Image/video compression: howto?

Aline ROUMY
INRIA Rennes

October 2016

1. **Why a need to compress video?**
2. How-to compress (lossless)?
3. Lossy compression
4. Transform-based compression
5. Prediction-based compression

1

Why a need to compress video?

Definition

Compression = store same image/video with less bits

Compression standards: given by

ITU International Telecommunication Union

ISO International Organization for Standardization

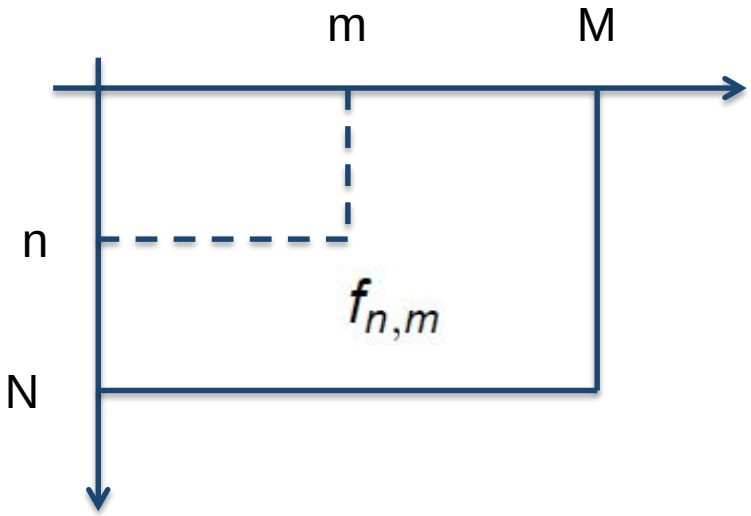
Joint teams:

JVT Joint Video Team (H.264/MPEG-4-AVC in 2003)

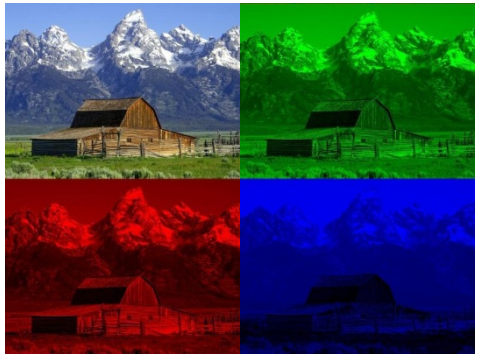
JCT-VC Joint Collaborative Team on Video Coding for development de HEVC (High efficiency video codec)
(Jan. 2013, ITU-T H.265 Avril 2013)

What is a digital image?

- Discrete grid with M (columns) x N (rows) pixels
- Canonical processing order: scanning row-by-row from left to right)



Color image decomposition



R G B (red green blue)

YCbCr (called YUV)

Y luminance = 0,3R + 0,6V + 0,1B

Cb Cr chrominance

Cb ~Y - B

Cr ~Y - R

backward compatible TV B&W

eye +sensitive to Y than CbCr

Chroma sub-sampling

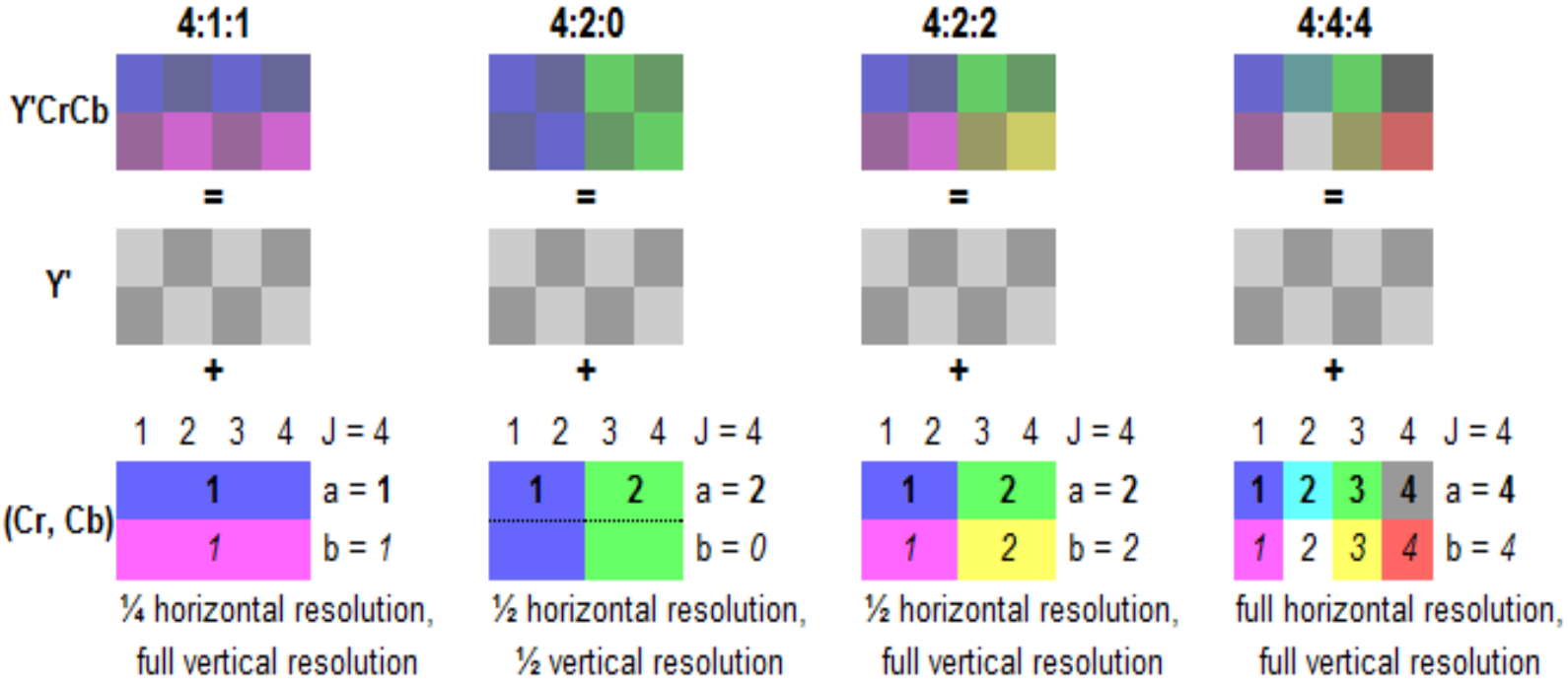
Color sampling is defined by three terms: ***J* : *a* : *b***

J Horizontal width of the sampling reference region, usually 4

a Number of chrominance samples (Cr, Cb) in the first row of *J* pixels

b Number of **new** chrominance samples (Cr, Cb) in the second row of *J* pixels

Chroma sub-sampling



Why a need to compress?

i.e. transmit same image with less bits.
Example:
HD 1080p = 1920 × 1080, 50 Hz (images/sec)

	Y luminance	Cb Chrominance	Cr Chrominance
Nb pixels	1920x1080= 2.1 M	960x540= .5 M	960x540= .5 M
Nb bits par pixel	8 bit	8 bit	8 bit
Raw data rate	**1.2 Gbps**		

Some data rate vs storage capacity (under evolution)

- DVD 4.7 GB — 3s
- TNT — 20 **M**bps
- Ethernet/Fast Ethernet — < 10 / 100 / 1000 **M**bps
- DSL downlink — 384 … 2048 **k**bps
- Mobile phone (true datarate 2G /3G /4G) — 9.05 / 384 **k**bps / 40 **M**bps

Quality measure

Objective criteria to measure the quality of image compression**,**

mapping that depends on

$f_{n,m}$: original image

$\tilde{f}_{n,m}$: reconstructed (after compression)

Mean square error (MSE):

$$\mathcal{D} = \frac{1}{N\,M} \sum_{n=1}^{N} \sum_{m=1}^{M} (f_{n,m} - \tilde{f}_{n,m})^2$$

Peak signal-to-noise ratio:

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathcal{D}} \right) \quad \text{dB}$$

1. Why a need to compress video?
2. **How-to compress (lossless)?**
3. Lossy compression
4. Transform-based compression
5. Prediction-based compression

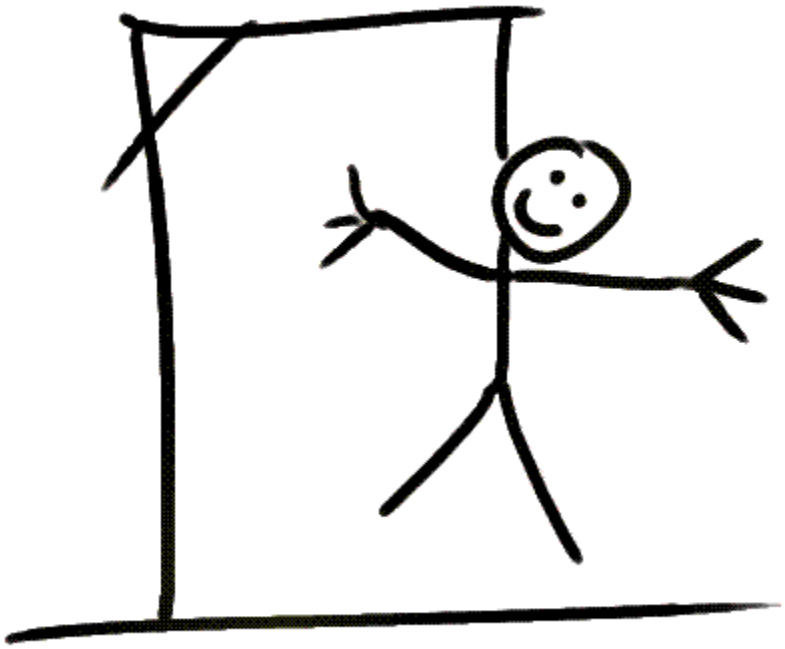
2

How-to compress?

Hangman game

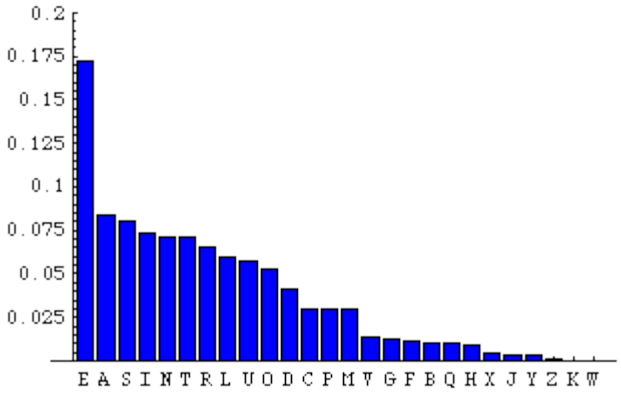
Objective = play… and explain your strategy

_ _ _ _ _



2 winning ideas

1. Letter frequency — probability



2. Correlation between successive letters — dependence

	_A	_B	_C	_D	_E	_F	_G	_H	_I	_J	_K	_L	_M	_N	_O	_P	_Q	_R	_S
A_	31	242	392	208	48	135	232	37	1255	32	7	663	350	1378	17	412	44	905	409
B_	158	2	1	2	130	1	2	0	132	4	10	181	1	1	146	1	3	187	29
C_	312	0	73	19	765	2	2	411	209	3	5	124	5	1	677	11	7	100	14
D_	427	1	8	24	2409	2	5	25	378	3	0	14	21	5	231	4	6	134	64
E_	616	176	917	998	782	258	209	67	179	96	8	1382	1056	2121	136	699	190	1514	3318

Analogy hangman game- compression

- word
- Answer to a question (yes/no).
- Goal: propose a minimum number of letter

- image
- 1 bit of the bistream that represents image
- Goal: propose a minimum number of bits

Exploit idea 1:
Use “non uniformity” of the probabilities

Consider a car race:

Name	A natole	B arnabe	C hef	D arius
Probability to win	1/4	1/2	1/8	1/8

A wins 1 out of 4, etc

For instance, let us consider the winner sequence:

ABBCBBDABCBABBCBBDAABABC…

Question: encode the winner sequence into a bitsequence…

in an efficient way.

Exploit idea 1:
Use “non uniformity” of the probabilities

Consider a car race:

Nom	A natole	B arnabé	C hef	D arius
Probability	1/4	1/2	1/8	1/8
Codeword	00	01	10	11

First try to encode the winner sequence:

ABBCBBDABCBABBCBBDAABABC…

00 01 01 10 01 01 … etc

Average length 2 bits/race

Better idea?

Exploit idea 1:
Use “non uniformity” of the probabilities

Consider a car race:

Nom	A natole	B arnabé	C hef	D arius
probability	1/4	1/2	1/8	1/8
codeword	10	0	110	111

Second try to encode the winner sequence:

ABBCBBDABCBABBCBBDAABABC…

Average length (fixed length code) 2 bits/race

Average length 2*1/4+1*1/2+3*1/8+3*1/8 1,75 bits/race

Optimal length [*information theory Shannon48*] 1,75 bits/race

Idea 2: use dependence between data

Assume that the car races results are dependent

course 2 / course 1	_A	_B	_C	_D		
A_	.25	.5	.125	.125		
B_	.25	.5	.125	.125		
C_	.25	.5	0	.25		
D_	.25	.5	.25	0		

Idea 2: use dependence between data

	_A	_B	_C	_D	average length	Optimal code
A_	.25	.5	.125	.125		
	10	0	110	111	1.75	1.75
B_	.25	.5	.125	.125		
	10	0	110	111	1.75	1.75
C_	.25	.5	0	.25		
	10	0		11	1.5	1.5
D_	.25	.5	.25	0		
	10	0	11		1.5	1.5

Idea 2: use dependence between data

	_A	_B	_C	_D	Average length	Optimal code
A_	.25	.5	.125	.125		
.25	10	0	110	111	1.75	1.75
B_	.25	.5	.125	.125		
.5	10	0	110	111	1.75	1.75
C_	.25	.5	0	.25		
.125	10	0		11	1.5	1.5
D_	.25	.5	.25	0		
.125	10	0	11		1.5	1.5
					1.69	1.69

Complexity: storage grows *exponentially* with the number of letters jointly processed i.e. the memory: from 4 to 16 $=4^2$ codewords

Information theory: a primer

Random process (source) $\{X_n\}$ Example: model of the pixels

Minimal average length (in bits) to represent one sample

- **discrete i.i.d.** source [Shannon]: **ENTROPY**

$$H(X_1) = -\sum_x p(x) \log_2 p(x) \quad \text{bits/sample}$$

- **discrete** stationary **ergodic** source [Shannon-McMillan-Breiman]

$$\lim_{n \to +\infty} \frac{1}{n} H(X_1, ..., X_n) \quad \text{bits/sample}$$

$$H(X_1, ..., X_n) = -\sum_{x_1, ..., x_n} p(x_1, \ldots, x_n) \log_2 p(x_1, \ldots, x_n)$$

Information theory

Theorem: independence bound for joint entropy

$$H(X_1, ..., X_n) \leq H(X_1) + ... + H(X_n)$$

equality iff $X_1, \ldots, X_n$ independents

Interpretation

1. Exploit the dependency allows to reduce the number of bits. BUT requires to store the joint law (complexity grows exponentially with n)
2. Separate encoding of independent data is optimal.

Information theory

Exploit idea: encode SEPARATELY dependent data

- Transform the random vector X into an equivalent vector Y

$$H(X_1, ..., X_n) = H(Y_1, ..., Y_n) \lesssim H(Y_1) + ... + H(Y_n)$$

Invertible transform

Y independents

Transform the data s.t. the transformed data are independent and s.t. one can compress the data separately (scalar entropy coding = coding with the marginal law).

1. Why a need to compress video?
2. How-to compress (lossless)?
3. **Lossy compression**
4. Transform-based compression
5. Prediction-based compression

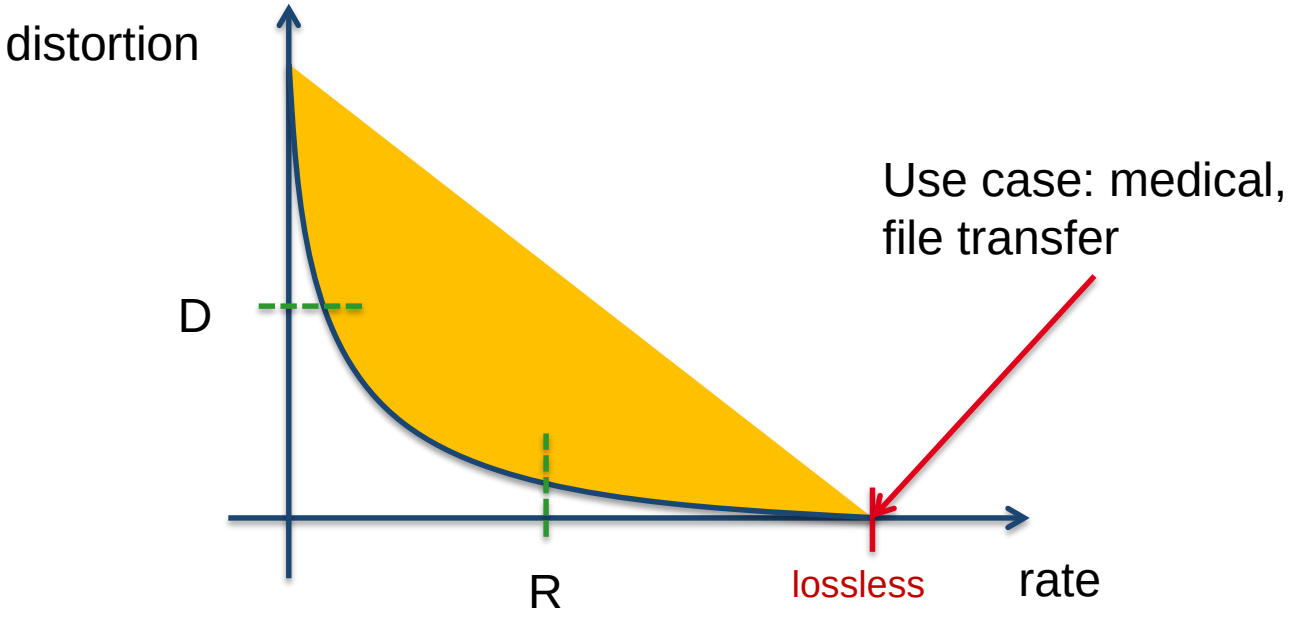
3

Lossy compression

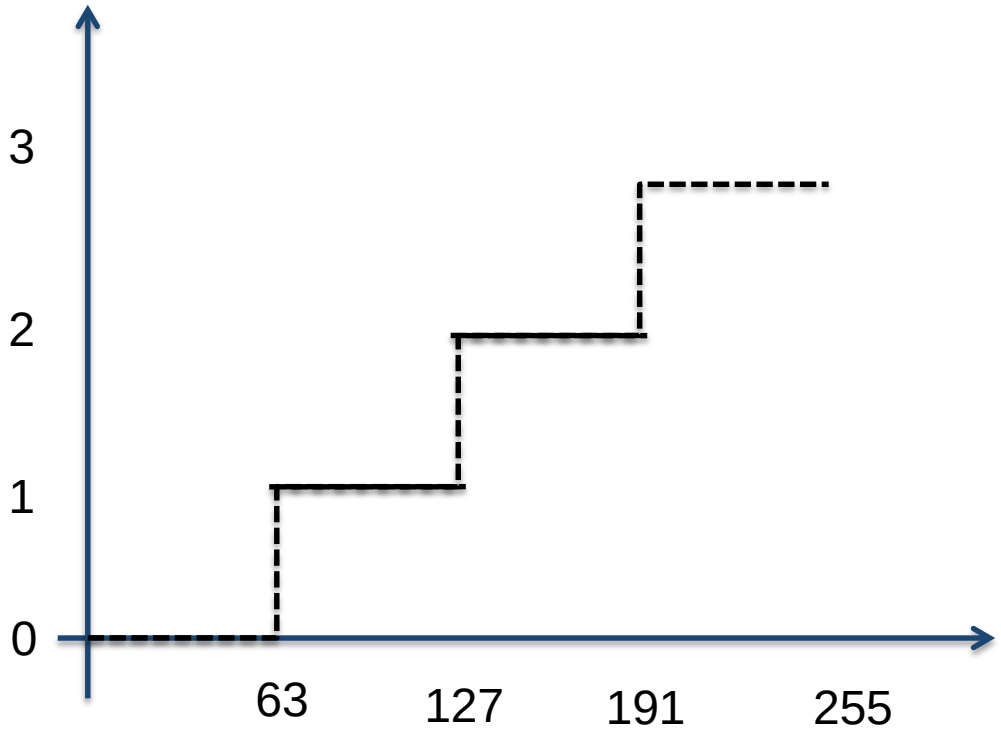
Compression with losses

Compression under a rate constraint

rate = nb bits / pixel (or letter)
distortion = distance between original and compressed image



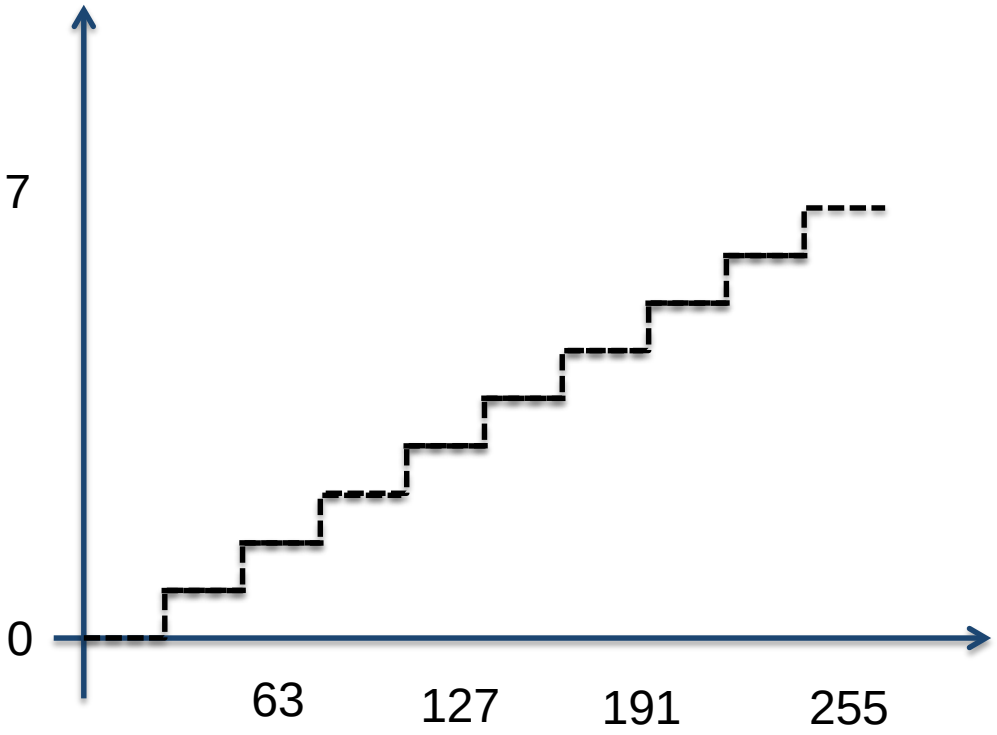
Implementing lossy compression



Uniform scalar quantization

8 bits → 2 bits

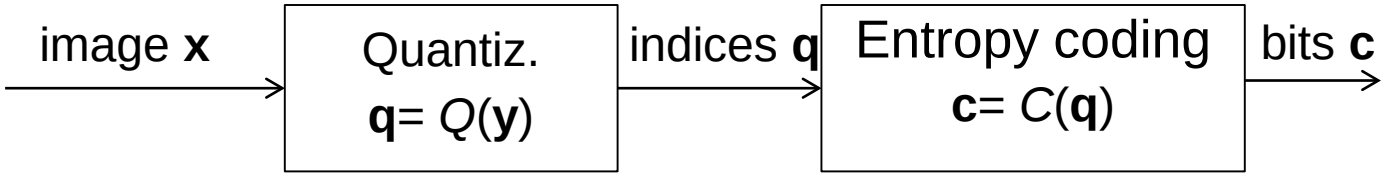
Implementing lossy compression

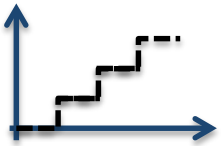


Uniform scalar quantization

8 bits → 3 bits

Implementing efficiently lossy compression





A	B	C	D
1/4	1/2	1/8	1/8
10	0	110	111

Uniform scalar quantization + entropy coding

→ degrades by **1.53 dB** with respect to optimal

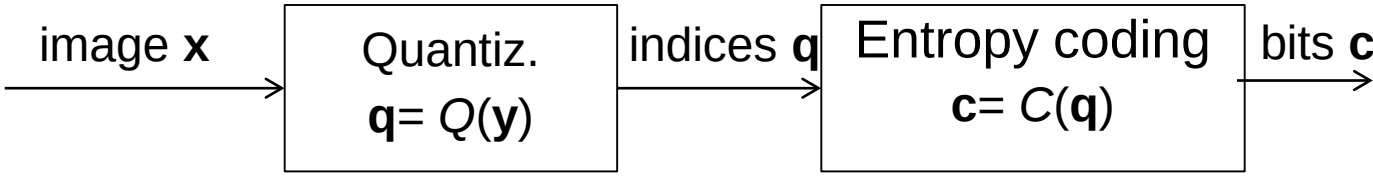
1. Why a need to compress video?
2. How-to compress (lossless)?
3. Lossy compression
4. **Transform-based compression**
5. Prediction-based compression

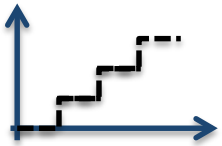
4

Dependent data

transform coding

Where do we address dependency?



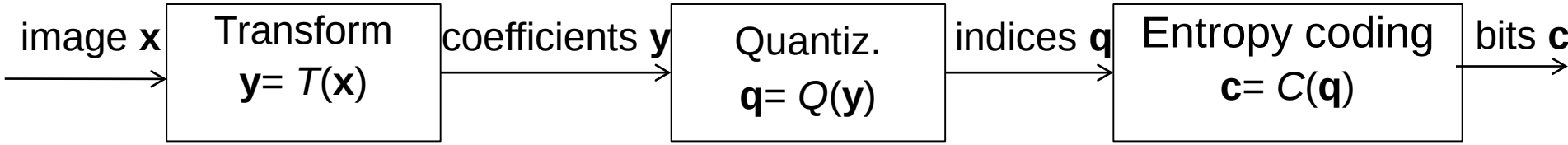


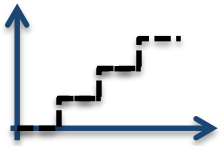
A	B	C	D
1/4	1/2	1/8	1/8
10	0	110	111

Scheme is quasi-optimal if pixels were independent

.....

Where do we address dependency?



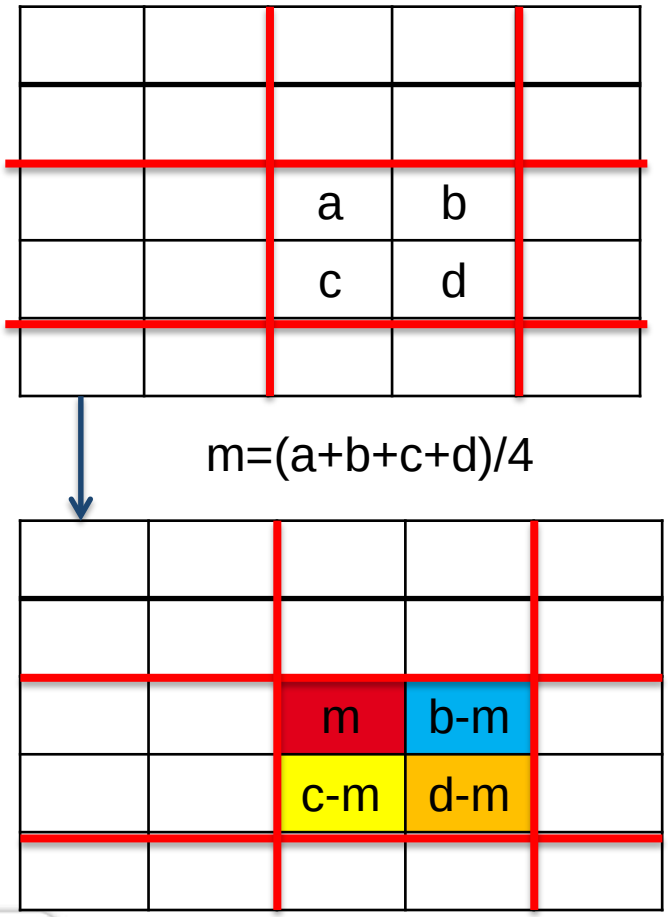


A	B	C	D
1/4	1/2	1/8	1/8
10	0	110	111

Scheme is quasi-optimal if pixels were independent

..... transform the pixels into independent data…

Exploit data dependency



(a,b,c,d) -> (m, b-m, c-m, d-m): invertible transform

Dependence =

Neighboring pixels have similar value

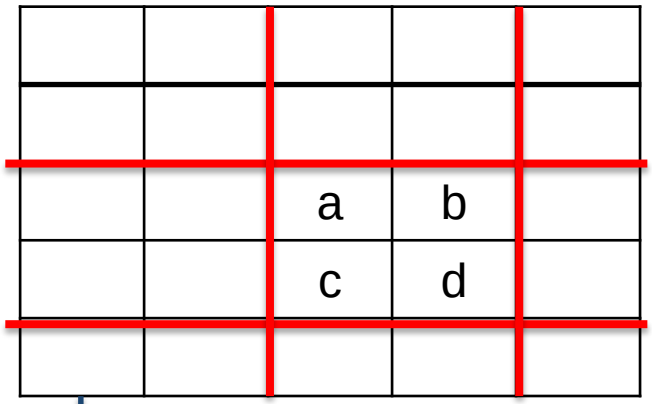
Mean difference:

- small amplitude
- break dependence within a block?

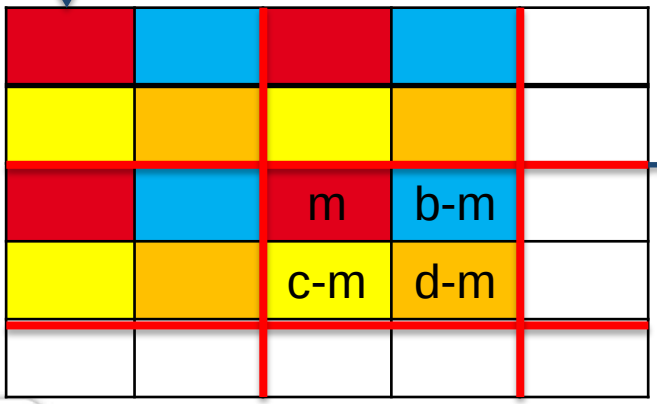
Exploit data dependency

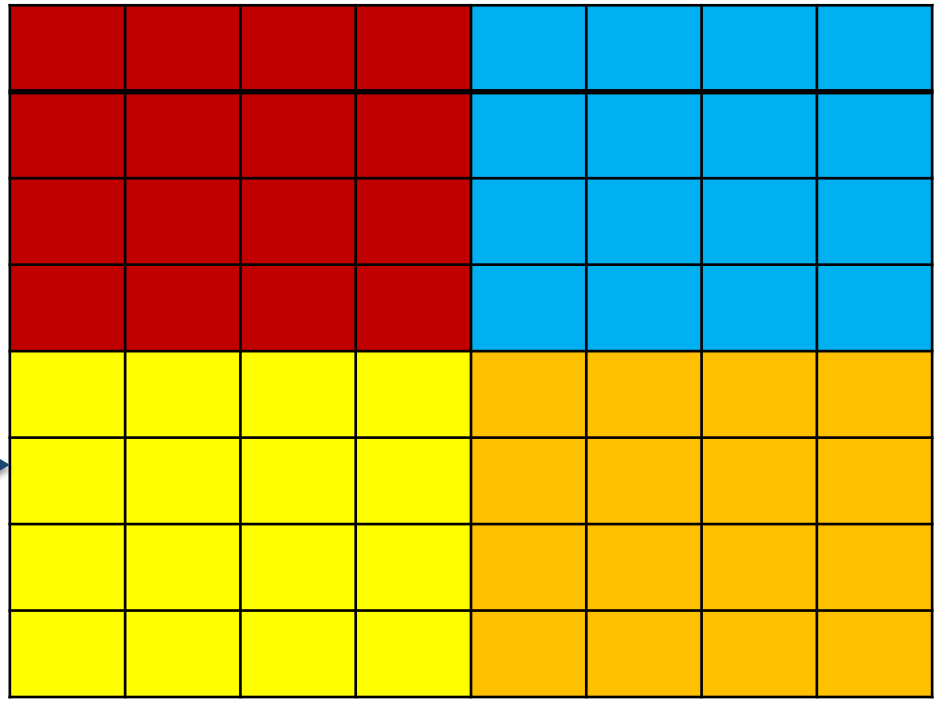
This transform is a particular case:

- discrete cosine transform (DCT) (MPEG 1, 2, 4, HEVC)
- wavelets (JPEG-2000, Dirac)



m=(a+b+c+d)/4





Original Lena (512 x 512 Pixels, 8-Bit Y/ pixel, 257Ko)

Gray = 0 since m-d coded by m-d+128

Dependences remain

Gray = 0

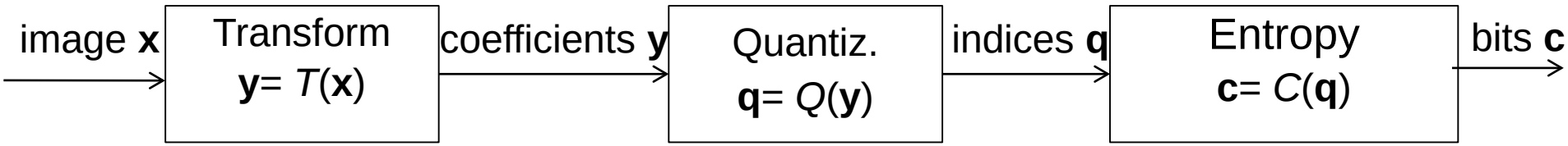
Gray = 0

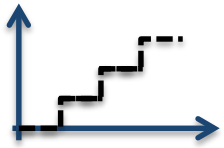
Wavelet example

Transform used in JPEG 2000

Less dependence… but still some remains

JPEG2000



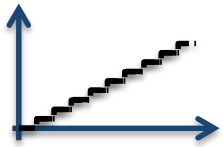


A	B	C	D
1/4	1/2	1/8	1/8
10	0	110	111

Compression results JPEG2000

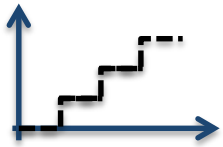
Original Carol (512 x 512 Pixels, 24-Bit RGB/pixel, Size 786kbytes)

Compression results JPEG2000



75:1, 10.6 kbytes

Compression results JPEG2000



150:1, 5.3 kbytes

Compression results JPEG2000

300:1, 2.6 kbytes

Comparison JPEG (1992) vs. JPEG2000

Lena, 256x256 RGB

Baseline JPEG: 4572 bytes

Lena, 256x256 RGB

JPEG-2000: 4572 bytes

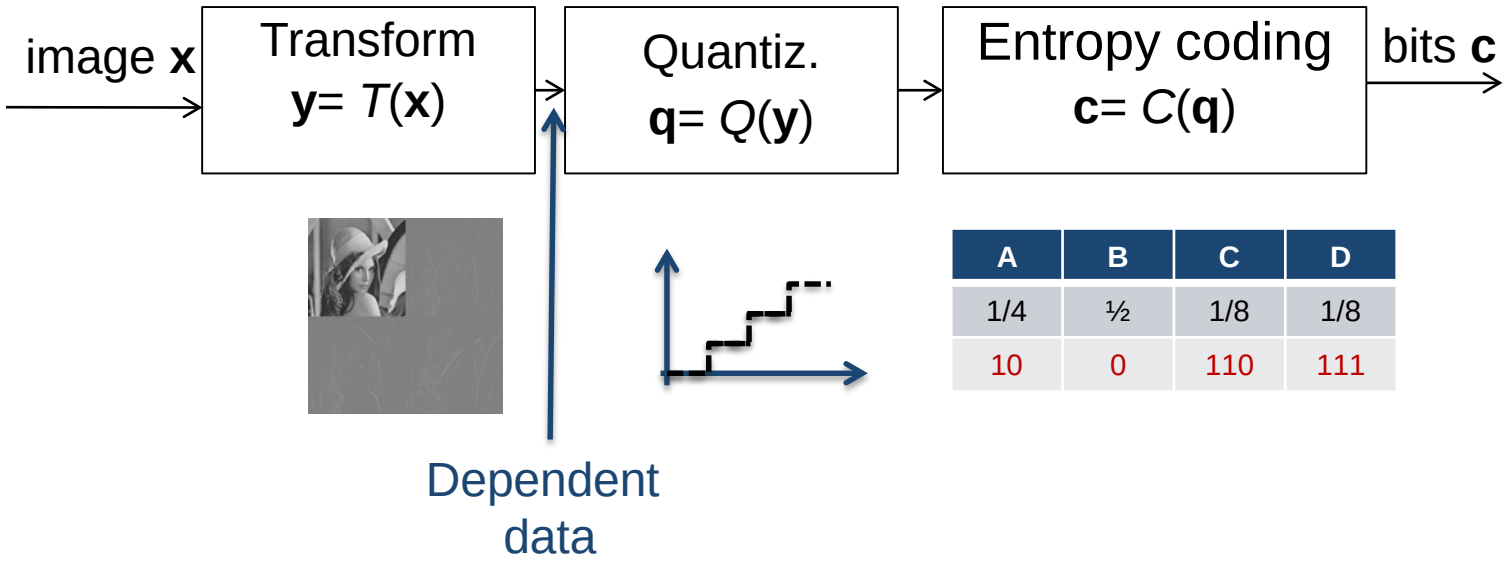
1. Why a need to compress video?
2. How-to compress (lossless)?
3. Lossy compression
4. Transform-based compression
5. **Prediction-based compression**

5

Dependent data

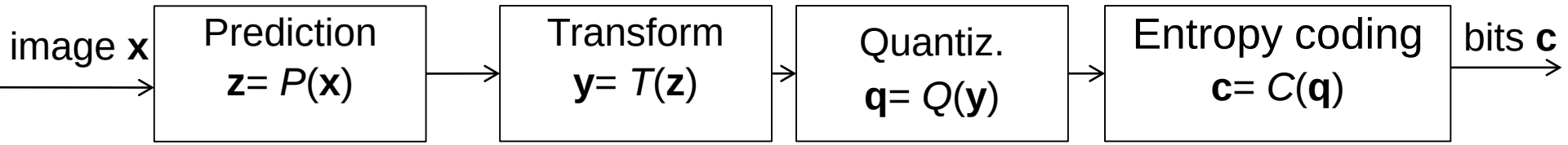
prediction

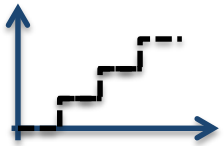
Dependency remains



A	B	C	D
1/4	½	1/8	1/8
10	0	110	111

Dependency remains

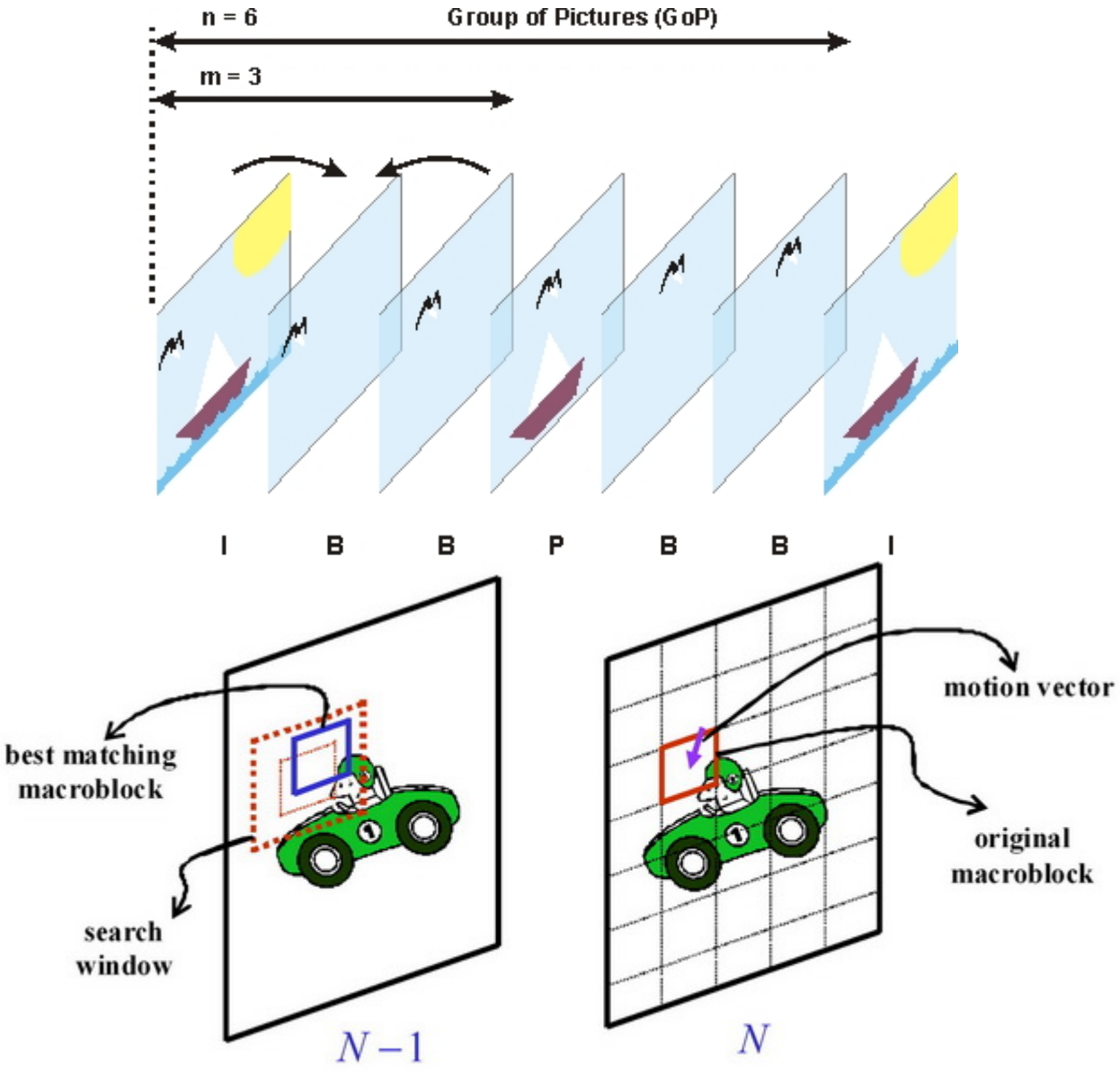




A	B	C	D
1/4	½	1/8	1/8
10	0	110	111

Prediction: further/better suppress dependencies

Prediction… between images

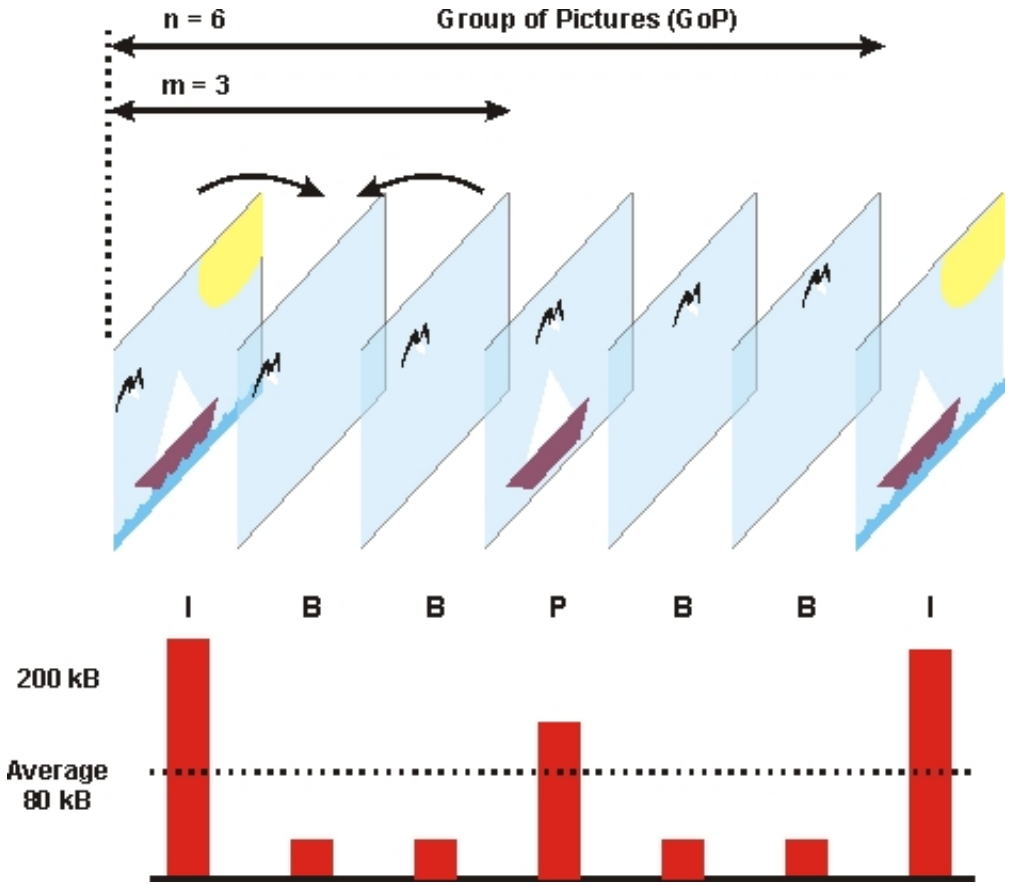


Intra: image coded independently of other images (synchro)

Inter: (P/B) image expressed in terms of other images

Used in MPEG-2, -4, HEVC

Prediction… between images: inter prediction



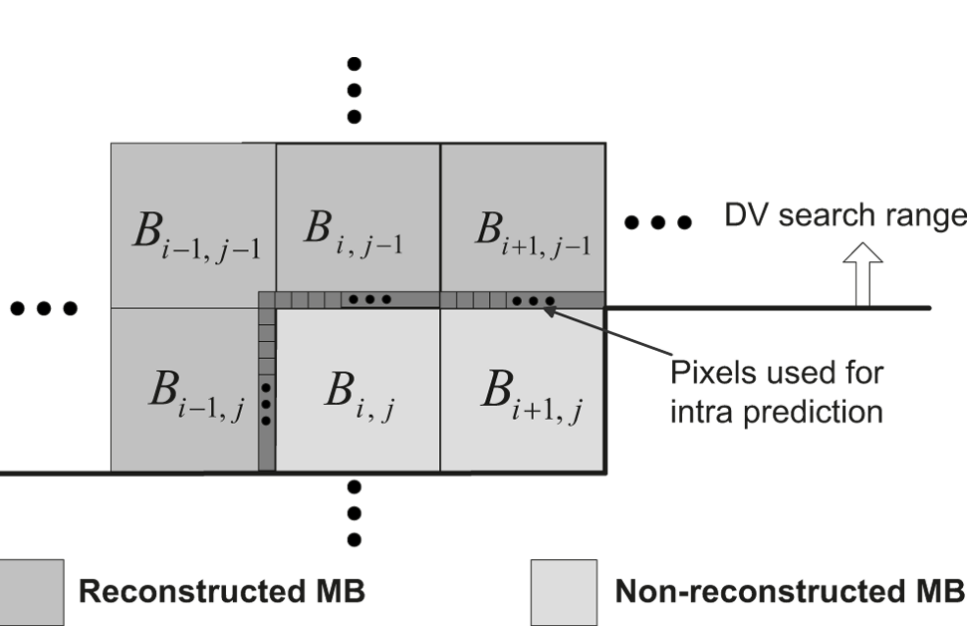
Intra: image codée indépendamment des autres (synchro)

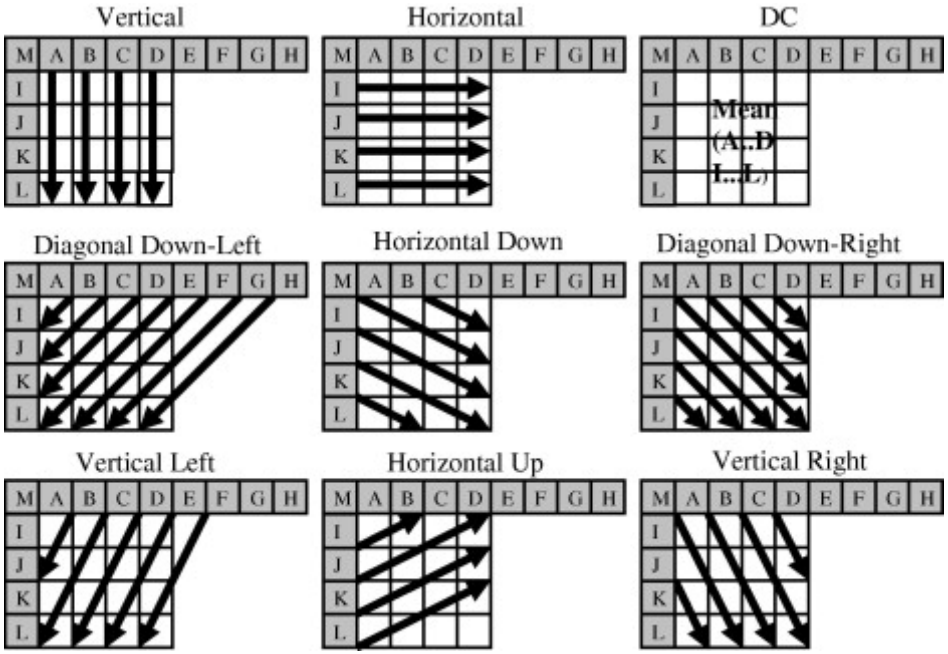
Inter: (P/B) image codée en fonction d'autres images

Utilisé en MPEG-2, -4, HEVC

Very efficient: rate (inter) << rate (intra)

Intra prediction (MPEG-4, HEVC)





If efficient prediction, difference between original and prediction (residue) :

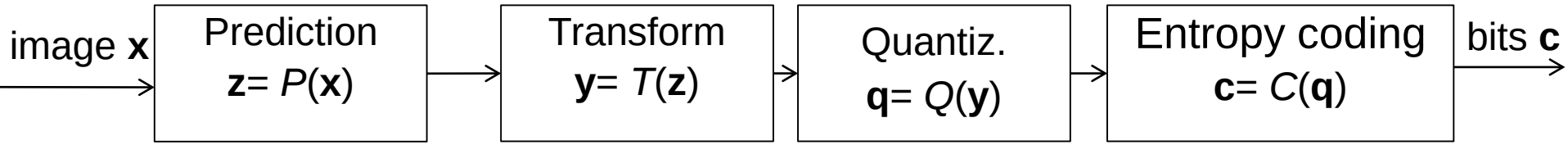
- independent samples [estimation theory]

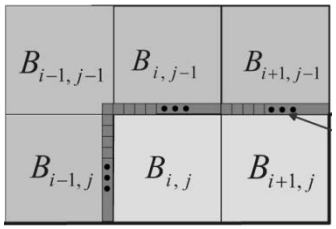
6

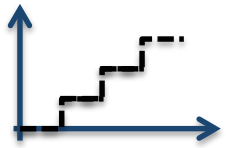
Conclusion

Take Home Messages

Take-home-message







A	B	C	D
1/4	½	1/8	1/8
10	0	110	111

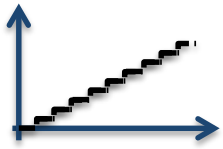
Transform and prediction: exploit **pixel dependency**

Quantization: introduce **losses**

Entropy coding: exploit **non uniform** pixel distribution

Results with HEVC

Vidéo BasketBallPass 416x240 pixels 50Hz (8 + (8+8)/4 bits) 60 Mbps



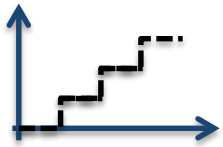
Quantization step	Rate	Compression ratio
uncompressed	**60 Mbps**	
22	**1 Mbps**	**60:1**
32	340 kbps	176:1
51	30 kbps	2000:1

original

60:1

Results with HEVC

Vidéo BasketBallPass 416x240 pixels 50Hz (8 + (8+8)/4 bits) 60 Mbps



Quantization step	Rate	Compression ratio
uncompressed	**60 Mbps**	
22	1 Mbps	60:1
32	**340 kbps**	**176:1**
51	30 kbps	2000:1

original

176:1

Results with HEVC

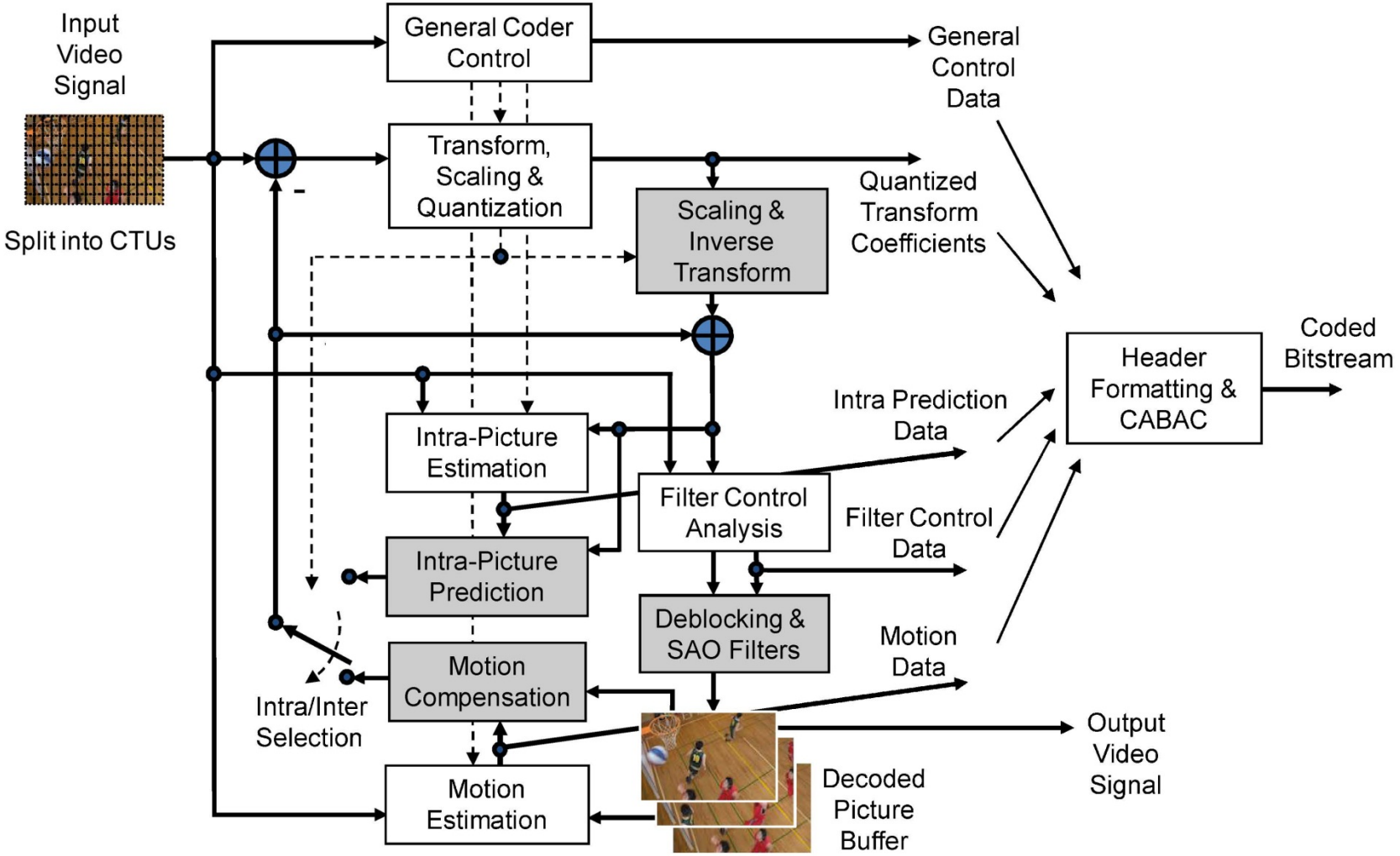
Vidéo BasketBallPass 416x240 pixels 50Hz (8 + (8+8)/4 bits) 60 Mbps

Quantization step	Rate	Compression ratio
uncompressed	**60 Mbps**	
22	1 Mbps	60:1
32	340 kbps	176:1
51	**30 kbps**	**2000:1**

original

2000:1

Encoding scheme HEVC



Elements in gray correspond to the decoder (*Overview of HEVC Standard*, TCSVT Dec 2012)

Pour en savoir plus…

1. Thomas Wiegand and Heiko Schwarz:

   Source Coding: Part I of Fundamentals of Source and Video Coding,

   Foundations and Trends in Signal Processing, vol. 4, no. 1-2, pp. 1-222, January 2011.

   http://iphome.hhi.de/wiegand/pubs.htm

2. Ian Richardson

   The Advanced H 264 video compression standard, Wiley 2010.

3. Vivienne Sze, Madhukar Budagavi, Gary J. Sullivan

   High Efficiency Video Coding (HEVC), Springer 2014**.**

Thank you

www.inria.fr