Delta_y^2}} \tag{625}$$

- Power spectral density $\Phi_{ss}(\omega_x, \omega_y)$ for image signal

$$\begin{aligned} \Phi_{ss}(\omega_x, \omega_y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R_{ss}(\Delta_x, \Delta_y) \cdot e^{-j(\omega_x \Delta_x + \omega_y \Delta_y)} \, \mathrm{d}\Delta_x \, \mathrm{d}\Delta_y \\ &= K \cdot \sigma_s^2 \cdot \left(1 + \frac{\omega_x^2 + \omega_y^2}{(\ln \varrho)^2}\right)^{-\frac{3}{2}} \end{aligned} \tag{626}$$

- Consider band-limited image signal (sampled at Nyquist rate)

$$\Phi_{ss}(\omega_x, \omega_y) = \begin{cases} K \cdot \sigma_s^2 \cdot \left(1 + \frac{\omega_x^2+\omega_y^2}{(\ln \varrho)^2}\right)^{-\frac{3}{2}} & : \quad |\omega_x| \leq \pi \text{ and } |\omega_y| \leq \pi \\ 0 & : \quad \text{otherwise} \end{cases} \tag{627}$$

where the constant K has to be determined by

$$\sigma_s^2 = \frac{1}{4\pi^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \Phi_{ss}(\omega_x, \omega_y) \, \mathrm{d}\omega_x \, \mathrm{d}\omega_y \tag{628}$$

Complete Model for Power Spectral Density of Prediction Error

Apply models for displacement error pdf and image signal

- Remember

$$\Phi_{uu}(\omega_x, \omega_y) = 2 \cdot \Phi_{ss}(\omega_x, \omega_y) \cdot (1 - \Re\{P(\omega_x, \omega_y)\}) + \Phi_{nn}(\omega_x, \omega_y) \tag{629}$$

- Assume constant noise $n(x, y)$ inside $[-\pi, \pi] \times [-\pi, \pi]$

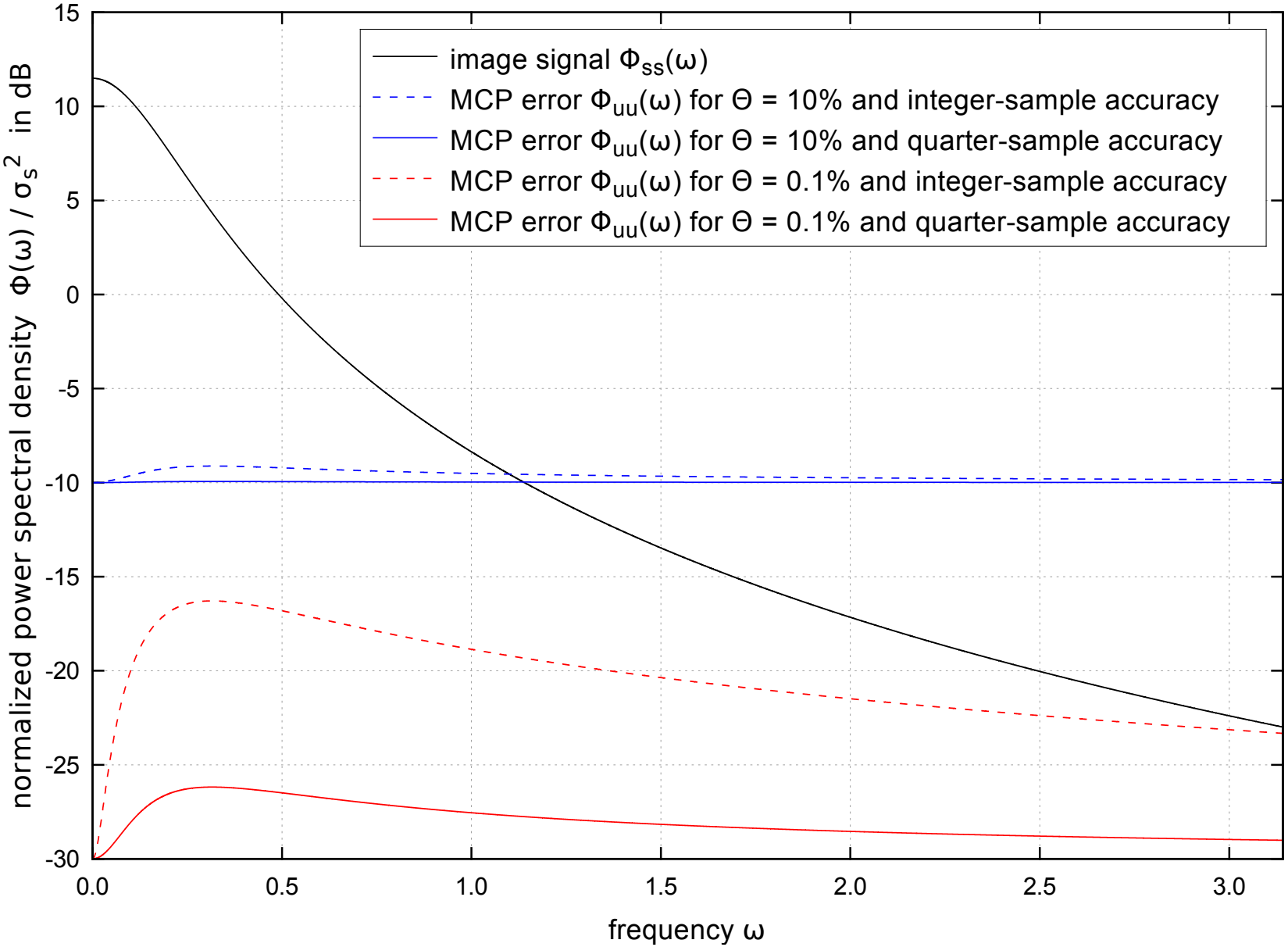
$$\Phi_{nn}(\omega_x, \omega_y) = \sigma_n^2 = \Theta \cdot \sigma_s^2 \tag{630}$$

  with Θ being the ratio of noise and signal power

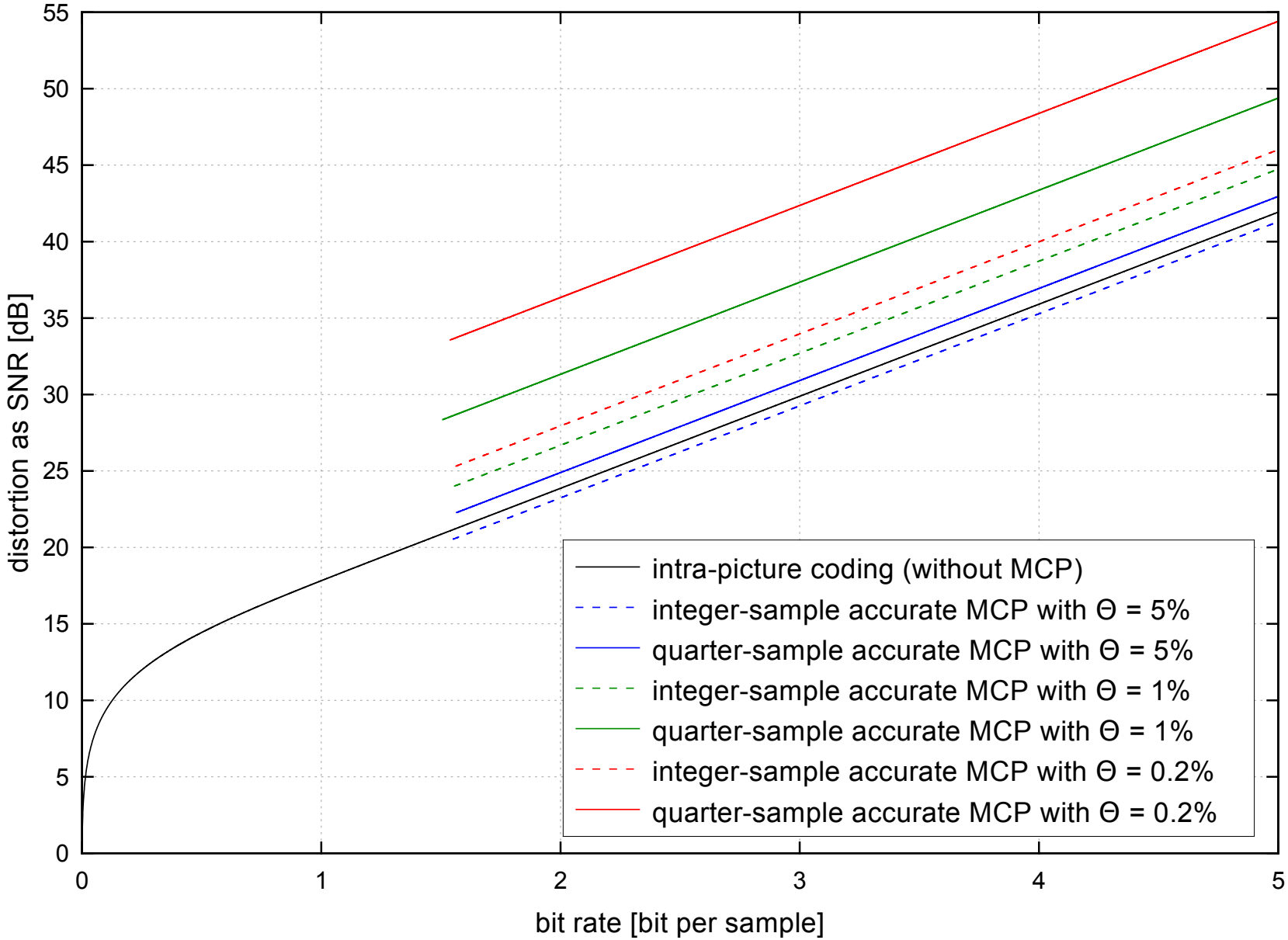
- Inserting the models for $\Phi_{ss}(\omega_x, \omega_y)$, $\Phi_{nn}(\omega_x, \omega_y)$ and $P(\omega_x, \omega_y)$ yields the normalized power spectral density inside interval $[-\pi, \pi] \times [-\pi, \pi]$

$$\begin{aligned} \frac{\Phi_{uu}(\omega_x, \omega_y)}{\sigma_s^2} &= K \cdot \left(1 + \frac{\omega_x^2 + \omega_y^2}{(\ln \varrho)^2}\right)^{-\frac{3}{2}} \cdot \\ &\quad \left(1 - \mathrm{sinc}(2^{-(1+\beta)} \cdot \omega_x) \cdot \mathrm{sinc}(2^{-(1+\beta)} \cdot \omega_x)\right) + \Theta \end{aligned} \tag{631}$$

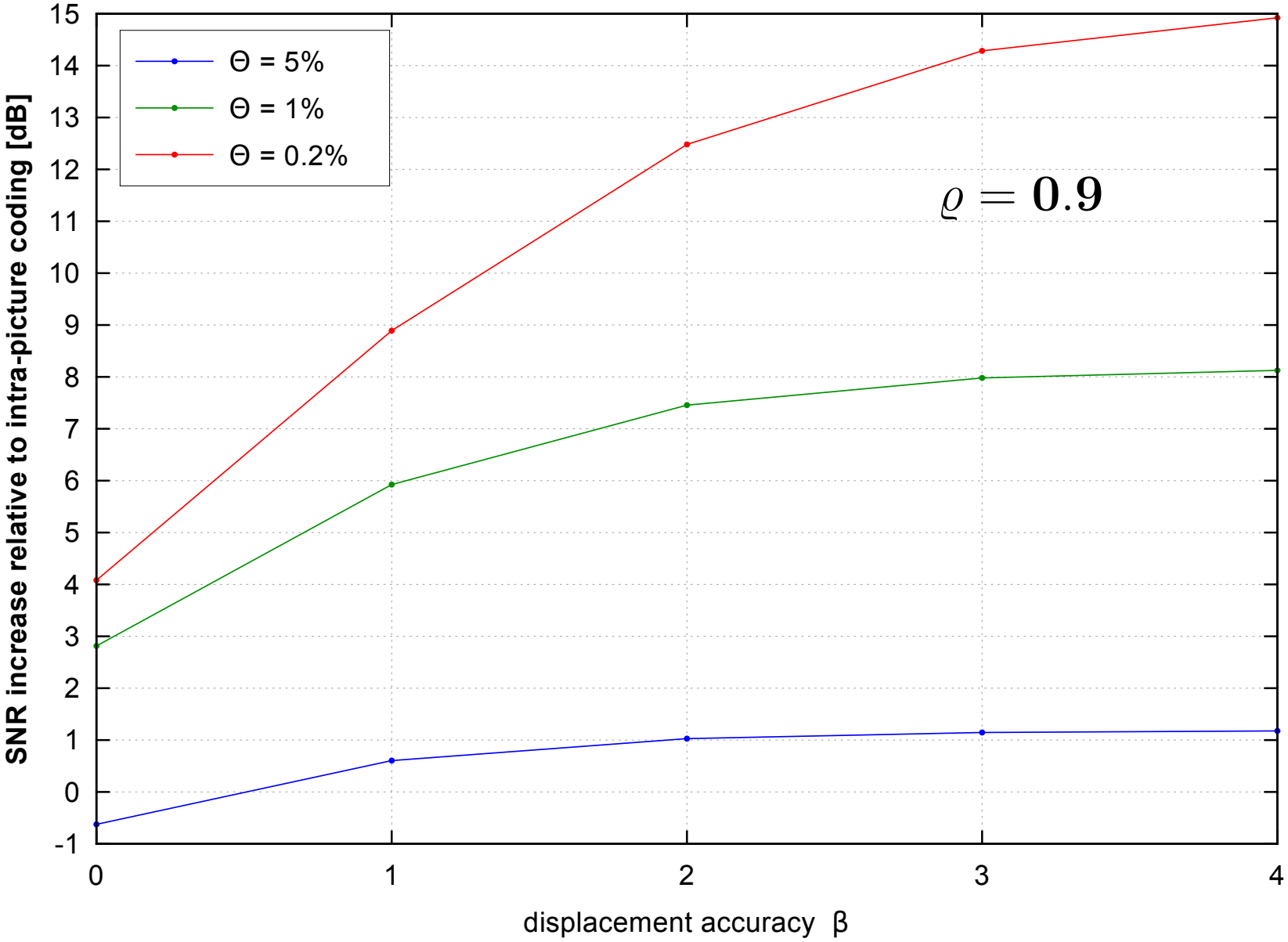
Power Spectral Densities for 1-D Signal with $\varrho = 0.8$



High-Rate Performance of MCP for 2-d Signals with $\varrho = 0.9$



Impact of Displacement Accuracy and Noise at High Rates



Interpretation of Theoretical Results

Theoretical analysis showed

- Motion-compensated prediction typically improves coding efficiency
- Efficiency of motion-compensated prediction increases with increasing accuracy of displacement vectors
- Accuracy increase is mainly useful for video signals with low noise

Motion-compensated prediction in practice

- Bit rate required for transmitting displacement vectors increases with increasing displacement vector accuracy

  $\Longrightarrow$ There is an "optimal" displacement vector accuracy for a given noise level

  $\Longrightarrow$ For typical sequences, an accuracy higher than quarter-sample displacements does not provide noticeable coding efficiency gains
  (for low noise data: eight-sample accuracy can provide gain)

- Interpolation filters are required for sub-sample accurate MCP

  $\Longrightarrow$ Interpolation filters have a significant impact on coding efficiency

  $\Longrightarrow$ More accurate interpolation filters require higher complexity

Displacement Vector Accuracy in Video Coding Standards

H.262 | MPEG-2 Video, H.263 and MPEG-4 Visual (Version 1)

- Half-sample accurate displacement vectors
- Prediction signal at half-sample positions is obtained by bi-linear interpolation

Advanced Simple profile of MPEG-4 Visual

- Quarter-sample accurate displacement vectors
- Prediction signal at half-sample positions: Separable 8-tap FIR filter
- Prediction signal at quarter-sample positions: Bi-linear interpolation of integer- and half-sample positions

H.264 | MPEG-4 AVC

- Quarter-sample accurate displacement vectors
- Prediction signal at half-sample positions: Separable 6-tap FIR filter
- Prediction signal at quarter-sample positions: Averaging of two integer- and half-sample positions

H.265 | MPEG-H HEVC

- Quarter-sample accurate displacement vectors
- Prediction signal at half- and quarter sample positions: Separable 8- and 7-tap FIR filter (depending on sub-sample shift)

Experimental Analysis of MCP and Displacement Accuracy

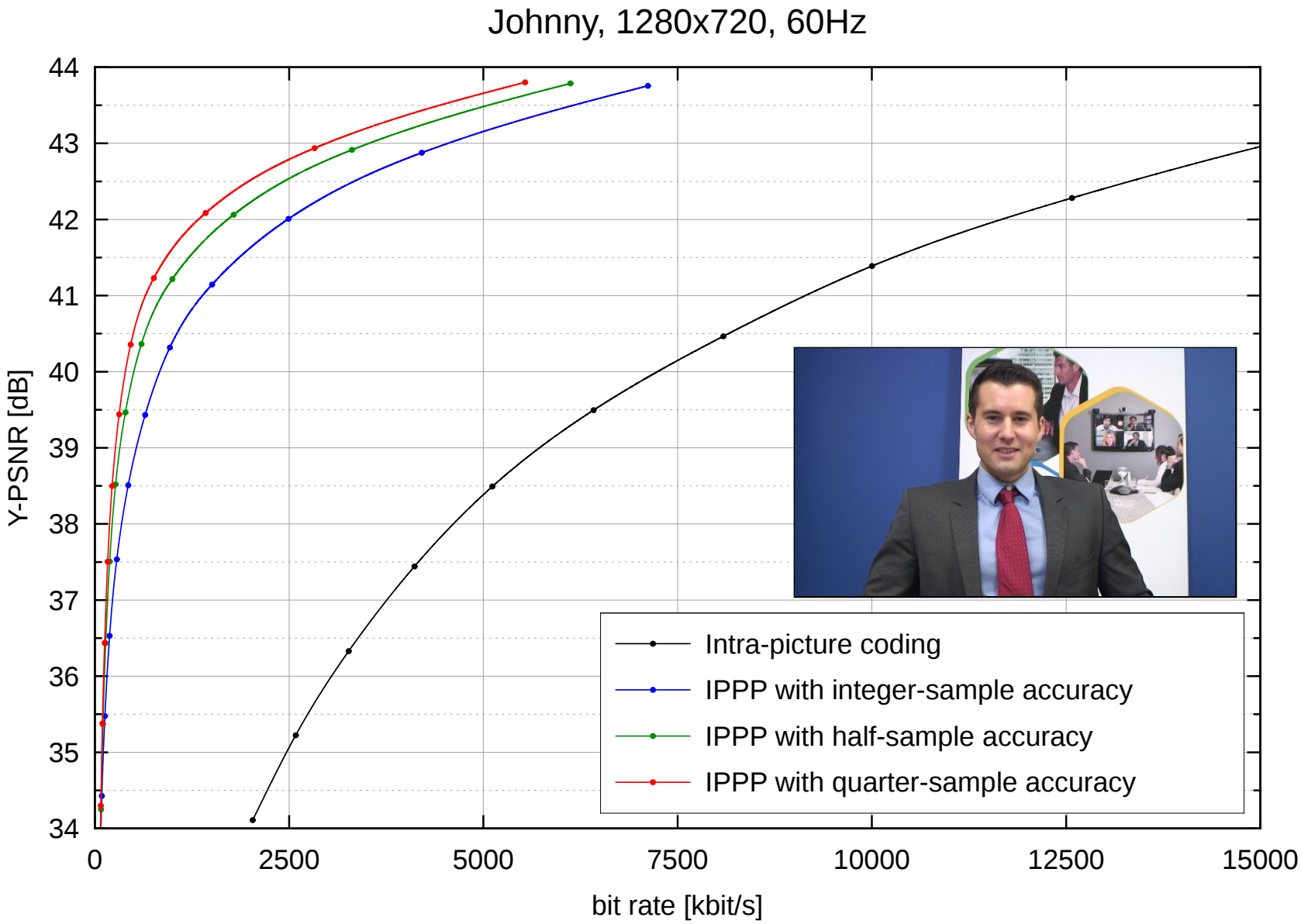
Comparison of different codecs

- HEVC intra-only coding
- HEVC inter coding (quarter-sample accuracy)
- modified HEVC with half-sample accuracy (adapted motion vector coding)
- modified HEVC with integer-sample accuracy (adapted motion vector coding)

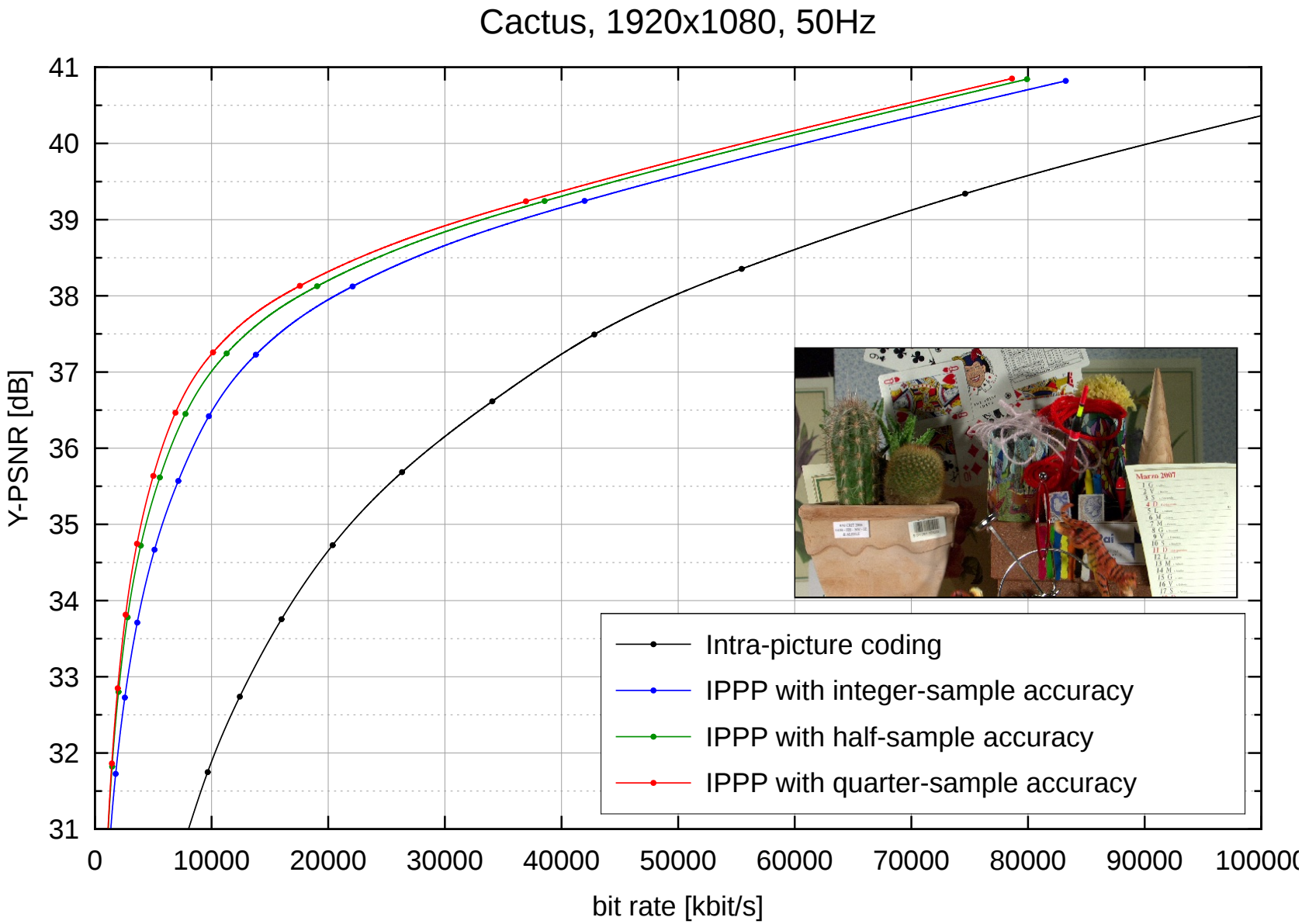
Configuration

- 12 test sequences
  - 6 sequences in 720p with video conferencing content
  - 6 sequences in 1080p with entertainment content
- Only the first picture is intra-picture coded (except for intra-only coding)
- Picture are coded in display order (IPPP coding structure)
- Previous picture is used as reference picture
- Intra blocks are allowed in inter-picture coded frames
- Fixed quantization parameter
- All codecs are based on same HEVC encoder version
- Lagrangian encoder control

Efficiency of Motion-Compensated Prediction – “Johnny”



Efficiency of Motion-Compensated Prediction – “Cactus”



Efficiency of Motion-Compensated Prediction – Summary

Comparison of HEVC intra-picture coding and HEVC-based motion-compensated coding with different motion vector accuracy

- Bit-rate saving at a PSNR value is obtained by interpolating the r-d curves
- Average bit-rate savings are obtained by averaging the savings for 100 PSNR values
- Average bit-rate savings for all sequences are summarized below

codec version	average bit rate savings relative to ...		
	half-sample	integer-sample	intra-picture
quarter-sample	10.3 %	29.9 %	77.7 %
half-sample		22.8 %	75.6 %
integer-sample			69.7 %

Motion Models for Motion-Compensated Prediction

Translational motion in image plane

- Motion of a region is described by 2-d displacement vector $\mathbf{d} = [d_x, d_y]^{\mathrm{T}}$

$$\boldsymbol{d}(x, y) = \boldsymbol{d} = \text{const} \tag{632}$$

- Used in all video coding standards of ITU-T and ISO/IEC
- Can only describe small amount of "real motion"

Higher order motion models

- Motion in image plane is caused by motion in 3-d space
- Assuming reasonable restrictions for the motion in 3-d space (e.g. rigid body motion), motion in image plane can be described by a parametric model

$$\boldsymbol{d}(x, y) = f(\boldsymbol{a}, x, y) \qquad \text{with } \boldsymbol{a} \text{ being a constant parameter vector} \tag{633}$$

- Advantage of higher order motion models

  $\Longrightarrow$ Better approximation of "real motion" than translational model
- Disadvantages of higher order motion models

  $\Longrightarrow$ Increased bit rate for transmitting motion parameters

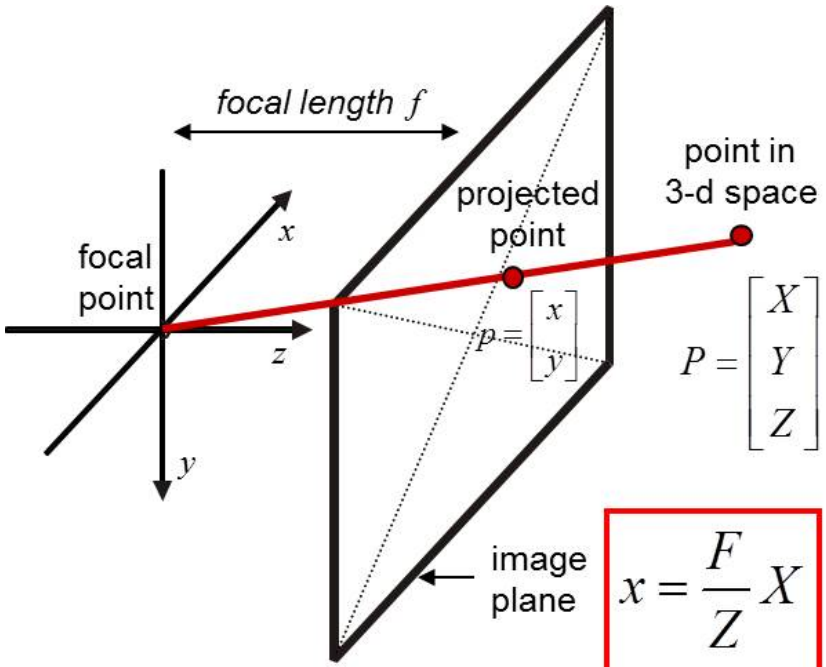
  $\Longrightarrow$ Increased complexity and decreased robustness of motion estimation

Models for Projection of 3-D Space onto Image Plane

Projection of 3-d world onto 2-d image plane by camera lens

- **Perspective projection**: Neglecting image distortions and blurring
- **Orthographic projection**: All rays are parallel to image plane (valid for $Z \gg f$)

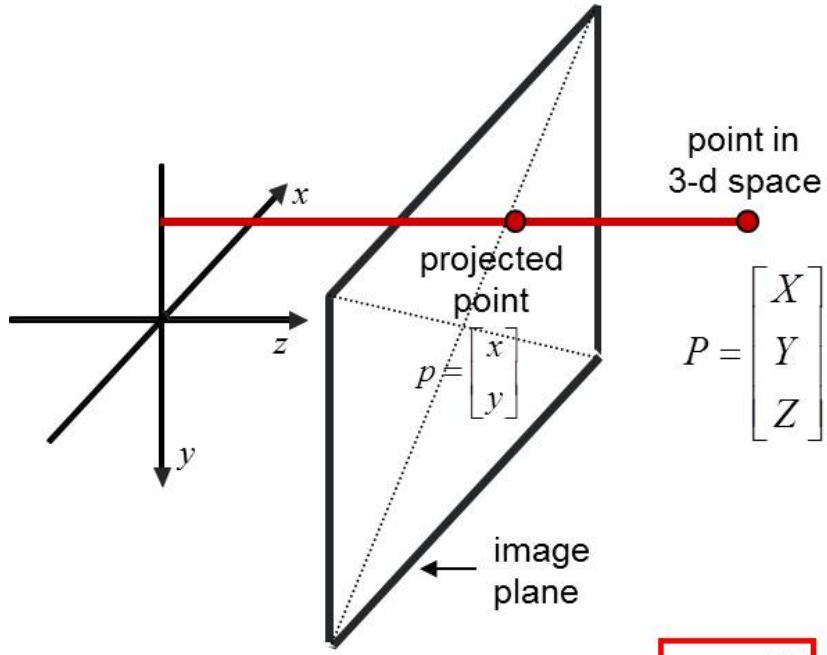
perspective projection model



$$x = \frac{F}{Z} X$$

$$y = \frac{F}{Z} Y$$

orthographic projection model



Valid approximation for $Z >> f$

$$x = X$$

$$y = Y$$

Perspective Motion Model

Perspective model for motion in image plane

- Mathematical model for motion field $\boldsymbol{d} = [d_x, d_y]^{\mathrm{T}}$ inside a region

$$d_x(x,y) = \frac{a_0 + a_1 \cdot x + a_2 \cdot y}{1 + c_1 \cdot x + c_2 \cdot y} \tag{634}$$

$$d_y(x,y) = \frac{b_0 + b_1 \cdot x + b_2 \cdot y}{1 + c_1 \cdot x + c_2 \cdot y} \tag{635}$$

- Assumptions / restrictions:
  - Rotation and scaling of rigid body in 3-d space, but no translation
  - Translation, rotation and scaling of a plane in 3-d space
- Advantage:
  - Corresponds to perspective projection model
- Disadvantage:
  - Hyperbolic model is non-linear with respect to motion parameters
  - Difficult to estimate

Orthographic Motion Models

Motion models with different degrees of freedom

- **Translational model**: Translation parallel to image plane (or in image plane)

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} \tag{636}$$

- **4-parameter model**: Translation, zoom and rotation in image plane

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} a_0 + a_1 \cdot x + a_2 \cdot y \\ b_0 - a_2 \cdot x + a_1 \cdot y \end{bmatrix} \tag{637}$$

- **Affine model**: Translation and rotation of a plane in 3-d space

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} a_0 + a_1 \cdot x + a_2 \cdot y \\ b_0 + b_1 \cdot x + b_2 \cdot y \end{bmatrix} \tag{638}$$

- **Parabolic model**: Includes approximation of perspective distortions

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} a_0 + a_1 \cdot x + a_2 \cdot y + a_3 \cdot x^2 + a_4 \cdot y^2 + a_5 \cdot xy \\ b_0 + b_1 \cdot x + b_2 \cdot y + b_3 \cdot x^2 + b_4 \cdot y^2 + b_5 \cdot xy \end{bmatrix} \tag{639}$$

- Orthographic models are linear with respect to parameters (easier to estimate)
- Can be interpreted as Taylor expansions of perspective motion model

Illustration of Impact of Affine Parameters

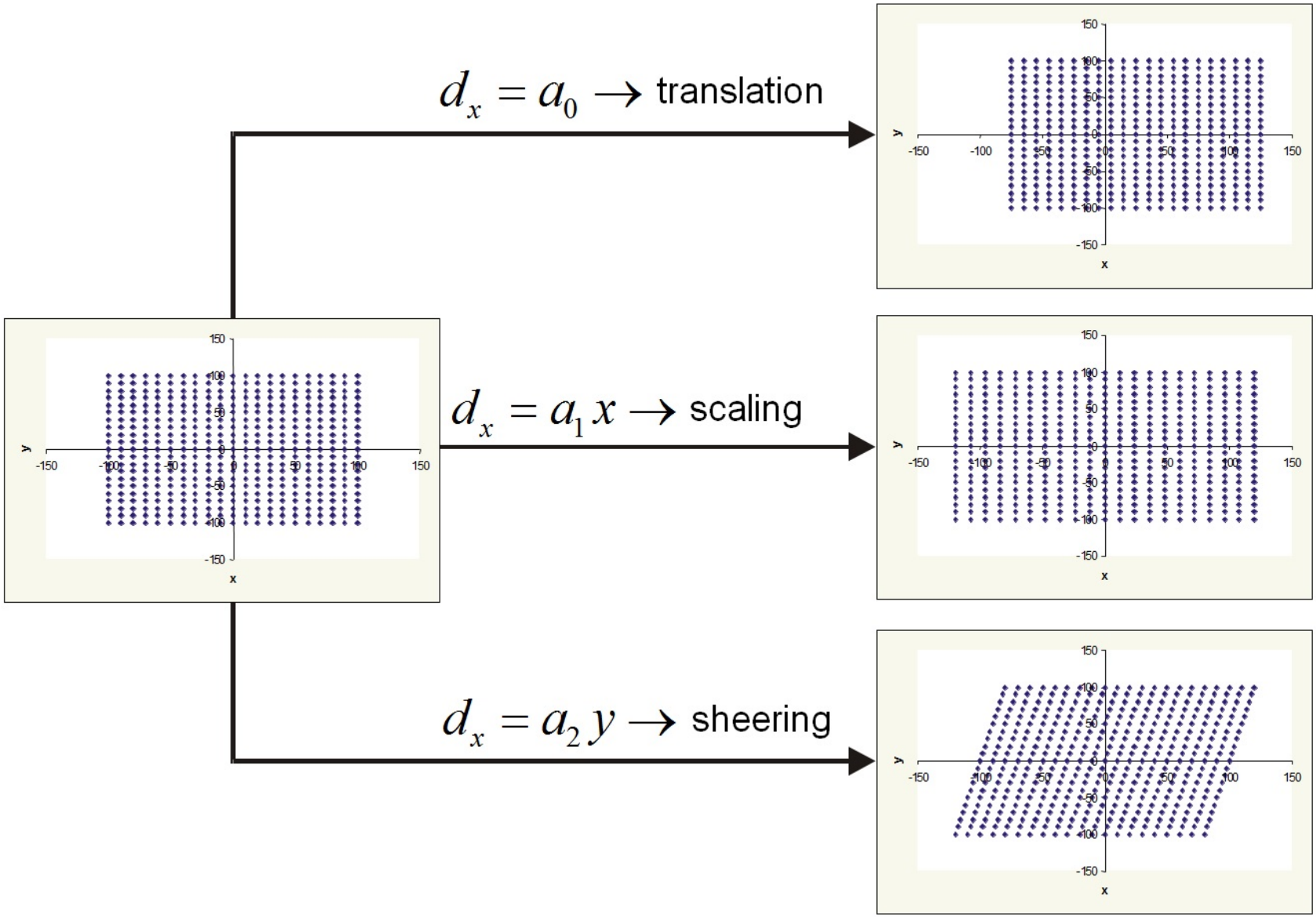


Illustration of Impact of Parabolic Parameters

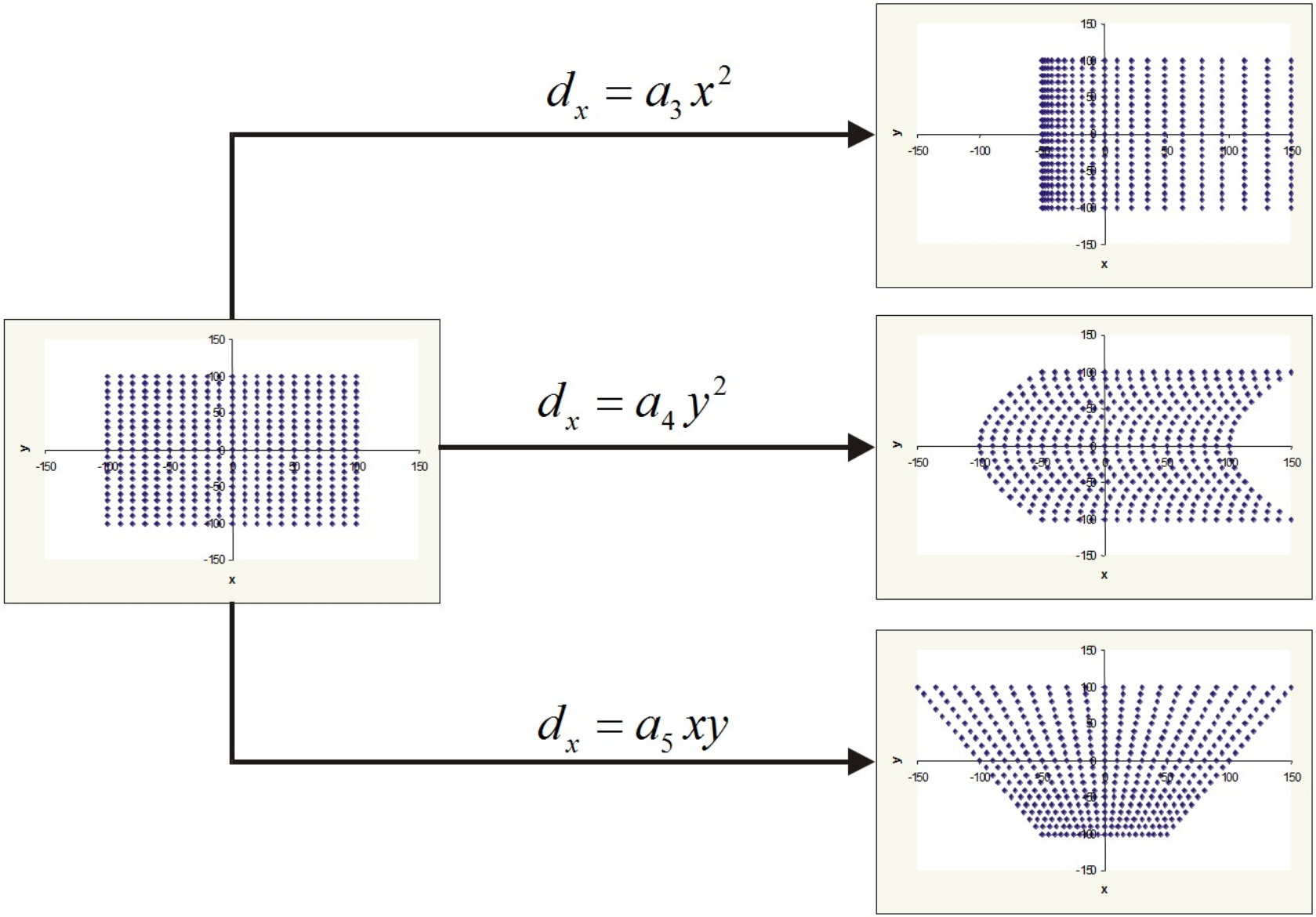
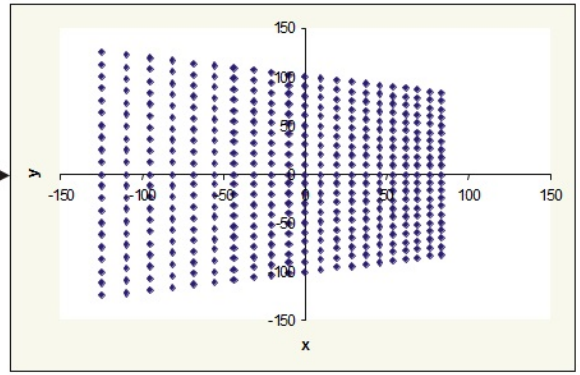
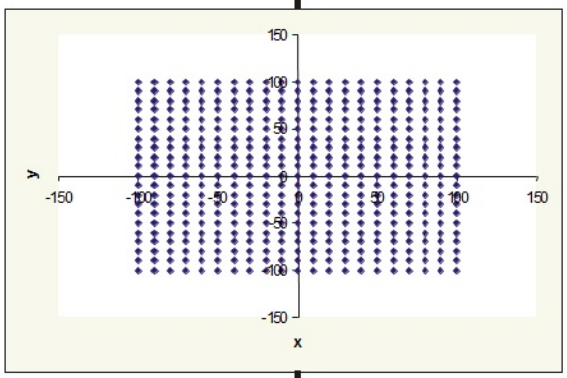


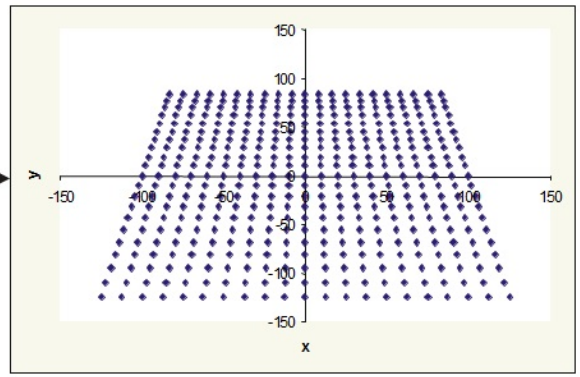
Illustration of Impact of Perspective Parameters

$$d_x = \frac{1}{1 + c_1 x}$$





$$d_x = \frac{1}{1 + c_2 y}$$



Usage of Higher Order Motion Models for Video Coding

Video coding standards of ITU-T and ISO/IEC

- All standards use simple translational motion model
- Exception: MPEG-4 Visual supports also affine and perspective model for global motion compensation (single motion model for background)
  - Difficult to estimate suitable motion parameters for background
  - Rarely used in practice

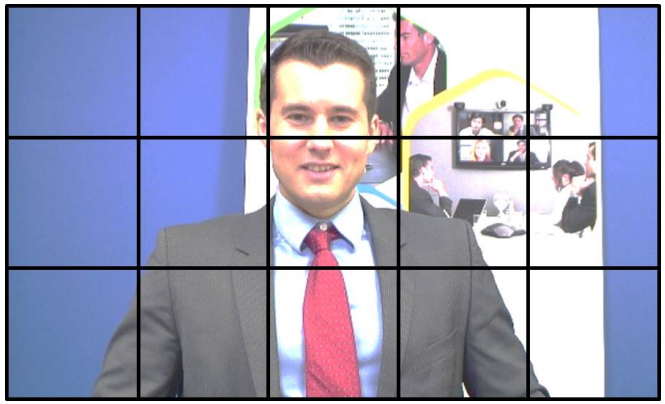
Scientific papers

- Higher-order motion models for background motion (or large regions)
- Higher-order motion models for block motion
- Example [Lakshman, et al, 2010]
  - Switching between translational and affine motion model on block basis
  - On average, bit rate savings of 1-2% compared to translational mode
  - Maximum bit rate savings of about 4%

Picture Partitioning for Motion-Compensated Prediction

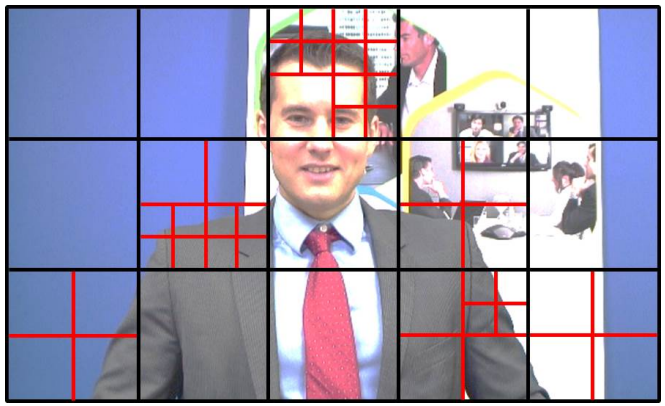
Partitioning into fixed-size square blocks

- Partitioning does not need to be transmitted
- Low flexibility
- H.262 | MPEG-2 Video:
  - One motion vector per 16×16 macroblocks



Partitioning into variable-size square blocks

- Partitioning has to be transmitted
- Simple approach: Quadtree partitioning
- Increased flexibility
- H.263 and MPEG-4 Visual:
  - 16×16 or 8×8 blocks for MCP
- H.264 | MPEG-4 AVC:
  - 16×16 to 4×4 blocks + non-square blocks
- H.265 | MPEG-H HEVC:
  - 64×64 to 8×8 blocks + non-square blocks

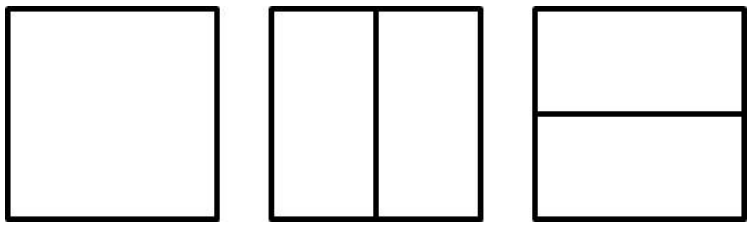


Non-Square Blocks for Motion-Compensated Prediction

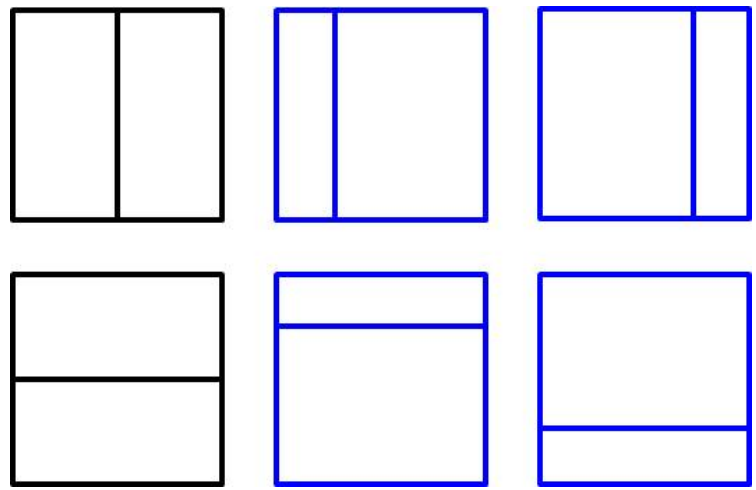
Partitioning into variable-size rectangular blocks

- Typically combined with quadtree-based partitioning into square blocks
- Square blocks can also be partitioned into 2 rectangular blocks
- Flexibility is further increased (side information rate is also increased)
- H.264 | MPEG-4 AVC:
  - Symmetric vertical and horizontal subdivision (for 16×16 and 8×8 blocks)
- H.265 | MPEG-H HEVC:
  - Symmetric and asymmetric subdivisions (for 64×64 to $8\times8/16\times16$ blocks)

Partitioning into non-square blocks for H.264/AVC (for 16×16 and 8×8)



Partitioning into non-square blocks for H.265/HEVC (for 64×64 to $8\times8/16\times16$)



Impact of Block Sizes for Motion-Compensated Prediction

Framework for analysis

- HEVC codec with partially disabling certain block sizes
- IPPP coding structure with quarter-sample accurate motion vectors
- Previous frame is used as reference frame

Side effect of restricting block sizes for MCP

- Impact on applicable transform sizes
- Transforms are not applied across coding unit boundaries in HEVC

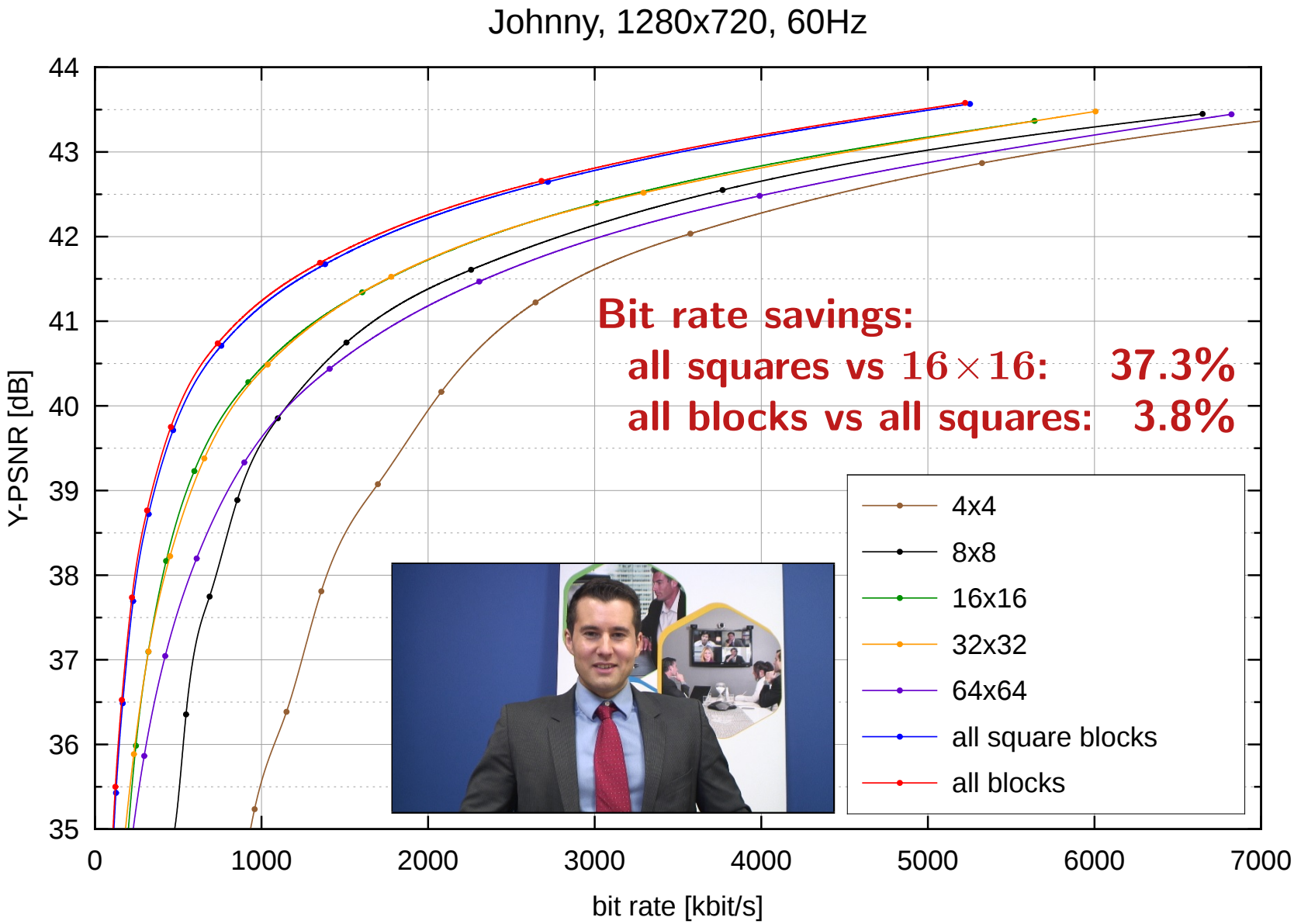
Experiment 1: Exclude effect of different transform sizes

- The same 4×4 transform coding is applied for all coding units, independently of the block sizes used for motion-compensated prediction

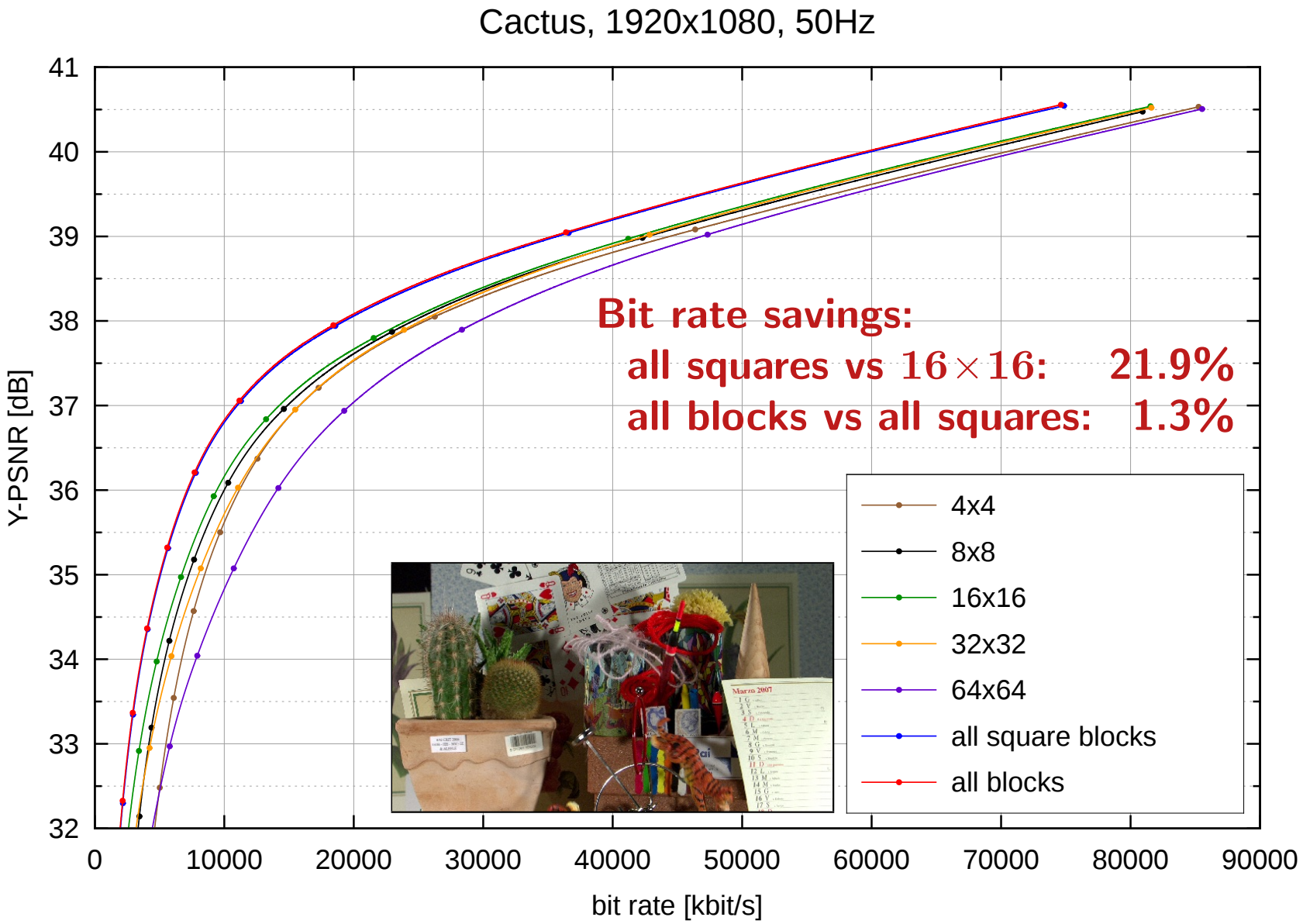
Experiment 2: Combined effect of block sizes for prediction and transform coding

- No restriction of transform sizes beyond that given by HEVC syntax

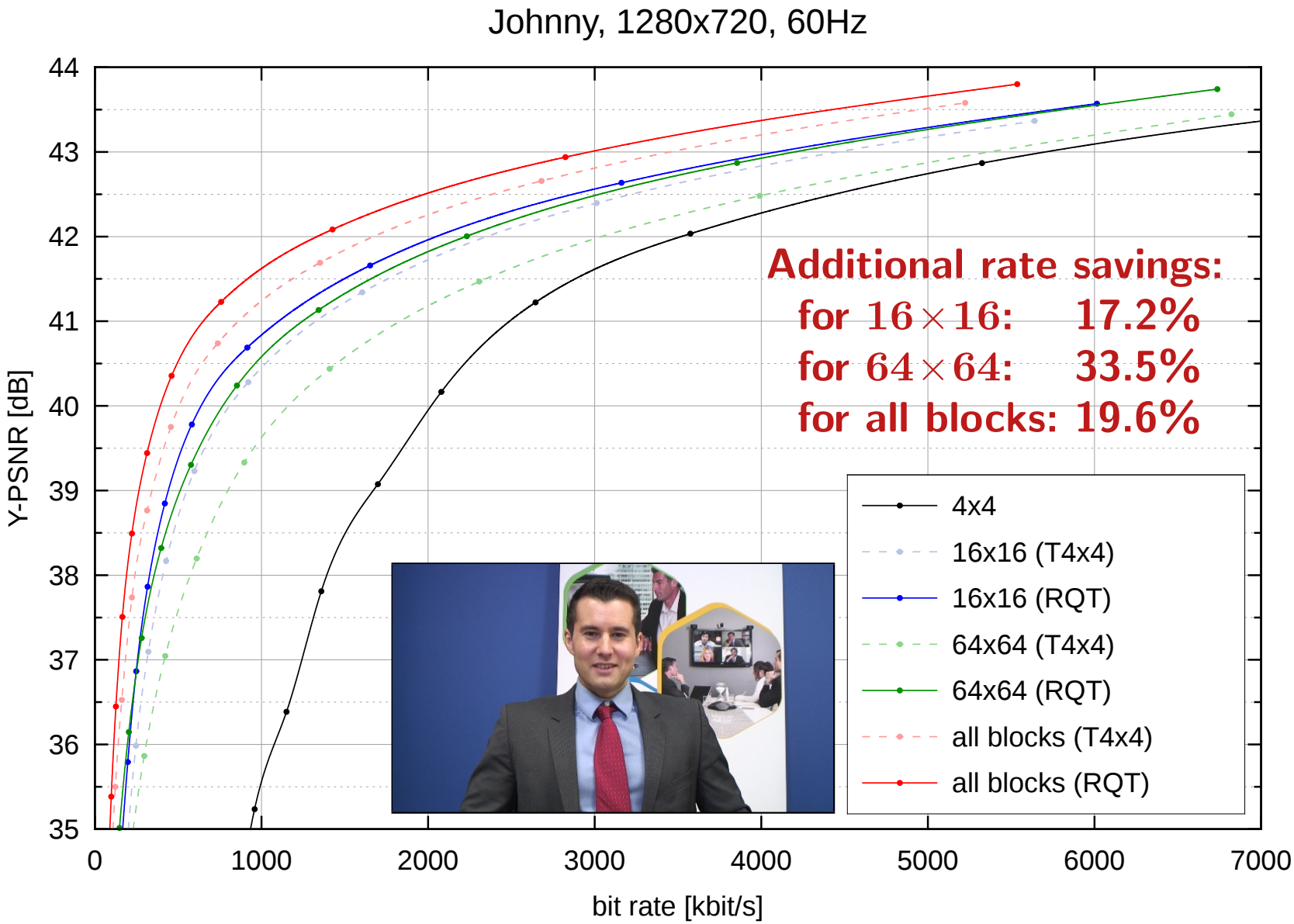
Block Sizes for MCP (4×4 Transform) – Sequence "Johnny"



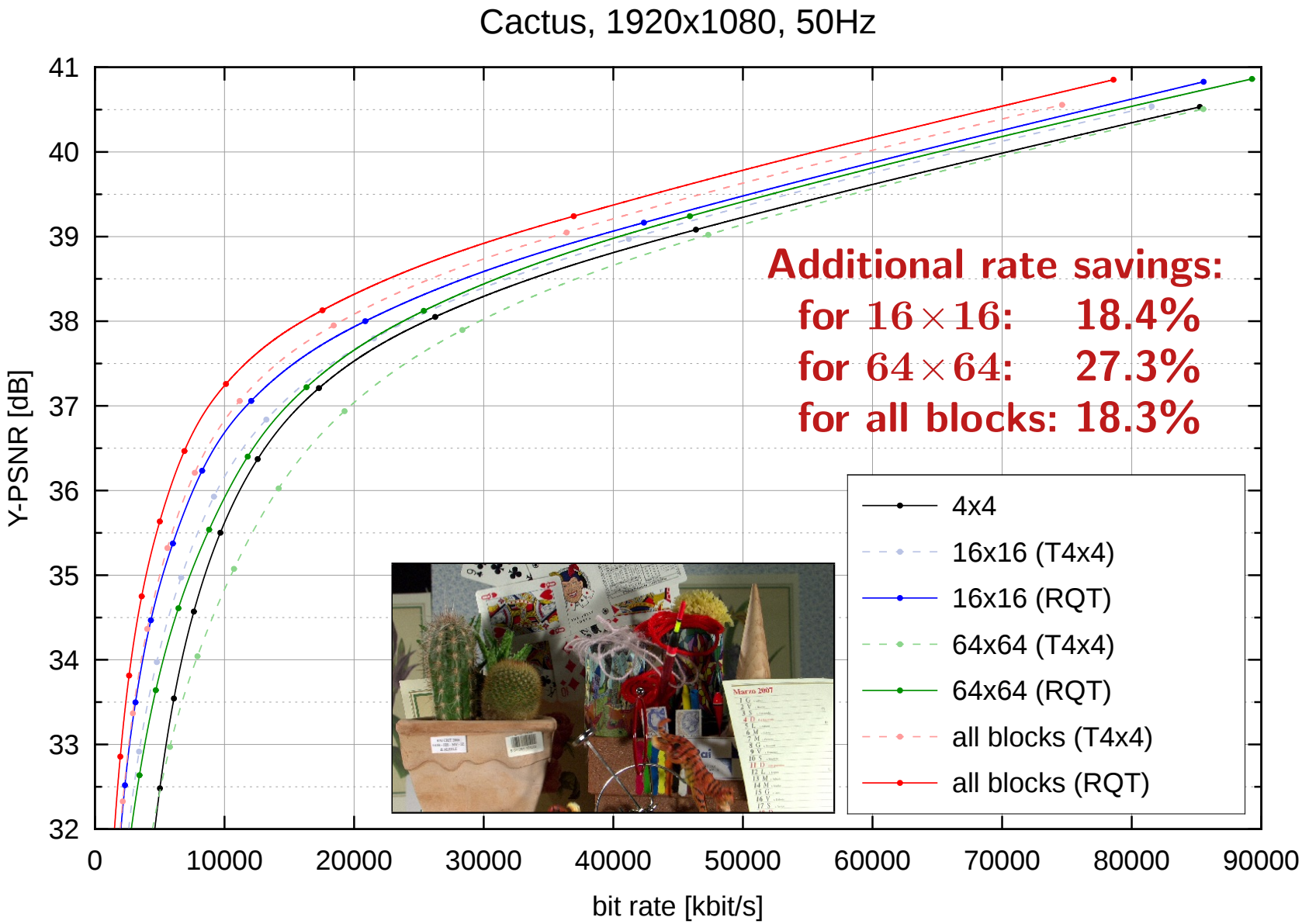
Block Sizes for MCP (4×4 Transform) – Sequence "Cactus"



Without Restricting Transform Coding – Sequence "Johnny"



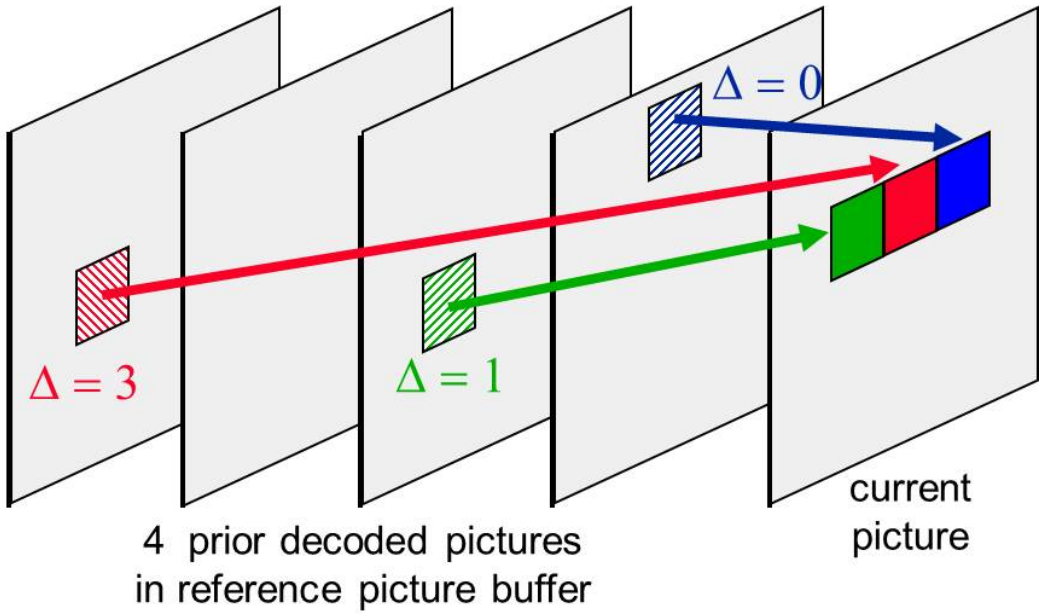
Without Restricting Transform Coding – Sequence "Cactus"



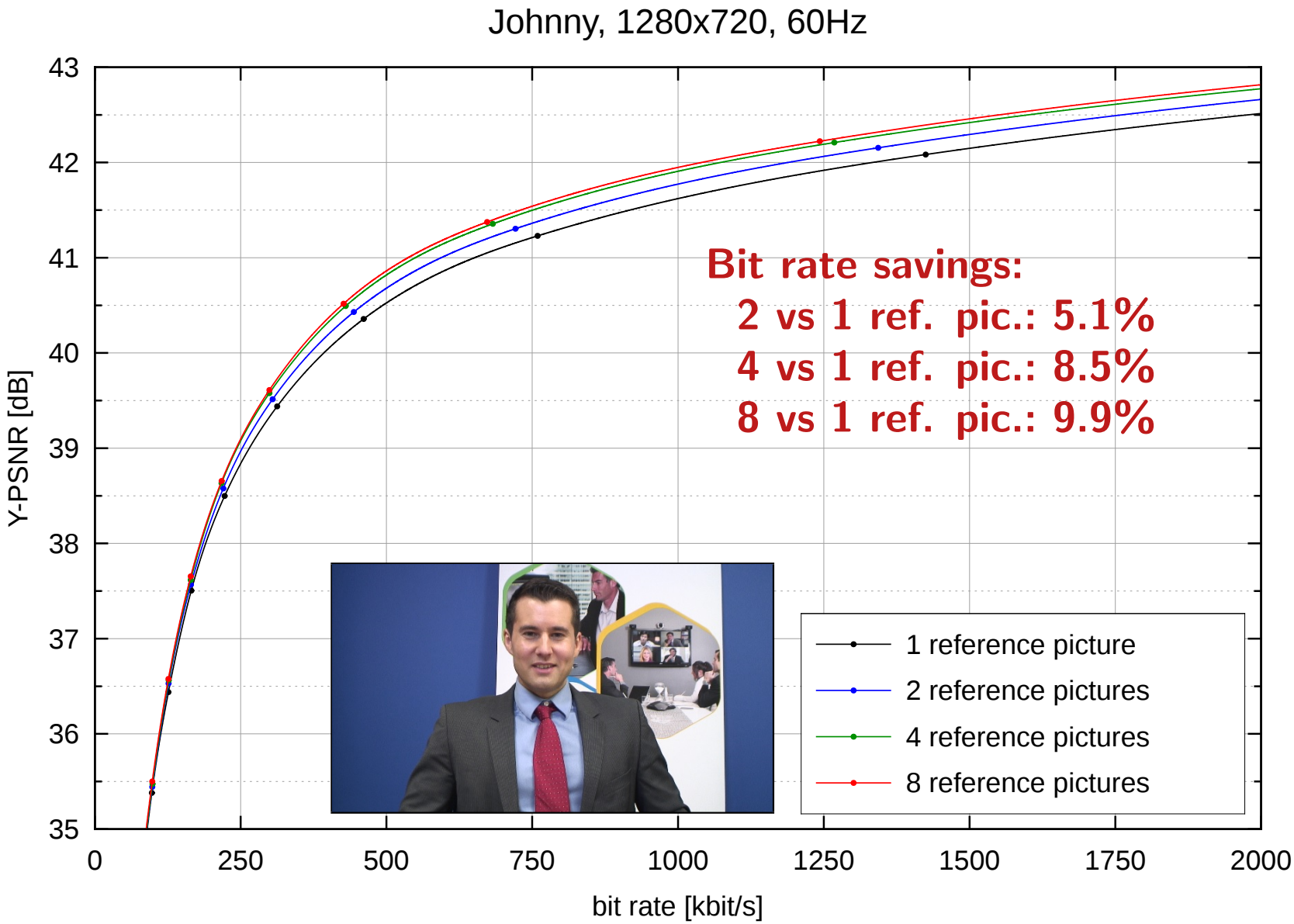
Selection of Reference Picture for MCP

Multiple reference pictures

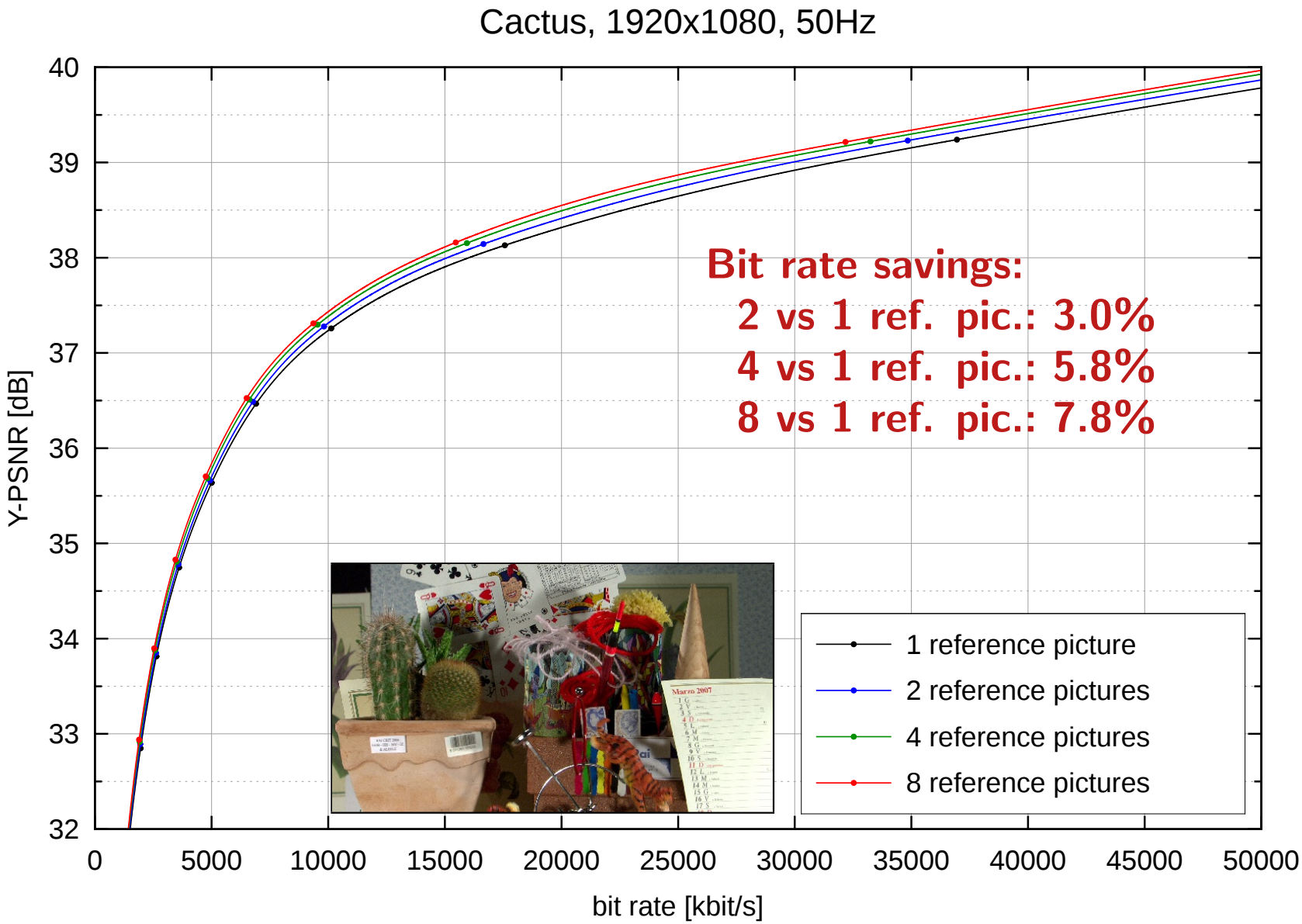
- Motion-compensated prediction is not restricted to use previous decoded picture
- Multiple decoded pictures can be hold in a reference picture buffer
- Employed reference picture is indicated by coding an index Δ
- Side information rate is increased, but prediction may be improved
- Can exploit effects such as scene cuts, aliasing and uncovered background
- Concept is supported in H.263++, H.264/AVC and H.265/HEVC



Number of Reference Pictures – Sequence "Johnny"



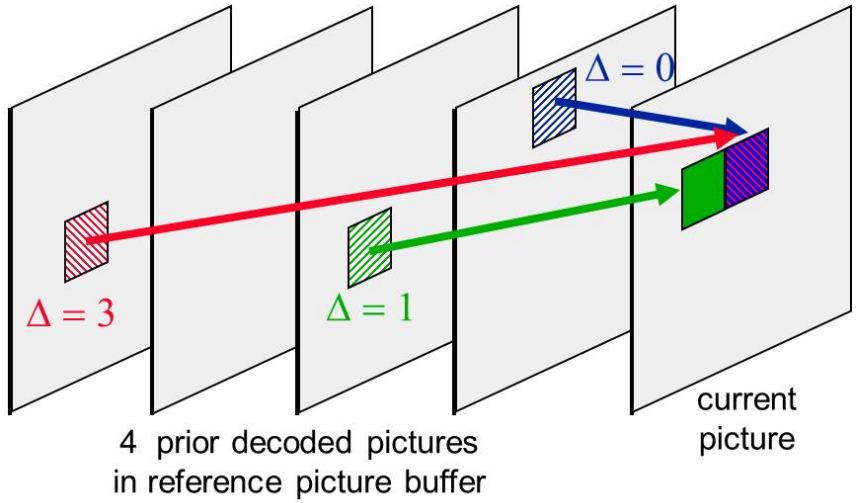
Number of Reference Pictures – Sequence "Cactus"



Number of Motion Hypotheses

Multi-hypotheses motion-compensated prediction

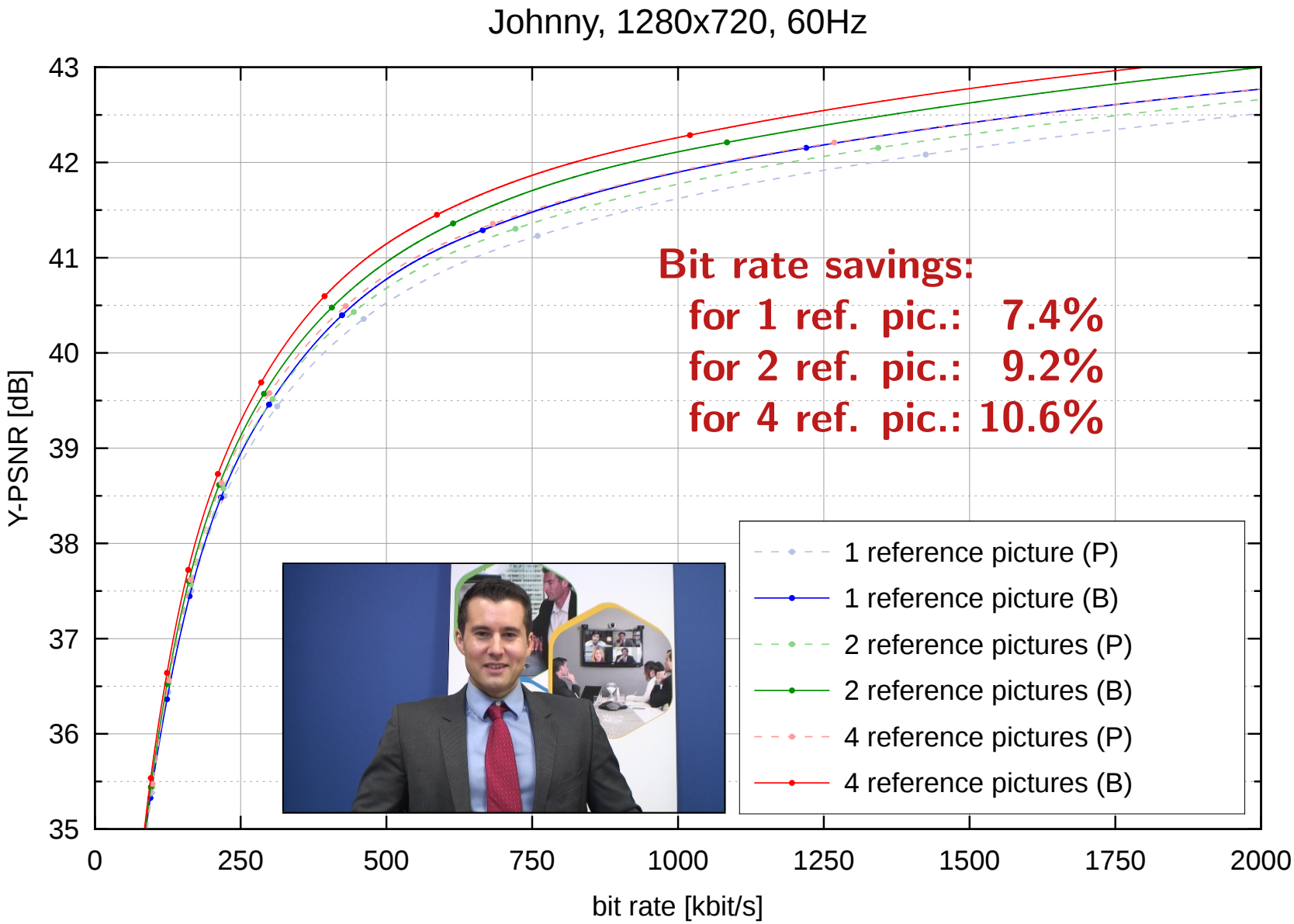
- Average (or weighted average) of multiple motion-compensated prediction signals
- Typically combined with multiple reference pictures
- Side information rate is increased, but prediction may be improved



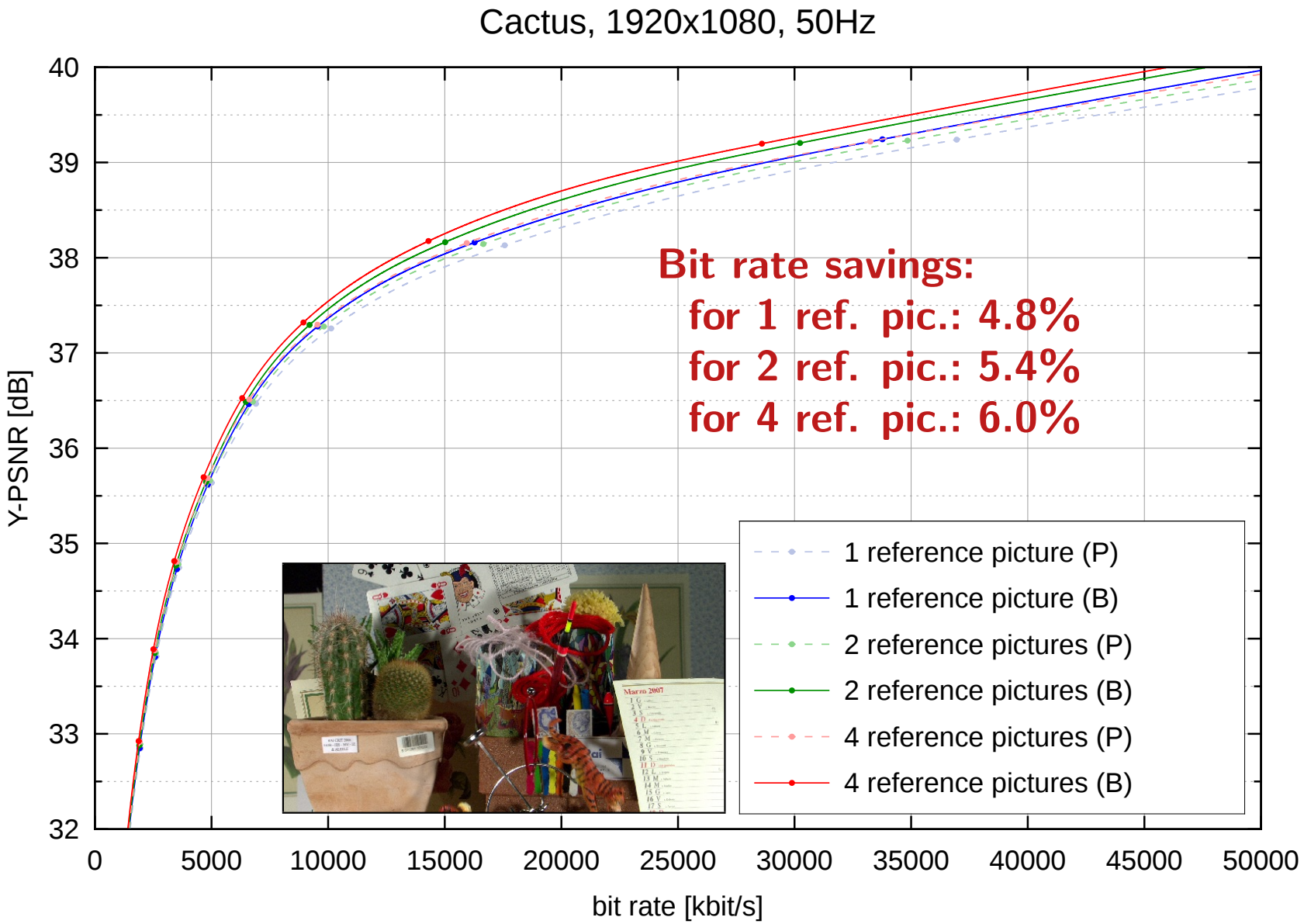
Multi-hypotheses prediction in video coding standards

- Restricted to two motion hypotheses
- Block-based switching between 1 and 2 hypotheses
- MPEG-2, MPEG-4: Restricted two "bi-directional" prediction
- H.263++, H.264/AVC, H.265/HEVC: Generalized B slices

Two Motion Hypotheses (Bi-Prediction) – Sequence "Johnny"



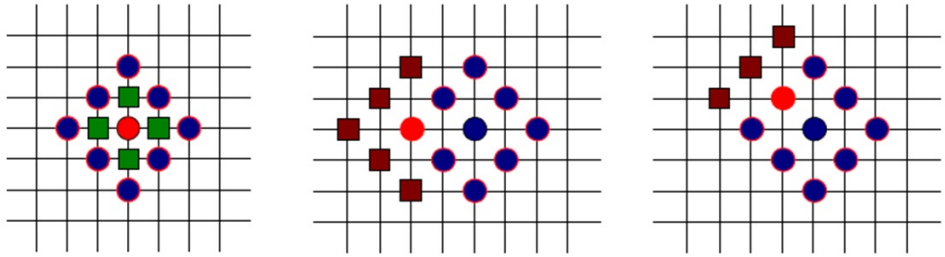
Two Motion Hypotheses (Bi-Prediction) – Sequence "Cactus"



Summary on Motion-Comp. Prediction & Hybrid Video Coding

- Motion-compensated prediction: Exploiting similarities between pictures
- Hybrid video coding
  - Motion-compensated prediction with transform coding of prediction error
  - Concept used in all ITU-T and ISO/IEC video coding standards
- Theoretical analysis of MCP using simple models
  - Motion-compensated prediction improves coding efficiency
  - Sub-sample accurate motion vectors improve coding efficiency
- Design aspects for motion-compensated prediction
  - Accuracy of displacement vector: Integer-, half, quarter-sample accuracy
  - Motion models: Translational, affine, perspective
  - Regions with constant motion: Fixed and variable block sizes
  - Selection of reference picture: Multiple reference pictures
  - Number of motion hypotheses: Bi-prediction
- Motion-compensated prediction in newest standard H.265/HEVC
  - Variable block sizes from 64×64 to $8\times4/4\times8$
  - Translational motion with quarter-sample accurate vectors
  - Multiple reference pictures and up to 2 motion hypotheses

Encoder Control



$$\min \quad D + \lambda \cdot R$$

Review of Lagrangian Encoder Control

Optimal bitstream for given set of constraints (bit rate, delay, etc.)

- With $\mathcal{B}_c$ being the set of *conforming* bitstreams $\boldsymbol{b}$ that fulfill all given constraints, the *optimal* bitstream is given by

$$\boldsymbol{b}^* = \arg\min_{\boldsymbol{b}\in\mathcal{B}_c} D\big(\boldsymbol{s}, \boldsymbol{s}'(\boldsymbol{b})\big) \tag{640}$$

  where $\boldsymbol{s}$ and $\boldsymbol{s}'$ are the original and reconstructed video, respectively

- The optimization is not feasible due to huge parameter space

  $\Longrightarrow$ Split into smaller optimization problems by partially ignoring dependencies

Lagrangian encoder control

- Consider coding of set of samples $\boldsymbol{s}_k$ (e.g. picture, macroblock) and optimized with respect to coding parameters $\boldsymbol{p}_k$ (e.g. coding modes, motion vectors)

$$\min_{\boldsymbol{p}_k} D\big(\boldsymbol{s}_k, \boldsymbol{s}'_k(\boldsymbol{p}_k)\big) \qquad \text{subject to} \qquad R(\boldsymbol{p}_k) \le R_c \tag{641}$$

- Reformulate constraint optimization problem as unconstrained problem

$$\min_{\boldsymbol{p}_k} D\big(\boldsymbol{s}_k, \boldsymbol{s}'_k(\boldsymbol{p}_k)\big) + \lambda \cdot R(\boldsymbol{p}_k) \tag{642}$$

Determination of Coding Parameters for Subsets

Determination of coding parameters for smaller units

- Consider partition of $\boldsymbol{s}_k$ into smaller subsets $\boldsymbol{s}_{k,i}$ (e.g. smaller blocks)
- For **independent coding decisions** and **additive distortion measures**, we have

$$\sum_i \left(\min_{\boldsymbol{p}_{k,i}} D\big(\boldsymbol{s}_{k,i}, \boldsymbol{s}'_{k,i}(\boldsymbol{p}_{k,i})\big) + \lambda \cdot R(\boldsymbol{p}_{k,i}) \right) \tag{643}$$

$\Longrightarrow$ **Independent selection of coding parameters** $\boldsymbol{p}_{k,i}$ for each subset

Coding decisions in image and video coding

- Coding decisions are typically not independent (e.g. due to prediction)
- For practical applicability: Partially ignore dependencies
  $\Longrightarrow$ Consider past decisions (correct predictors for samples and coding parameters)
  $\Longrightarrow$ Ignore impact on future decisions
- Typically used distortion measures

$$D\big(\boldsymbol{s}, \boldsymbol{s}'\big) = \sum_i \big| s_i - s'_i \big|^{\beta} \tag{644}$$

with $\beta = 1$: Sum of absolute differences (SAD)
$\beta = 2$: Sum of squared differences (SSD)

Application of Lagrange Optimization in Video Coding

Quantization of the transform coefficients of a block

- Select vector $\boldsymbol{q}$ of transform coefficient levels according to

$$\boldsymbol{q}^* = \arg\min_{\boldsymbol{q}} \; D(\boldsymbol{q}) + \lambda \cdot R(\boldsymbol{q}) \tag{645}$$

with $D(\boldsymbol{q})$: SSD distortion for choosing transform coefficient level vector $\boldsymbol{q}$
$R(\boldsymbol{q})$: Number of bits required for representing $\boldsymbol{q}$

$\Longrightarrow$ Rate-distortion optimized quantization (as discussed for run-level coding)
$\Longrightarrow$ Discussed algorithm considers dependencies inside a block
$\Longrightarrow$ Can be adapted to other coding schemes for transform coefficient levels

Mode decision (e.g. macroblock mode, intra prediction mode, block partitioning)

- Select coding mode c for a block

$$c^* = \arg\min_{c} \; D(c) + \lambda \cdot R(c) \tag{646}$$

with $D(c)$: SSD distortion for choosing coding mode c for the block
$R(c)$: Number of bits for block when coded with mode c

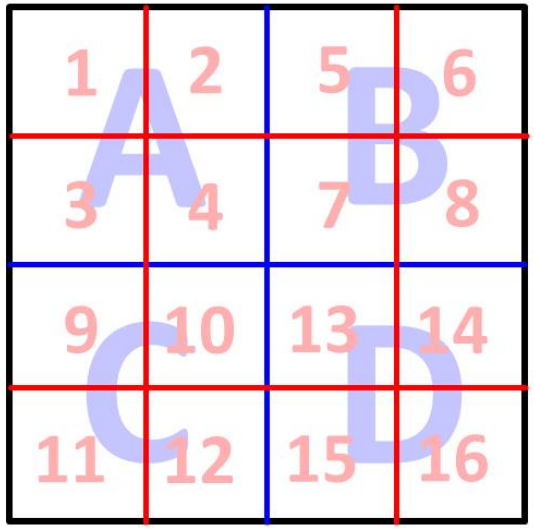
$\Longrightarrow$ Considers quantization for both distortion D and rate R
$\Longrightarrow$ If applicable, coding parameters for sub-blocks have to determined in advanced (e.g., for tree-based partitioning)

Mode Decision for Hierarchical Block Structures

Exhaustive evaluation of all partitionings

- Evaluate blocks in *depth-first order*
- Example: Two quadtree levels for a 16×16 block
  (1) Select best partitioning for first 8×8 block A
  (2) Select best partitioning for second 8×8 block B
  (3) Select best partitioning for third 8×8 block C
  (4) Select best partitioning for fourth 8×8 block D
  (5) Choose between 16×16 block and sub-division

  Note: Predictors are set according to prior decisions



Fast mode decision strategies for Hierarchical Structures

- Terminate decision process as soon as a "quality criterion" is met
  - Distortion (or Lagrangian cost) less than a threshold
  - Number of significant transform coefficients less than a threshold
- Example: Top-down approach
  (1) Evaluate 16×16 block and stop if quality criterion is met
  (2) Evaluate first 8×8 block and check quality criterion
  $\Longrightarrow$ Check 4×4 partitioning only if quality criterion is not met
  (3) Proceed with next 8×8 block, etc.

Lagrange Optimization for Choosing Motion Vectors

Straightforward application of Lagrange optimization

- Treat each motion vector $\boldsymbol{m} = [m_x, m_y]^{\mathrm{T}}$ out of a considered set $\mathcal{M}$ of motion vectors as a coding mode and apply mode decision process

$$\boldsymbol{m}^* = \arg \min_{\boldsymbol{m} \in \mathcal{M}} D(\boldsymbol{m}) + \lambda_m \cdot R(\boldsymbol{m}) \tag{647}$$

with $D(\boldsymbol{m})$: SSD distortion for choosing motion vector $\boldsymbol{m}$ for the block
$R(\boldsymbol{m})$: Number of bits for block when coded using motion vector $\boldsymbol{m}$

$\Longrightarrow$ Considering quantization for each possible motion vector is way too complex

Practical rate-distortion optimization for motion vector selection

- Assume that transmitted residual is equal to zero (important case in practice)
- Use the Lagrange minimization with less complex cost measure
  - $R(\boldsymbol{m})$ is the rate for coding only the motion vector $\boldsymbol{m}$
  - $D(\boldsymbol{m})$ is the distortion between original and prediction signal $D(\boldsymbol{s}, \hat{\boldsymbol{s}})$
- As distortion measure, the SAD distortion is typically used in practice

$\Longrightarrow$ The Lagrange parameter is different than for mode decision and quantization

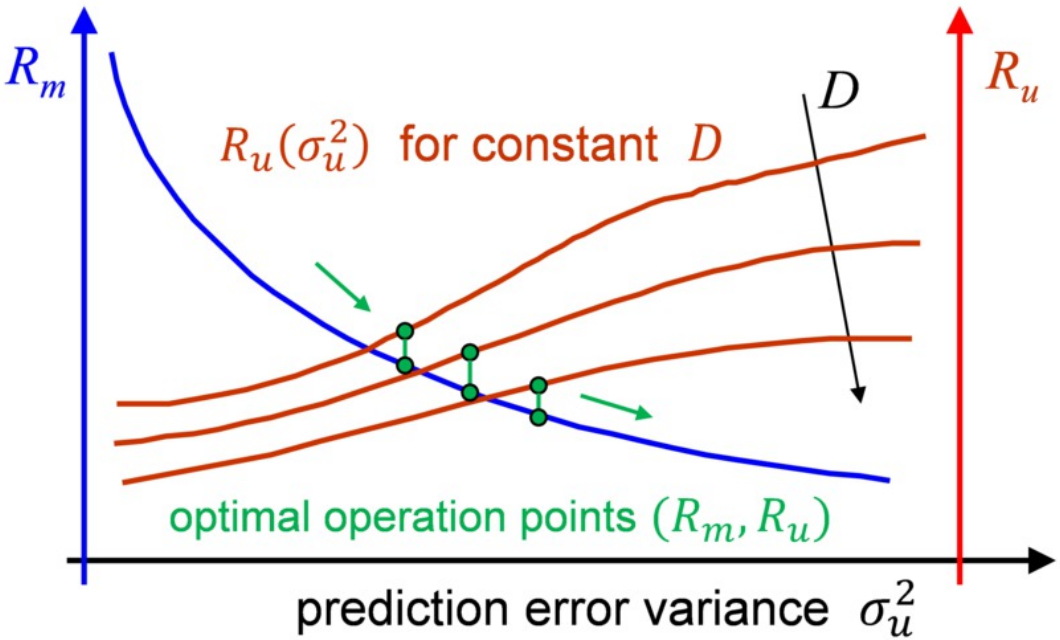
$\Longrightarrow$ The choice $\lambda_m = \sqrt{\lambda}$ is typically used in this case

Importance of Rate-Constrained Decisions for Modes & Motion

Distortion D of reconstructed signal is influence by two factors

- Side information rate R_m: Increasing R_m improves prediction and reduces distortion
- Rate for residual signal R_u: Increasing R_u reduces distortion
- Optimal rate allocation: Minimization of $J = D(R_m, R_u) + \lambda \cdot (R_m + R_u)$

$$\left.\begin{array}{lll} \frac{\partial}{\partial R_u} J = 0 & \Longrightarrow & \frac{\partial}{\partial R_u} D = -\lambda \\ \frac{\partial}{\partial R_m} J = 0 & \Longrightarrow & \frac{\partial}{\partial R_m} D = -\lambda \end{array}\right\} \Longrightarrow \boxed{\frac{\partial}{\partial R_m} D = \frac{\partial}{\partial R_u} D} \qquad (648)$$



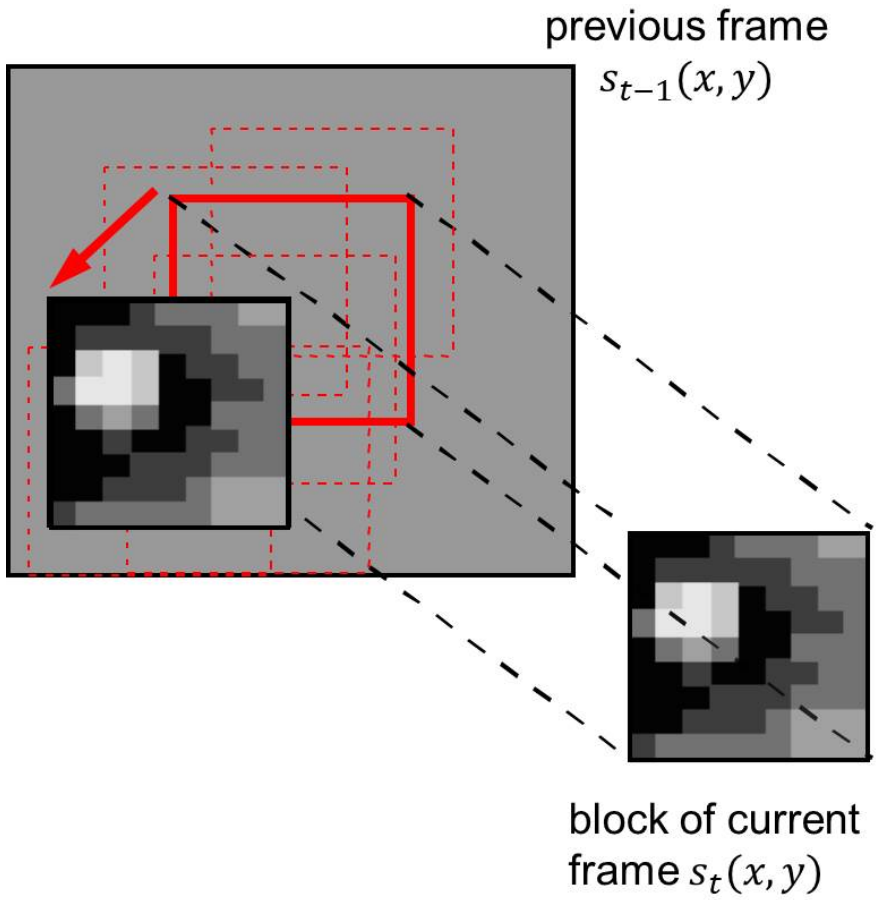
Prediction error variance σ_u^2 can be influenced by

- Number of intra pred. modes
- Block sizes for intra prediction
- Block sizes for motion comp.
- Accuracy of motion vectors
- Number of reference pictures
- Number of motion hypotheses
- Choice of motion model

Estimation of Translational Motion: Block Matching

Principle of block matching

- Subdivide current frame into blocks
- Determine one displacement vector for each block
- Find best match in reference frame by minimizing Lagrange cost $D + \lambda \cdot R$



Distortion measures for block matching

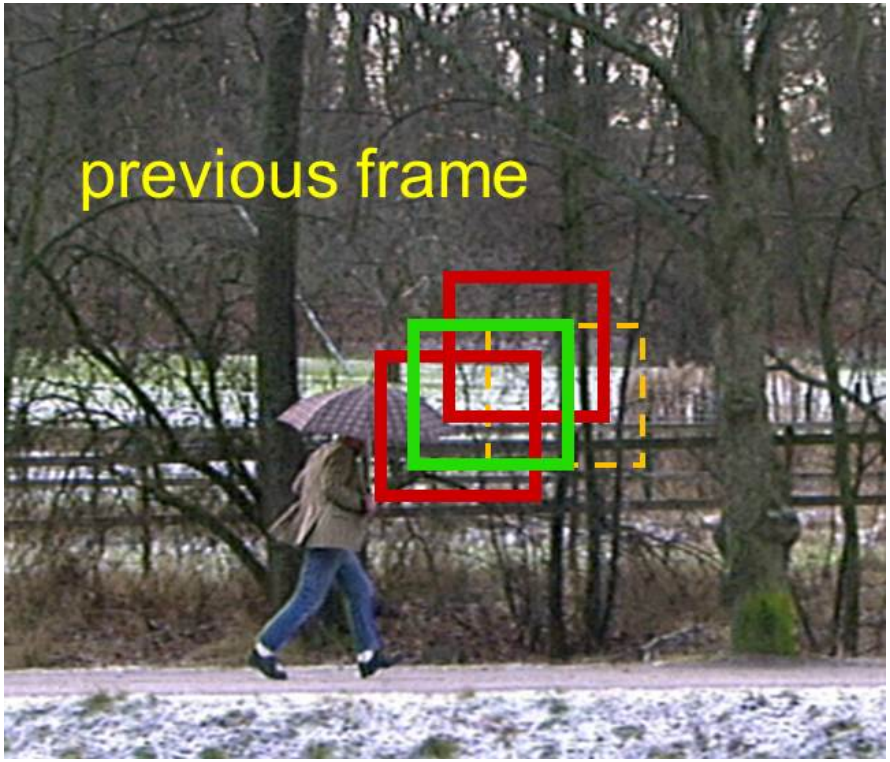
- Typically: SAD distortion
- Alternative measures:
  - SSD distortion
  - SAD in transform domain
  - cross correlation

Difficulty in determination of displacement parameters by block matching

- It is not feasible to evaluate all "possible motion vectors" (there are too many)

$\Longrightarrow$ Require intelligent search strategies (testing only most likely motion vectors)

Illustration of the Block Matching Algorithm

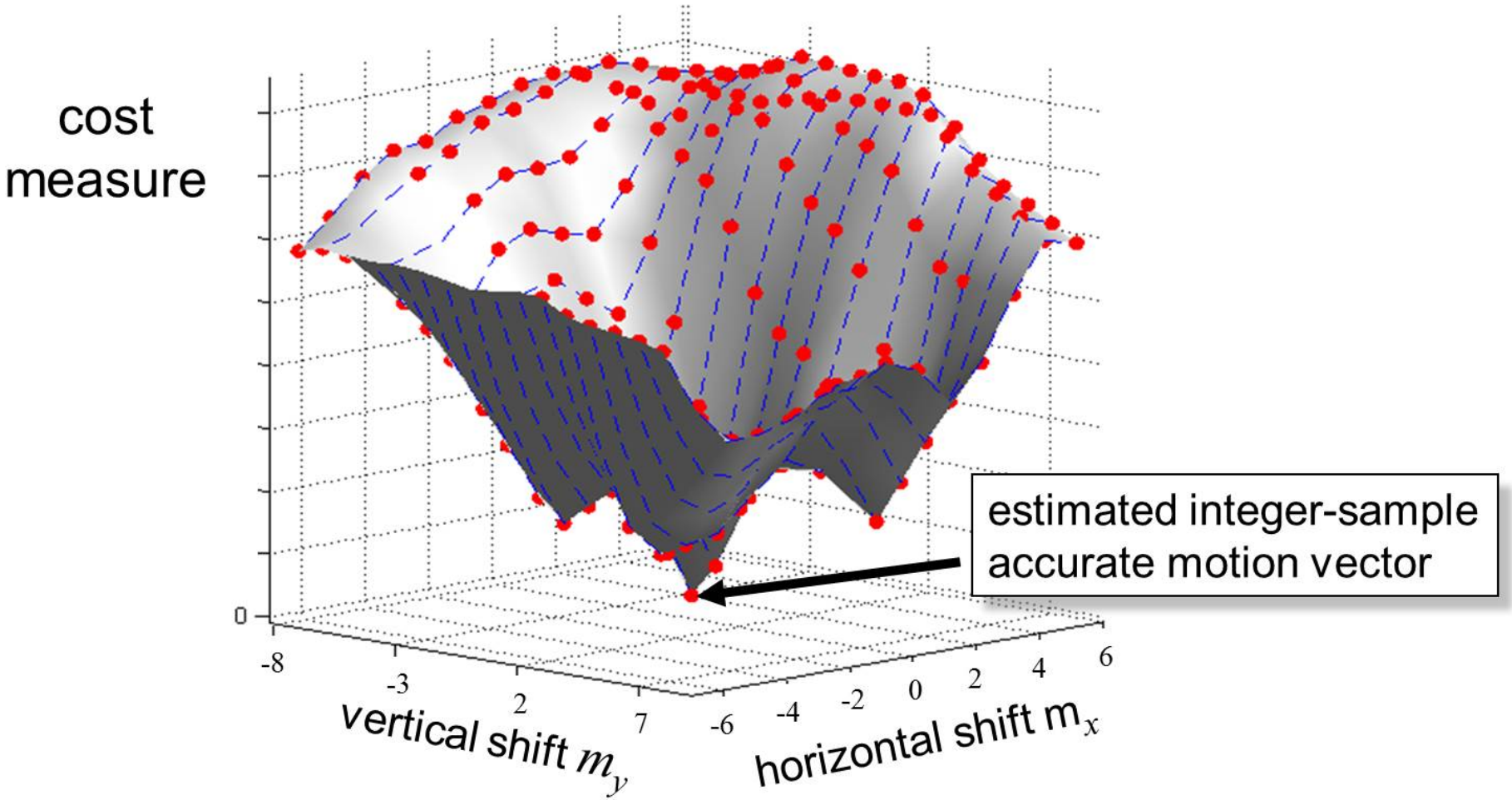


The measurement window is compared with **different shifted blocks** in the reference frame and the **best match** is determined

The considered block of samples in the current frame is selected as a measurement window

Cost Measures Values inside a Search Window

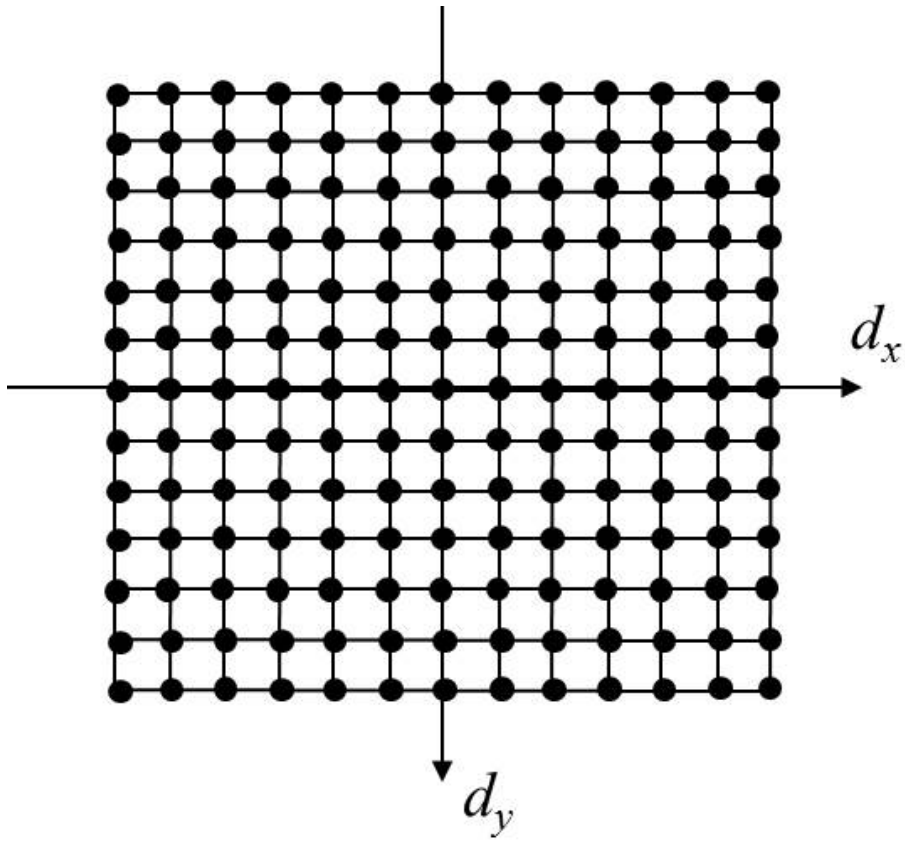
Example: Cost measure values inside a search window



Search Strategies: Exhaustive Search

Exhaustive search

- Evaluate all possible motion vectors (displacements) inside a rectangular search window
- Computationally very complex
- Highly regular, parallelizable



Selection of search window

- Often centered around zero motion vector
- Can also be centered around motion vector predictor
- Size can be adapted during encoding of a picture
- Size can be increased under certain circumstances

Search Strategies: Methods for Complexity Reduction

Complexity of block matching

- Evaluation of **complex error measure** for **many candidates**

Two approaches for reducing encoder complexity

Complexity of error measure

- Fast approximations
- Early termination
- Exclusion of candidates

Number of search candidates

- Skip unlikely areas in search
- Adaptively increase or decrease distance between search candidates

Combine both approaches

- Choose starting point and search order that maximizes likelihood for efficient approximations, early terminations and excluding candidates

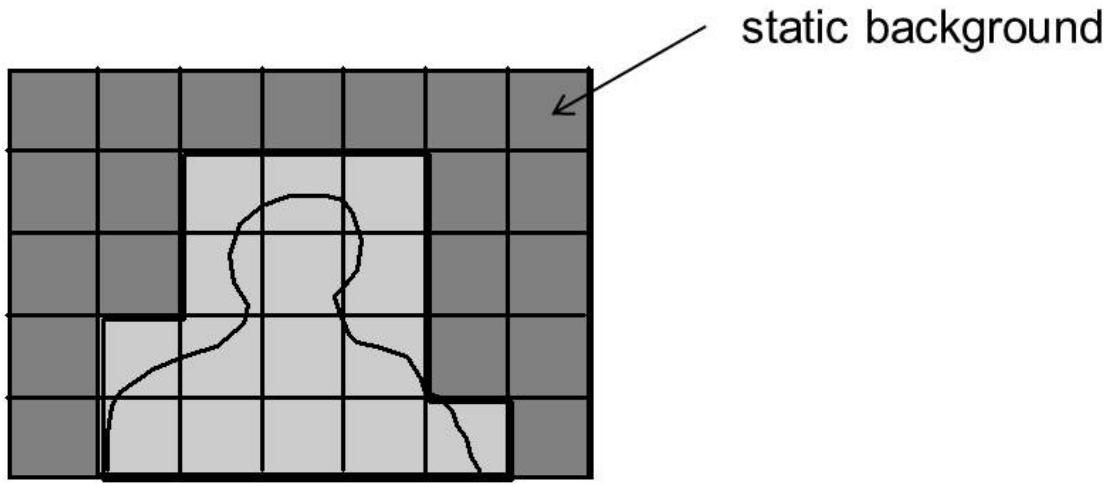
Search Strategies: Fast Approximations

Basic approach: Stop search if match is "good enough"

- Distortion measure D is less than a threshold
- Lagrange cost $D + \lambda \cdot R$ is less than a threshold

Practical method in video conferencing (static background)

- Evaluate zero vector and stop search if the match is good enough



Search Strategies: Early Termination

Compare partial cost measures

- Partial distortion measure D_K for block size $B_x \times B_y$, with $K = 1 \ldots B_y$

$$D_K(m_x, m_y) = \sum_{y=0}^{K-1} \sum_{x=0}^{B_x-1} \left| s_t(x,y) - s'_{t-1}(x - m_x, y - m_y) \right|^{\beta} \qquad (649)$$

- Compare partial cost measure with previously determined minimum cost $J_{\min}$
- Early termination without loss

$$\text{Stop if:} \quad D_K(m_x, m_y) + \lambda_m \cdot R(m_x, m_y) \geq J_{\min} \qquad (650)$$

- Early termination with possible loss (but higher speedup)

$$\text{Stop if:} \quad D_K(m_x, m_y) + \lambda_m \cdot R(m_x, m_y) \geq \alpha(K) \cdot J_{\min} \qquad (651)$$

Example for weighting function: $a(K) = \sqrt{\frac{K}{B_y}}$

Search Strategies: Early Exclusion of Candidates

Speed-up for block comparison

- Triangle inequality for samples in a block $\mathcal{B}$ (here, for SAD)

$$\sum_{k \in \mathcal{B}} \left| s_k - \hat{s}_k \right| \geq \left| \sum_{k \in \mathcal{B}} (s_k - \hat{s}_k) \right| = \left| \left(\sum_{k \in \mathcal{B}} s_k \right) - \left(\sum_{k \in \mathcal{B}} \hat{s}_k \right) \right| \tag{652}$$

- Basic strategy
  1. Compute sum of samples values for all block locations in reference frame (sliding window average can be calculated in a very easy way)
  2. Compute sum of samples values for current block
  3. Omit complete distortion calculation if difference between sums of samples values yields larger cost measure than previous minimum

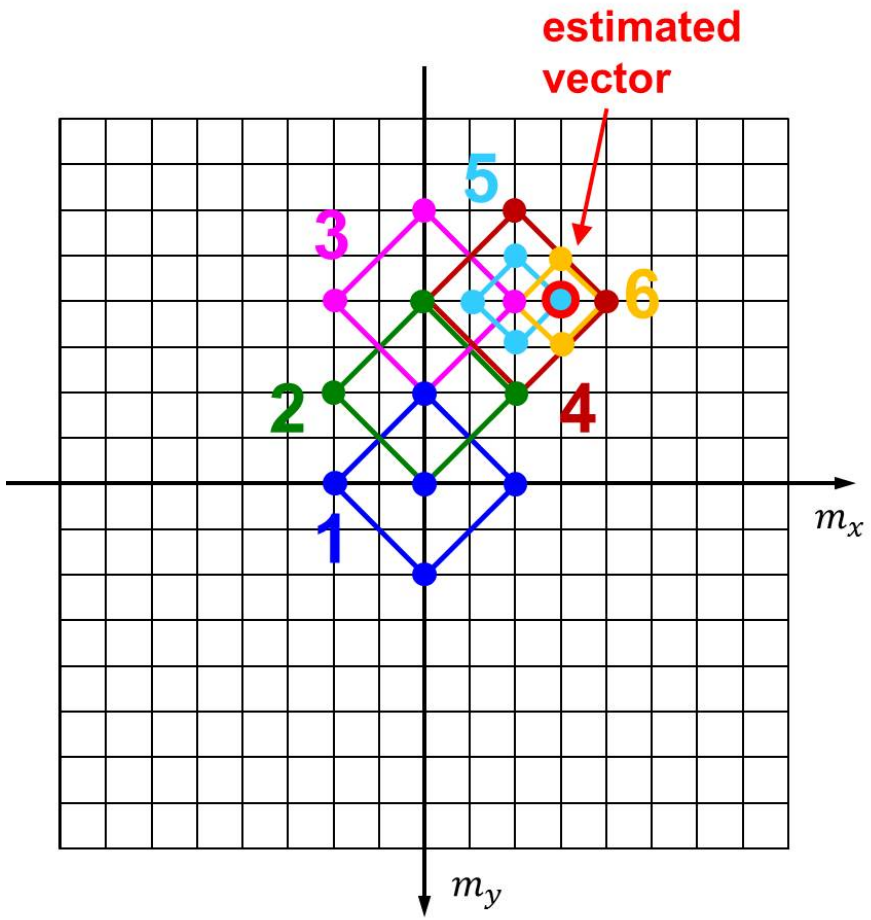
- Combination with variable size prediction blocks (H.264/AVC, H.265/HEVC)
  - Start with computation of sample sums for the smallest supported block size
  - Sums for larger blocks are obtained by adding up the sums for smaller blocks

  $\Longrightarrow$ Increases speed-up for nested block sizes as found in modern video codecs

Search Strategies: 2-D Logarithmic Search

2-d logarithmic search [Jain, Jain, 1981]

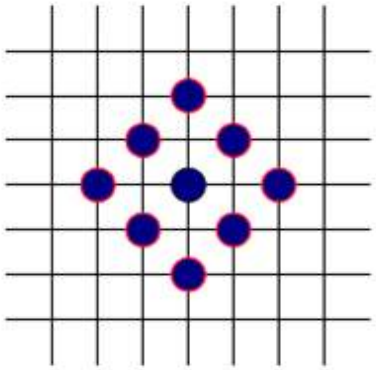
- Iterative comparison of the cost measures at 5 points (corners and center) of a diamond-shaped pattern
- Move pattern so that pattern in centered around best match
  - No more than 3 new candidates
- Logarithmic refinement of search pattern (4 new candidates) if
  - Best match is in center of pattern
  - Or best match is at the border of the search range
- Motion search is terminated if
  - Best match is in center of pattern
  - And smallest pattern size is used



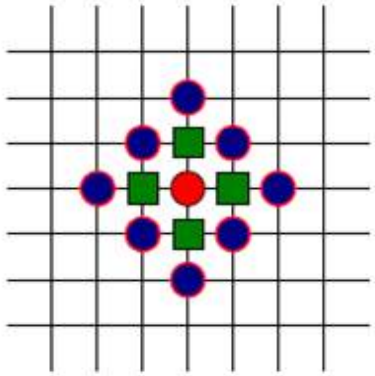
Search Strategies: Diamond Search

Diamond search [Li, Zeng, Liou, 1994] and [Zhu, Ma, 1997]

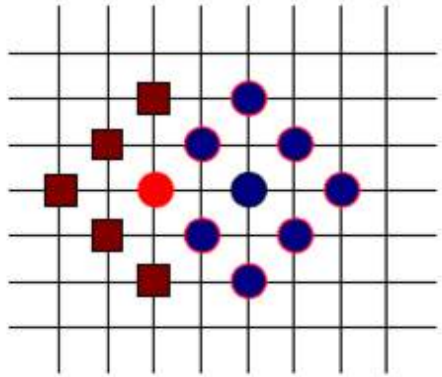
- Iterative search with 9 points of a diamond pattern
- Similar search strategy as 2-d logarithmic search

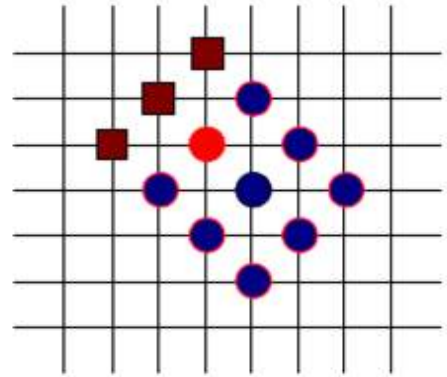


Start with large diamond pattern at motion vector (0,0) or at a predicted vector



If best match is in the center of a large diamond, proceed with a smaller diamond



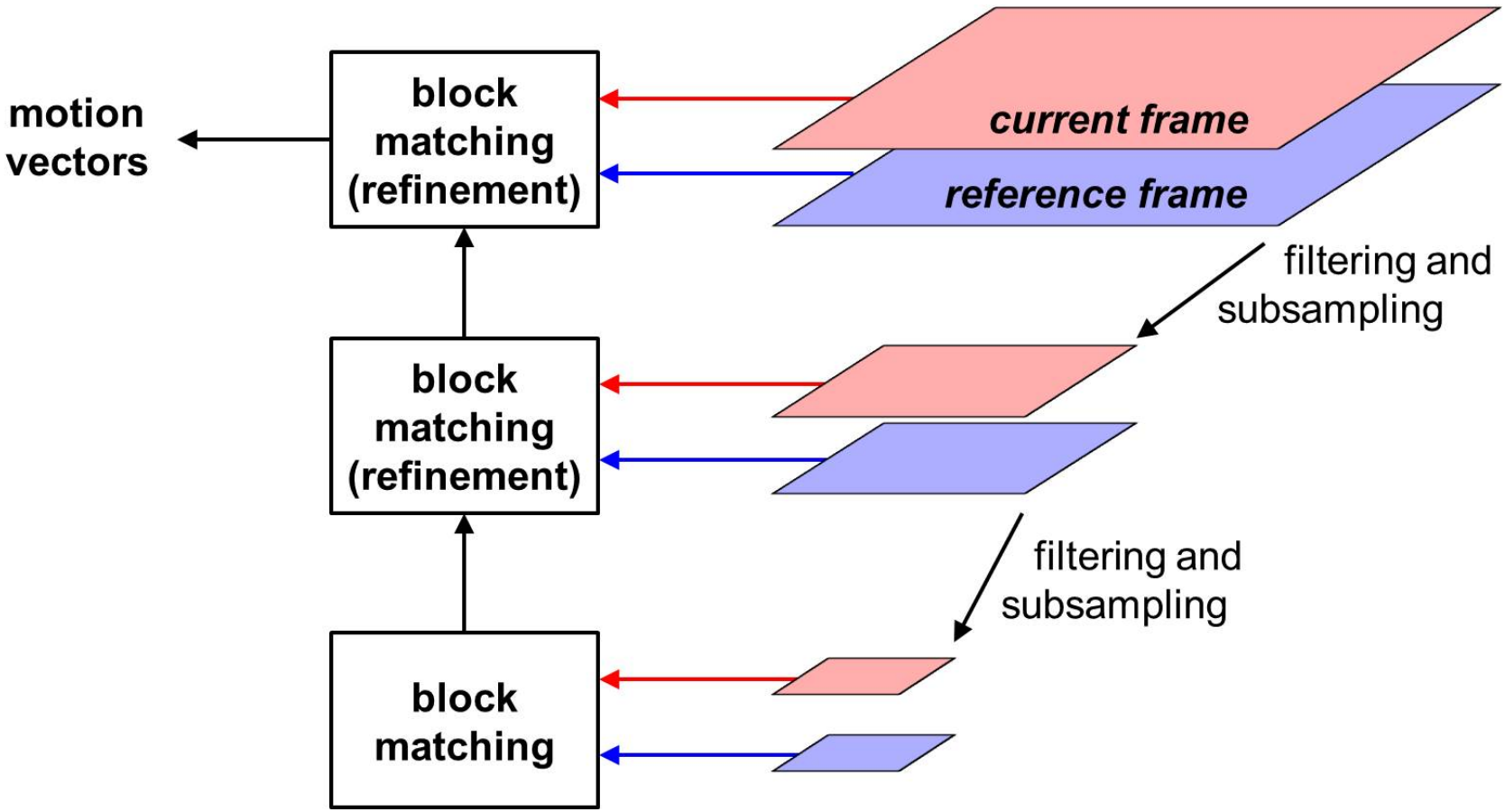


If best match does not lie in the center of the diamond pattern, center next diamond pattern at the best match

Search Strategies: Hierarchical Block Matching

Hierarchical block matching

- Start with dyadically downsampled pictures
- Refine motion vectors from one hierarchy level to the next



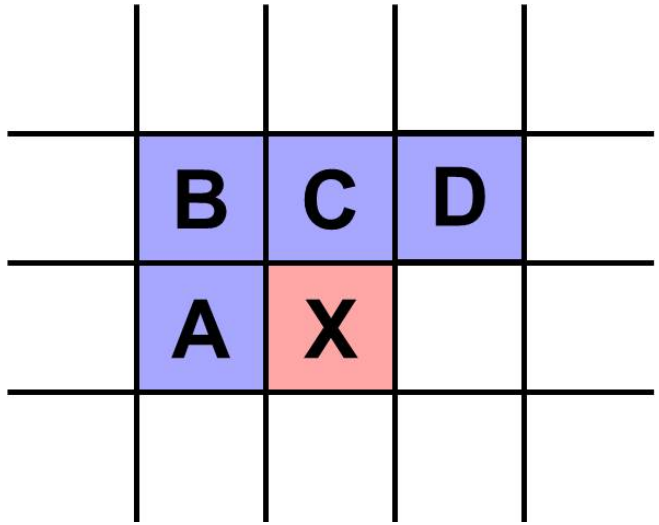
Search Strategies: Choosing of Start Point

Non-adaptive choices of start point

- Use motion vector (0,0) as start point of motion search
  - $\Longrightarrow$ Suitable for applications like video conferencing
  - $\Longrightarrow$ Problematic if large motions occur in video sequence
- Use motion vector predictor as start point for motion search
  - $\Longrightarrow$ Typically results in faster termination of motion search

Adaptive choice of start point

- General idea: Motion of a block is similar to at least one of the neighboring blocks
- First evaluate the motion vectors of the already estimated neighboring blocks
  - Example: Blocks A, B, C and D
  - Candidates can also include a temporally predicted motion vector
- Choose best match among the candidates as start point of the motion search



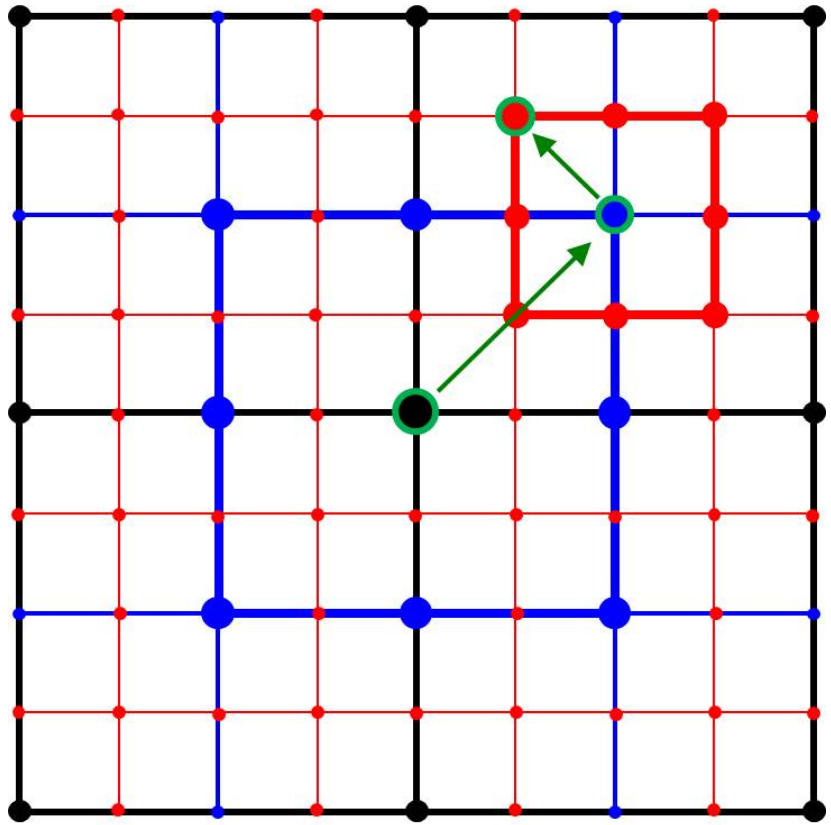
Estimation of Sub-Sample Accurate Motion Vectors

Sub-sample accurate motion vectors

- Motion vectors are often not restricted to integer-sample accurate displacements
- Typical sub-sample accuracies: Half- and quarter-sample

Estimation of sub-sample shifts

- Typical: Iterative sub-sample refinement using best integer-sample displacement
  - Test 8 half-sample candidates around best integer-sample match
  - Test 8 quarter-sample candidates around best half-sample match
- Requires interpolation of sample values at sub-sample locations



- integer-sample positions
- half-sample positions
- quarter-sample positions

Estimation of Higher-Order Motion Parameters

Linear approximation of prediction error signal

- Interpolated reconstructed reference picture is denoted by $s'_{\text{ref}}(x, y)$
- Assume: Estimate $\hat{\boldsymbol{d}} = [\hat{d}_x, \hat{d}_y]^{\text{T}}$ of displacement vector $\boldsymbol{d} = [d_x, d_y]^{\text{T}}$ is given
- Assume: Displacement errors $\Delta d_x = d_x - \hat{d}_x$ and $\Delta d_y = d_y - \hat{d}_y$ are small
- Prediction error for a sample location (x, y) can be approximated by

$$\begin{aligned} u[x,y] &= s[x,y] - s'_{\text{ref}}(x - d_x, y - d_y) \\ &= s[x,y] - s'_{\text{ref}}(x - \hat{d}_x - \Delta d_x, y - \hat{d}_y - \Delta d_y) \\ &= s[x,y] - s'_{\text{ref}}(\hat{x} - \Delta d_x, \hat{x} - \Delta d_y) \\ &\approx s[x,y] - s'_{\text{ref}}(\hat{x}, \hat{y}) + \frac{\partial s'_{\text{ref}}}{\partial x}(\hat{x}, \hat{y}) \cdot \Delta d_x + \frac{\partial s'_{\text{ref}}}{\partial y}(\hat{x}, \hat{y}) \cdot \Delta d_y \end{aligned} \tag{653}$$

with

$(\hat{x}, \hat{y})$: Predicted reference sample location $(x - \hat{d}_x, y - \hat{d}_y)$

$\frac{\partial s'_{\text{ref}}}{\partial x}$: Gradient in x direction of interpolated reference picture $s'_{\text{ref}}(x, y)$

$\frac{\partial s'_{\text{ref}}}{\partial y}$: Gradient in y direction of interpolated reference picture $s'_{\text{ref}}(x, y)$

Linear Parametric Motion Models

Consider motion models that are linear with respect to the motion parameters

- Displacement vector field can be expressed using a matrix multiplication

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \boldsymbol{B}(x,y) \cdot \boldsymbol{a} \tag{654}$$

- Example: Affine motion model

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \tag{655}$$

- Prediction error for a location $\boldsymbol{x} = [x, y]^{\mathrm{T}}$ can be written as

$$u[\boldsymbol{x}] = s[\boldsymbol{x}] - s'_{\mathrm{ref}}(\hat{\boldsymbol{x}}) + \frac{\partial s'_{\mathrm{ref}}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) \cdot \boldsymbol{B}(\boldsymbol{x}) \cdot \boldsymbol{\Delta a} \tag{656}$$

with

$$\frac{\partial s'_{\mathrm{ref}}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial s'_{\mathrm{ref}}}{\partial x} & \frac{\partial s'_{\mathrm{ref}}}{\partial y} \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{B}(\boldsymbol{x}) \cdot \hat{\boldsymbol{a}} \tag{657}$$

Minimizing SSD Distortion for Linear Motion Models

SSD distortion using linear approximation for small displacement errors

- SSD distortion for a region $\mathcal{R}$ with unique parametric motion

$$D(\boldsymbol{\Delta a}) = \sum_{\boldsymbol{x} \in \mathcal{R}} \left(s[\boldsymbol{x}] - s'_{\text{ref}}(\hat{\boldsymbol{x}}) + \frac{\partial s'_{\text{ref}}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) \cdot \boldsymbol{B}(\boldsymbol{x}) \cdot \boldsymbol{\Delta a} \right)^2 \tag{658}$$

- Minimization with respect to parameter update $\boldsymbol{\Delta a}$ yields a linear equation system

$$\frac{\partial D(\boldsymbol{\Delta a})}{\partial \boldsymbol{\Delta a}} = \boldsymbol{0} \quad \Longrightarrow \quad \boxed{\boldsymbol{G}(\hat{\boldsymbol{a}}) \cdot \boldsymbol{\Delta a} = \boldsymbol{g}(\hat{\boldsymbol{a}})} \tag{659}$$

  with the matrix $\boldsymbol{G}(\hat{\boldsymbol{a}})$ and the vector $\boldsymbol{g}(\hat{\boldsymbol{a}})$ being given by

$$\boldsymbol{G}(\hat{\boldsymbol{a}}) = \sum_{\boldsymbol{x} \in \mathcal{R}} \boldsymbol{B}(\boldsymbol{x})^{\mathrm{T}} \left(\frac{\partial s'_{\text{ref}}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) \right)^{\mathrm{T}} \left(\frac{\partial s'_{\text{ref}}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) \right) \boldsymbol{B}(\boldsymbol{x}) \tag{660}$$

$$\boldsymbol{g}(\hat{\boldsymbol{a}}) = -\sum_{\boldsymbol{x} \in \mathcal{R}} \boldsymbol{B}(\boldsymbol{x})^{\mathrm{T}} \left(\frac{\partial s'_{\text{ref}}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) \right)^{\mathrm{T}} \left(s[\boldsymbol{x}] - s'_{\text{ref}}(\hat{\boldsymbol{x}}) \right) \tag{661}$$

- Can be solved by conventional methods, e.g. Gauss algorithm

Iterative Estimation of Motion Parameters

Iterative algorithm for parameter estimation of linear models and SSD distortion

1. Initialize parameter estimate $\hat{\boldsymbol{a}}$, e.g. with zero vector
2. Determine matrix $\boldsymbol{G}(\hat{\boldsymbol{a}})$ and vector $\boldsymbol{g}(\hat{\boldsymbol{a}})$
3. Determine parameter update $\boldsymbol{\Delta a}$ by solving the linear equation system

$$\boldsymbol{G}(\hat{\boldsymbol{a}}) \cdot \boldsymbol{\Delta a} = \boldsymbol{g}(\hat{\boldsymbol{a}})$$

4. Update parameter estimate

$$\hat{\boldsymbol{a}} = \hat{\boldsymbol{a}} + \boldsymbol{\Delta s}$$

5. Repeat the last three steps until algorithm converges

Difficulties

- Approximation only valid for small parameter errors $\boldsymbol{\Delta a}$

  $\Longrightarrow$ Initialize translational part with block matching result

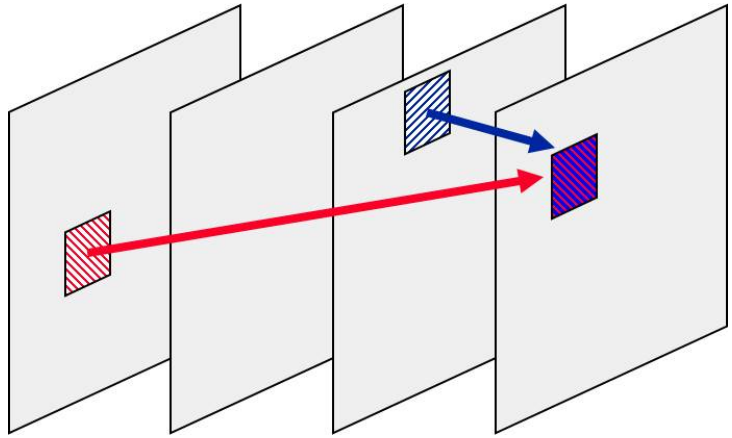
- Aperture problem: Estimate $\hat{\boldsymbol{a}}$ has a large relative error when the smallest eigenvalue of the gradient matrix $\boldsymbol{G}$ is small

  $\Longrightarrow$ Block has to contain large gradients in both directions for a reliable estimate

Motion Estimation for Multi-Hypotheses Prediction

Motion estimation for multiple motion vectors

- Need to estimate multiple motion vectors (typically for different reference pictures) for a block in current frame
- Independent estimation is sub-optimal
- Estimation in product space is too complex



Independent estimation of motion hypotheses is not optimal

- Example: SSD distortion for bi-prediction

$$
\begin{aligned}
D_{\mathrm{Bi}} &= \sum_{x,y} \Big(s[x,y] - \frac{1}{2}\big(\hat{s}_1[x,y] + \hat{s}_2[x,y]\big) \Big)^2 \\
&= \frac{1}{4} \sum_{x,y} \Big(\big(s[x,y] - \hat{s}_1[x,y]\big) + \big(s[x,y] - \hat{s}_2[x,y]\big) \Big)^2 \\
&= \frac{1}{4} D_1 + \frac{1}{4} D_2 + \frac{1}{2} \sum_{x,y} \big(s[x,y] - \hat{s}_1[x,y]\big)\big(s[x,y] - \hat{s}_2[x,y]\big) \qquad (662)
\end{aligned}
$$

$\Longrightarrow$ Minimization of D_1 and D_2 does not minimize D_{Bi}

Iterative Motion Estimation for Multi-Hypotheses Prediction

Iterative motion estimation for multiple motion hypotheses

- Example: Bi-prediction with the following assumptions
  - Motion vector for one hypothesis is given and yields prediction signal $\hat{s}_1[x,y]$
  - Want to estimate motion vector $[m_x^{(2)}, m_y^{(2)}]^{\mathrm{T}}$ for the second hypothesis
- Distortion for bi-prediction can be written as

$$\begin{aligned} D_{\mathrm{Bi}} &= \sum_{x,y} \left| s[x,y] - \frac{1}{2}\left(\hat{s}_1[x,y] + s_{\mathrm{ref}}^{(2)}\left(x - m_x^{(2)}, y - m_y^{(2)}\right)\right)\right|^{\beta} \\ &= \frac{1}{2^{\beta}} \cdot \sum_{x,y} \left| \left(2 \cdot s[x,y] - \hat{s}_1[x,y]\right) - s_{\mathrm{ref}}^{(2)}\left(x - m_x^{(2)}, y - m_y^{(2)}\right)\right|^{\beta} \\ &= \frac{1}{2^{\beta}} \cdot \sum_{x,y} \left| s^*[x,y] - s_{\mathrm{ref}}^{(2)}\left(x - m_x^{(2)}, y - m_y^{(2)}\right)\right|^{\beta} \end{aligned} \tag{663}$$

  $\Longrightarrow$ Conventional motion search, but with modified original signal $s^*[x,y]$
- Iterative algorithm for bi-prediction
  1. Independent estimation of first motion hypothesis
  2. Conditional estimation of second/first motion hypothesis (alternately)
  3. Repeat last step until convergence
- Algorithm can be extended to more than two hypotheses

Summary on Encoder Control

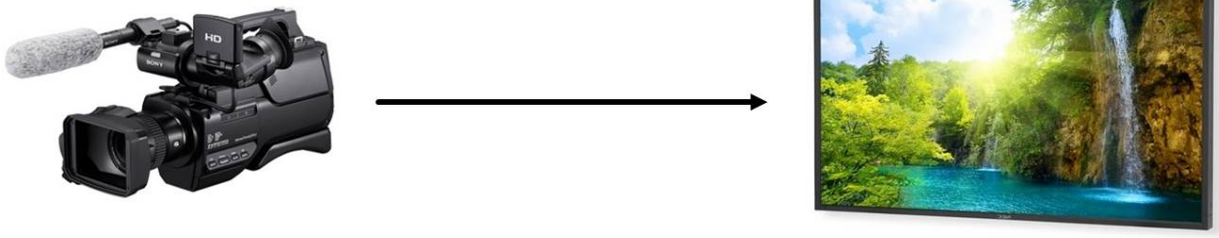
Lagrangian encoder control

- Minimization of cost function $D + \lambda \cdot R$
- In video coding: Need to partially neglect interdependencies
- Applications in video coding
  - Rate-distortion optimized quantization
  - Rate-distortion optimized mode decision
  - Rate-distortion optimized motion estimation

Motion estimation

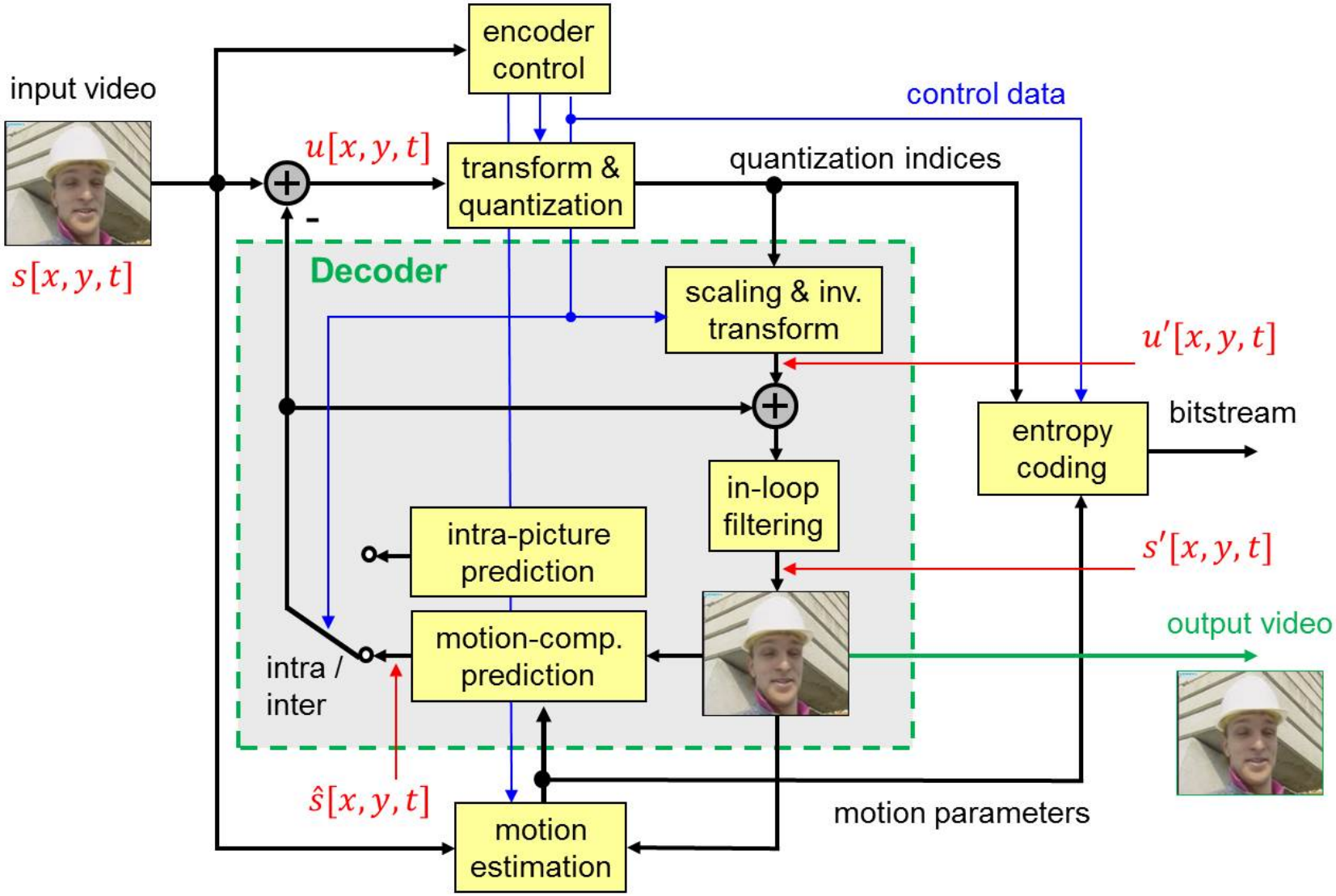
- Translational motion: Block matching
  - Early termination of distortion calculation
  - Fast search strategies
- Higher-Order motion models
  - Iterative differential motion search for linear motion models
  - Initialization with block matching result
- Motion estimation for multi-hypotheses prediction
  - Iterative motion search
  - Alternatively refinement of motion hypotheses

Video Coding Standards



Basic Coding Approach: Hybrid Video Coding

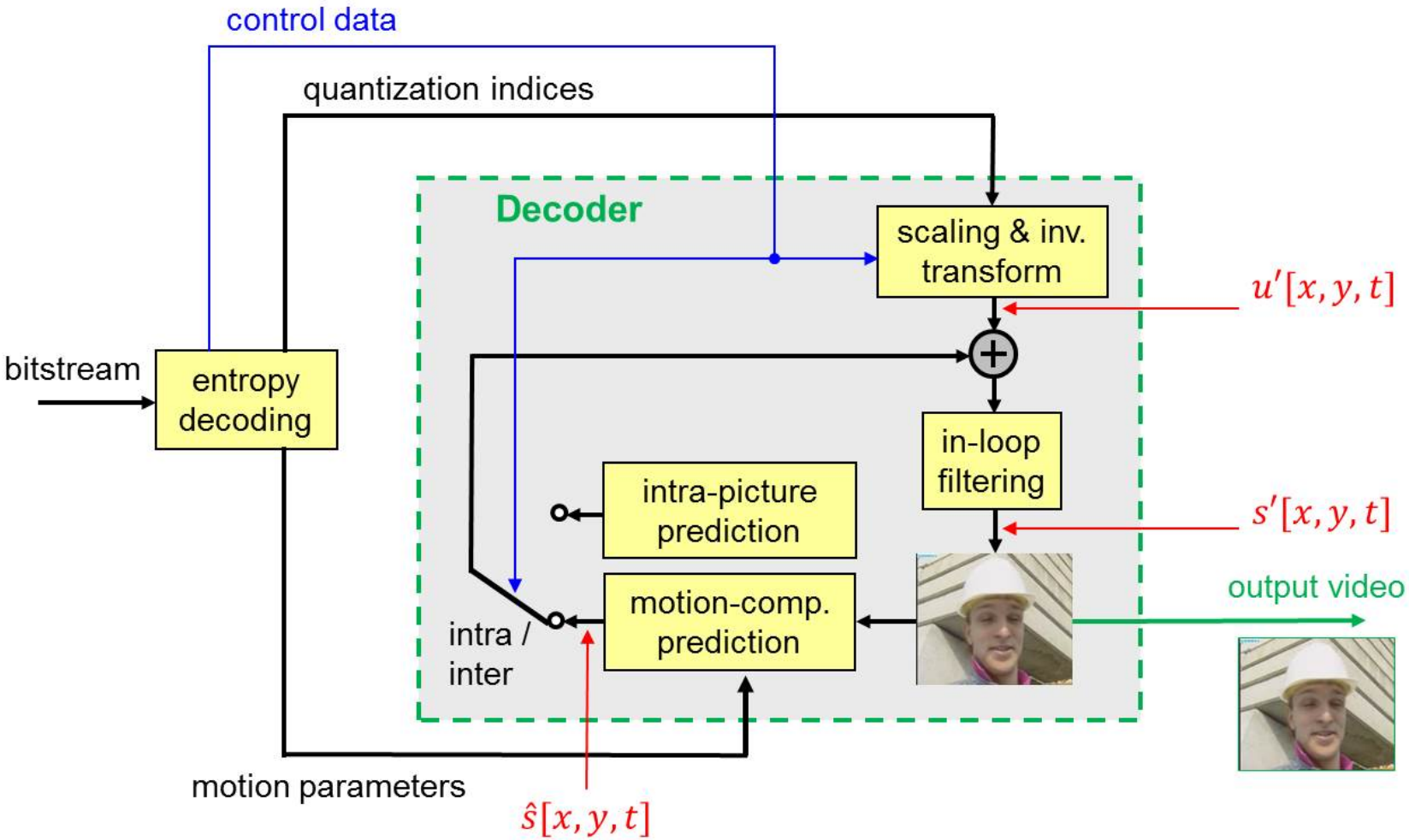
H.261, H.262/MPEG-2, H.263, MPEG-4, H.264/AVC, H.265/HEVC



Specification of Video Coding Standards

Video coding standards specify **bitstream syntax** and **decoding result**

- Enables interoperability between devices of different manufactures
- Leaves room for optimization (but does not guarantee any quality)

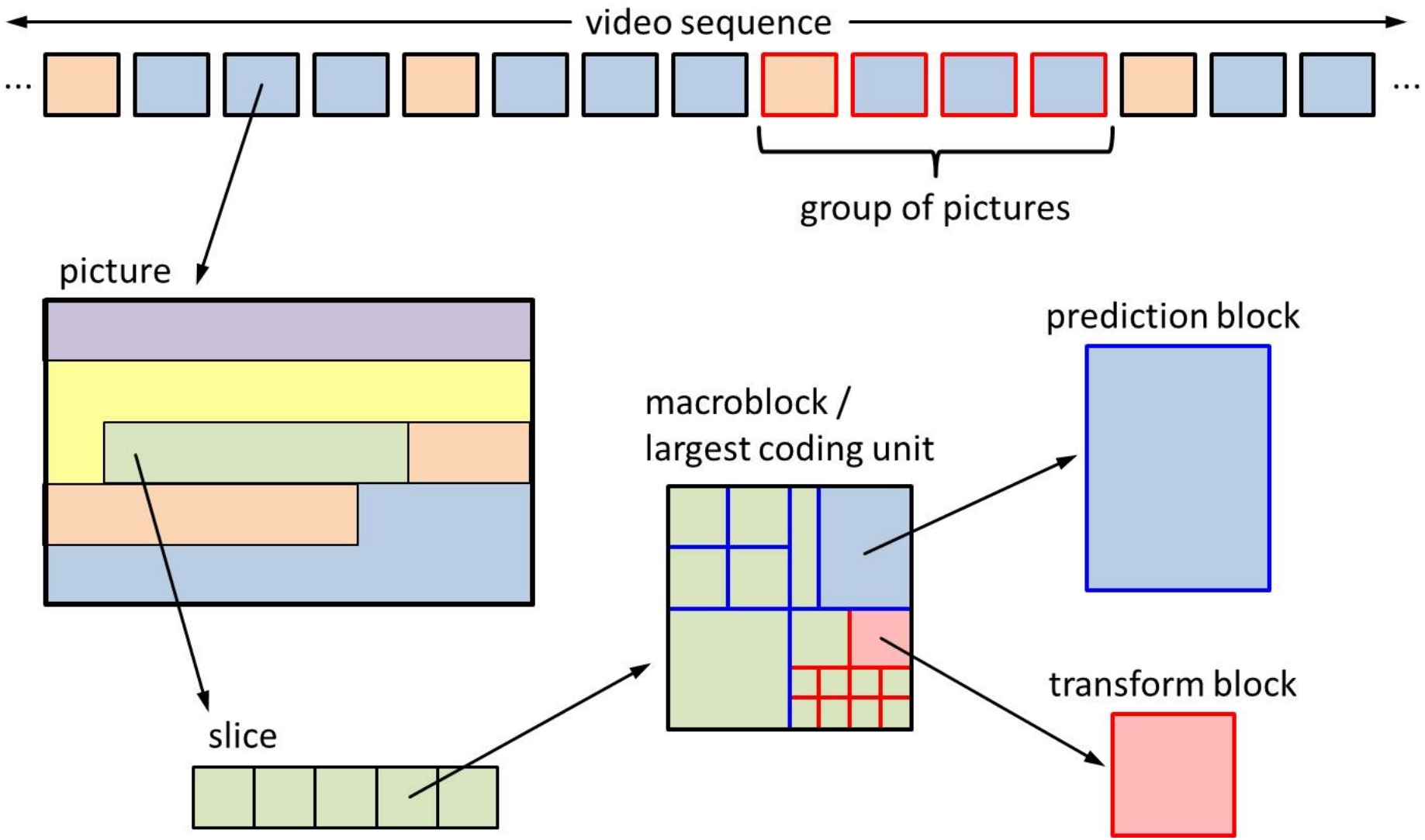


Application Areas of Video Coding Standards

digital television broadcasting	SD: 1.5 ... 6 Mbps HD: 5 ... 20 Mbps	MPEG-2, H.264/AVC
DVD-Video Blu-ray disc	5 ... 20 Mbps up to 40 Mbps	MPEG-2 MPEG-2, H.264/AVC, VC-1
Internet video streaming	100 ... 2000 kbps	H.264/AVC or proprietary codecs
video telephony video conferencing	20 ... 2000 kbps	H.263, H.264/AVC (incl. SVC extension)
video over 3G wireless networks	100 ... 500 kbps	H.263, MPEG-4, H.264/AVC

Note: H.265/HEVC is expected to be used in a significant number of application areas in near future

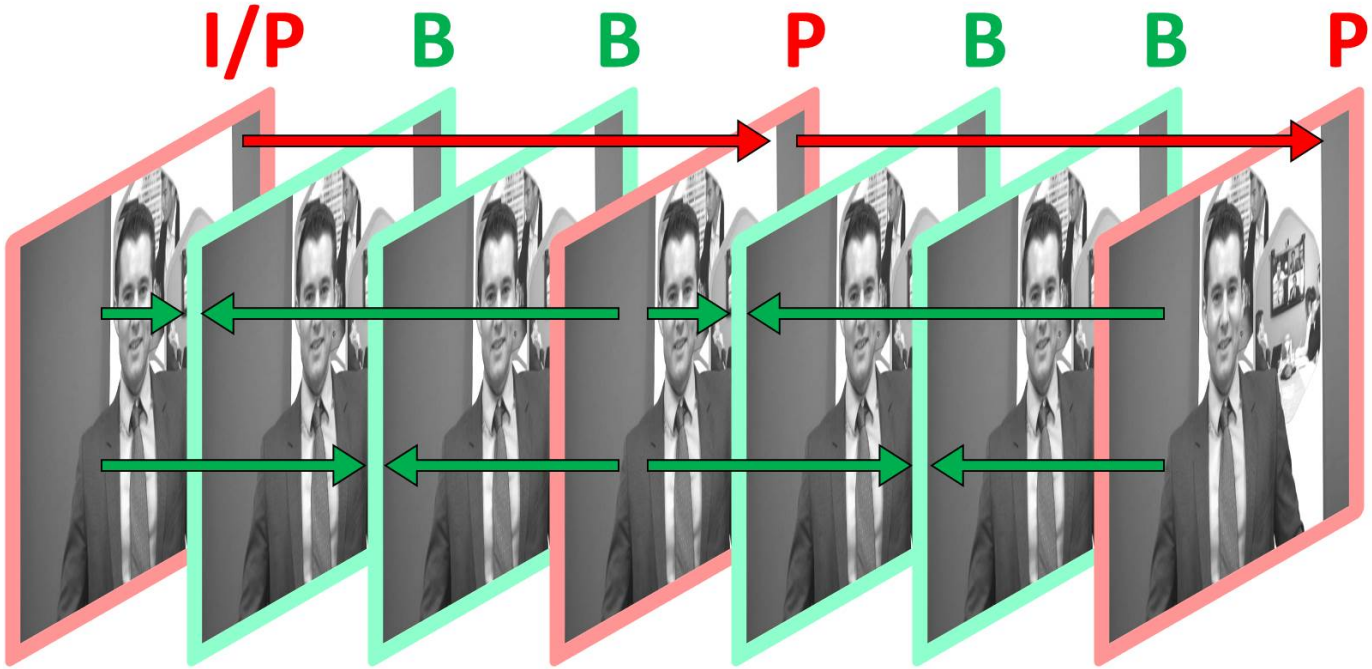
Hierarchical Bitstream Syntax



H.262 | MPEG-2 Video

(ITU-T Rec. H.262 | ISO/IEC 13818-2)

Pictures Types in H.262 | MPEG-2 Video



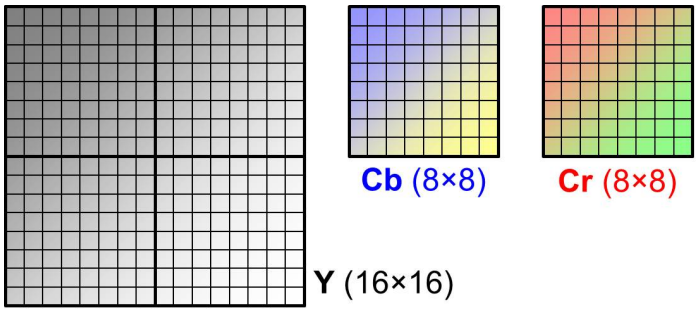
H.262 | MPEG-2 Video supports 3 pictures types:

- **I picture**: Intra-only coding (random access point)
- **P picture**: Predicted using previous I/P picture
- **B picture**: Predicted using previous I/P and next I/P picture

Picture Partitioning using Macroblocks

Partitioning of pictures into macroblocks

- Partitioning into fixed-size macroblocks
- Macroblock in 4:2:0 chroma format:
  - one 16×16 luma block
  - two 8×8 chroma blocks
- Slices: Consecutive MBs inside an MB row



For each macroblock, a coding mode can be selected

- Supported coding modes depends on picture type
- Supported modes are summarized below
  (without mentioning the modes which additionally support a quantizer change)

I picture	P picture	B picture
Intra	Intra	Intra
	P-Skip	B-Skip
	No MC, coded	Fwd, not coded
	MC, not coded	Fwd, coded
	MC, coded	Bwd, not coded
		Bwd, coded
		Interp, not coded
		Interp, coded

Macroblock Coding Modes in I and P Pictures

Intra coding mode in H.262 | MPEG-2 Video (brief review)

- Transform coding of 8×8 blocks with separable DCT and scalar quantization
- DC coefficient: Coding of difference to DC coefficient of previous block
- AC coefficients: Zig-zag scan and run-level coding with EOB symbol

Inter-picture macroblock coding modes in P pictures

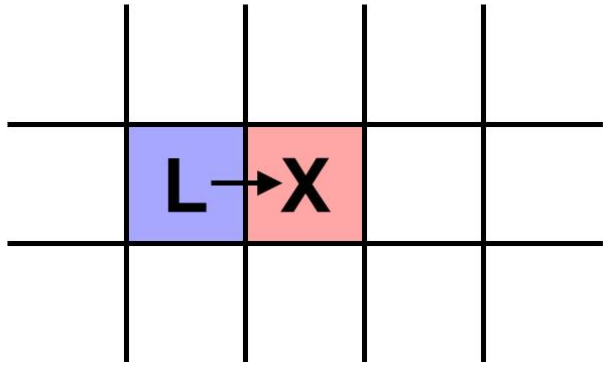
- Motion-compensated prediction using the previous I/P picture
- One motion vector for the entire macroblock
- Residual is transmitted using transform coding (similar to Intra)
  - Transform coding of 8×8 blocks using separable DCT and scalar quantization
  - Coded block pattern (VLC code indicating non-zero transform blocks)
  - Zig-zag scan and run-level coding (different table than for Intra)
- Special modes indicating that the motion vector and/or residual is zero

coding mode	motion vector	residual signal
P-Skip	inferred to be zero	inferred to be zero
No MC, coded	inferred to be zero	transmitted
MC, not coded	transmitted	inferred to be zero
MC, coded	transmitted	transmitted

Coding of Motion Vectors and Sub-Sample Interpolation

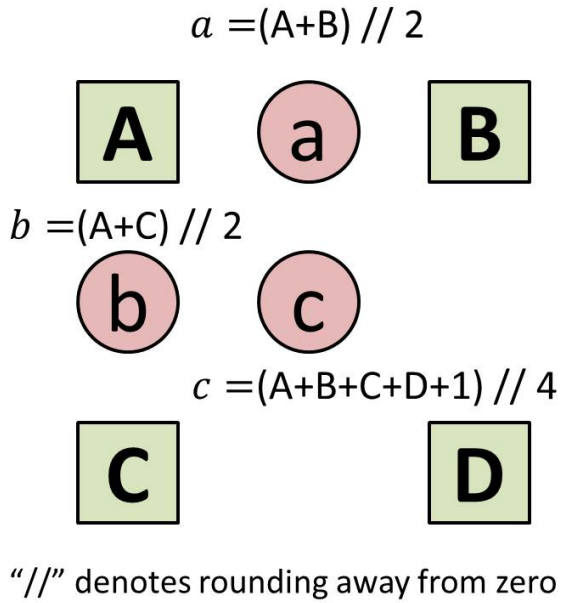
Coding of motion vectors

- Motion vector are transmitted with an accuracy of a half-luma sample
- Differential coding using the motion vector of the left macroblock as predictor
- Predictor is reset at beginning of a slice
- Motion vectors can only reference blocks inside the picture area



Sub-sample interpolation

- Samples at the integer grid are directly copied from the reference frame
- Bi-linear interpolation is used for sub-sample locations
- For chroma
  - Motion vector are first rounded to half-chroma sample precision
  - Then, bi-linear interpolation is used



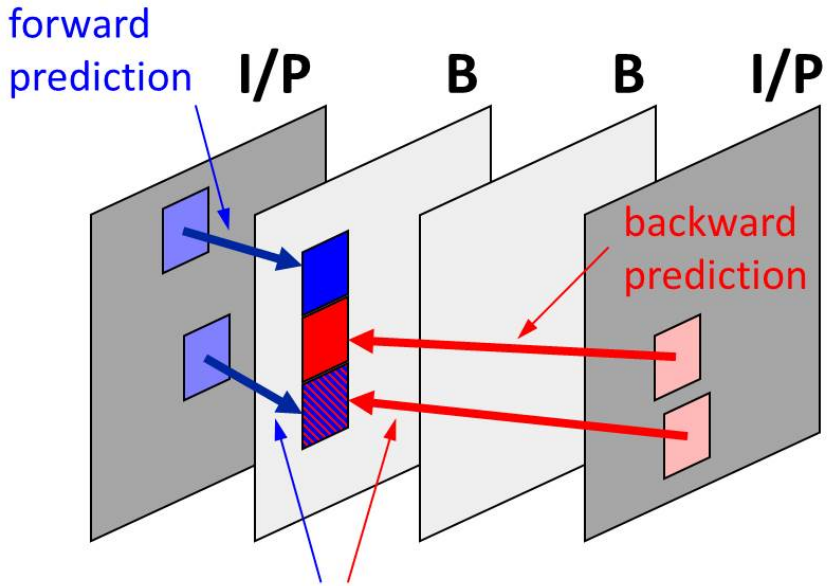
Motion-Compensated Prediction in B Pictures

Two reference pictures

- Previous I/P picture in display order
- Next I/P picture in display order (but preceding in coding order)

Three types of prediction

- Forward: Using previous I/P picture
- Backward: Using next I/P picture
- Bi-directional: Average of forward and backward prediction signal



bi-directional (interpolative) prediction

Inter-picture macroblock coding modes in B pictures

- One or two motion hypotheses
- One motion vector per hypothesis
- Coding mode signals type of prediction and if residual is zero
- B-Skip: Same motion hypotheses as macroblock to the left

coding mode	prediction	resiudal
B-Skip	inferred (left)	not coded
Fwd, not coded	forward	not coded
Fwd, coded	forward	coded
Bwd, not coded	backward	not coded
Bwd, coded	backward	coded
Interp, not coded	bi-directional	not coded
Interp, coded	bi-directional	coded

ITU-Rec. H.263

Overview of Main Syntax Features in ITU-Rec. H.263

Main syntax features are similar to that of H.262 | MPEG-2 Video

- 3 pictures types: I, P and B pictures (B pictures enabled by optional Annex O)
- Macroblock coding modes: Intra & Inter (motion-compensated prediction)
- Transform coding of intra or residual signal
- 8×8 DCT and scalar quantization

Main improvements relative to H.262 | MPEG-2 Video

- 3-d run-level-last coding for transform coefficient levels
- Component-wise median prediction for motion vectors
- Annex D: Motion vectors outside picture boundaries
- Annex F: Motion-compensated prediction with 8×8 blocks
- Annex I: Prediction of intra AC coefficient and adaptive scanning
- Annex J: Deblocking filter inside motion compensation loop
- Annex U: Multiple reference pictures

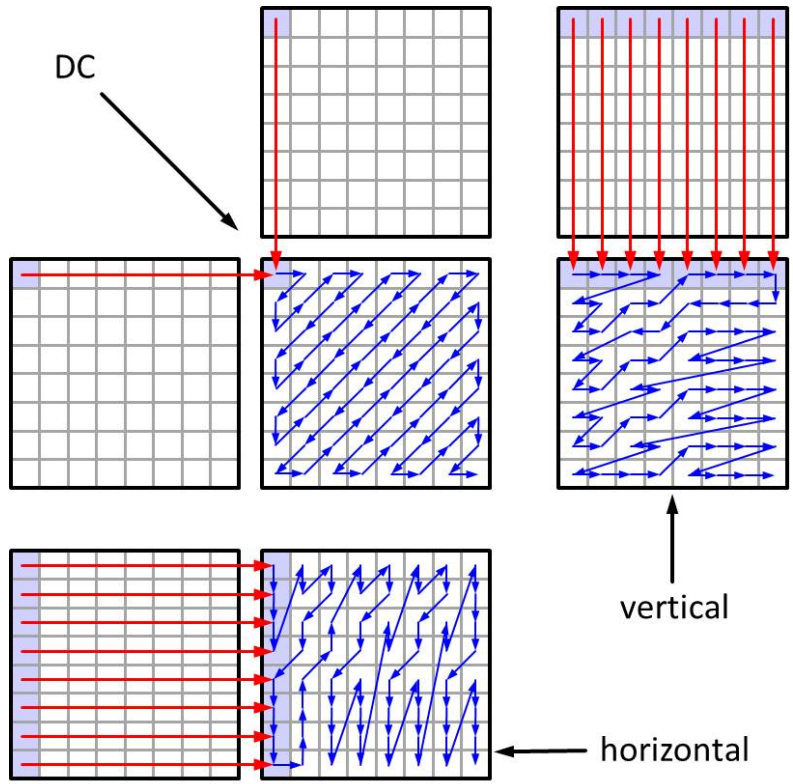
Improvements for Residual Coding and Intra Coding

Coding of the residual signal for a macroblock

- Same 8×8 DCT as in H.262 | MPEG-2 Video
- Scalar quantization (no support of quantization weighting matrices)
- **Run-level-last coding** of transform coefficient levels (instead of run-level coding)

Advanced intra coding mode (Annex I)

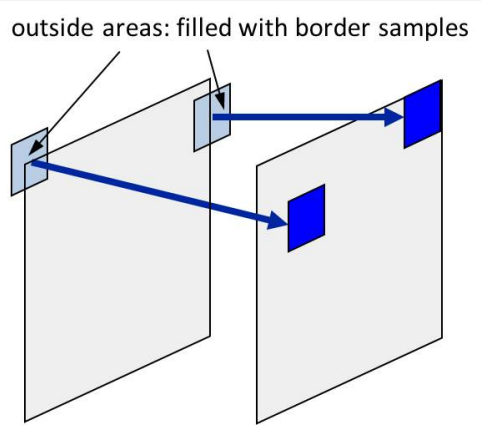
- 8×8 DCT and scalar quantization
- **Prediction of DC and AC coefficients** (signaled at macroblock level)
  - DC prediction
  - Vertical prediction
  - Horizontal prediction
- **Adaptive scanning** of transform coefficient levels
  - Determined by chosen intra prediction mode
- Run-level-last coding of transform coefficient levels



Improvements for Motion-Compensated Prediction

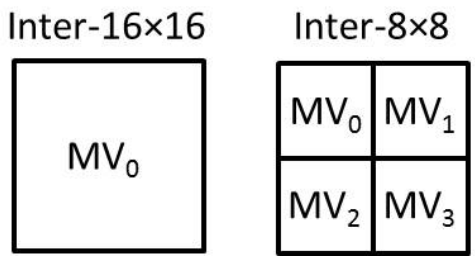
Motion vectors outside picture boundaries

- Specified in optional Annex D
- Motion vectors can reference blocks outside the picture area (not supported in MPEG-2)
- Outside areas are filled with corresponding border samples



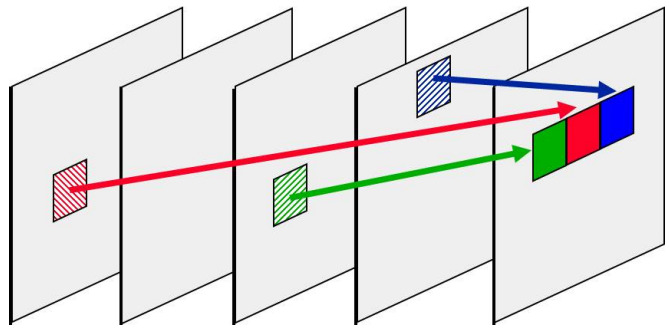
Variable block size motion compensation

- Supported in P pictures (optional Annex F)
- Inter-16×16: One motion vector per macroblock
- Inter-8×8: Four motion vectors per macroblock



Multiple reference pictures (optional Annex U)

- Transmit reference picture index in addition to motion vector
- Management of reference picture buffer
  - Sliding window operation
  - Explicit commands

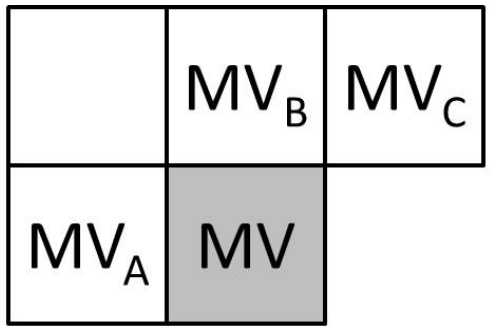


Further Improvements compared to H.262 | MPEG-2 Video

Motion vector coding

- Slices can cover multiple rows of macroblocks
- Motion vectors are predicted by the component-wise median of 3 neighboring motion vectors

$$\hat{m}_x = \text{median}\left(m_x^{(A)}, m_x^{(B)}, m_x^{(C)}\right)$$
$$\hat{m}_y = \text{median}\left(m_y^{(A)}, m_y^{(B)}, m_y^{(C)}\right)$$



- B pictures: Separate predictor for forward and backward prediction

Optional deblocking filter (Annex J)

- Deblocking filter for reducing block-edge artifacts
- Strength of smoothing filter is controlled by quantization parameter
- Improves subjective quality of current picture
- Improves "quality" of motion-compensated prediction signal of following pictures

MPEG-4 Visual

(ISO/IEC 14496-2)

Overview of Coding Tools in MPEG-4 Visual

Similar features as MPEG-2 Video or H.263

- I, P and B pictures and 16×16 macroblocks
- Intra and residual coding: 8×8 DCT, scalar quantization and run-level-last coding

Intra coding

- Prediction of transform coefficients and adaptive scanning (similar to H.263)
- DC is always predicted, prediction of AC coefficients can be selected on MB basis

Motion-compensated prediction

- Support of Inter-16×16 and Inter-8×8 mode
- Component-wise median prediction of motion vectors
- **No support of multiple reference pictures**
- **Quarter-sample accurate motion vectors** (Advanced Simple profile)
  - Half-sample interpolation: 8-tap filter
  - Quarter-sample interpolation: Bi-linear interpolation of half-sample grid
- **Global motion compensation** (rarely supported)
  - Perspective or affine motion model for background of picture
  - Prediction signal for macroblock is generated by warping

H.264 | MPEG-4 AVC

(ITU-T Rec. H.264 | ISO/IEC 14496-10)

Overview of Main Syntax Features in H.264 | MPEG-4 AVC

Commonalities with prior coding standards

- I, P and B pictures (actually I, P and B slices in H.264 | MPEG-4 AVC)
- 16×16 macroblocks supporting different coding modes
- Intra coding, uni-directional prediction or bi-prediction
- Motion vector coding with component-wise median prediction
- Transform coding with scalar quantization of residual signal

Main improvements relative to prior standards

- Decoupling of picture type, coding order and display order
- Spatial intra prediction
- Multiple reference pictures (more general than Annex U of H.263)
- More flexible partitioning of a macroblock for motion compensation
- Adaptive selection of transform size (High profile)
- Improved coding of transform coefficient levels
- Optional context-adaptive binary arithmetic coding (High profile)
- Deblocking filter (improved relative to Annex J of H.263)

Intra Coding and Residual Coding in H.264 | MPEG-4 AVC

Review of intra coding

- **Spatial intra prediction** & transform coding of residual signal
- Intra-4×4: Prediction and transform of 4×4 blocks (9 prediction modes)
- Intra-8×8: Prediction and transform of 8×8 blocks (9 prediction modes)
- Intra-16×16: Prediction of 16×16 block (4 prediction modes), transform of 4×4 blocks and second level Hadamard transform
- Intra-PCM: Direct coding of samples (fallback for high rates)

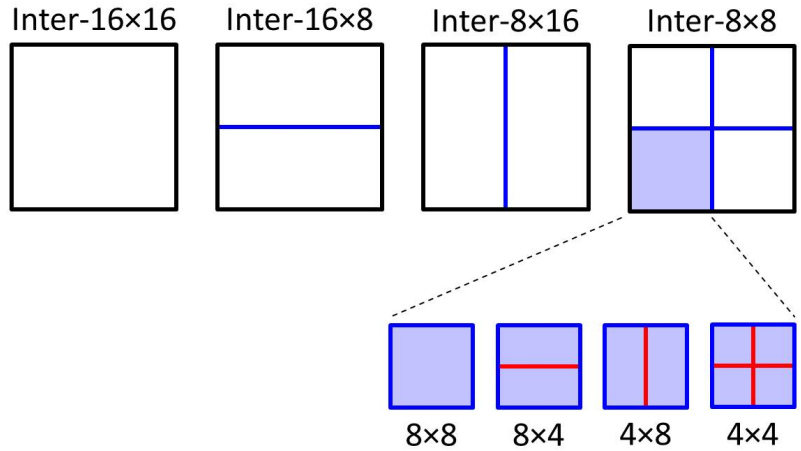
Transform coding of prediction residuals

- Transform coding using 4×4 or 8×8 transform and scalar quantization
- **Transform selection** on macroblock basis (if no MC blocks smaller than 8×8)
- Transforms are integer approximations of DCT
- **Inverse transform is specified by exact integer operations**
  - No accumulation of inverse transform mismatches
  - Encoder does not need to insert frequent intra updates
- Improved coding of transform coefficient levels (as discussed for intra)
  - Context-adaptive variable length coding (CAVLC)
  - **Context-adaptive binary arithmetic coding** (CABAC)

Improvements of Motion-Compensated Prediction

Flexible macroblock partitioning

- 4 inter coding modes with block sizes of 16×16, 16×8, 8×16 and 8×8
- For Inter-8×8 mode, sub-macroblock mode is transmitted for each 8×8 block
- Sub-macroblock mode indicates usage of 8×8, 8×4, 4×8 or 4×4 blocks



Multiple reference pictures

- Reference picture index is transmitted for each 16×16, 16×8, 8×16 or 8×8 block
- Reference picture buffer is managed by sliding window operation or explicit picture management commands (MMCO commands)
- Arbitrary construction of reference picture list using the available reference pictures

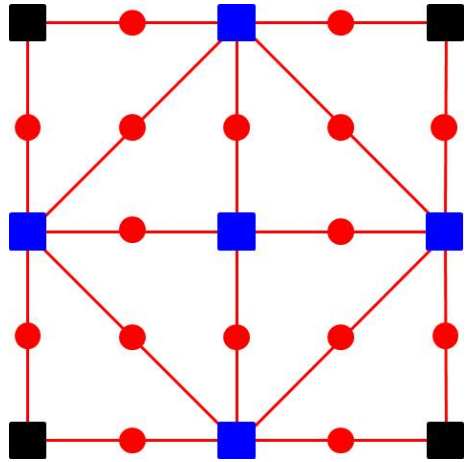
Motion-compensated prediction in B slices

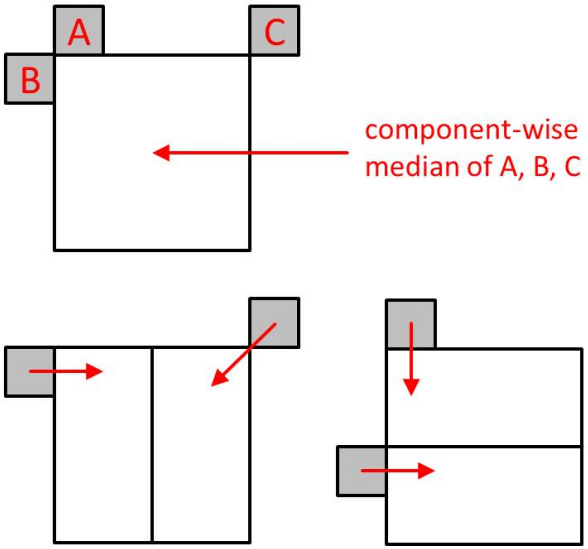
- Two reference lists (list 0 and list 1) can be arbitrarily constructed
- Prediction type (list 0, list 1 or bi-prediction) is transmitted for 16×16, 16×8, 8×16 blocks and 8×8 sub-macroblocks

Motion Vector Coding and Sub-Sample Interpolation

Motion vector accuracy and sub-sample interpolation

- Motion vector accuracy: Quarter luma sample
- Sub-sample interpolation for luma
  - Separable 6-tap filter for half-sample locations
  - Quarter-sample locations: Averaging two samples at integer and half-sample locations
- Sub-sample interpolation for chroma: Bi-linear





Coding of motion vectors

- Differential coding using a predictor
- Independent prediction per reference list
- In general: Component-wise median of the motion vectors of 3 neighboring blocks
- Some special conditions based on available motion vectors and reference picture indexes
- Special predictors: Inter-16×8 and Inter-8×16

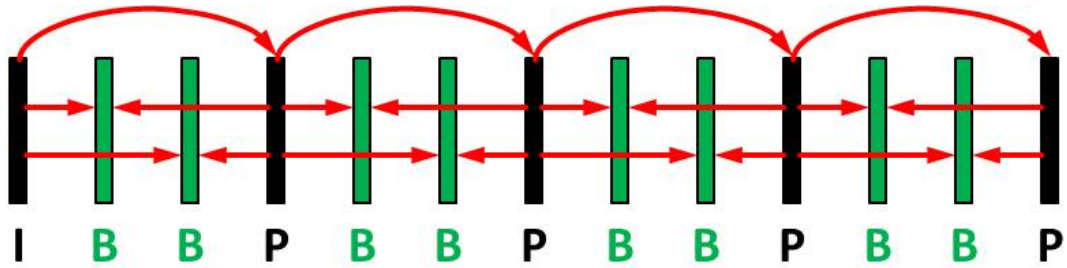
Decoupling of Picture Type, Coding and Display Order

Generalization of dependencies between pictures

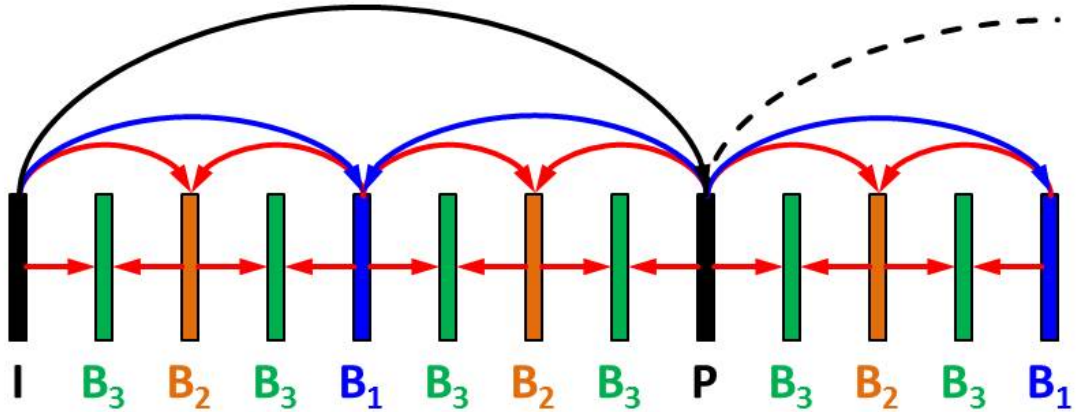
- A picture can consist of slices with different slice coding types (I, P, B)
- Each picture can be used as reference picture (as indicated in bitstream)
- Flexible coding order and construction of reference picture lists

$\Longrightarrow$ **Allows new types of prediction structures**

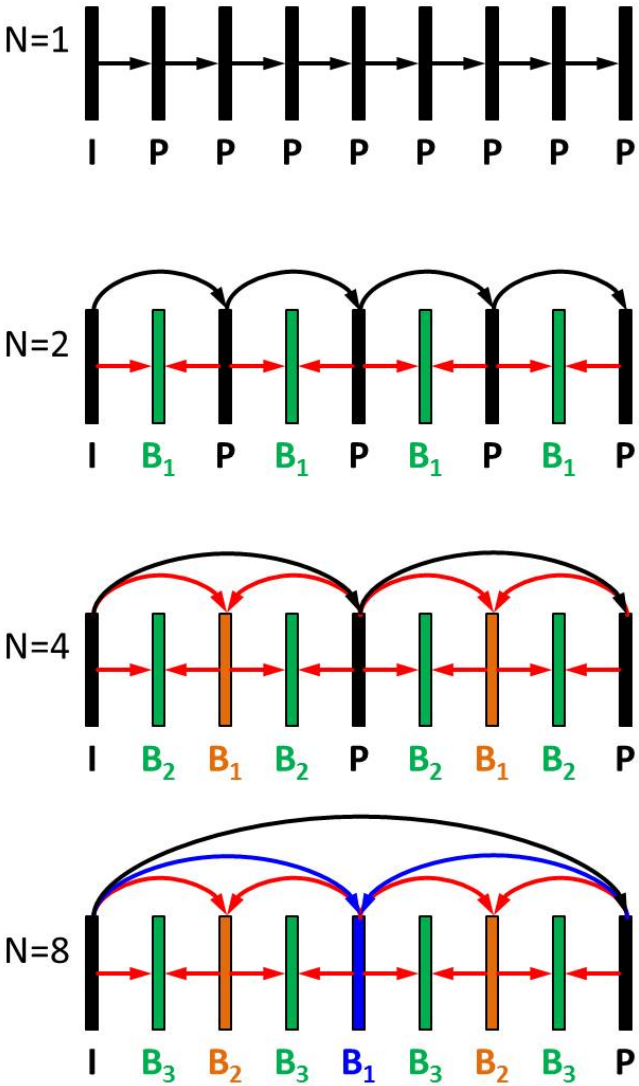
"traditional" prediction structure (2 B pictures)

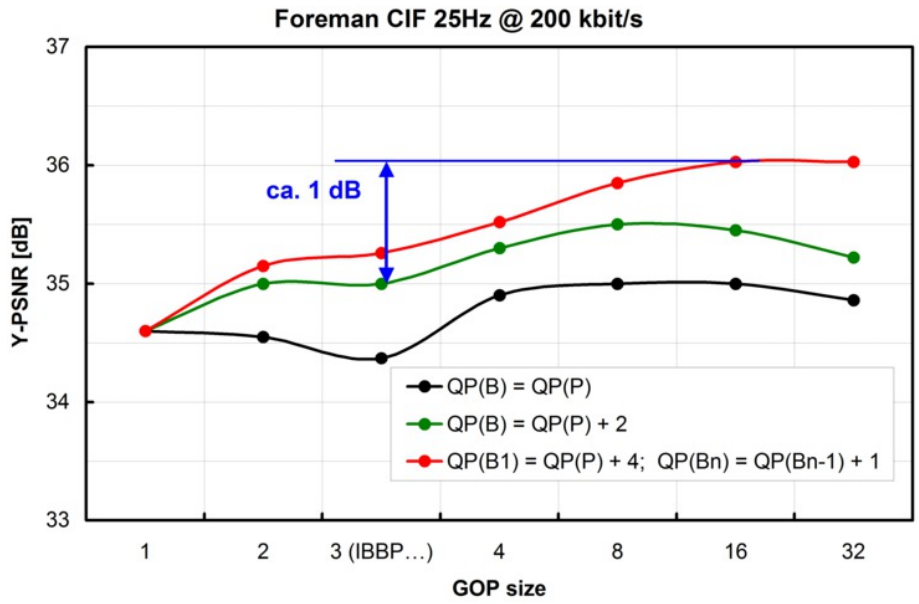


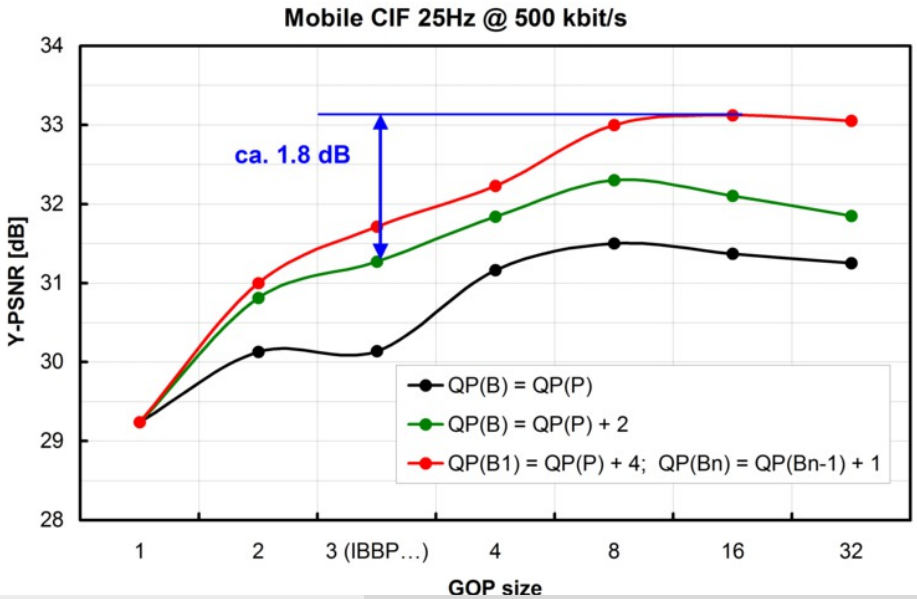
Example of a more general prediction structure:
Hierarchical B pictures



Performance of Hierarchical Prediction Structures







Subjective Quality Using Hierarchical Prediction Structures

Example: Sequence "Football" (CIF, 30Hz) at about 500 kbit/s

- Comparison of subjective quality
- Frame #206: Frame with highest QP (low PSNR) in hierarchical structure

conventional IBBP

Hierarchical B with GOP 16

Deblocking filter

Illustration of the filtering operation at block boundaries

- Filtering of $\mathbf{p_0}$ and $\mathbf{q_0}$ if all of the following conditions are fulfilled

$$\begin{aligned} |p_0 - q_0| &< \alpha(\mathrm{QP}) \\ |p_1 - p_0| &< \beta(\mathrm{QP}) \\ |q_1 - q_0| &< \beta(\mathrm{QP}) \end{aligned}$$

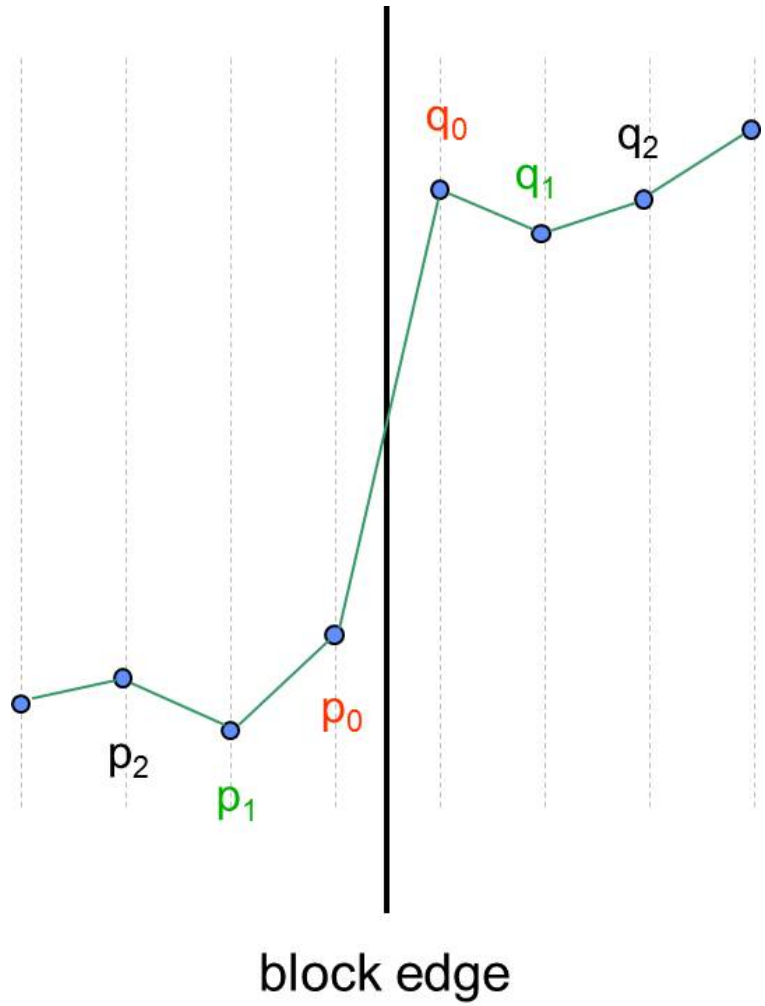
  where
  - $\alpha(\mathrm{QP})$ and $\beta(\mathrm{QP})$ increase with QP
  - $\alpha(\mathrm{QP})$ is larger than $\beta(\mathrm{QP})$

- The sample $\mathbf{p_1}$ is additionally filtered if

$$|p_2 - p_0| < \beta(\mathrm{QP})$$

- The sample $\mathbf{q_1}$ is additionally filtered if

$$|q_2 - q_0| < \beta(\mathrm{QP})$$



Subjective Quality Improvement due to Deblocking Filtering

Example: Highly compressed decoded picture

without deblocking filter

with deblocking filter

H.265 | MPEG-H HEVC

(ITU-T Rec. H.265 | ISO/IEC 23008-2)

Overview of Main Syntax Features in H.265 | MPEG-H HEVC

Commonalities with H.264 | MPEG-4 AVC

- I, P and B slices and similar high-level syntax concepts
- Conceptually similar reference picture buffer management
- Conceptually similar construction of reference picture lists
- Spatial intra prediction
- Transform coding of residual with scalar quantization
- Inverse transform specification by exact integer operations
- Quarter-sample accurate motion vectors
- Deblocking filter inside motion compensation loop

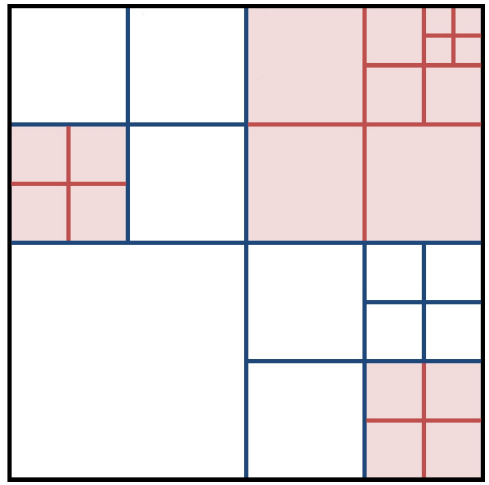
Main improvements relative to H.264 | MPEG-4 AVC

- Larger transform sizes and more flexible partitioning for transform coding
- Larger block sizes and more flexible partitioning for motion compensated prediction
- Larger number of spatial intra prediction modes
- Improved sub-sample interpolation filters
- Improved motion parameter coding
- Improved transform coefficient coding (particularly for larger transform blocks)
- Optional sample-adaptive offset filter inside motion compensation loop

Picture Partitioning, Residual and Intra Coding

Picture partitioning into coding units

- Picture partitioning into fixed-size coding tree units (CTUs) of 64×64, 32×32 or 16×16 luma samples
- Quadtree partitioning of CTUs into coding units (CUs)
- Coding units can be coded in **Intra** or **Inter** mode

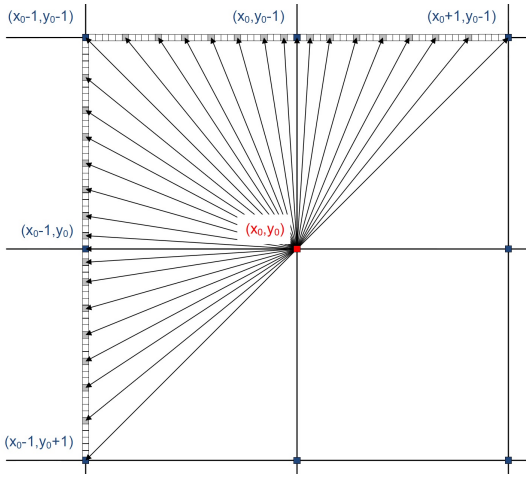


Residual coding of Inter CUs

- Quadtree partitioning of CUs into transform units (TUs)
- Transform coding of TUs with scalar quantization
- Transform sizes: 32×32, 16×16, 8×8 and 4×4
- Coding of transform coeff. levels based on 4×4 blocks
- Context-adaptive arithmetic coding (CABAC)

Coding of Intra CUs

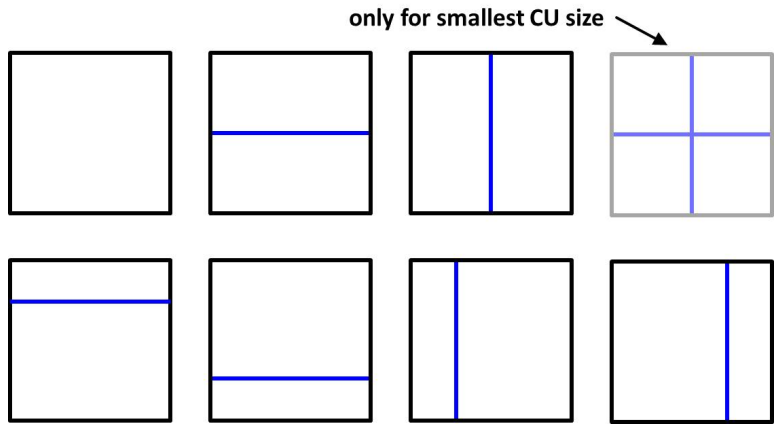
- Spatial intra prediction of TUs
- Transmission of 1 or 4 intra prediction modes per CU
- 35 intra prediction modes supported
- Same residual coding as for Inter CUs



Improvements for Motion-Compensated Prediction

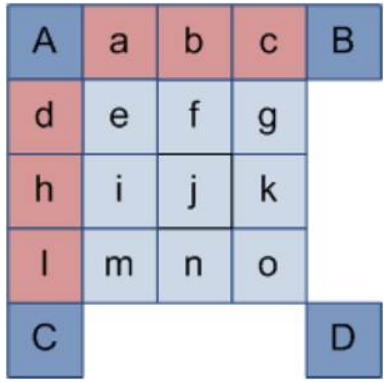
Partitioning of a CU for MCP

- Up to 8 possibilities for the partitioning of a CU into prediction units (PUs)
  - Splitting into 4 blocks only for smallest CU size
  - Asymmetric partitionings only for CUs larger than 16×16
- Selection of prediction type (list 0, list 1 or bi-prediction), reference picture(s) and motion vectors per PU



Motion vector accuracy and sample interpolation

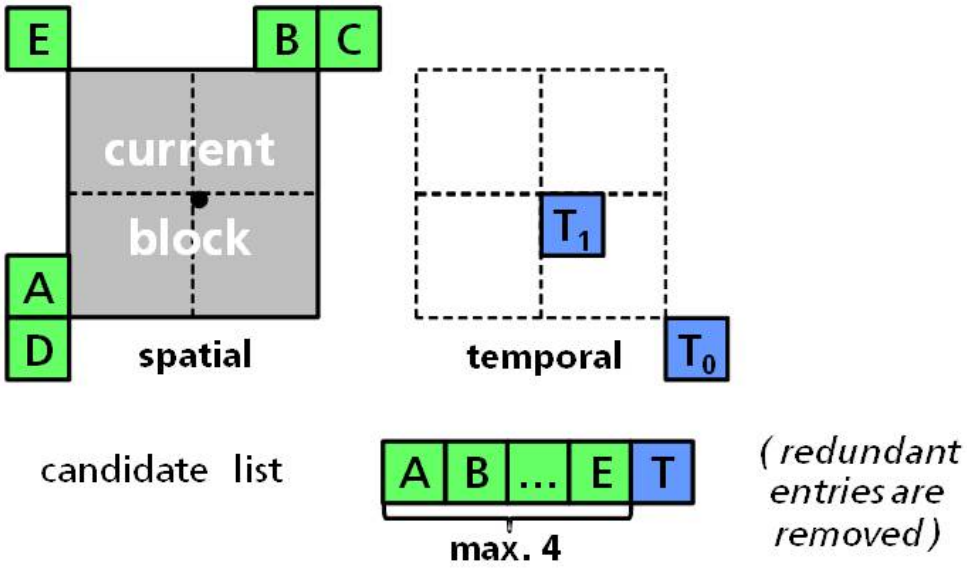
- Quarter-luma sample precision motion vectors
- Sub-sample interpolation for luma:
  - Separable 7- or 8-tap filters for all sub-sample positions
- Sub-sample interpolation for chroma:
  - Separable 4-tap filters

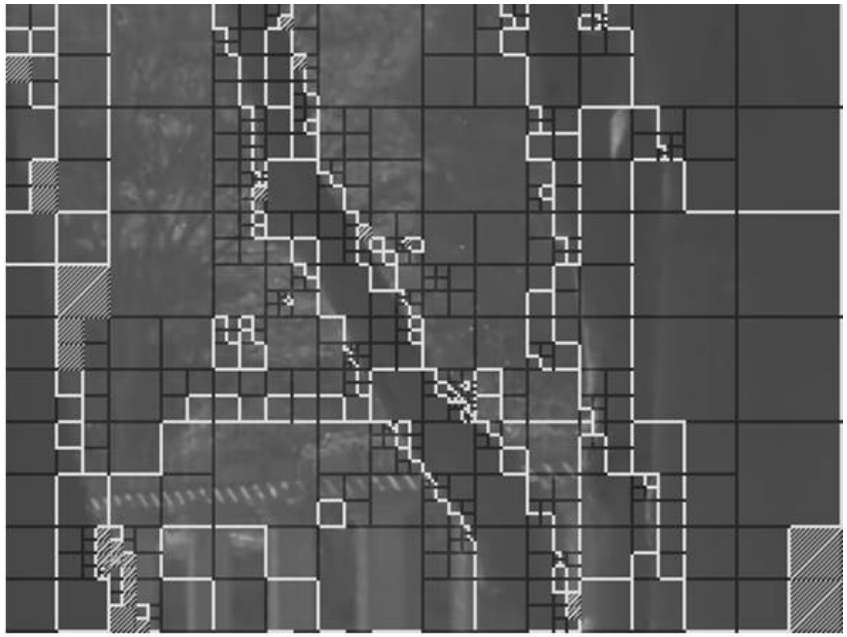


Coding of Motion Parameters: Merge Mode

Coding of prediction units in merge mode

- No transmission of prediction type, reference index or motion vector
- Prediction parameters are inferred from an already coded block
- Construction of a candidate list with up to five candidates:
  - Up to four spatially neighboring blocks
  - Up to one candidate derived from a co-located block in a reference picture
- Transmission of an index into the candidate list

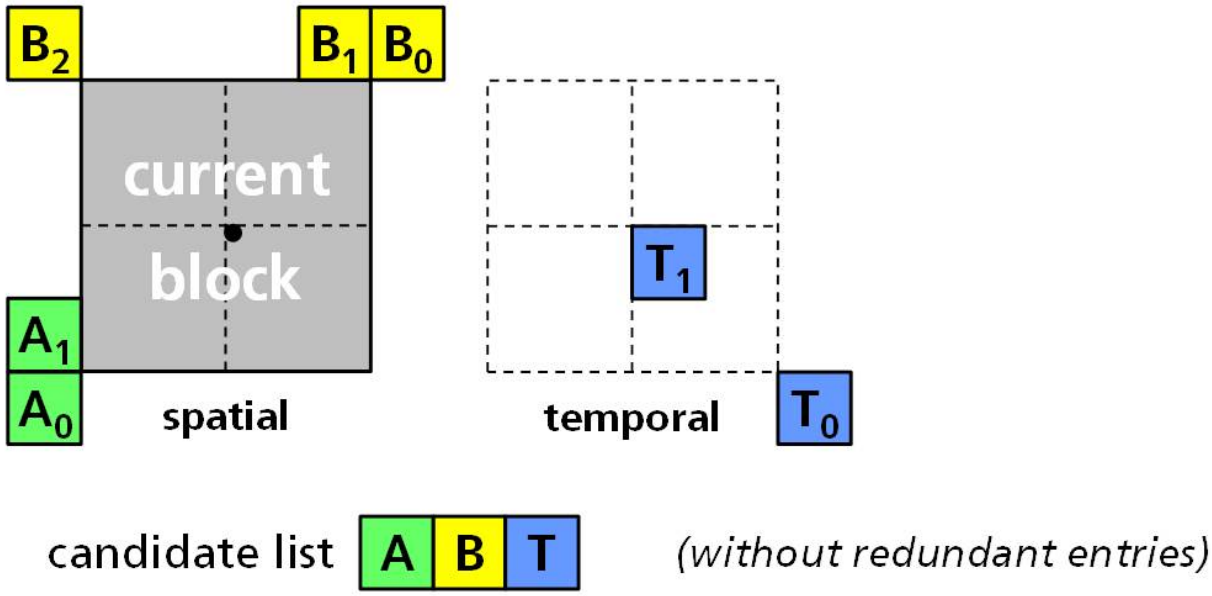




Coding of Motion Parameters: AMVP mode

Advanced motion vector prediction

- The following parameters are transmitted for PUs not coded in merge mode
  - Prediction type (list 0, list 1 or bi-prediction)
  - Per reference list: Reference index, motion vector difference, **predictor index**
- Motion vector predictor can be chosen between 3 predictors
  - Motion vector of 2 spatially neighboring blocks
  - A motion vector derived from co-located block in a reference picture



Coding Efficiency Comparison of Video Coding Standards

Coding Efficiency for Low-Delay Applications

Encoding constraints

- Bitstream characteristics suitable for low-delay applications
- Targeted application area: Video conferencing
- Constraint: Pictures are coded in display order

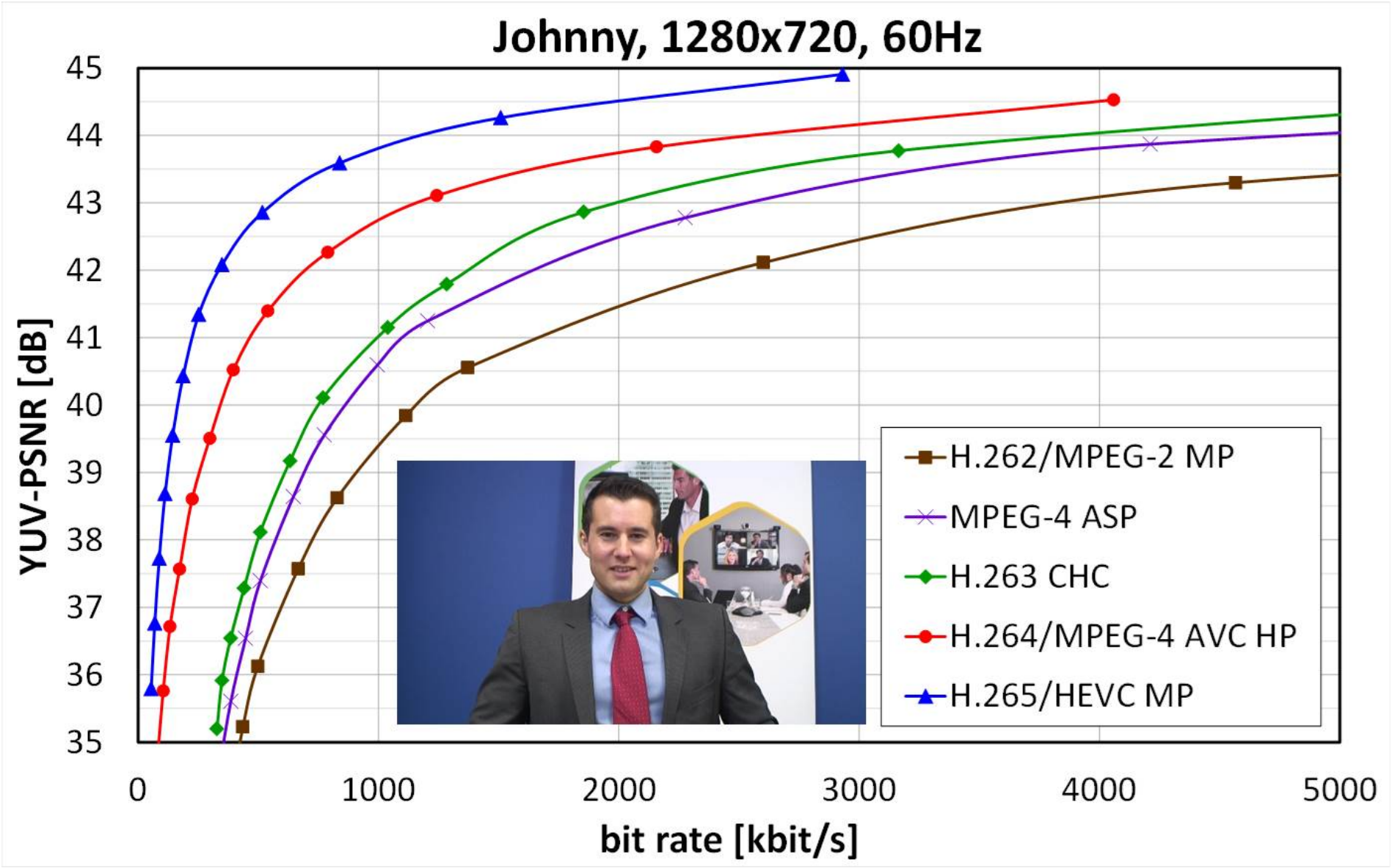
Investigated coding standards (best available configuration)

- MPEG-2 Main profile (IPPP coding structure)
- MPEG-4 Advanced Simple profile (IPPP coding structure)
- H.263 Conversational High Compression profile (IPPP coding structure)
- H.264/AVC High profile (low-delay GOP4 with P pictures)
- H.265/HEVC Main profile (low-delay GOP4 with B pictures)

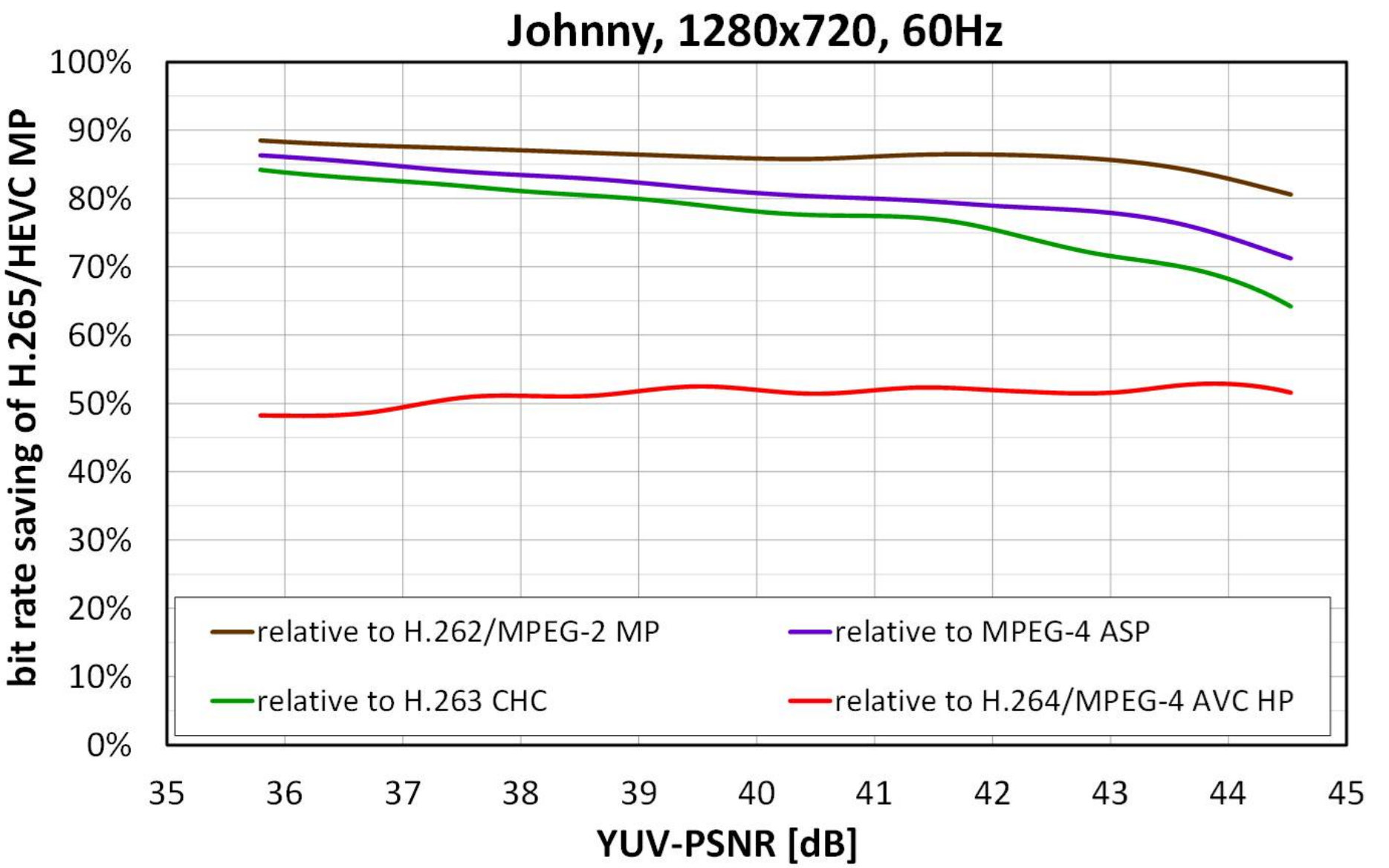
Encoder control

- Same Lagrangian encoder optimization for all encoders
- Same motion search strategy

Coding Efficiency for Low Delay: Example R-D Curves



Coding Efficiency for Low Delay: Example Rate Savings



Coding Efficiency for Low Delay: Summary

Average bit rate savings

- Averaged over covered PSNR range
- Averaged over test set of 6 video conferencing sequences

codec version	average bit rate savings relative to ...			
	H.264/AVC	H.263 CHC	MPEG-4 ASP	MPEG-2 MP
H.265/HEVC	40.3 %	67.9 %	72.3 %	80.1 %
H.264/AVC		46.8 %	54.1 %	67.0 %
H.263 CHC			13.2 %	37.4 %
MPEG-4 ASP				27.8 %

Coding Efficiency for Entertainment Applications

Encoding constraints

- Bitstream characteristics suitable for applications requiring random access
- Targeted application area: Broadcast, streaming, optical discs
- Constraint: Random access about every second (no delay constraint)

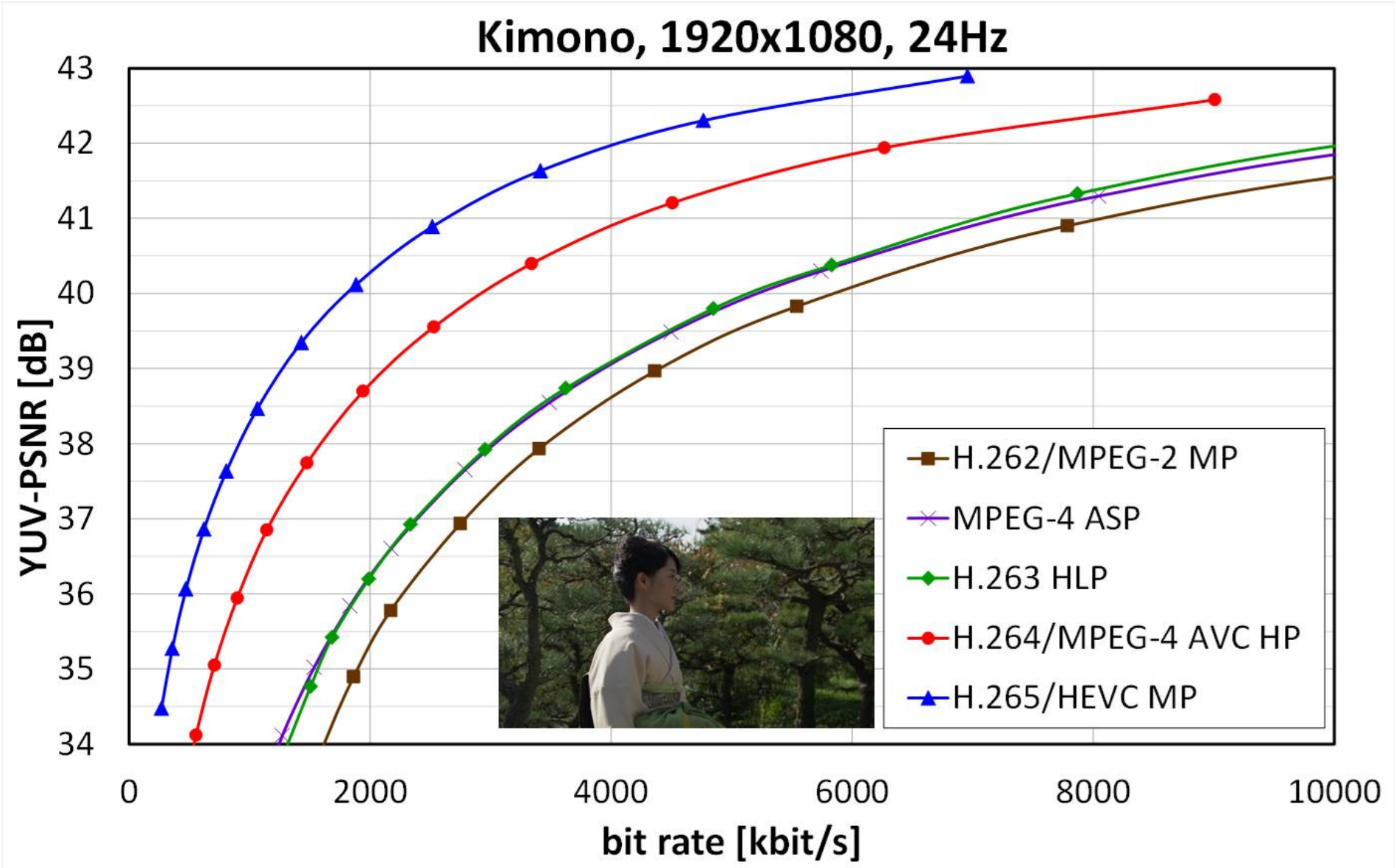
Investigated coding standards (best available configuration)

- MPEG-2 Main profile (IBBBP coding structure)
- MPEG-4 Advanced Simple profile (IBBBP coding structure)
- H.263 High Latency profile (IBBBP coding structure)
- H.264/AVC High profile (hierarchical B pictures with GOP8)
- H.265/HEVC Main profile (hierarchical B pictures with GOP8)

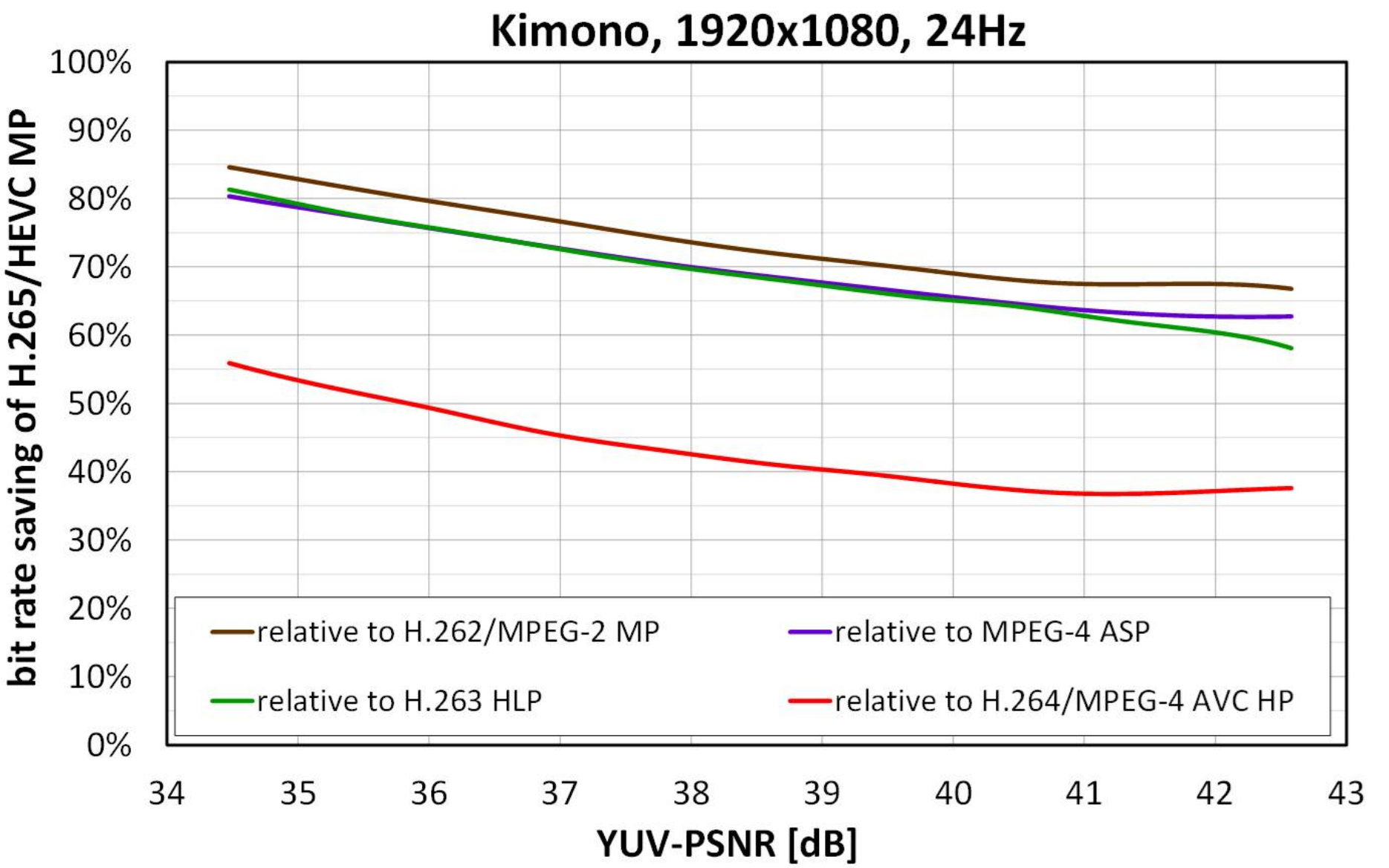
Encoder control

- Same Lagrangian encoder optimization for all encoders
- Same motion search strategy

Coding Efficiency for Random Access: Example R-D Curves



Coding Efficiency for Random Access: Example Rate Savings



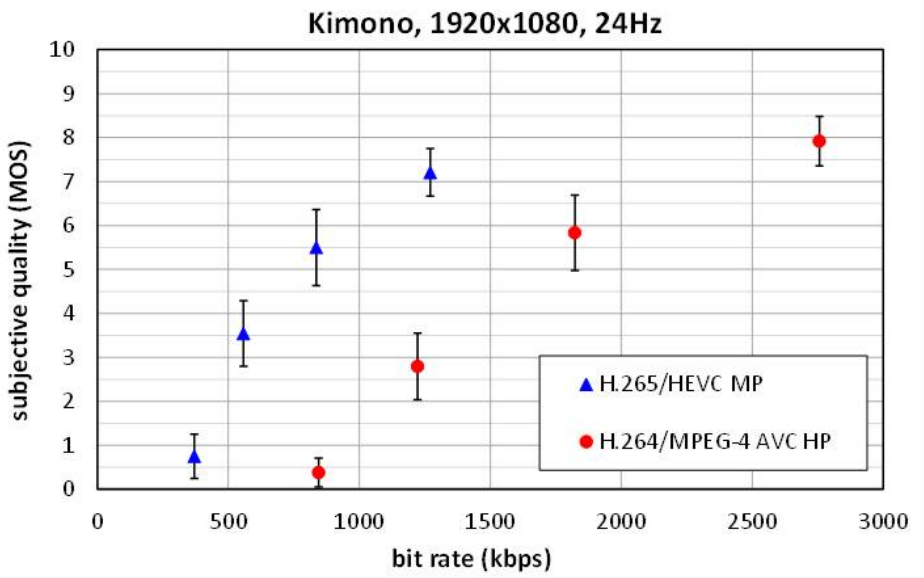
Coding Efficiency for Random Access: Summary

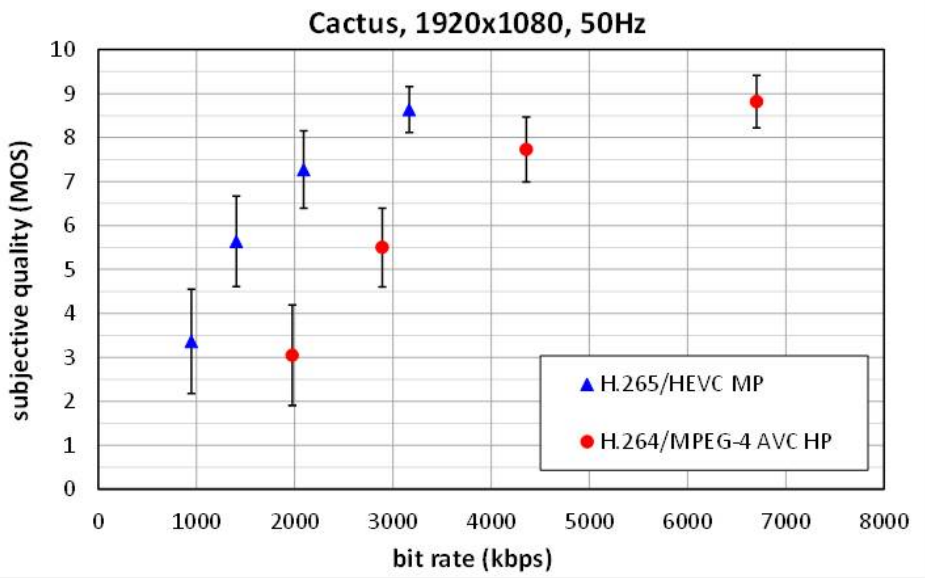
Average bit rate savings

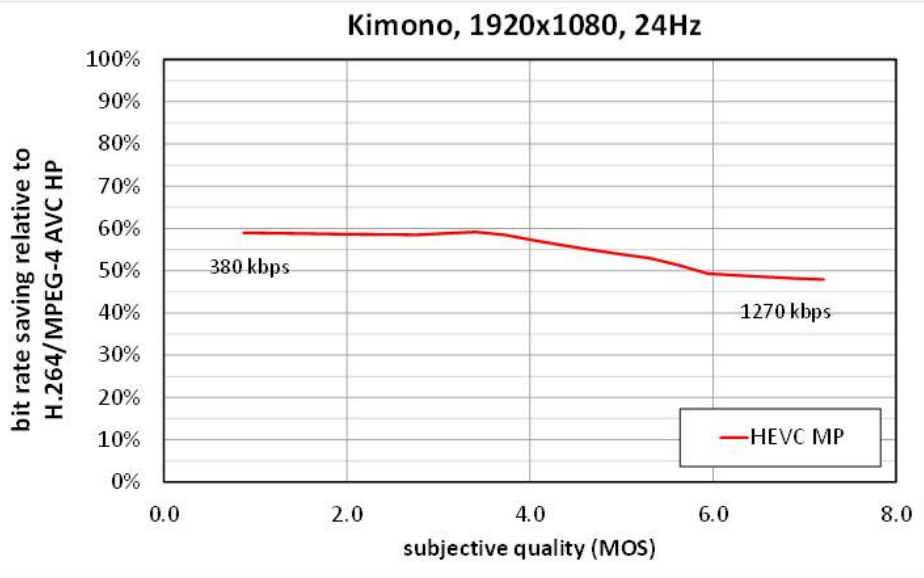
- Averaged over covered PSNR range
- Averaged over test set of 9 video sequences

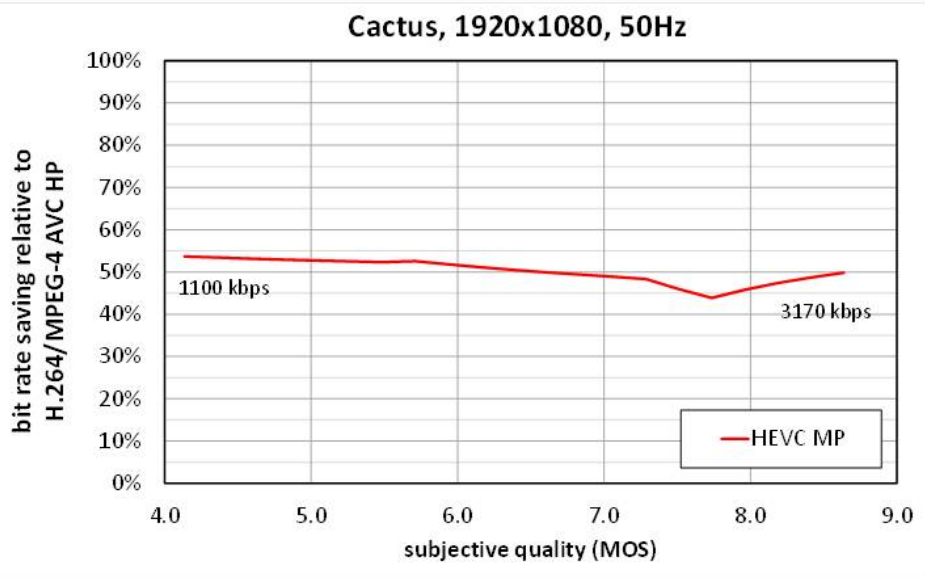
codec version	average bit rate savings relative to ...			
	H.264/AVC	MPEG-4 ASP	H.263 HLP	MPEG-2 MP
H.265/HEVC	35.4 %	63.7 %	65.1 %	70.8 %
H.264/AVC		44.5 %	46.6 %	55.4 %
MPEG-4 ASP			3.9	19.7 %
H.263 HLP				16.2 %

Subjective Comparison of H.264/AVC and H.265/HEVC









Summary on Video Coding Standards

Video coding standards

- All standards follow the hybrid video coding design
- Continuous improvement of coding efficiency
- To a large extend enabled by complexity increases

Key features for improving the coding efficiency

- Accuracy of motion vectors
- Interpolation filters
- Coding of motion vectors
- Partitioning for motion compensation, intra prediction and transform coding
- Spatial intra prediction
- Coding of transform coefficient levels
- Entropy coding
- In-loop filtering
- Generalization of supported prediction structures