Multimedia Databases
Coding and Compression

Prof. (FH) PD Dr. Mario Döller

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 Image: JPEG Coding

5 Video: MPEG 1 & 2

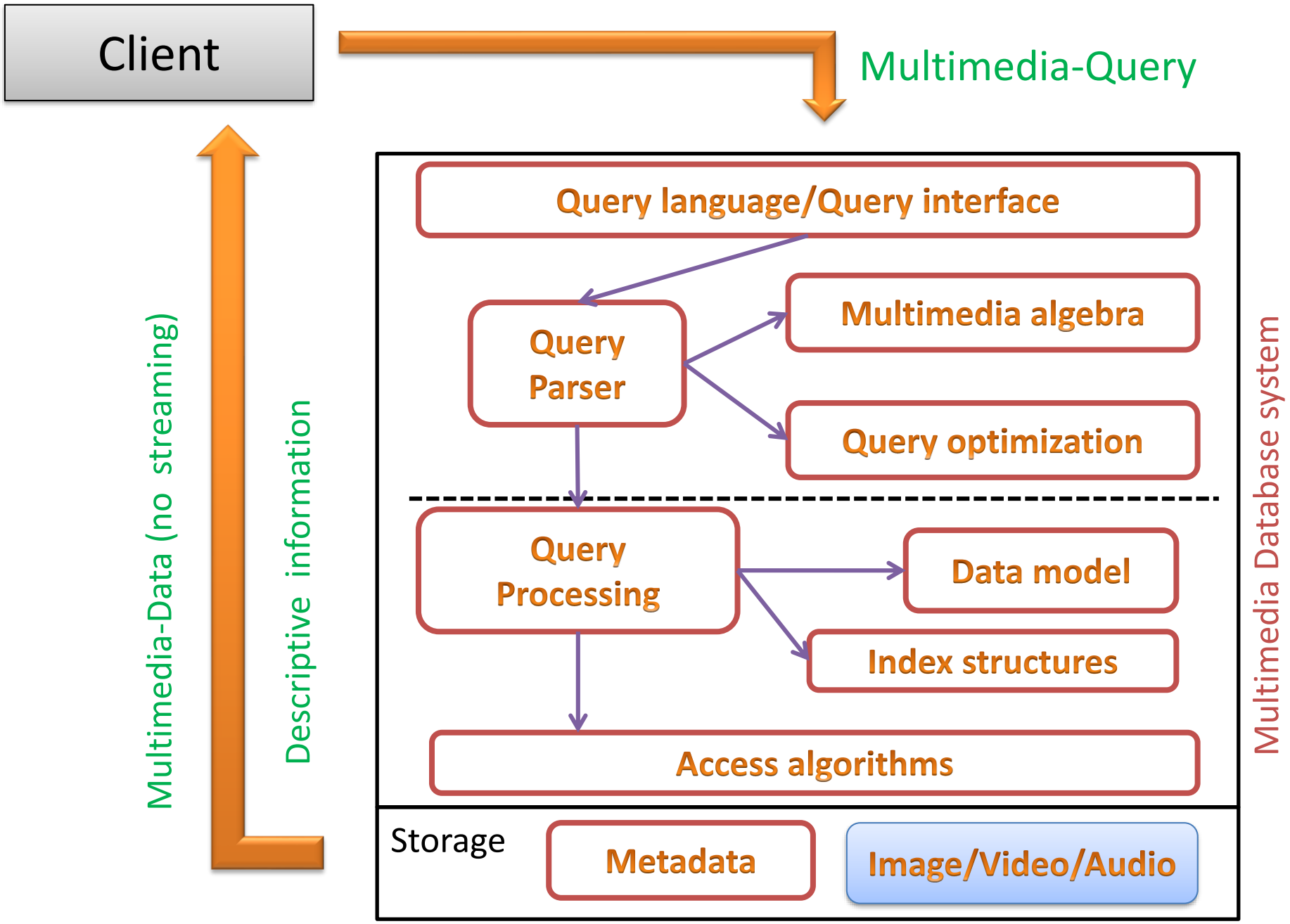

Client
Multimedia-Query
Multimedia-Data (no streaming)
Descriptive information
Query language/Query interface
Query Parser
Multimedia algebra
Query optimization
Query Processing
Data model
Index structures
Access algorithms
Storage
Metadata
Image/Video/Audio
Multimedia Database system

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2

Motivation

Media Data Size I

- **Storage space requirements by media type:**
  - (With ISO units: 1kB=10^3 Bytes, 1 MB=10^6 Bytes,…)
  - **Text**
    Assumption: 80 x 60 text, 2 B / character.
    Space requirement: 4800*2 = **9,6 kB**
  - **Vector Image**
    Assumption: a typical image contains 500 segments,
    Coordinate in x-Direction : 10 Bits,
    Coordinate in y-Direction : 9 Bits,
    Attribute vector / segment: 8 Bits.
    Bit pro line: (9+10+9+10+8) Bits = 46 Bits
    Space requirement: 500*46/8 B = **2,875 kB**

Motivation

Media Data Size II

- **Pixel image**
    - Assumption: HD ready 1080p: 1080 * 1920, True color = 3 B / Pixel
      Space requirement: 1080*1920*3 Bytes = **6,22 MB**
- **Video sequence**
    - Assumption: 60 Frames per second
      Data rate: 6,22 * 60 * 8 ≈ 3 Gbit/s
      Space requirement: **373 MB (1 second!)**

„Important" size reduction possible through compression

Motivation

Overview of size and ratio by media type

Medium	**Data amount**	**Compression ratio**
Text	80 x 60 chars 9.6 kB	Huffman 1:2 gzip 1:3
Image	HD ready 1080p 1080x1920 3 B/Pixel 6,22 MB	JPEG 1:15 (~lossless) 1:35 (lossy w. good quality)
Vector graphics	500 segments 2,9 kB	gzip 1:2 <-> 1:5
Video	HD ready 1080p à 60 Fps 373 MB/s	MPEG2 1:60 H264/AVC 1:120 HEVC 1:240

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

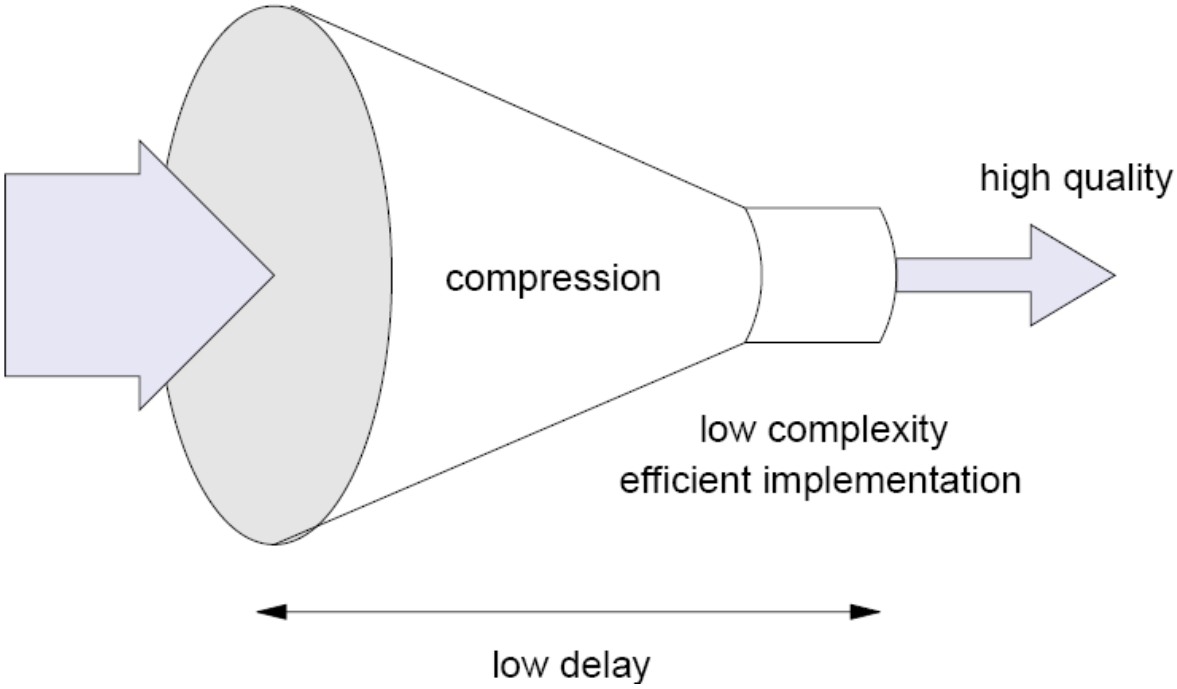
- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2

Data Compression

Criteria



- Good compression rate
- Good quality (lossy methods)
- Low computing and storage space requirements
- Low latency (fast)

Symmetrical / Asymmetrical Compression

Symmetrical Compression:

Symmetrical in execution time for compression and decompression

- e.g. for communicating systems (image transmission, videoconferencing, ..) ~: End-to-end delay ≤ 150 ms

Asymmetrical Compression:

Compression computationally intensive - Decompression fast

- e.g. multimedia data (non-live) distribution: data compressed once, decompressed many times, possibly in real-time (e.g. DVI)

Criteria for (Video) Compression II

- Symmetrical and asymmetrical methods - common requirements:
  - Independence of resolution and framerate
  - Possibility of different data rates for audio/video
  - Audio/Video must be synchronizable, also with other media objects
  - Low cost, software rather than hardware
  - Interoperability

    Use of standards: de jure - de facto

Criteria for (Video) Compression III

- Specific requirements for asymmetrical methods:
  - fast forward/backward when displaying the data
  - Random access to individual images ≤ 0.5 sec
  - „Direct“ decompression of selected images or sequences without having to access „previous“ data.

    Possibility of editing after random access.

Classification

Data Coding

Direct Coding		PCM
Entropy Encoding (lossless)	Reduction of repeating sequences	Reduction of nulls
		Run-length encoding
	Statistical coding	Pattern replacement
		Huffman coding
Source Coding	Transformation	FFT
		DCT
		Others
	Prediction (Difference coding)	DPCM
		Delta modulation
		ADPCM
Channel coding		HDLC, …..

Classification

Entropie vs. Source Coding

- Entropy Coding:
  - Properties of the data ignored
  - Redundancy in the signal gets reduced/deleted
  - Statistical methods, e.g. Huffman-Coding
  - Lossless and reversible
  - Low compression factors (e.g. ca. 2 for audio data)

- Source Coding (source encoding):
  - Consider the specificities of the data source and recipient
  - e.g. Frequency spectrum / masking / noise thresholds of the human ear
  - Lossless or lossy for better compression (e.g. MP3 up to ca. 10:1)

Classification
Lossless Compression

- Classification of the techniques
  - Static
  - Adaptive (Dynamic)
  - Hybrid

- Static methods are Two-Pass:
  - First pass to determine probability distributions / frequencies
  - Second pass for coding
    - Ex: Static Huffman Coding

- Adaptive methods are One-Pass
  - Only one scan of the message for coding is required.
    - Ex: LZ77, LZ78

Codec

- Data compression:

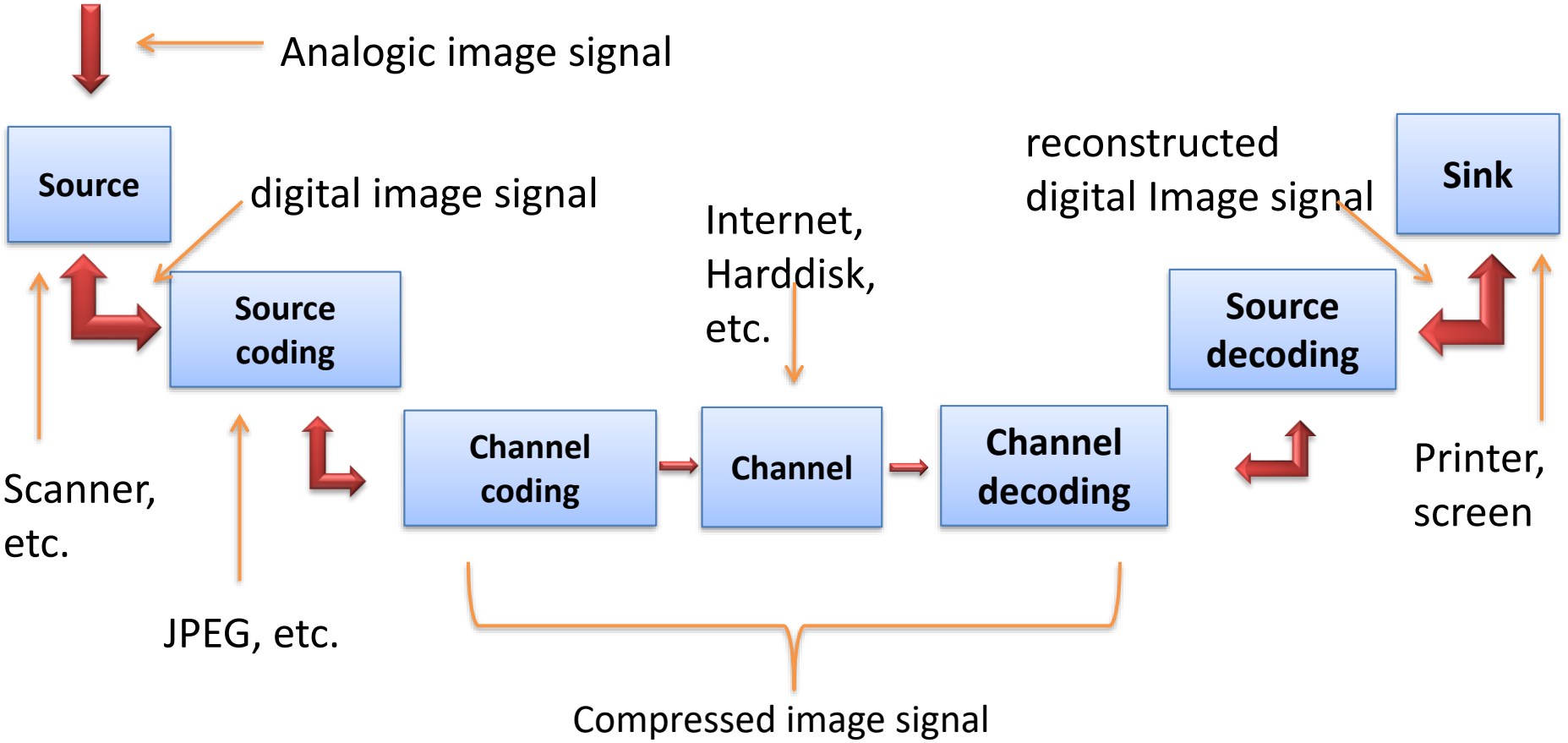
decode(encode(x)) = x

- Each method must offer a pair of algorithms, coder and decoder := **CODEC**

decode1(decode2(channel(encode2(encode1(x))))) = x

  - Outer / inner layers do not need knowledge of the (other) codec
  - Further nesting possible

Digital Image Transmission steps



Coding

- Essential step of data compression for Audio and Video (here shown for a single image):

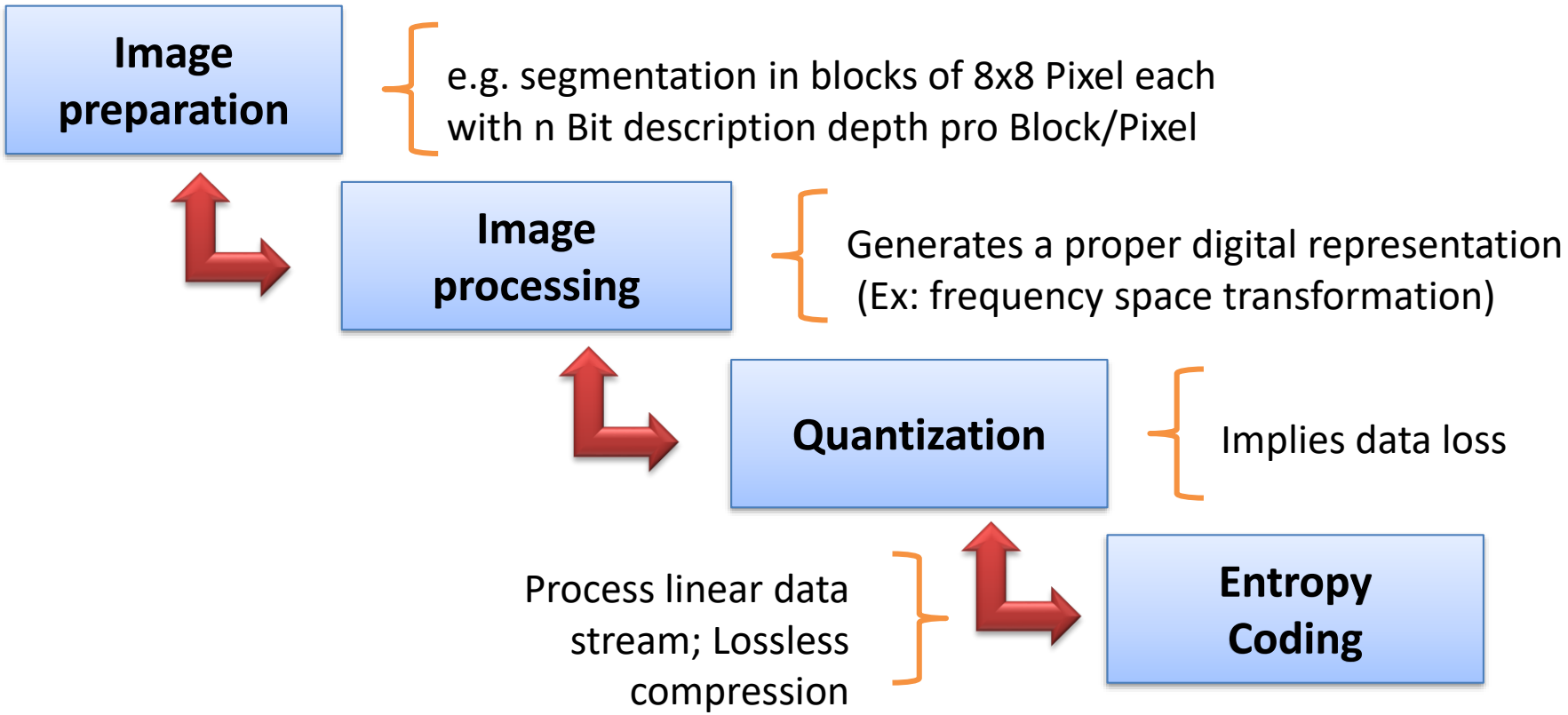


Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2

Basic Methods

Run-length Encoding

- Lossless compression method
- Suitable for data with long sequences of identical characters (bytes)
  - Ex: Black and white images
- Replaces sequences of identical bytes by the byte code followed by sequence length
- Special character used to separate sequences

- Method
  - If a byte appears more often than x times in sequence (x: Offset, x=4 usual), compress as follows:
    - Sequence-Byte
    - Flag
    - Number of apparitions (-offset)

Basic Methods

Run-length Encoding: Example

- Assumption: Separation char='!', offset=4
- Example text:

- RLE: ABC!6DEFG!0H

- Compression rate: 4 - 259 Bytes in 3 Bytes

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2

Basic Methods

Statistical Coding

- Symbols may be coded by sequences of varying length.
- Frequent symbols -> short code, rare symbols -> longer code.
- Important: No ambiguity in decoding allowed.
  - Ex. Code book 1<->A, 11<->B
    Coded data 111 = AAA, AB, BA?
- Example methods: Huffman Coding, arithmetic coding.

Basic Methods

Huffman Coding

- Given: List of symbols with probabilities/(relative) frequencies
- Idea: Creation of an optimal code-tree in two steps
  - 1. sort symbols by frequency
  - 2. create tree „bottom-up“ by repeatedly merging the 2 least frequent nodes
- Ex: Text = AABAACDAAEABACD

Basic Methods

Huffman Code Example

- Example: Text = AABAACDAAEABACD

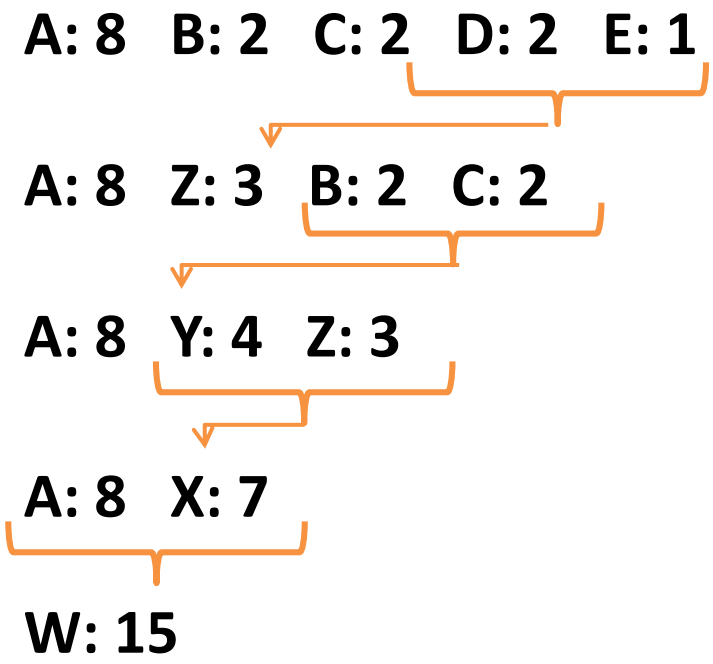
(1) Computation of frequencies and sorting

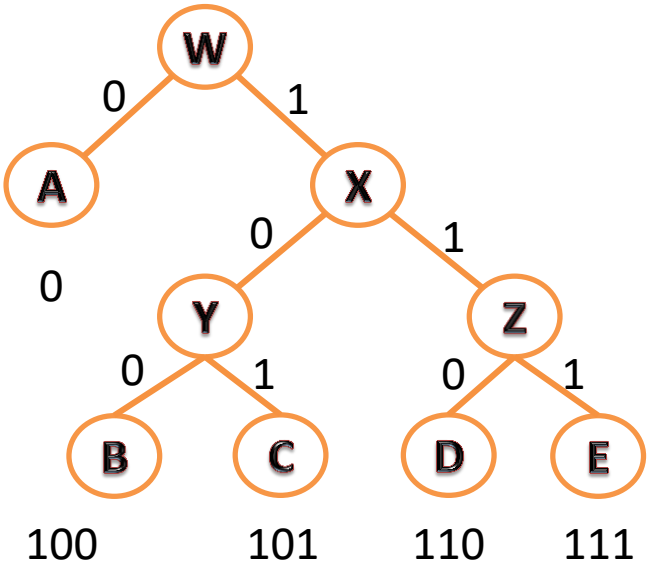
Symbols	Frequency	Probability
A	8	8/15
B	2	2/15
C	2	2/15
D	2	2/15
E	1	1/15

Basic Methods

Huffman Code Example

(2) Creation of the tree





AABAACDAAEABACD
→ 00100001011100011101000101110
= 29 bits

Basic Methods

Huffman Coding Algorithm

1. Creation of code tree
   - Computation of frequencies by going through the text
   - Heap for the nodes, sorted by frequency
   - Binary trie for the code tree

2. Creation of code table
   - Traversal of the tree

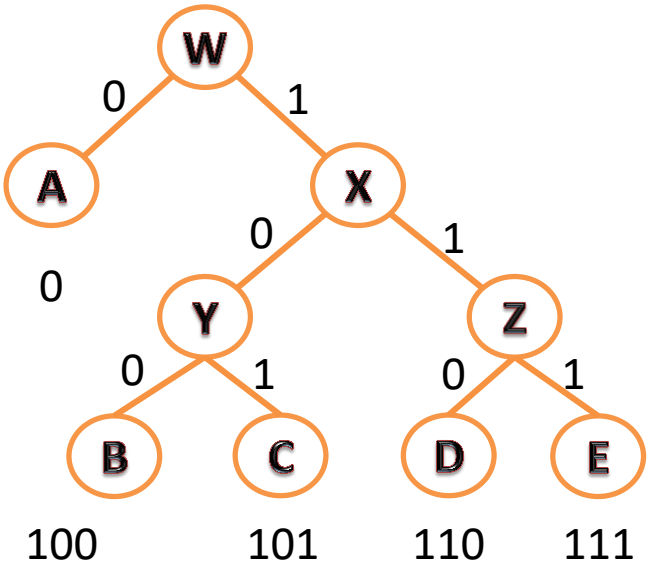
3. Coding the text
   - Look-up of the code by symbols in code table

4. Storage of the code tree with the coded text

Basic Methods

Huffman Codes Decoding

- Reconstruction of the code tree
  - binary trie
- Decoding of bit sequences
  - Start at the root of the tree
  - Go through the tree based on encountered bits
  - When reaching a leaf output the corresponding symbol and start again from the root



00100001011100011101000101110
→ AABAACDAAEABACD

Basic Methods
Code Book Coding

- Goal: Compact representation
  - Simplest option:
    - Pro sign k+n Bits, k constant
    - k Bits to represent the number n, n Bits to represent the code
    - Not efficient

- Better:
  - Traversal of the trie (e.g. in preorder)
  - Packets of symbols plus code-length (both with constant number of bits)

Basic Methods

Properties of the Huffman code

- In a sense (!) optimal, i.e. minimal code
  - In specific conditions such as: apparition of the symbols independent from one another
- No compression for random data
  - all symbols having similar frequencies
- Requires two text traversals
  - No possibility of stream processing
- Code book must be stored in addition to coded data
  - Constant overhead
  - Negligible for large data
  - Or use „Standard“-code book (cf. JPEG)

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2

Basic Methods

Transformation Coding

- Applies a transform to the data, expressing them into another mathematical space in which they can (hopefully) be encoded in a more efficient way e.g.:
  - Discrete Cosine Transformation DCT (see below JPEG)
  - Wavelets
  - Fourier transformation FFT

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2

JPEG-1 I

- "Joint Photographic Expert Group"
  - Joint activity of ISO/IEC JTC1/SC2/WG 1 and commission Q.16 of CCITT SGVIII (today ITU)
  - ISO norm (International Standard) since 1992
- Standardized Format for Raster Images
  - High compression rate possible (adjustable), but lossy
  - Also used for videos ("Motion-JPEG„), basis for MPEG video coding (more later)
- History:
  - JPEG 1, Motion-JPEG
  - Since 2001: JPEG 2000
  - Since 2010: JPEG XR
  - Since 2012: JPSearch

JPEG-1 II

- Four modes:
  - Baseline (Minimal requirement for decoder compliance)
    - 8x8 Discrete Cosine Transform (DCT)
    - Huffman-Coding (optional: optimized tables)
    - Sequential (blockwise) processing
  - Progressive mode (instead of sequential processing)
    - Progressive spectral refinement (from lower to higher frequency DCT-coefficients)
    - Progressive refinement of the exactitude of DCT-coefficients
  - Lossless mode (poorly supported)
    - DPCM-based (no use of the DCT)
  - Hierarchical mode
    - progressive wrt. image resolution

JPEG Compression: why?

Start

580 x 848 Pixel
24 Bit Color depth (RGB)
1,5 MBytes

Goal

580 x 848 Pixel
24 Bit Color depth
468 kBytes

580 x 848 Pixel
24 Bit Color depth
26 kBytes

JPEG slides: Source: Karl Jonas, NEC

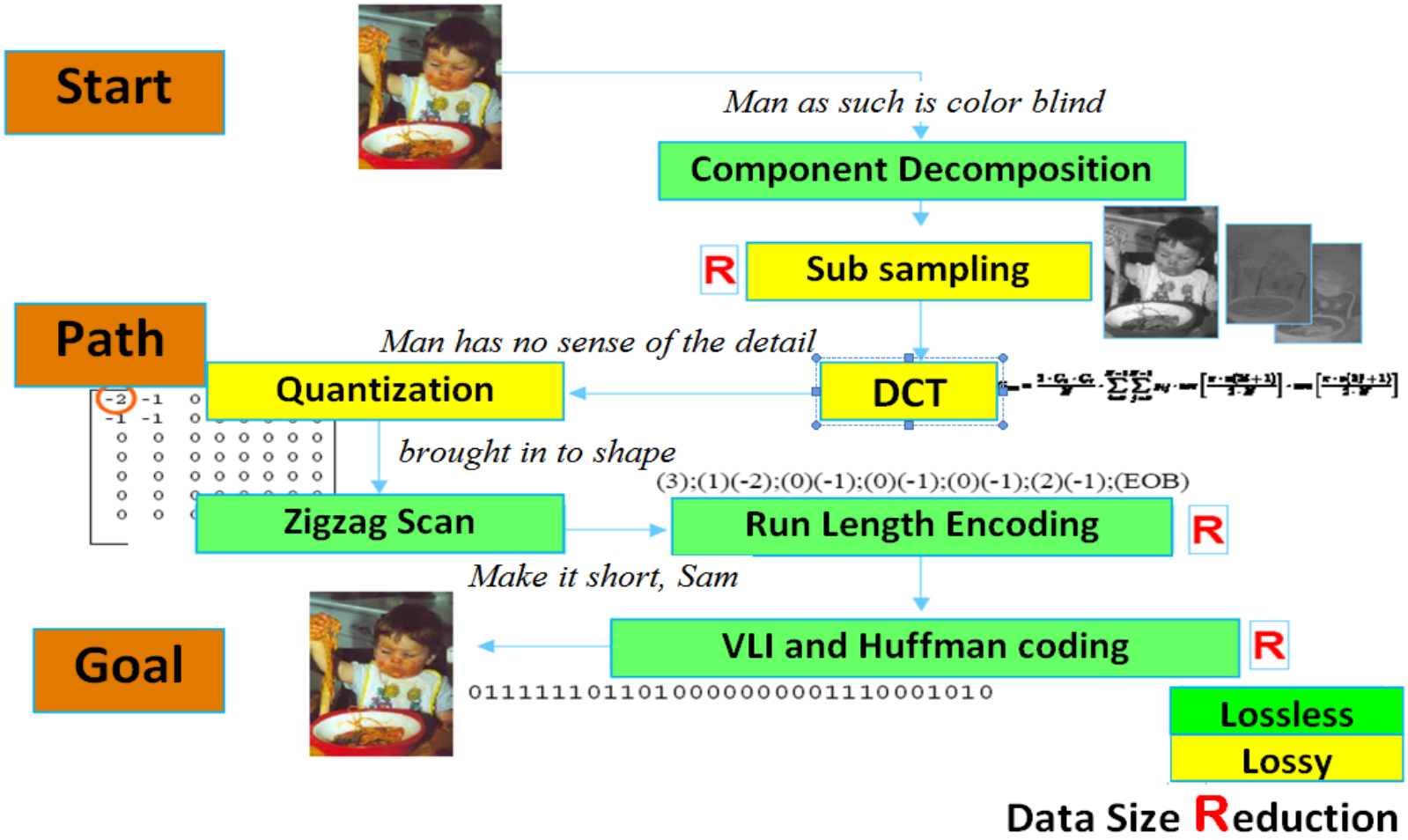
Principles of JPEG Baseline

- Identify the image information that is less important for human vision
  - Use of the Discrete Cosine Transform
- Process the less important info
  - Discard them or represent them with lower precision

  ⇛ Lossy compression
- The resulting data have higher redundancy
  - Apply entropy coding

  ⇛ Lossless compression

JPEG Compression: Process



JPEG Compression: Color Space Conversion

- RGB Color space
  - 3 values describe a color [R,G,B]
  - Hardware oriented model
- YCbCr Color space
  - Y is the luminance, Cb and Cr are the chrominance values of the color space
  - Separates intensity and color information
  - Perception oriented model

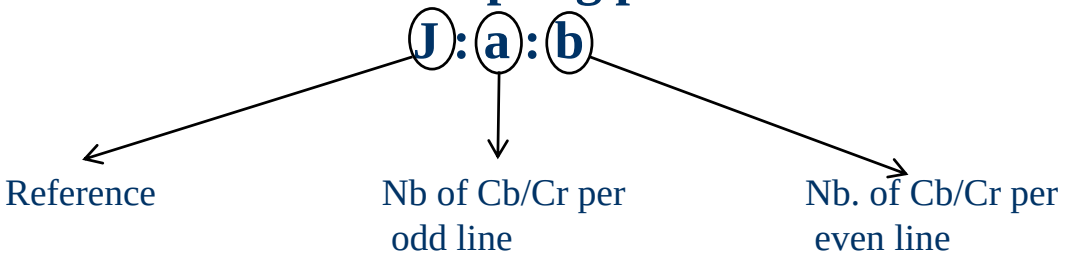
Conversion:

- Y = 0.30R +0.59 G+ 0.11B
- Cr = R-Y
- Cb = B-Y

JPEG Compression: Subsampling

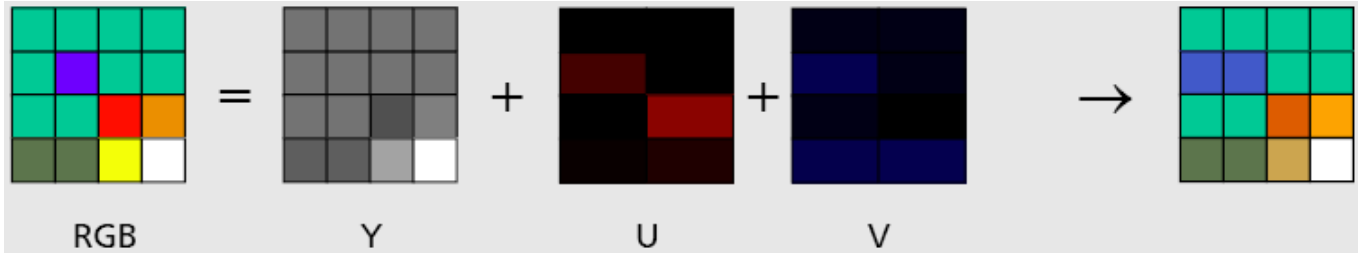
- Subsampling keeps all luminance coeffs but only 50% or 25% of chrominance coeffs.
- Data reduction with low quality loss

Subsampling process:



Sampling ratio:

Effect:

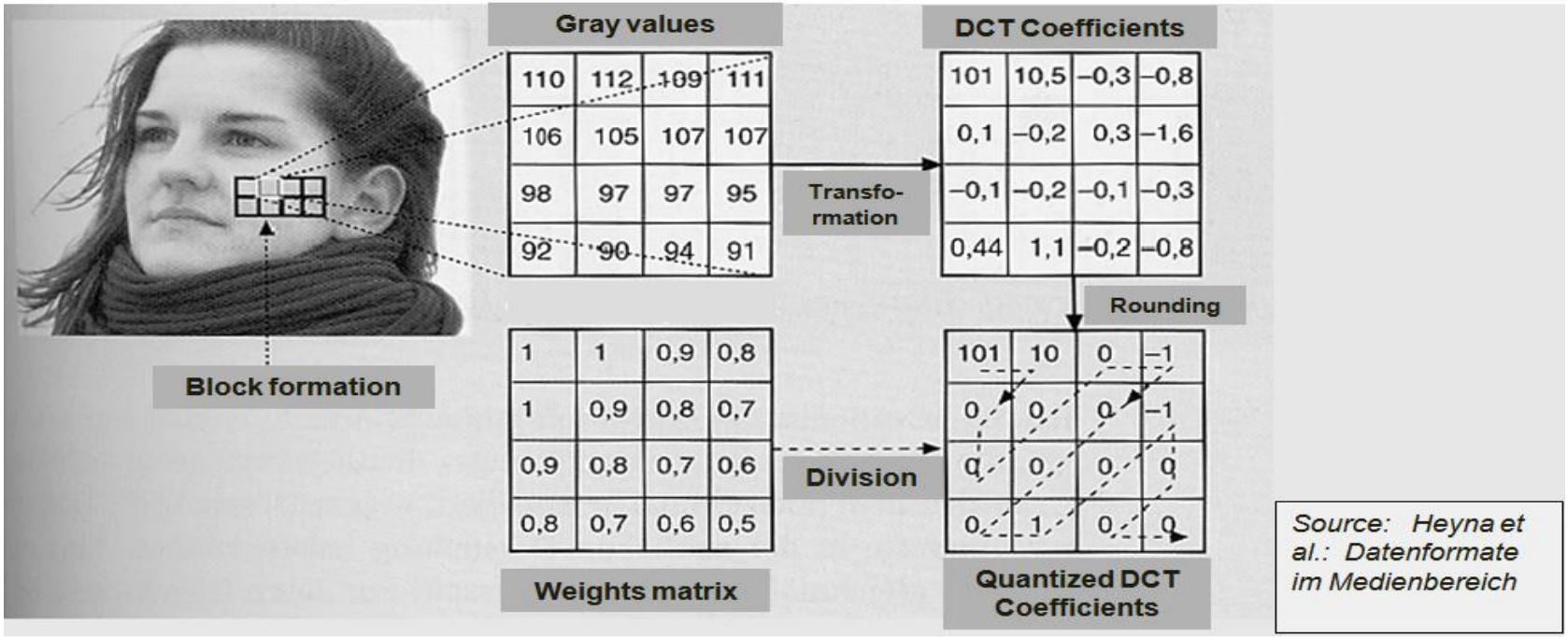


JPEG Compression: DCT

- Application of the Discrete Cosine Transform
  - Applied to blocks of 8x8 pixel, for each component Y, Cb, Cr
  - The DCT is theoretically reversible (but information loss due to rounding errors).
  - Transformation from (2D) spatial space (x,y) to (2D) frequency space (u,v)
    - Similar to Fourier-Transform (Time -> Frequency)
    - Result: DCT-coefficients
      - S_{00}: DC-coefficient = average of pixel values of the block
      - $S_{u,v}$: AC-coefficients represent the "strength" of a given frequency in the block; frequencies increase when going towards coefficient S_{77}
      - High frequencies = details of the image
      - Low frequencies = main structures of the image

JPEG Compression: DCT

- Image example: DCT + Quantization



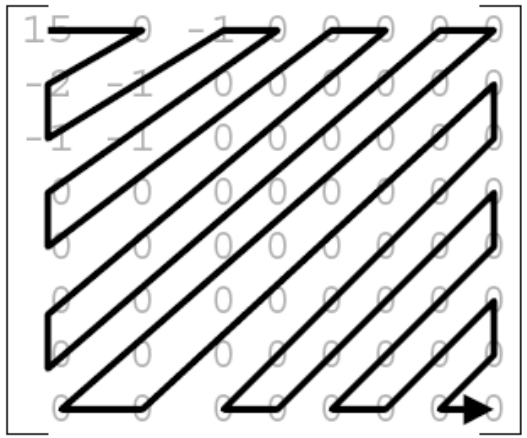
Quantization:

- Coefficients of high frequencies are quantized more strongly
- Values are rounded

JPEG Compression: Zig-Zag Scan

- Zig-Zag Scan

$$\begin{bmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Resultat after scanning:

15 0 -2 -1 -1 -1 0 0 -1 0 0 0 0 0 0 0 ... 0

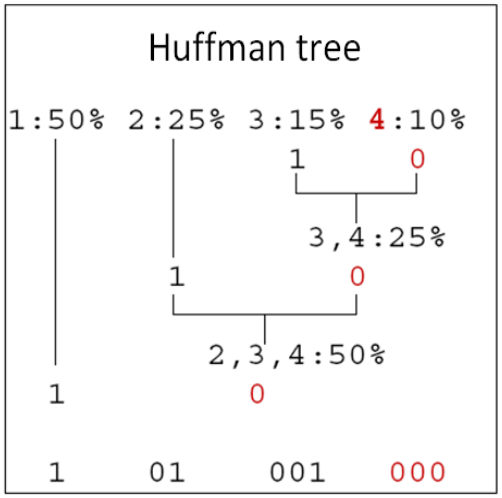
JPEG Compression: Entropy Coding

Difference Coding: 10 12 15 18 12 → 10 2 3 3 -6
(for DC-coefficients)

Run-length Encoding: 0 -2 -1 -1 -1 0 0 -1 0..0 →1 -2 0 -1 0 -1 0 -1 2 -1 EOB
(for AC coefficients)

Not a „true“ RLE, only considers sequences of „0“.

Huffman Coding:



Output values (range of values -4...-1, 1...4)
11211441322113312221 1

Relative frequency
1: 50% 2: 25% 3: 15% 4: 10%

Binary Coding

1	1	2	1	1	4
0001	0001	0010	0001	0001	0100

Huffman coding

1	1	2	1	1	4
1	1	01	1	1	000

JPEG Compression: Example of data change

Original Data :

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Decoded Data :

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

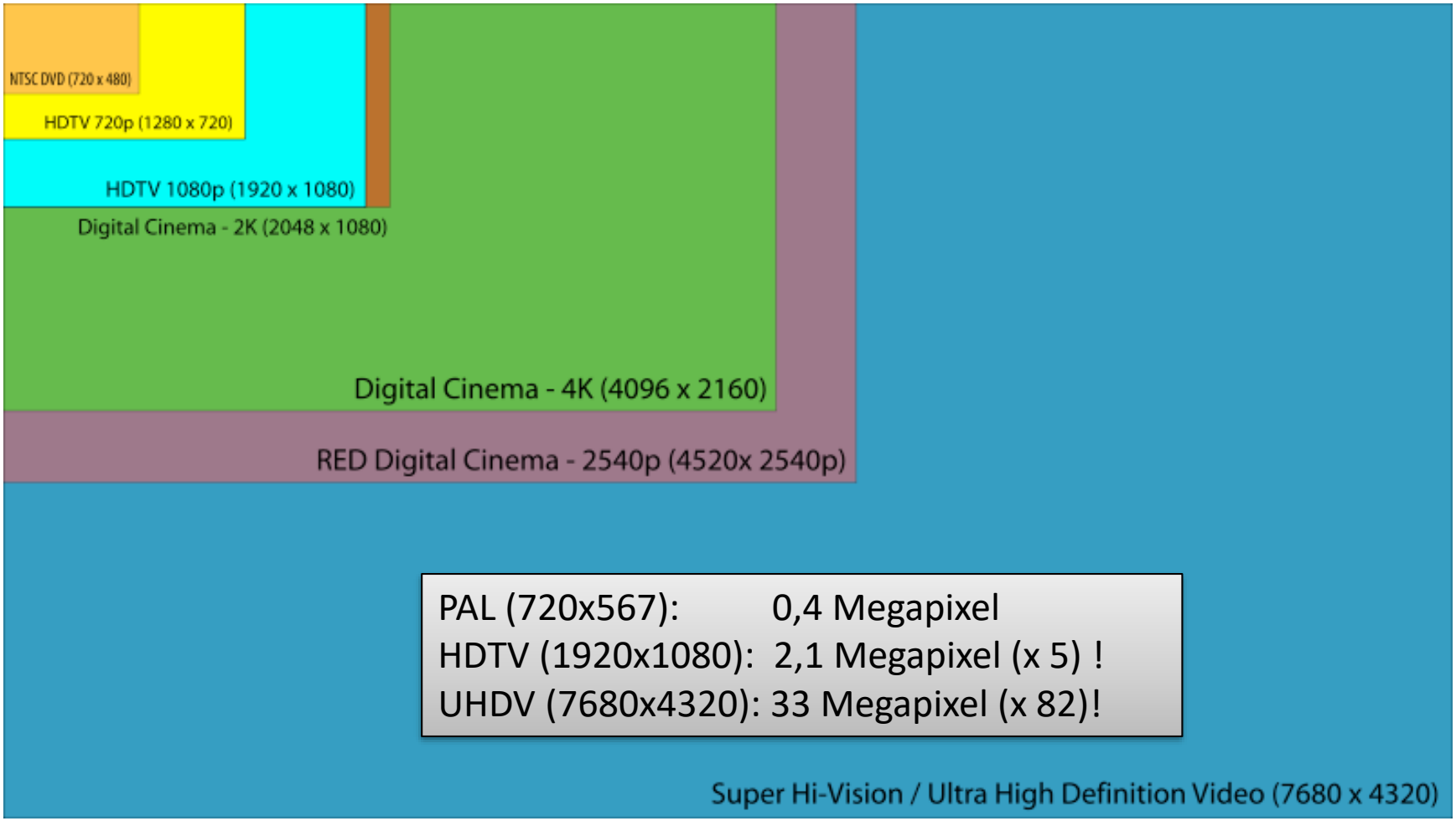
- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2 (slides partly © Klaus Schöffmann, University Klagenfurt)

- 5.1 Overview and Motion compensation
- 5.2 MPEG-4
- 5.3 H.264
- 5.4 Further Video Codecs

Space consumption of raw video data



PAL (720x567): 0,4 Megapixel
HDTV (1920x1080): 2,1 Megapixel (x 5) !
UHDV (7680x4320): 33 Megapixel (x 82)!

Video Compression

- Simplest form
  - Every image saved as JPEG
  - MJPEG = **Motion JPEG**
  - Advantage
    - Fast
    - Simple
    - Video can be startet at any frame position
    - Transmission error are not propagated
  - Disadvantage
    - Only reduced compression rate

Video Compression

- Two classes of compression
  - **Intra** Coding
    - **Spatial** redundances are reduced
    - E.g. JPEG
  - **Inter** Coding
    - **Temporal** redundances are reduced
    - Main idea
      - One image is used as basis (reference image)
      - Only difference to the reference image is saved (*Residual*)
      - E.g. as JPEGs

Temporal redundances

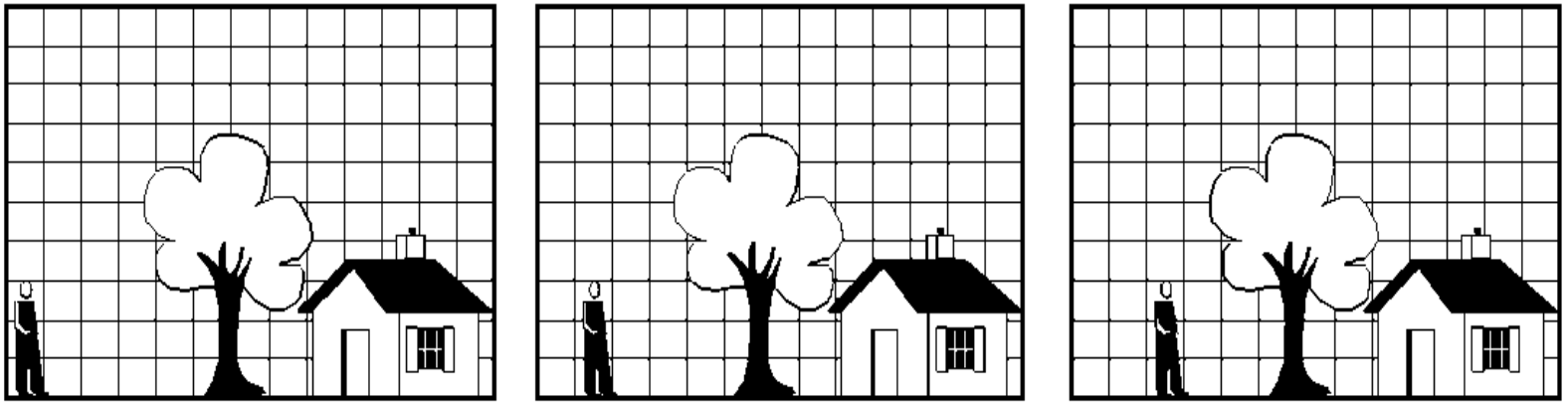
- How do calculate similarities of frames?
- Simple Approach:
  - Current Frame is substracted from previous one
    - Pixel by pixel
  - Only difference values are saved
  - Disadvantage
    - Motion is not taken into consideration!
    - Fast pan shots result in situation that all pixel are different
  - → Motion model is used
    - „*Motion Estimation*“

Residual

= difference between original and compressed image

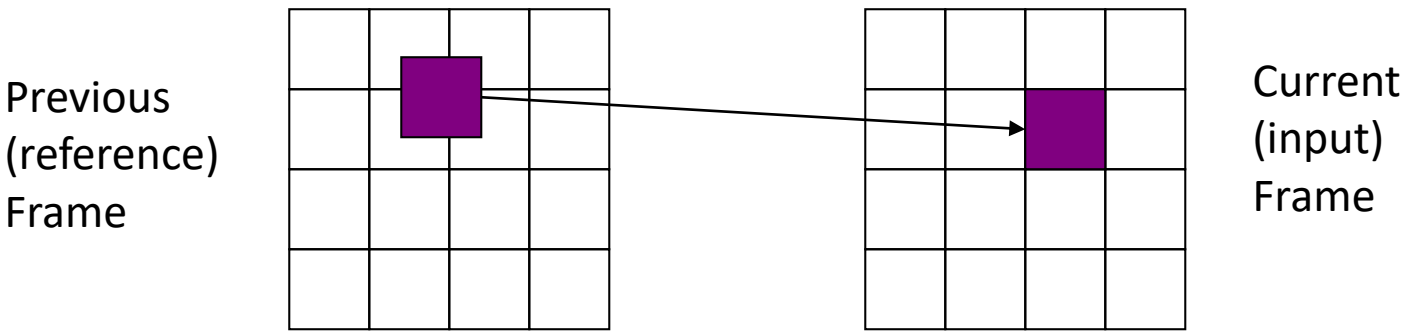
Block-based Coding

- Every frame is separated into blocks (z.B. 8x8 Pixel-Block, vgl. JPEG)
- Differences between frames is coded on the basis of blocks.



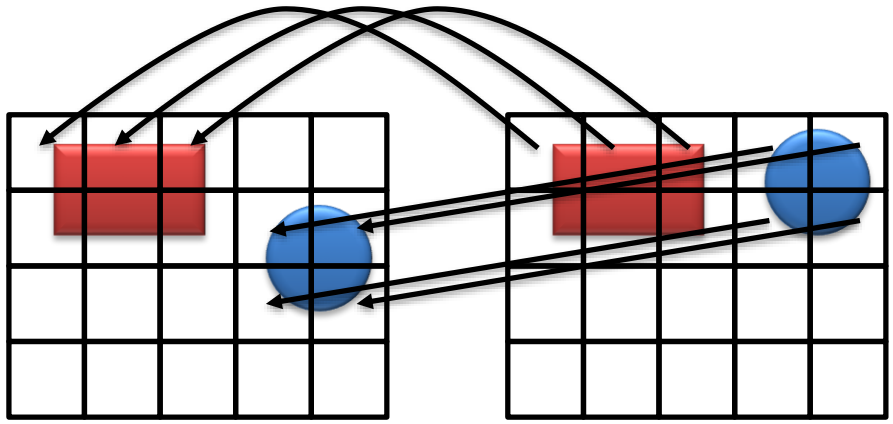
Motion Estimation

- Calculate a motion vector for every block
  - Specified the reference block in the reference frame
- In case an optimal block is existing :
  - Only motion vector + residual need to be saved. (**Inter-coding**)
- In case no optimal block is existing:
  - Block is coded without references (e.g. by JPEG). (**Intra-coding**)



Motion Estimation

- Search for a fitting block == motion estimation
  - Is processed by every block separatly
  - Looks for the most similar block in previous/suceeding frame!

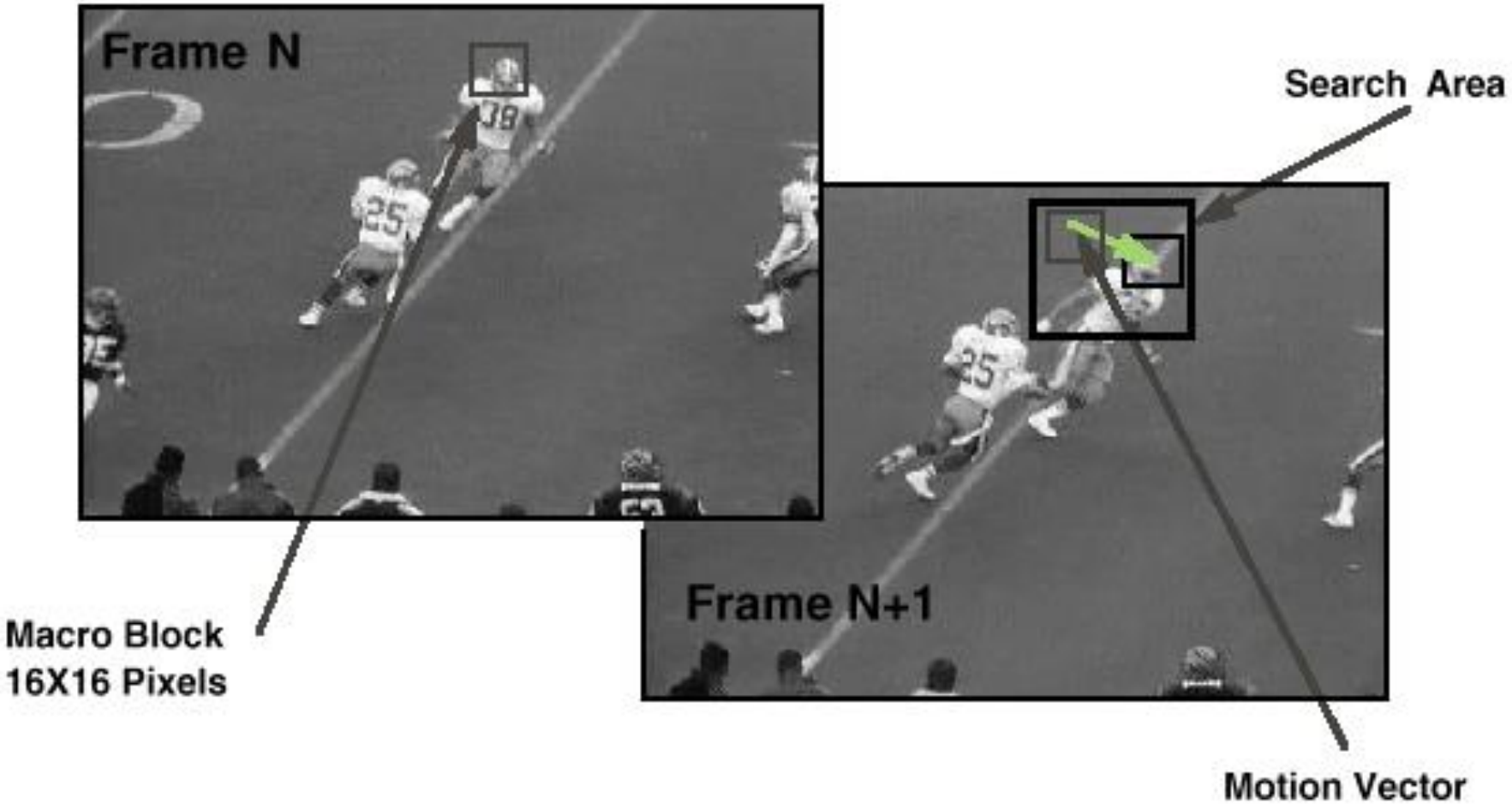


References can be found in succeeding frames as well

Motion Estimation

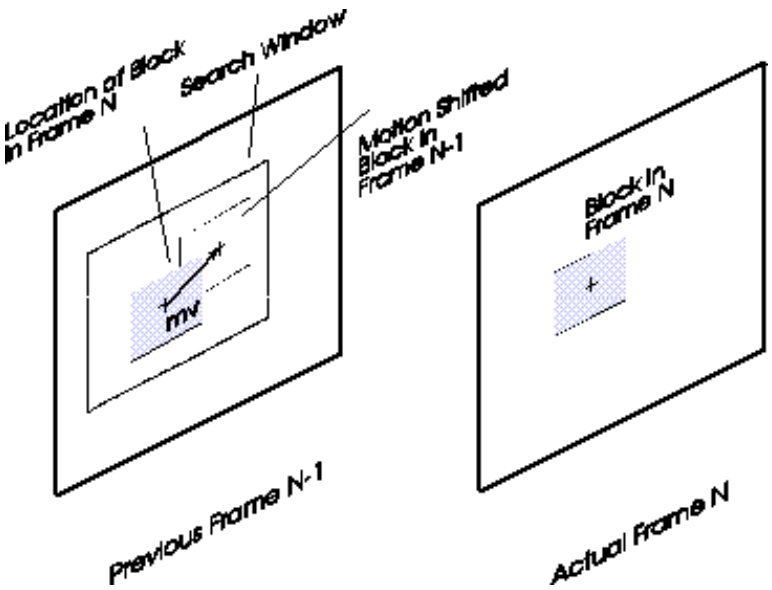
- The calculation of the error between blocks is called **Motion Compensation**.
- For smaller blocks we have the following characteristics:
  - higher probability to find an optimal block.
  - More motion vectors need to be saved.
- Be aware that MV + Residual might be larger as coding the blocks without references
  - Decision about coding is realised by encoder based on bit rate (Rate Control)

MPEG-2: Motion Vectors



Motion Compensation Prediction (MCP)

- An area of the preceding frame is the basis (Prediction) for this macro block
- The area is shifted by (x,y) pixels compared to the actual position (Motion)
- The motion vector and the difference are encoded (Compensation)



Motion Compensation Prediction (MCP)

Algorithms

The MPEG standard specifies the syntax of the encoded data (and therefore the decoding process as well).

On the other hand, the coding algorithms can be freely chosen.

Challenge:

fast detection of the best matches for MCP

MCP – Algorithms

Simple method

- Try all possible 16x16 fields – the best one wins
- Compare error-coding with intra-coding -> the shortest one wins

- ➢ This method is solid, but not fast.
- ➢ Too slow if high-speed coding (e.g. real-time) is required.

MCP Algorithms

More intelligent Methods

- Scanning of a given pixel window around the current position

- Choice of an error tolerance threshold: do not continue the search after exceeding it

➢ Does not necessarily find the best match, but needs less time.

MCP Algorithms

More intelligent Methods

- Search for the best reference block in a given pixel window
  - E.g. by a star shape
    - Provides good results (motion in most cases smooth)
    - Processed in several steps (e.g. three steps)

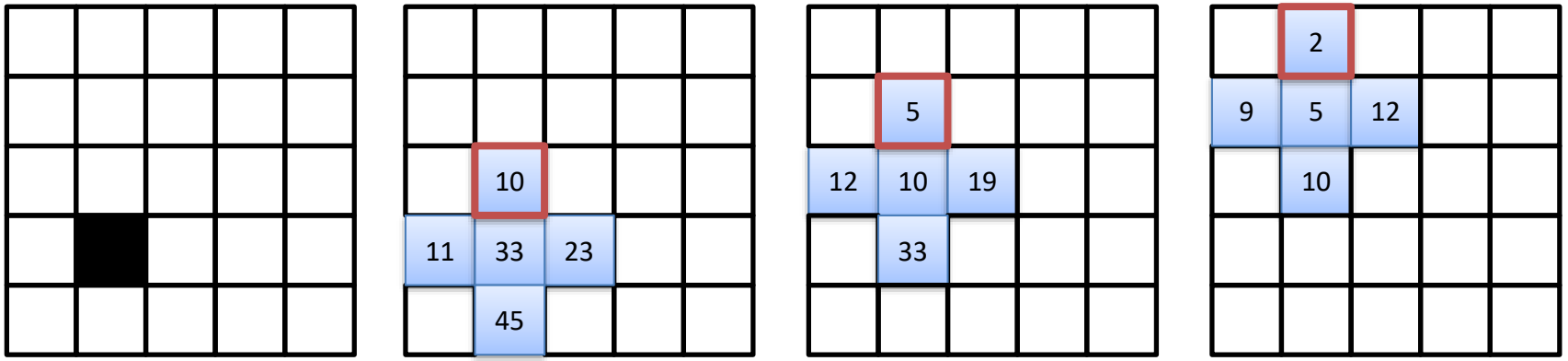


Bild n | **Bild n-1** | **Bild n-1** | **Bild n-1**

d.h. motion vector for the block is n = (0/-3)

Block Matching

- Mean Squared Error (MSE)

$$MSE = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (Current_{i,j} - Ref_{i,j})^2 / N^2$$

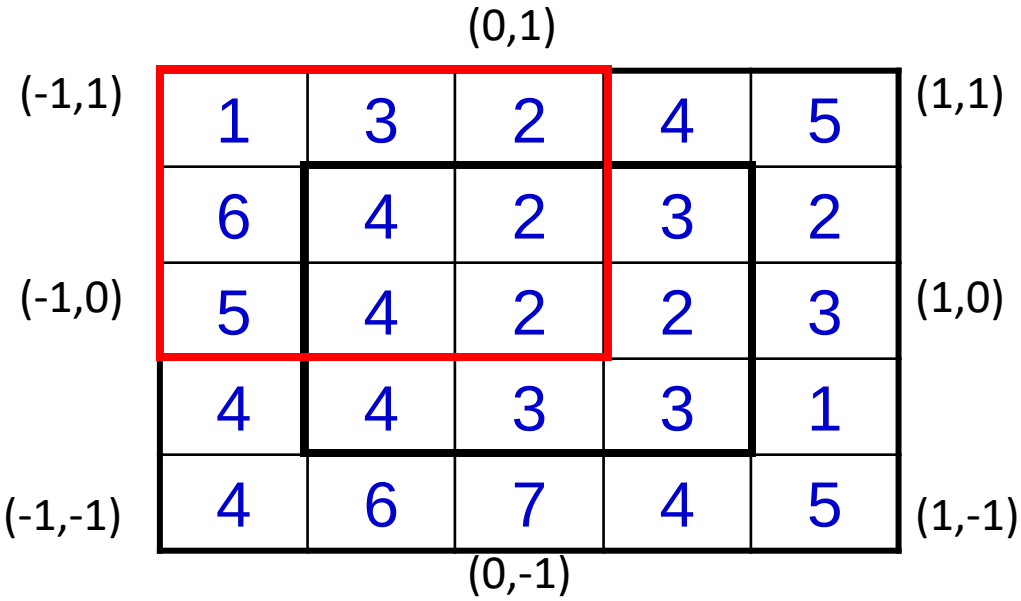
- Sum of Absolute Errors (SAE)

$$SAE = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |Current_{i,j} - Ref_{i,j}|$$

Block Matching with Mean-Squared-Error

1	3	2
6	4	3
5	4	3

Current block



MSE for middle position (0, 0):

$((1-4)^2 + (3-2)^2 + (2-3)^2 + (6-4)^2 + (4-2)^2 + (3-2)^2 + (5-4)^2 + (4-3)^2 + (3-3)^2) / 9 = 2.44$

Motion vector

(x,y)	-1,-1	0,-1	1,-1	-1,0	0,0	1,0	-1,1	0,1	1,1
MSE	4.67	2.89	2.78	3.22	2.44	3.33	0.22	2.56	5.33

MCP Algorithms

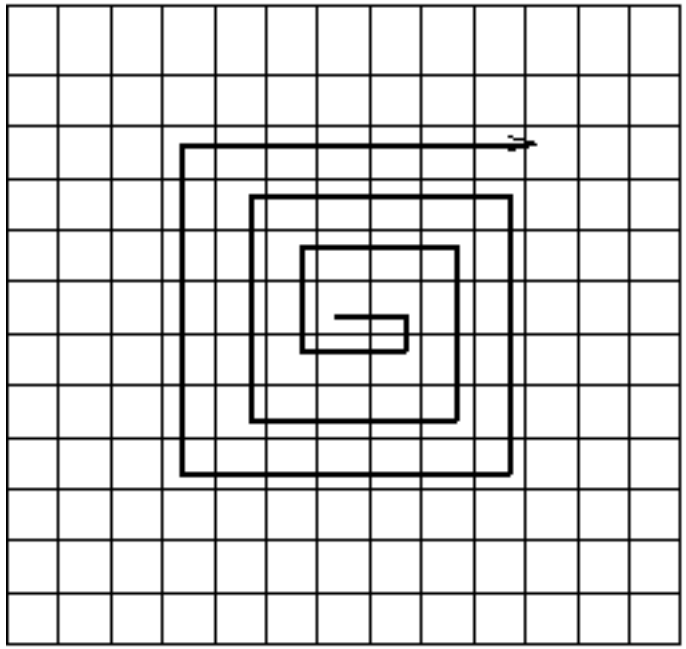
Rough Estimation

- Computation of the average values of all considered potential matches
- Fast comparison with the average of the current macro-block
- Enables to discard many candidates before more computationally intensive methods are applied

MCP – Algorithms

Spiral Search

- Searching of neighbor blocks according to a spiral pattern centered on the current macroblock in order to quickly find good candidates

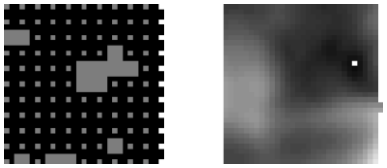


Centered spiral

MCP – Algorithms

Further algorithms

- Use of previous motion vectors
  - Consider the motion vector of the same block in preceding frames
  - Shift the search window by that vector
- Sub-Sampling
  - First search with a “rough grid”
  - Then performer a finer search around the best samples



- Fourier-Transformation
  - Perform a partial Fourier transformation of the lowest frequencies of the spectrum
  - Comparing the coefficients gives a good estimation of the error which is to be expected

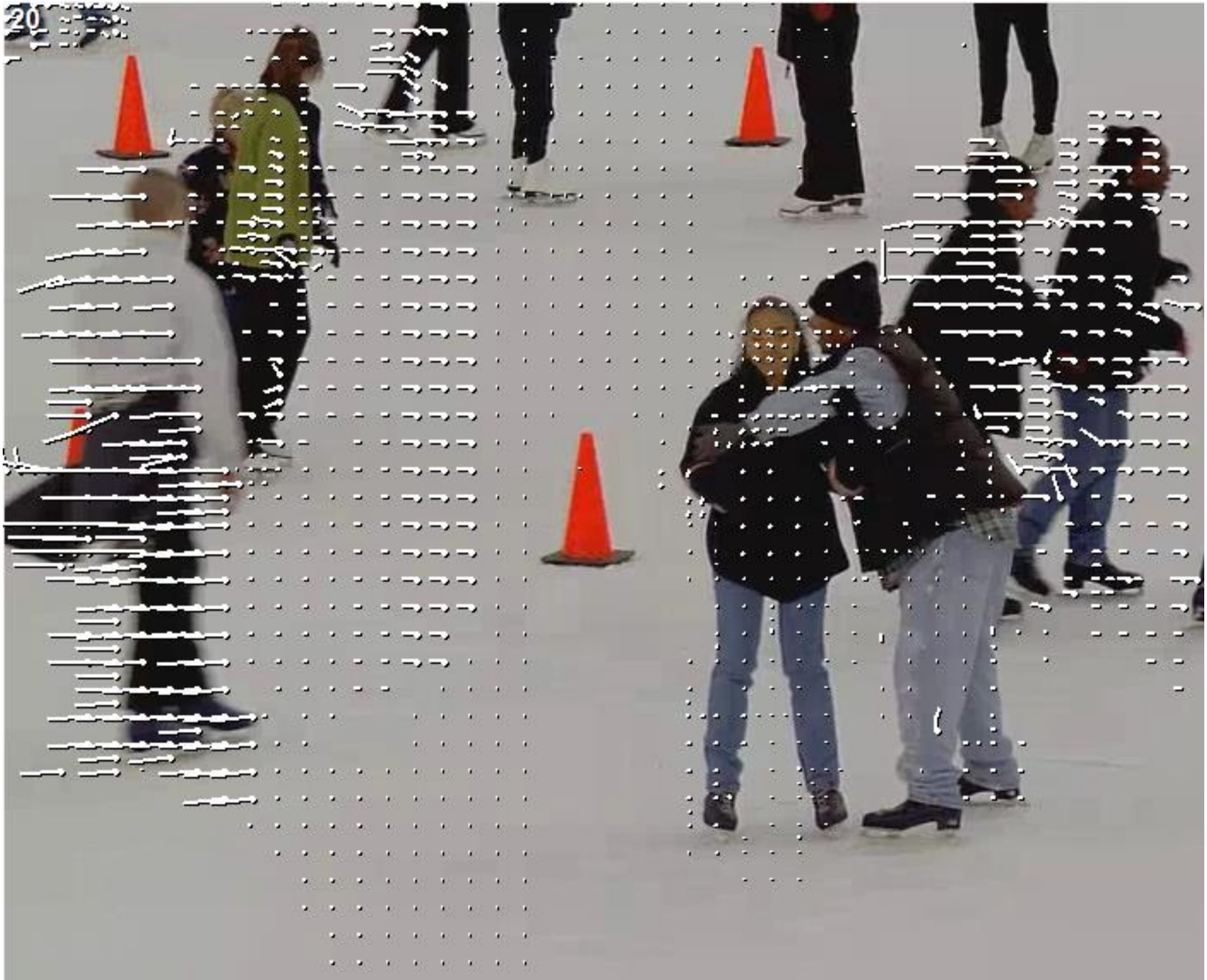

20

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2 (slides partly © Klaus Schöffmann, University Klagenfurt)

- 5.1 Overview and Motion compensation
- 5.2 MPEG-4
- 5.3 H.264
- 5.4 Further Video Codecs

MPEG – 1 & 2

- MPEG 1 : Int. Standard 11172 (1993, ISO/IEC)
  - CD-Video, CD-I for Multimedia applications
  - Video recorder-quality at ca 1.5 Mbit/s
  - MPEG-1 Compression is structured in layers.
    - MPEG-1 Layer 3 (or MP3) represents the audio part.

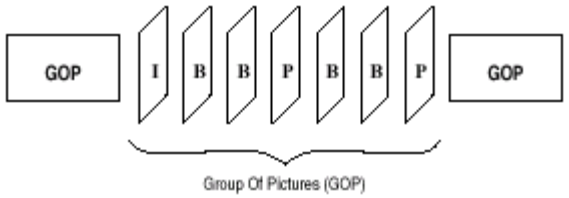
- MPEG 2 : Int. Standard 13818 (1994, ISO/IEC)
  - Improves MPEG-1 Compression
    - In particular through Motion Compensation
  - Digital Video with Broadcast-quality (NTSC) at 4 to 6 Mbit/s
  - High Definition TV at 15 – 30 Mbit/s

MPEG Compression Methods

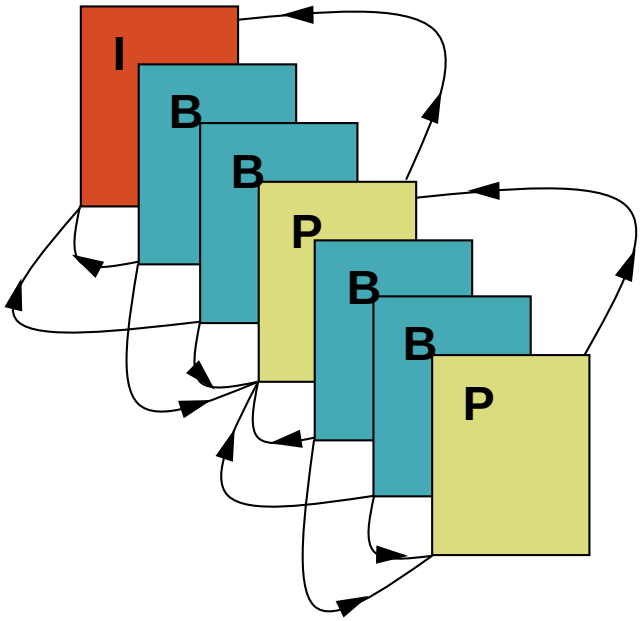
- Intraframe coding
  - Recognizes patterns inside the same frame
  - Based on Baseline JPEG Compression
  - DCT, Run-Length Encoding, …
- Interframe coding
  - Motion Compensation
  - I, P, B Frames
- Hierarchical Coding
  - Base Layer
  - Enhancement Layers

MPEG-2 Data Stream Structure I

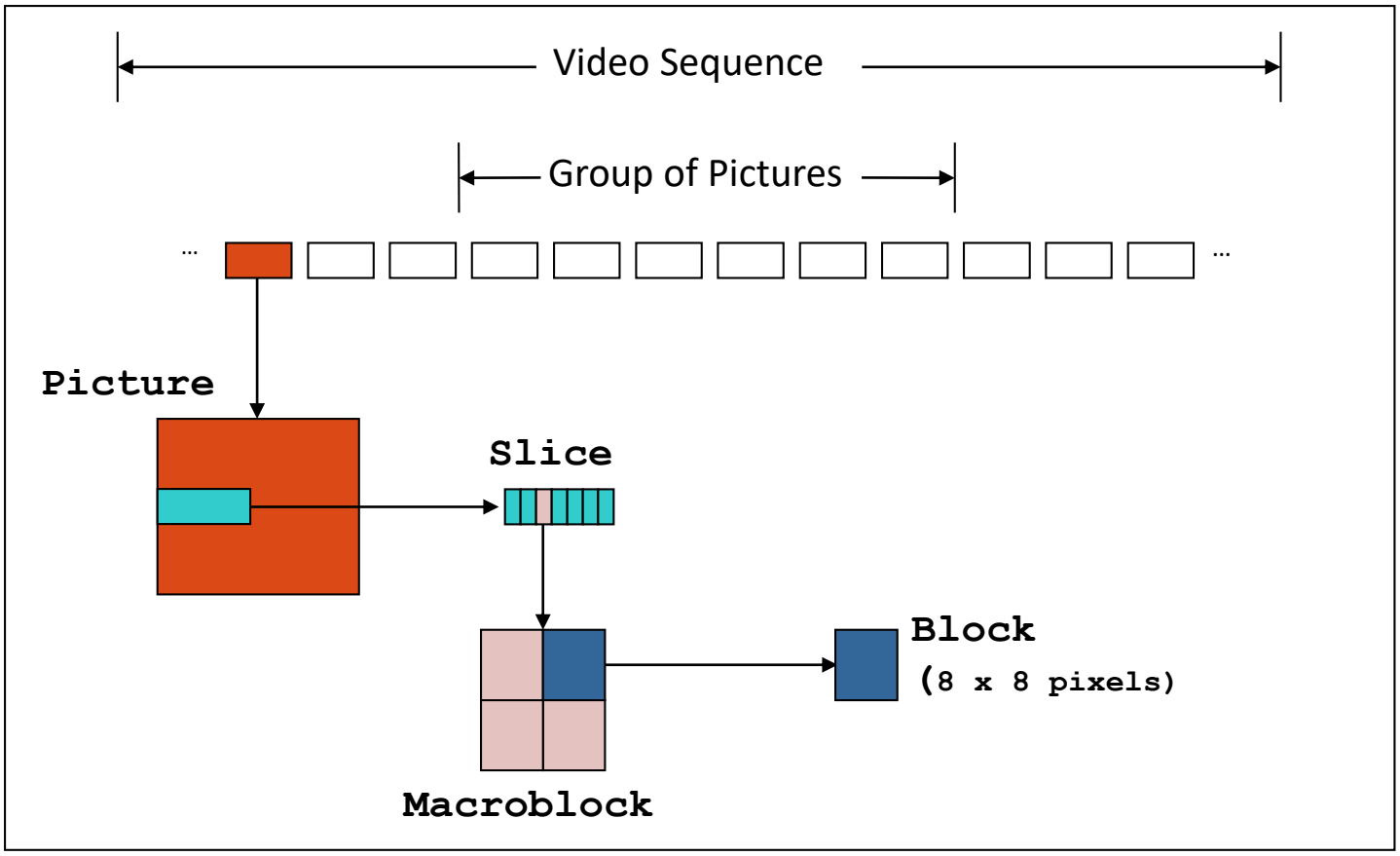
- Typical structure of a MPEG-2 stream:
  - I-BB-P-BB-P-BB-P-BB-I-BB…



  - **I-Frame :** coded without reference to other frames.; used for Fast Forward
  - **P-Frame :** coded predictively using the preceding I or P frame; can also contain intra-coded blocks (I)
  - **B-Frame** : bi-directionally coded based on the preceding and following I or P frame.; can also contain P- and I coded blocks

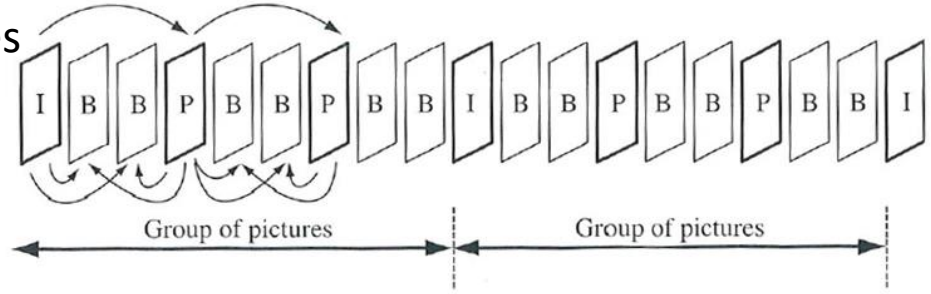


MPEG-2 Data Stream Structure II



Group of Pictures (GOP)

- GOP = Sequence of frames between two I-Frames
  - All frames in a GOP depend on the first frame (of the GOP)
  - Smallest unit for random access
    - Reduces costs for decoding single images of a video
      - In case one would need only one image of a GOP, then we can start at the I-frame
    - Reduces amount of storage of the decoder
      - Decoder can delete all previous images at the start of a GOP
    - Reduces error propagation
      - Transmission error only effects quality till the end of a GOP
  - Length of GOP can vary
    - Common length: 10-250 frames
    - E.g. DVB: 50 frames (2 secs)

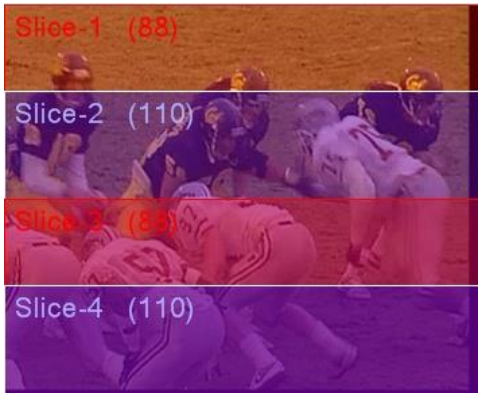


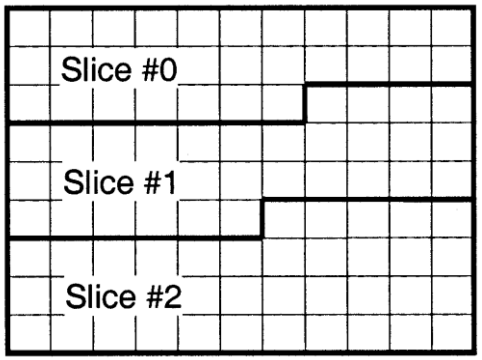
Slices

- Macro blocks are combined to one slice
- A slice can be coded independently
  - Does not contain references to blocks outside of a slice
  - Avoids error propagation
  - Allows parallel coding/decoding

2 Slices

4 Slices



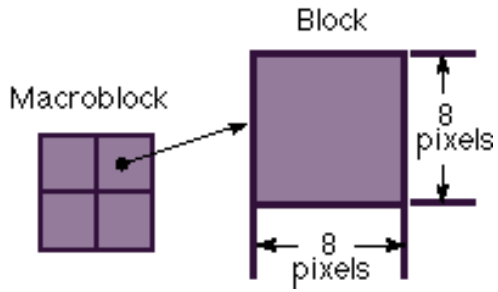


MPEG Data

Sample Blocks

The samples (pixel values) are organized in macro-blocks (16x16 Samples/Block)

Compromise : Data Overhead vs. Error



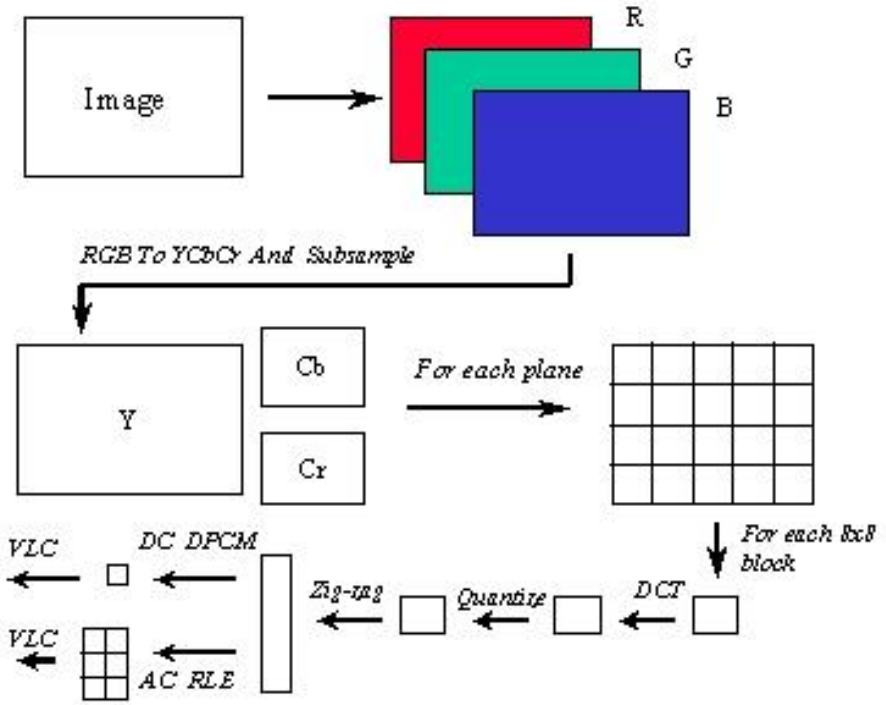
Each macro block can now be coded differently:

inter-coded or intra-coded.

Macro-blocks

intra-coded

- The block code contains the whole color information
- Coding similar to JPEG:
  - Transformation of the color space
  - Subsampling
  - Block generation
  - DCT
  - Quantification
  - Run-length encoding
  - Huffman coding

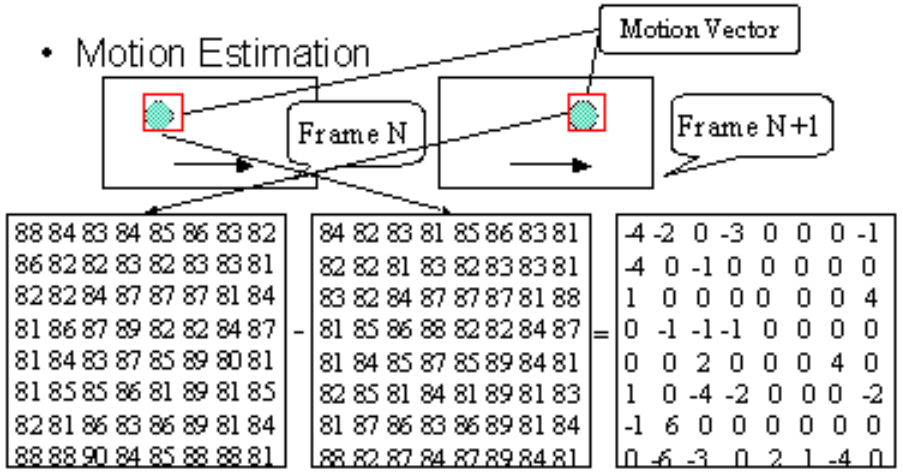


Macro-Blocks

inter-coded I

- An area, as similar as possible, of a different frame is referenced

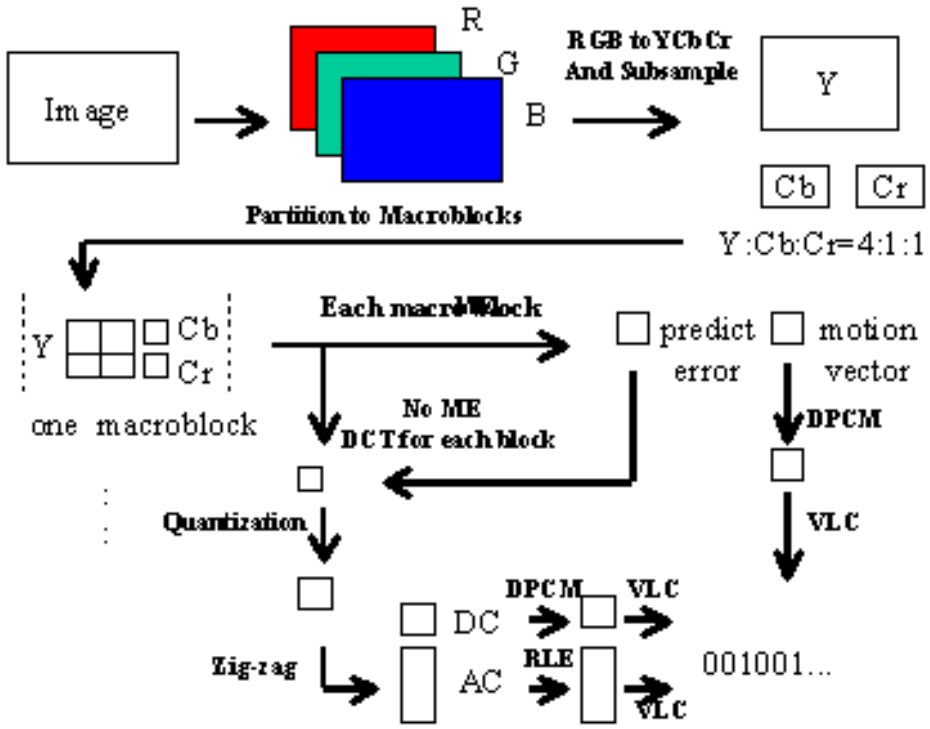
- Computation of motion vectors (Motion Compensation, MC)



Macro-Blocks

inter-coded II

- The sample difference is coded in the same way instead of the sample information



MPEG Video compression

- **M**oving **P**icture **E**xperts **G**roup (ISO/IEC JTC1)
  - Komitee of approx. 350 members (companies, research centers, Universities)

Standard	Description	Used for
MPEG-1 (1991)	compression of Audio and Video (to 1.5 Mbit) 5 parts (Systems, Video, Audio, Conf. Testing, Ref-SW)	VCD, MP3
MPEG-2 (1995)	Extension to MPEG-1, several Profile, interlacing, … 11 parts (Part 7 = AAC)	S-VCD, DVD (PS), DVB-S/ DVB-T (TS)
MPEG-4 (1998)	Extension of MPEG-1/2 by scence description (BIFS, LASeR), object-based compression, Ref- HW, ISO,MP4,AVC file format, Intellectual Property Mgmt and Protection (IPMP), 3D-Grafik,… 27 Teile	many cncoder: Quicktime 6, Xvid, DivX, Nero Digital, …
H.264/AVC (2003) = MPEG-4 Part 10	Video compression for high resolution and low bitrate, specialized for mobile devices Compression rate ca. 2x higher than MPEG-2 and MPEG-4 Part 2 (Video)	smartphone, Blue-ray video, DVB-S2, YouTube (HD), Many Encoder: Quicktime 7, x264, Nero Digital AVC, …

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2 (slides partly © Klaus Schöffmann, University Klagenfurt)

- 5.1 Overview and Motion compensation
- 5.2 MPEG-4
- 5.3 H.264
- 5.4 Further Video Codecs

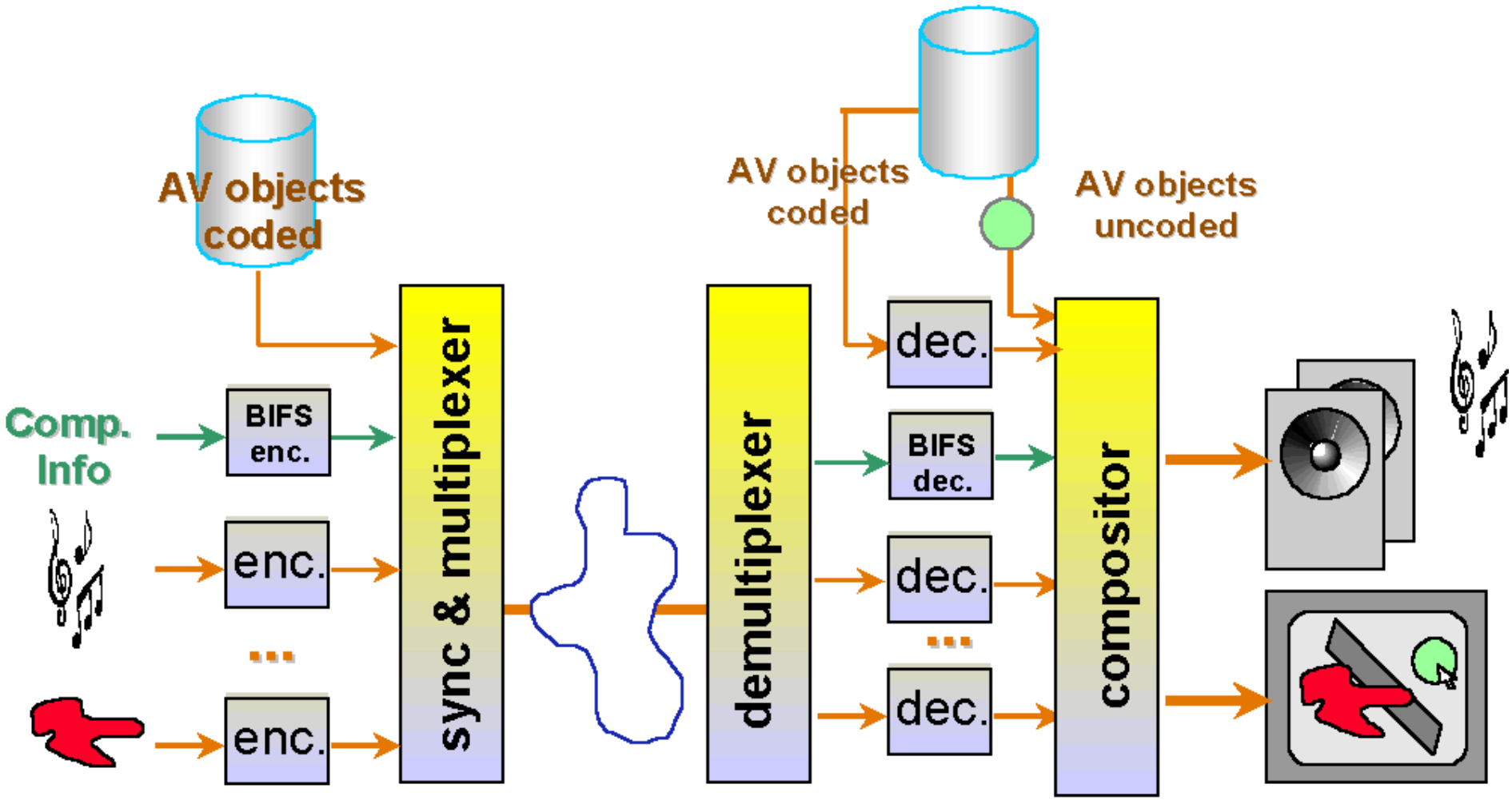
MPEG-4 Video compression

- Aims of MPEG-4 standard:
  - Optimized for Videos with low resolution and
    - E.g.176x144 with 10 fps
    - Use case: mobile devices, Video conferencing, …
  - Support for scences in videos and its description and mixing natural and scientific content
    - See next slides
- Some parts of MPEG-4 (25 in sum):
  - Format for scene description, Container-Format (.mp4), compression of 3D Graph, Intellectual property management & protection (IPMP)…

MPEG-4

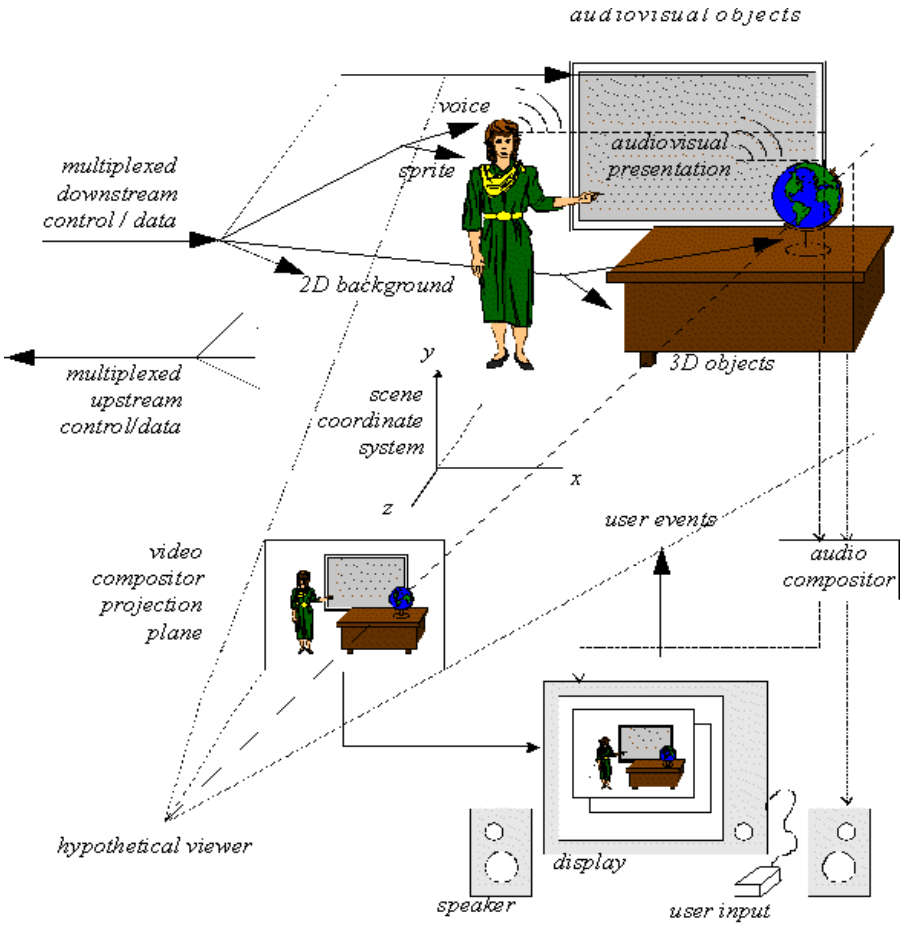
- Aims of MPEG-4
  - Focus on new Use Cases
  - Improvement of Compression was not in the main focus
- **Object-based coding**
  - Video consists of multiple streams, called AV-Objects
  - Every object is independent and can be coded with different codecs
    - E.g. common video with MPEG-2
    - Animation with another Codec
  - MPEG-4 Framework takes care of **(De-)Multiplexing**

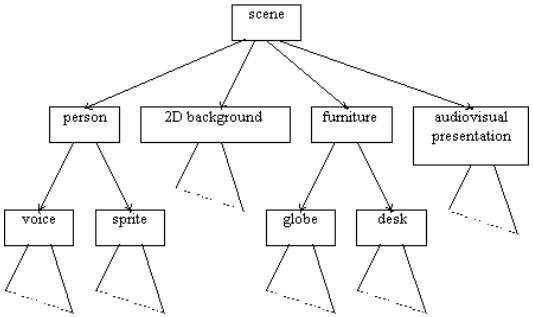
MPEG-4: Object-based Architectur



MPEG-4 Scenes

- A Video is a **Scene** and has a **Scene-Graph**
  - Can be changed interactively (e.g. user can move single objects)
  - **Interactive TV**
  - Own language for the description of scenes





Example: MPEG-4 Scene

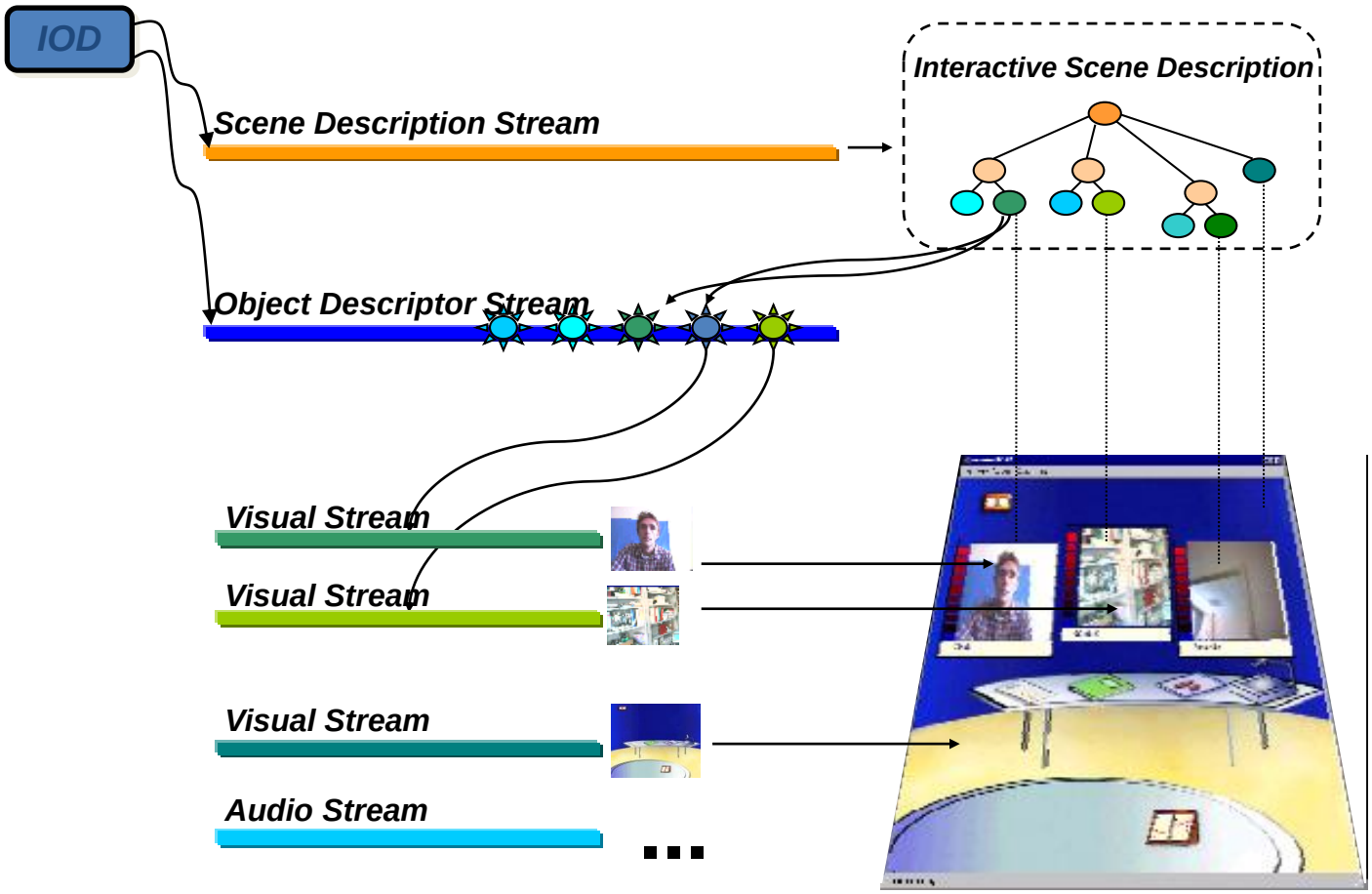


Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2 (slides partly © Klaus Schöffmann, University Klagenfurt)

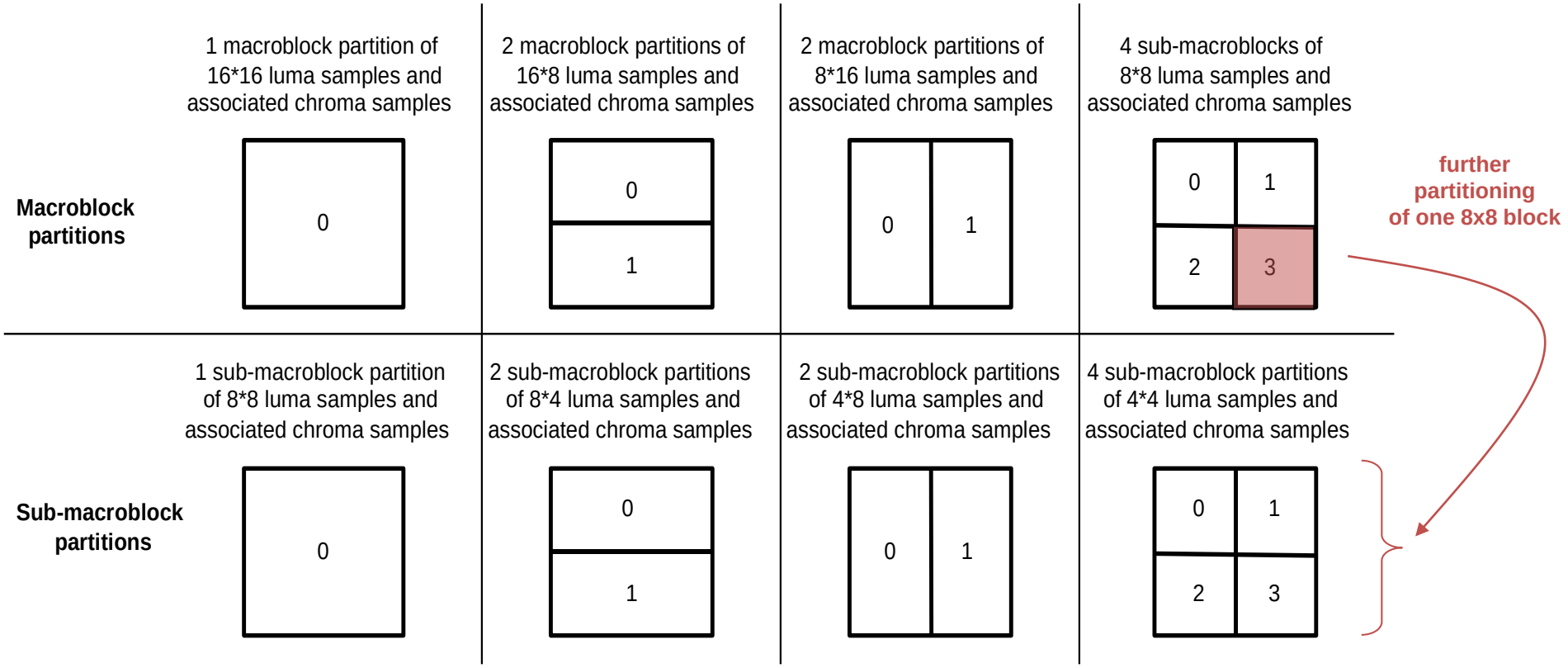
- 5.1 Overview and Motion compensation
- 5.2 MPEG-4
- 5.3 H.264
- 5.4 Further Video Codecs

H.264/AVC Video compression

- Part of MPEG-4 - Part 10: **Advanced Video Coding (AVC)**
  - Developed common with ITU (H.264)
  - 2003 finalized
- Many improvements in comparison to MPEG-2
  - Aims
    - Better quality with the same Bitrate/file size
    - Flexible support of resolution and Bitrate
    - Limit spreading of transmission errors
  - Used for
    - Videos on mobile devices (z.B. 3GPP)
    - DVB-S2 (HDTV)
    - Blu-ray and 3D Blu-ray

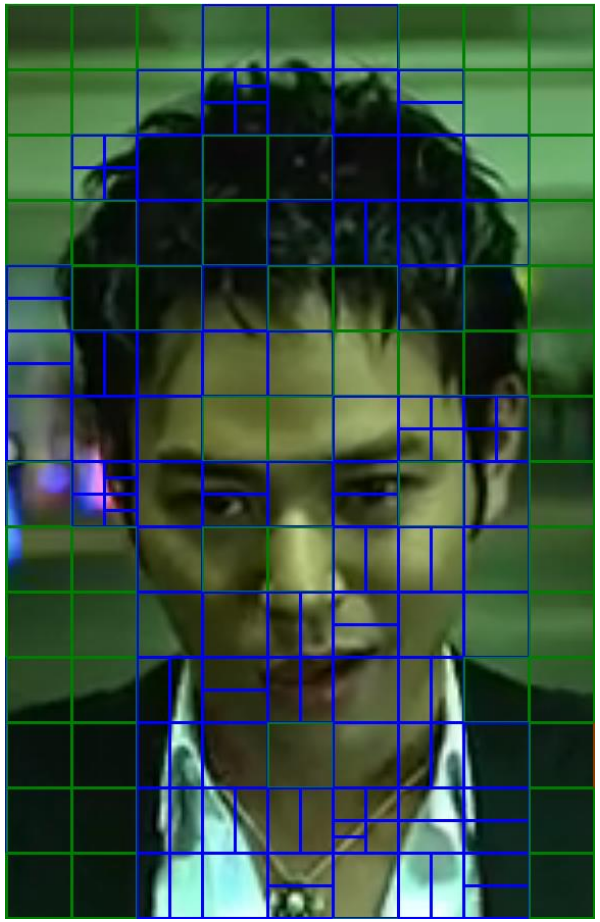
Macro blocks in H.264

- Macro block is divided into Partitions
  - 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4



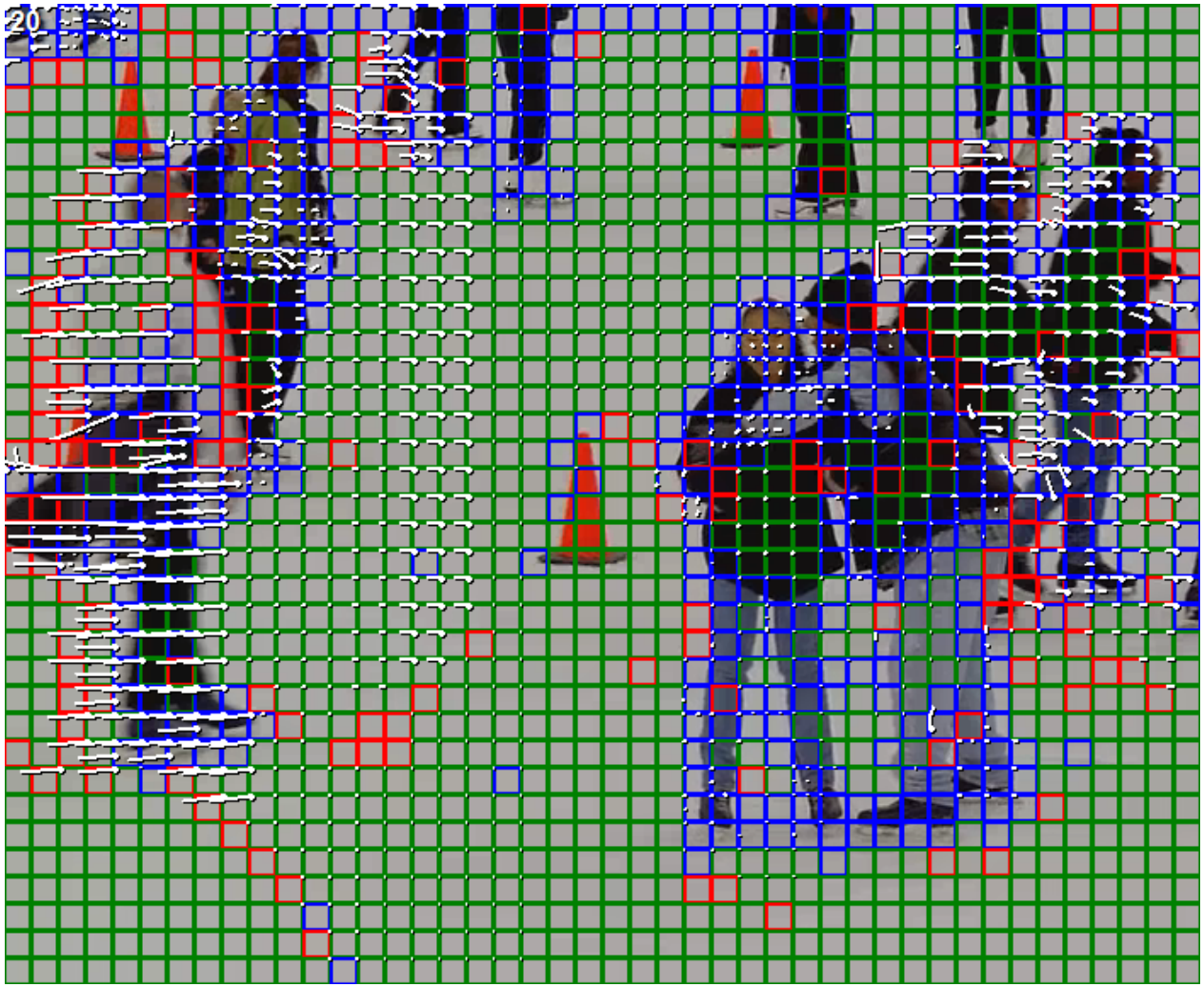
Macro blocks in H.264

- Arrangment of Partitions depends on pixel structure
- Partition is the unit for motion composition
- In addition to (I,P,B) also ***Skipped* (S)** Macro blocks
  - Only a motion vector is saved, no residual
  - **Motion Vector Prediction** is applied
    - Coherend blocks have similar motion vectors
    - See later

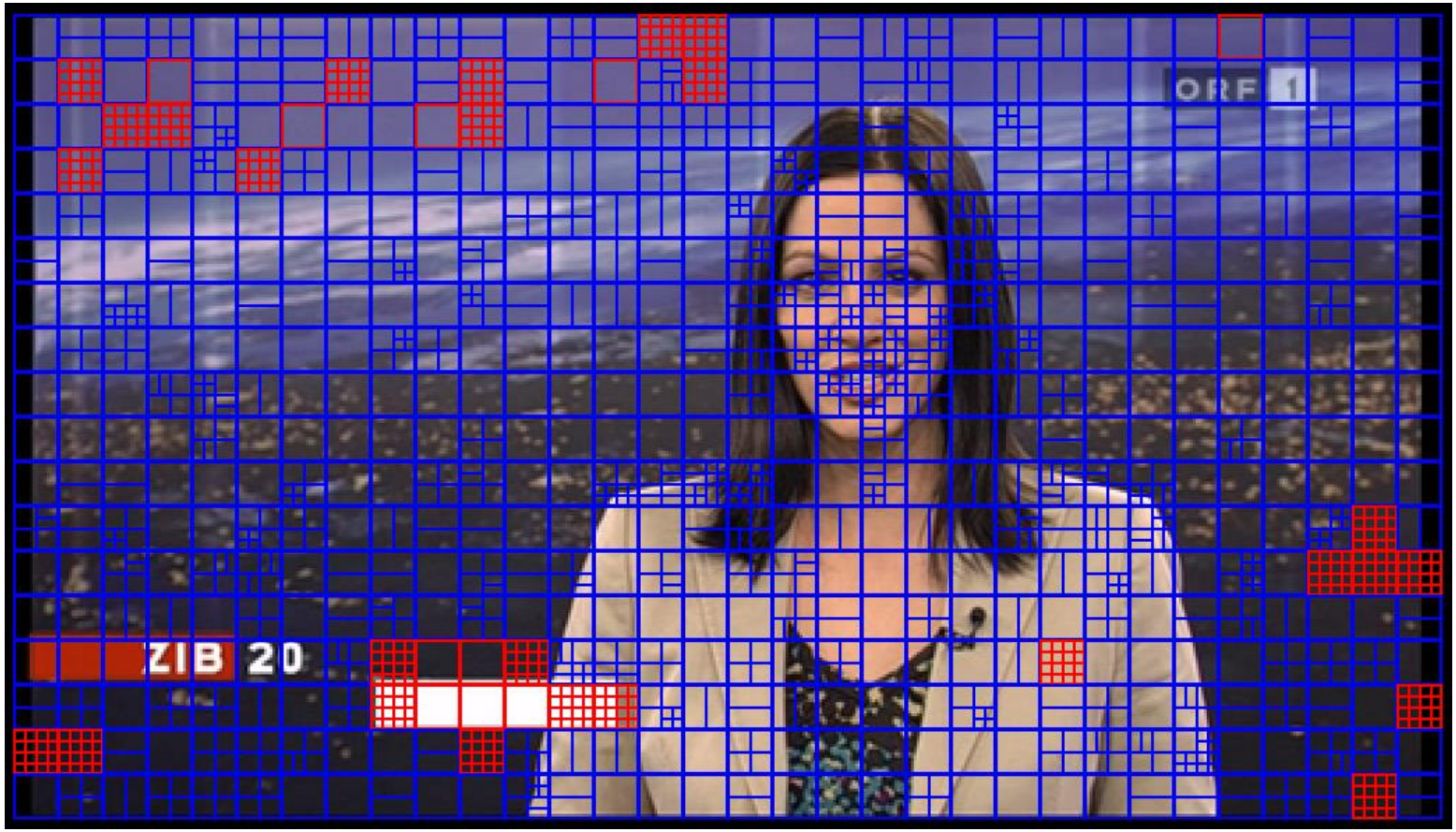


Macro blocks (H.264)

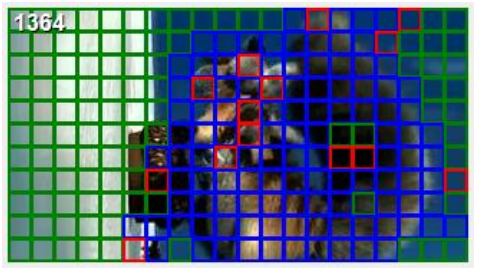
- Red = Intra (I)
- Blue = Inter (P)
- Green = Skipped (S)

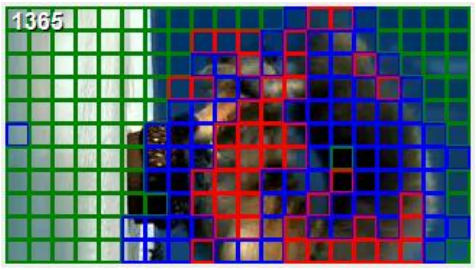


Macro block Partitions (H.264)



Example H.264/AVC





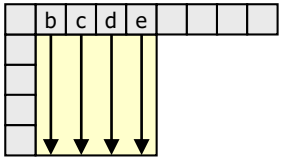
MB types
I (red)
P(blue)
S (green)

motion
vectors

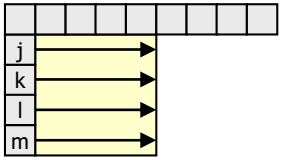
Extensions of H.264 (1)

- Intra-Prediction
  - Pixel of neighboring blocks are used as references
  - Only difference is saved (similar to P-Block)
  - Much better compression as JPEG
- In sum 9 possibilities
  - Depends on size of Partition (16x16 or 4x4)
  - See next slides

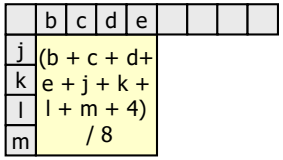
Intra Prediction



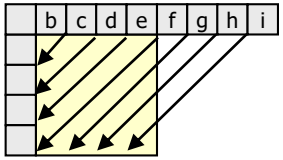
0: vertical



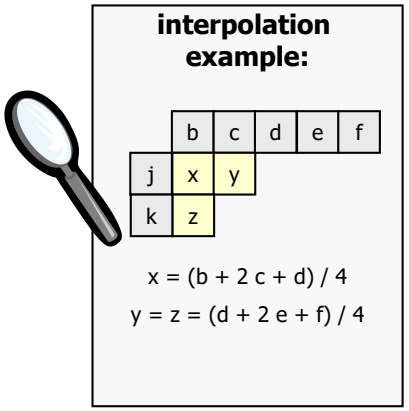
1: horizontal

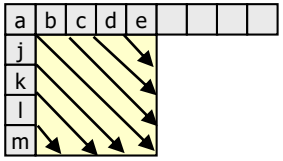


2: DC (average)

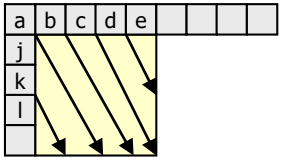


3: diagonal down left (45° to the left, interpolated)

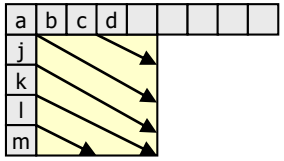




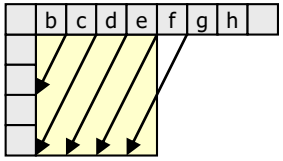
4: diagonal down right (45° to the right, interpolated)



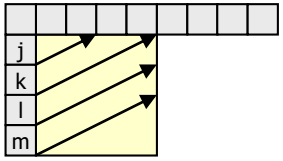
5: vertical right (26.6° to the left, interpolated)



6: horizontal down (26.6° below horizontal, interpolated)



7: vertical left (26.6° to the right of vert., interpolated)



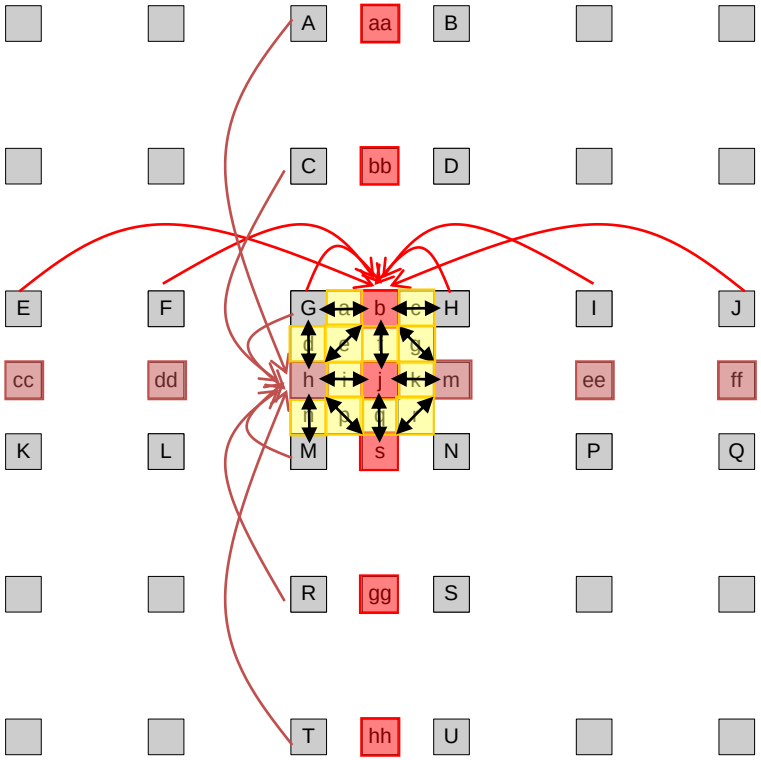
8: horizontal up (26.6° above horizontal, interpolated)

Extensions of H.264 (2)

- **QPEL (Quarter Pixel Estimation)**
  - Motion estimation bases on Pixel-interim values (by **Interpolation**)
  - Approach
    - Image is scaled up 4 times
    - Interim vaules are interpolated based on surrounding pixels
    - Motion estimation is determined on the enlarged image
      - Therefore a motion vector can point to a ¼ Pixel
  - Advantages ?
    - Let us assume that an object moves in 4 frames exactly one Pixel
      - This means per image ¼ Pixel
    - With only a basic pixel accuracy, the object would jump from image 3 to image 4 for one pixel which hinders a smooth video experience

Interpolation - Luma

- ¼ accuracy with the use of **6-Tap-Filter**



half-pel vert. pixel interpolation. e.g.:

$h = (A - 5\,C + 20\,G + 20\,M - 5\,R + T + 16) / 32$

half-pel hor. pixel interpolation. e.g.:

$b = (E - 5\,F + 20\,G + 20\,H - 5\,I + J + 16) / 32$

qpel pixel interpolation. e.g.:

$n = (m + h + 1) / 2$

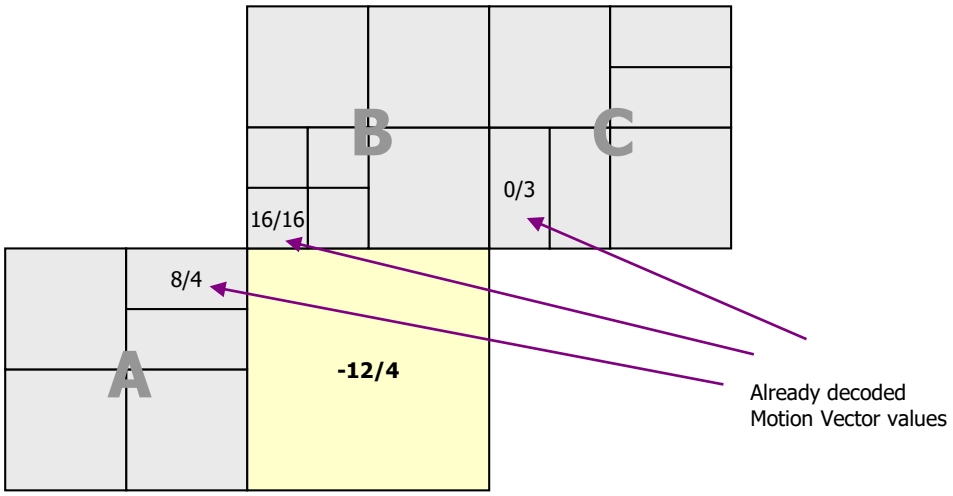
→ Very expensive!

Extension of H.264 (3)

- **Motion Vector Prediction**
  - Neighboring blocks have very often similar motion vectors
  - therefore, one predict the new motion vector based on neighboring blocks
  - Only difference is saved (MVD)
- Example: Decoder receives for block MV = (-12/4)
  - Prediction of neighboring blocks:
    - X = median(8,16,0) = 8
    - Y = median (4,16,3) = 4
    - MVP (X/Y) = (8/4)
  - Real value:
    - (8/4) + (-12/4) = (-4/8)

Information:

Motion Vector values in ¼ resolution (Luma)

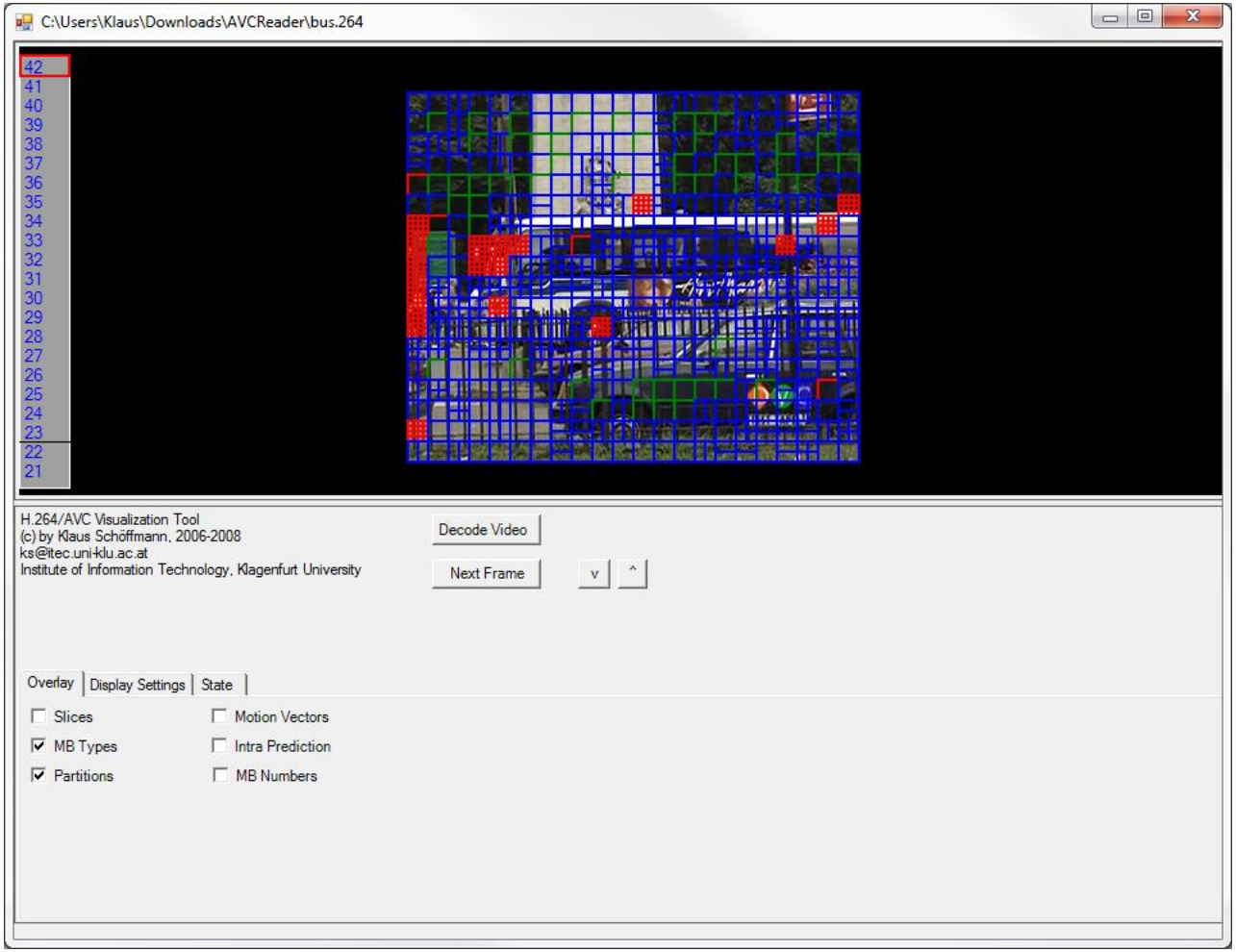


Extension of H.264 (4)

- Deblocking Filter
  - At the end of the decoding process a Pixel-Filter is applied
  - adjust strong edges between block borders
    - And reduces compression artefacts which occur through low bit rates

deblocked

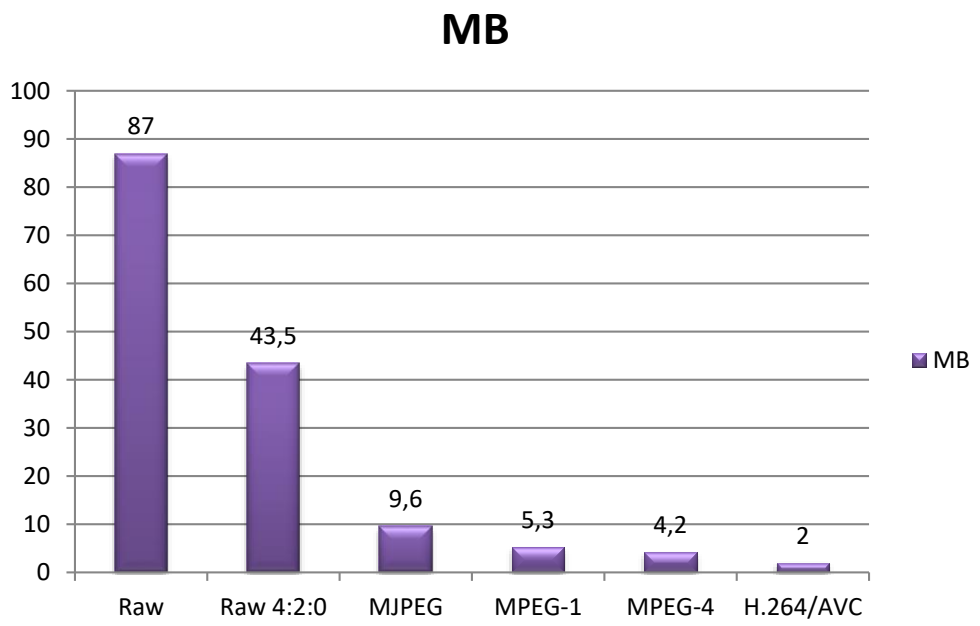
Demo



http://vidosearch.com/demos/AVCReader.zip

Video compression

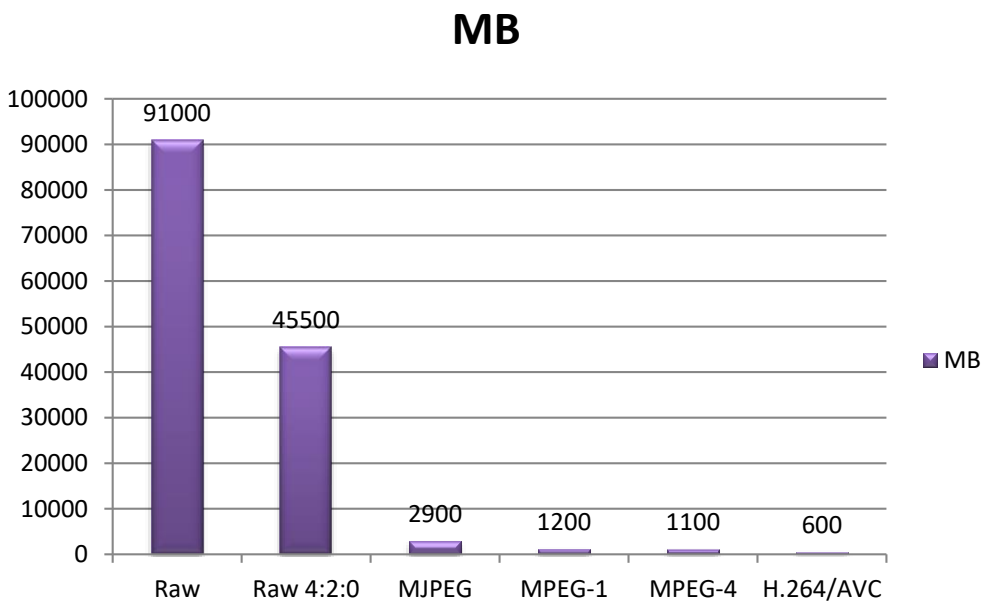
- Example video 1
  - 352 x 288 Pixel, 25 frames per second, 12 seconds in sum
  - Uncompressed size (with RGB, 24 Bit)?
    - 87 MB
  - Uncompressed size with 4:2:0 Subsampling (YUV420)?
    - 43,5 MB (2x)
  - Size of MJPEG *
    - 9,6 MB (9x)
  - Size of MPEG-1 *
    - 5,3 MB (16x)
  - Size of MPEG-4 *
    - 4,2 MB (20x)
  - Size of H.264/AVC *
    - Ca. 2 MB (43x)



* … same quality

Video compression

- Example video 2
  - 1920x 1080 Pixel, 24 frames per second, 654 seconds in sum
  - Uncompressed size (with RGB, 24 Bit)?
    - 91 GB
  - Uncompressed size with 4:2:0 Subsampling (YUV420)?
    - 45,5 GB (2x)
  - Size of MJPEG *
    - 2,9 GB (31x)
  - Size as MPEG-1 *
    - 1,2 GB (75x)
  - Size as MPEG-4 *
    - 1,1 GB (82x)
  - Size as H.264/AVC *
    - Ca. 600 MB (151x)



* … same quality

H.265/HEVC

- Extension of H.264/AVC
- Compression rate doubled
- Realized by several improvements
  - Macroblocks vs. Coding tree units
  - Better motion compensation and spatial prediction
  - Much more performant for 4k videos, which results in smaller bandwith requirement

  - See for more information: https://www.youtube.com/watch?v=Fawcboio6g4

Table of Contents

Coding and Compression

1 Motivation

2 Criteria for Compression

3 Basic Methods

- 3.1 Run-length Encoding
- 3.2 Statistical Coding
- 3.3 Transformation Coding

4 JPEG Coding

5 MPEG 1 & 2 (slides partly © Klaus Schöffmann, University Klagenfurt)

- 5.1 Overview and Motion compensation
- 5.2 MPEG-4
- 5.3 H.264
- 5.4 Further Video Codecs

Further video codecs

- DivX
  - proprietary
  - MPEG-4, H.264/AVC (DivX 7)
- Xvid
  - Open (GPL)
  - MPEG-4
- Ogg Theora
  - open Videocodec by Xiph.Org Foundation (BSD licence)
  - Similar to MPEG coding, no B-Frames, no interlacing
- VC-1
  - Successor of WMV (proprietary)
  - Supported by HD DVDs and Blu-rays
  - Very similar to H.264/AVC
- RealVideo, Flash (VP3), …

Tools and Libraries

- ffmpeg
  - Open-Source Multimedia Library (C/Linux)
  - Console for Video/Audio coding and convertion
  - Support for most well known Video/Audio Codecs
  - Runs on Windows
  - Is used by many other projects and tools
    - Mplayer, mencoder, VLC, HandBrake, PSPVideo9, ...
- mplayer und mencoder (based on ffmpeg)
  - Open-Source programs
  - Media Player
- gstreamer
  - Open-Source multimedia library (C/Linux)
- x264
  - Open-Source H.264/AVC encoder

Overview of Container-Formats

Typische Kombinationen

Container	Name	Videocodecs	Audiocodecs
3GP/3GP2	3rd Generation Mobile	H.263, MPEG-4, H.264	AMR-NB/WB, (HE-)AAC
AVI	Audio Video Interleave	MPEG-4, DV, MJPEG, Indeo, Cinepak	MP3, MP2, (AD)PCM, AC3
ASF	Advanced Streaming/Systems Format	Windows Media Video, VC-1, MS MPEG4 v3	Windows Media Audio
DIVX	DivX Media Format	DivX	MP3, AC3, PCM
DV	Digital Video	DV	PCM
DVR-MS	Microsoft Digital Video Recording	MPEG-2	MP2, AC3
EVO	Enhanced VOB	H.264, VC-1, MPEG-2	(E)AC3, DTS (HD), PCM
FLV	Flashvideo	H.263, VP6, H.264	MP3, AAC, ADPCM
M2TS/MTS	MPEG-2 Transport Stream (192 Byte)	H.264, VC-1, MPEG-2	(E)AC3, DTS (HD), PCM
MKV	Matroska	H.264, MPEG-4	MP3, AC3
MP4	MPEG-4	MPEG-4, H.264	AAC
MPG	MPEG Program Stream	MPEG-1, MPEG-2	MP2
MOV	QuickTime Movie	H.264, MPEG-4, MPEG-1, MJPEG, Sorenson Video	MP3, AC3, PCM
OGM	Ogg Media	Ogg Theora, Xvid	Ogg Vorbis, MP3, AC3
PS	MPEG-2 Program Stream	MPEG-2	MP2, AC3, DTS, PCM
RM(VB)	Real Media (variable Bitrate)	Real Video	Real Audio, AAC
TS/TP/TRP/PVR/VDR	MPEG-2 Transport Stream (188 Byte)	MPEG-2, H.264	MP2, AC3
VOB	Video Object	MPEG-2	AC3, DTS, PCM, MP2
WMV	Windows Media Video	Windows Media Video, VC-1	Windows Media Audio