Real-Time Audio and Video

NOTICE: This chapter is a prerequisite for the next chapter: *Multimedia over IP*

Multimedia Payloads

Raw Audio (uncompressed audio)

Telephony:

Speech signal: 20 Hz – 3.4 kHz → 4 kHz

PCM (Pulse Coded Modulation) → 8000 samples/sec x 8 bits = 64 kbps

Teleconferencing:

16000 samplea/sec x 8 bits = 128 kbps

Music (CD ROM):

Human hearing: 20 Hz – 22 kHz

44.1 samples/sec x 16 bits = 705.6 kbps (mono)

= 1.41 Mbps (stereo)

Multimedia Payloads (cont.)

Raw Video (uncompressed video)

Videoconferencing (MPEG-1):

Small screen
352 pixels x 240 lines x 12 bits x 30 frames/sec = 30.4 Mbps

TV (MPEG-2, NTSC):

720 pixels x 480 lines x 24 bits (3 x 8, red, green, blue)
x 30 frames/sec = 166 Mbps

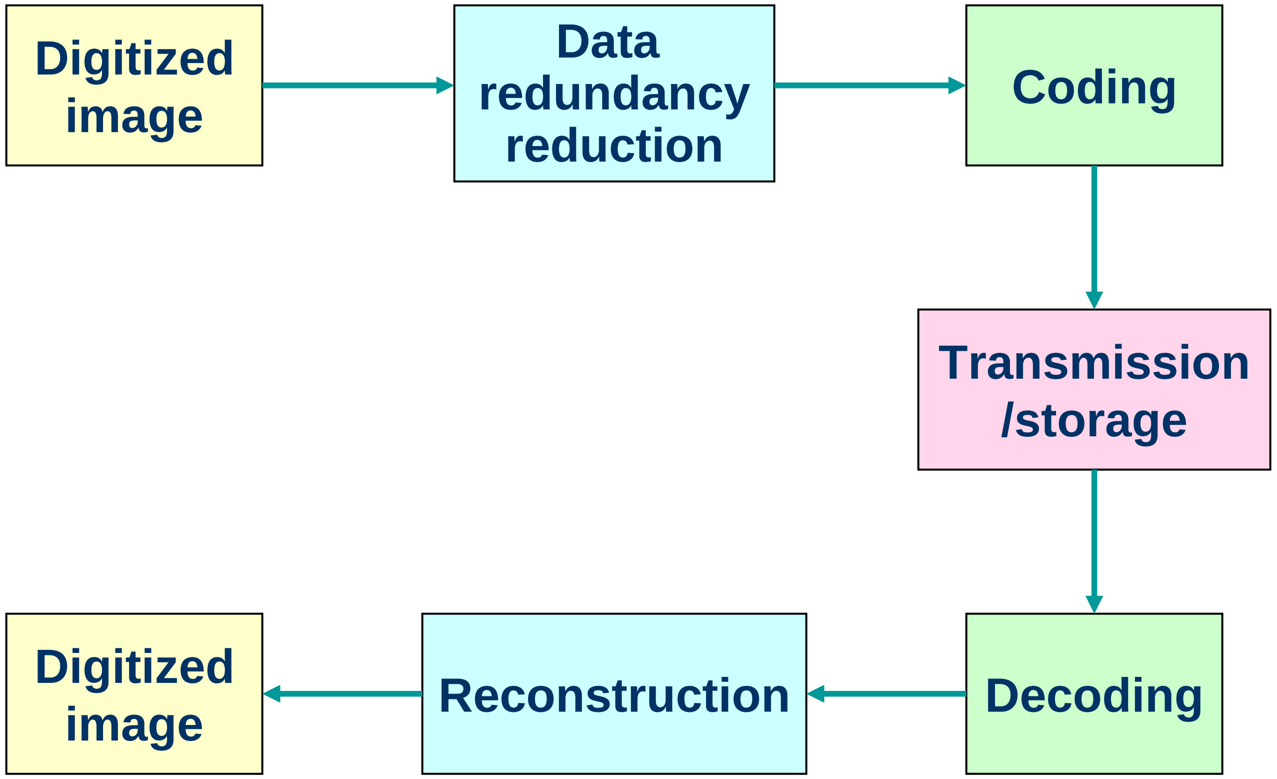
HDTV:

1920 pixels x 1080 lines x 24 bits (3 x 8, red, green, blue)
x 30 frames/sec = 1.493 Gbps

MPEG – Motion Pictures Expert Group

Compression

The previous two slides show that a transmission of raw signals, specially of video signals, would require enormous bandwidth. Given the fact that video signals have a high degree of redundancy, its is clear that this redundancy needs to be removed before transmitting or storing the signal. Removing or reducing the redundancy in signals is called compression.



Compression (cont.)

The key concepts used in video compression are:

- Discrete Cosine Transform (DCT)
- Quantization
- Zig-Zag Scan
- Run-Length Encoding (RLE)
- Entropy Coding
- YUV Color spaces and sub sampling
- Temporal compression

Spatial compression

The first five concepts will be applied on the compression of still grayscale images. The techniques will be then extended to the still color images, thus giving the foundation for JPEG, the major standard for transmission of still images in Internet.

The techniques of JPEG can further be extended to moving images (video) and audio giving MPEG, also the most common standard used in Internet.

Discrete Cosine Transform (DCT)

DCT belongs to the family of Fourier transforms – it is similar to discrete Fourier transform (DFT). Fourier transform of 1D signals (e.g. time-domain signals like audio signals) help us to see how the signal's energy (read: "signal's information contents") is distributed across different frequencies. For example, we can discover that most of the signal's energy is concentrated in a small range of frequencies. That gives a room for temporal compression of 1D temporal signals.

Fourier transform can also be applied to 2D signals – images. In the case of a single image (no temporal component assumed), the image is a function of two spatial variables. The reasoning about the 2D Fourier transform is similar to the reasoning about 1D Fourier transform: the signal energy concentration and spatial compression.

DCT (cont.)

As with the Fourioer transform there are two versions of DCT: one-dimensional and two-dimensional DCT.

One-dimensional DCT

Forward 1D DCT

Frequency-domain (transformed) signal

Time

$$S(f) = DCT(s(t))$$

$$s(t) = IDCT(S(f))$$

Time-domain (original) signal

Inverse 1D DCT

Frequency

Function *S(f)* shows explicitly the frequency components of the signal and the distribution of the signal's energy (i.e. signal's information contents) among the frequency components.

DCT is a lossless transformation, i.e. the original time-domain signal can be fully restored from the frequency-domain signal by applying the inverse DCT.

DCT (cont.)

Two-dimensional DCT

Forward 2D DCT

Frequency-domain (transformed) signal

$$S(p,q) = DCT(s(\overbrace{x,y}^{\text{Pixel position}}))$$

$$s(x,y) = IDCT(S(\underbrace{p,q}_{\text{2D frequency}}))$$

Spatial signal (pixel values of image)

Inverse 2D DCT

The concept of frequency of time-domain signal can be intuitively related to the rate of change of signal *s(t)* in time. In the case of spatial functions, the frequency can be related to the rate of change of pixel values along the spatial axes of *S(x,y)*

DCT (cont.)

We will consider the functions *s(x,y)* and *S(p,q)* to be discrete functions defined over the sets of integers

$$p, x \in [0, 1, \dots, M-1]$$

$$q, y \in [0, 1, \dots, N-1]$$

In that case *s(x,y)* and *S(p,q)* become matrices:

$$A = [s(x,y)]_M^N$$

$$B = [S(p,q)]_M^N$$

DCT (cont.)

The image compression is based on the fact that DCT tends to concentrate the signal's energy into a small regions of the frequency domain (exactly into the part of the frequency domain that is closer to the origin). We may pick a threshold value *h* and nullify all elements of *S(p,q)* that are smaller than *h*:

$$S'(p,q) = \begin{cases} S(p,q) & if\ |S(p,q)| > h \\ 0 & if\ |S(p,q)| \le h \end{cases}$$

The matrix *S'(p,q)* has fewer nonzero elements than the matrix *S(p,q)*. That means we have less information to transmit or store (we transmit the nonzero elements only). However the removal of small elements of *S(p,q)* leads to lossy compression, since *DCT(S'(p,q))* is no longer equal to *s(x,y)*.

By varying *h* we can compromise between the compression rate and the compression quality of the restored image.

DCT (cont.)

Forward Transform:

$$\begin{tikzcd} A = [A_{mn}]_M^N \arrow[r] & \boxed{\textbf{DCT}} \arrow[r] & B = [B_{pq}]_M^N \end{tikzcd}$$

MxN block
of grayscale pixels

MxN block
of DCT coefficients

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \qquad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

DCT (cont.)

Inverse Transform:

$$\begin{tikzcd} B = \left[B_{pq}\right]_M^N \arrow[r] & \boxed{\text{IDCT}} \arrow[r] & A = \left[A_{mn}\right]_M^N \\ \text{MxN block of DCT coefficients} \arrow[u] & & \text{MxN block of grayscale pixels} \arrow[u] \end{tikzcd}$$

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

DCT (cont.)

DCT is a lossless tranformation:

$$IDCT(DCT(A)) = A$$

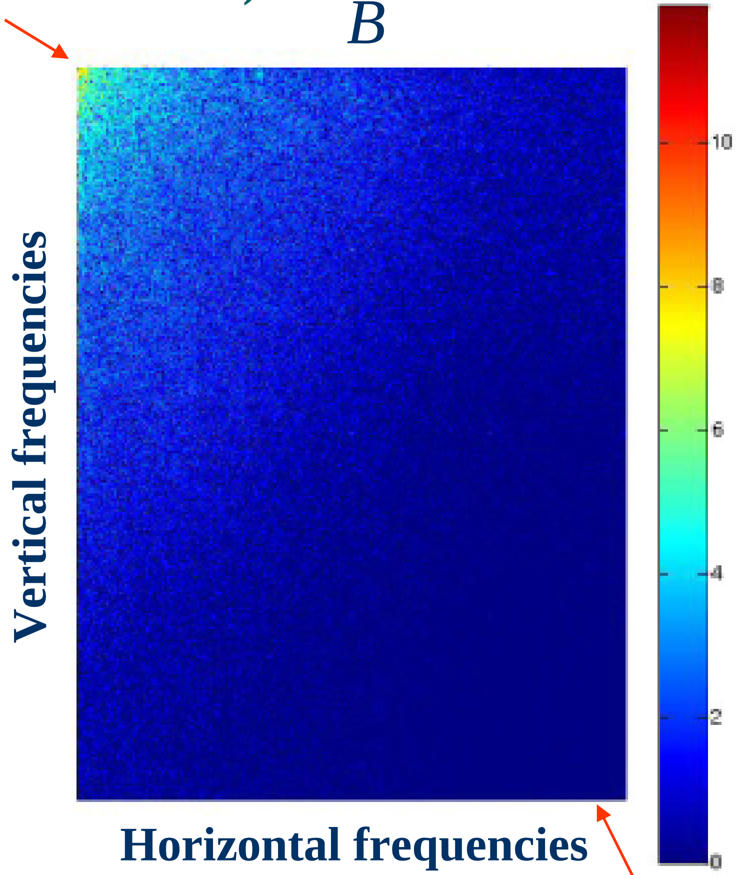
$$\begin{tikzcd}
A = \left[A_{mn}\right]_M^N \arrow[r] & \boxed{\textbf{DCT}} \arrow[r, "B = \left[B_{pq}\right]_M^N"] & \boxed{\textbf{IDCT}} \arrow[r] & A = \left[A_{mn}\right]_M^N
\end{tikzcd}$$

DCT (cont.)

Large coefficient values are concentrated in the upper left corner (smaller indices)

A

B

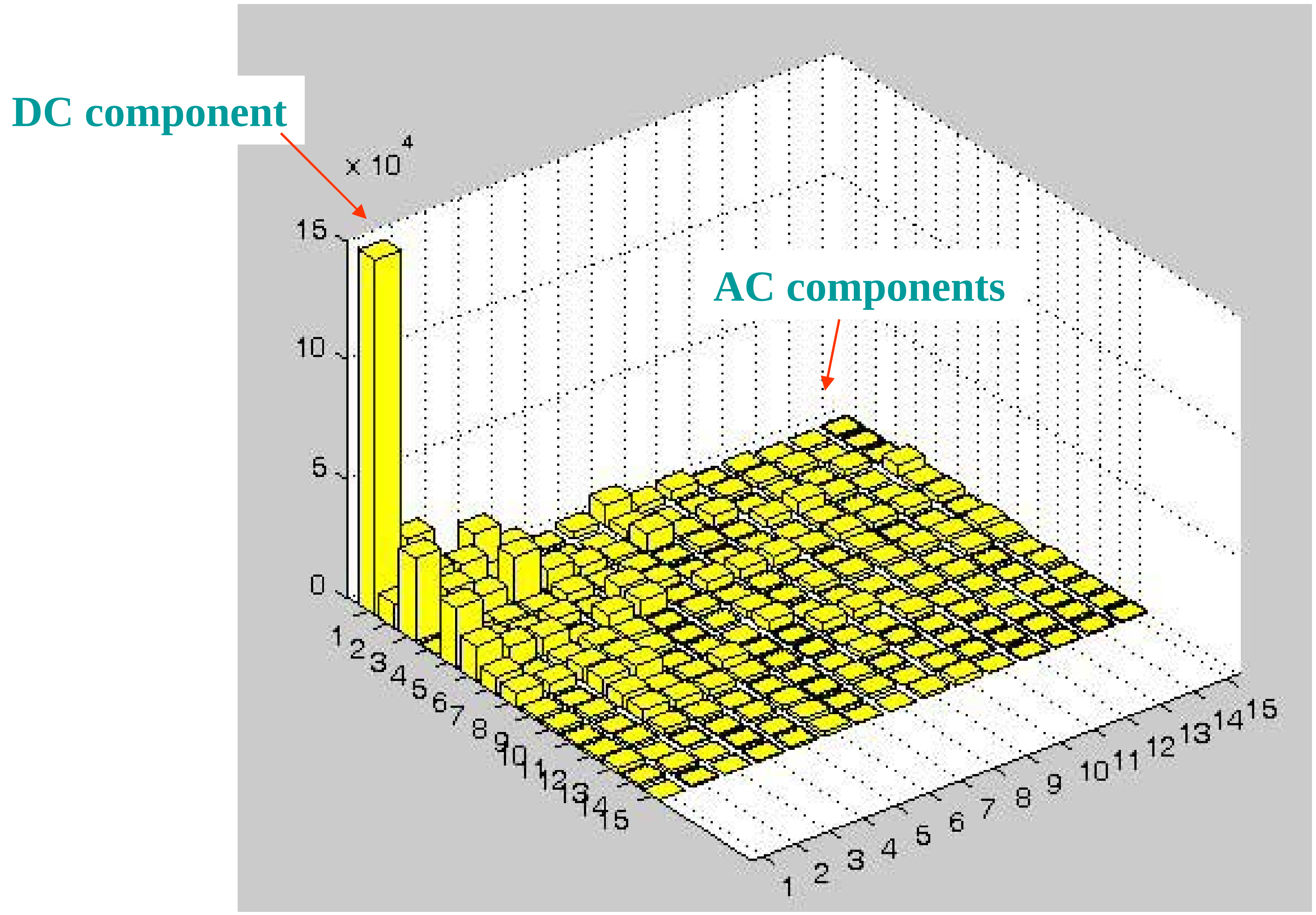


Very small or zero coefficient values are concentrated in the lower right corner (larger indices)

Notice: The plot above is obtained by using the log (abs(B)), where the elements of B smaller than 1 were set to 1, otherwise the plot would be entirely blue, since all larger elements would be concentrated at very small area in the upper left corner.

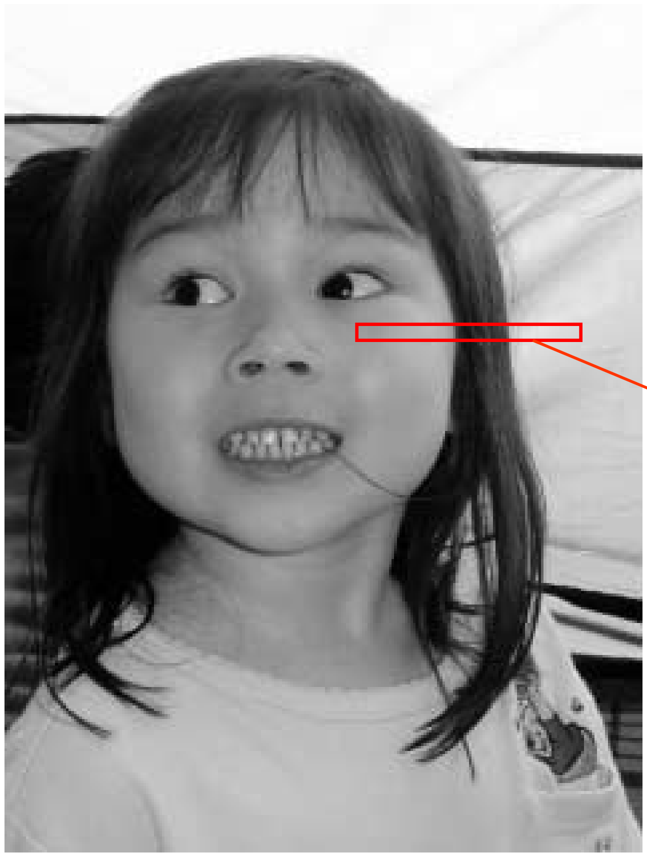
DCT (cont.)

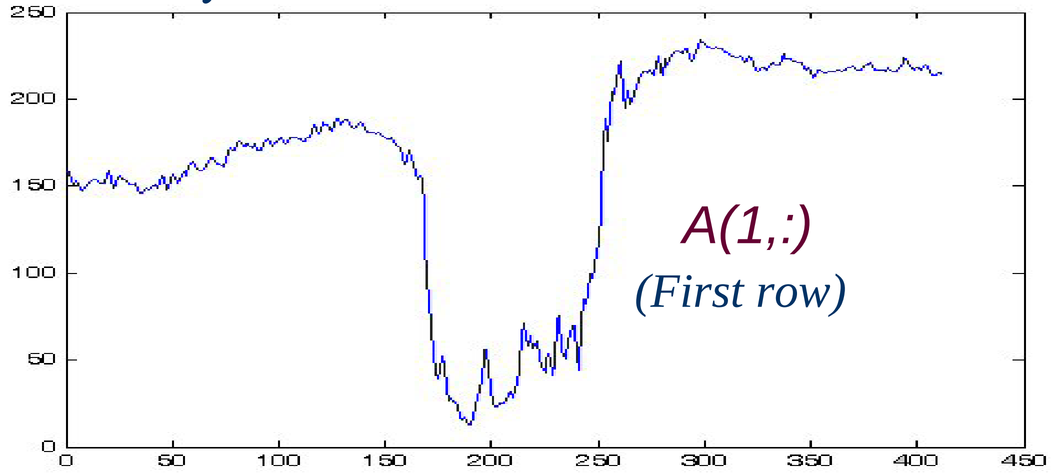
This bar graph shows the absolute values of 15 x 15 top left elements of *B*

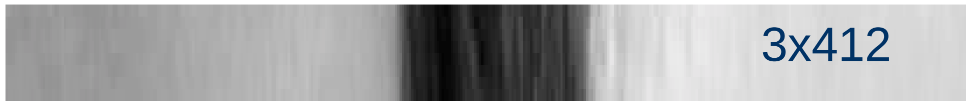


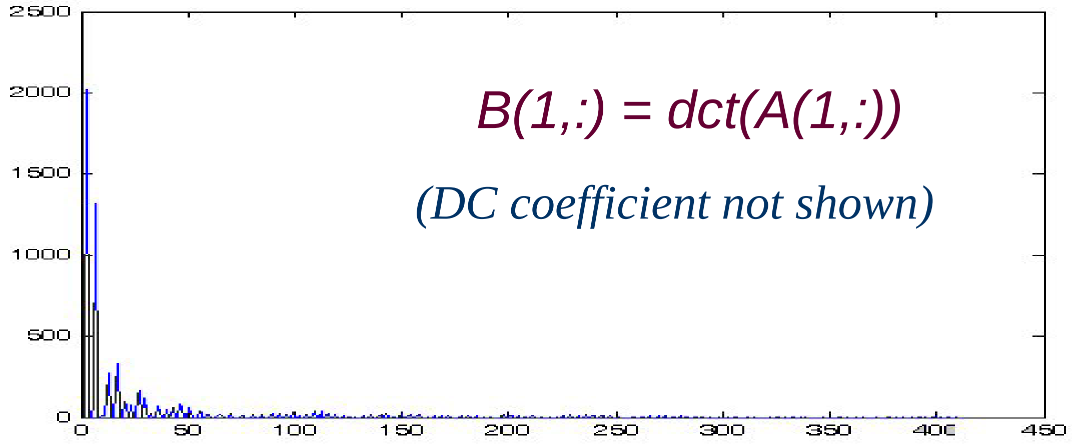
DCT (cont.)

This slide illustrates the benefit of DCT: typically an image consists of many smooth areas and fewer edges. Edges have sharp change of pixel values. The low-frequency DCT coefficients represent the smooth areas of the original image, while the high-frequency coefficients represent the edges. The excerpt from the image below contains two edges, therefore the high-frequency coefficients are very small.









DCT (cont.)

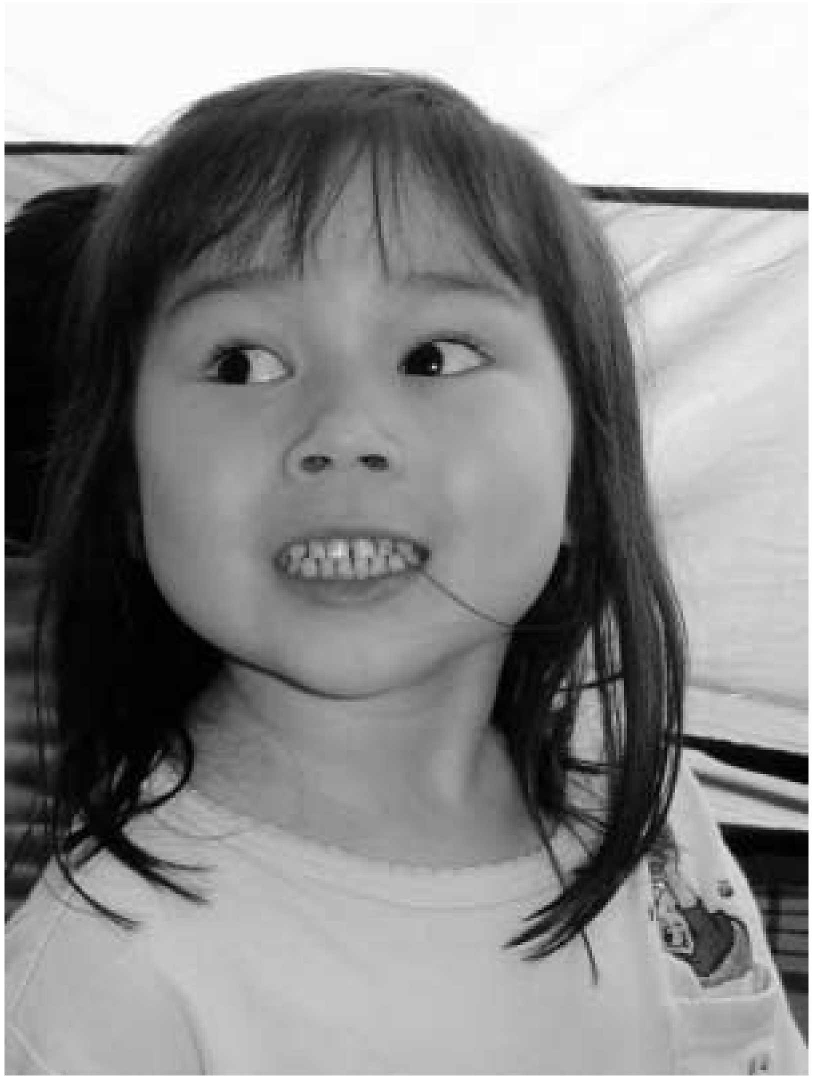
Since the DCT matrix has many small coefficient values, they can be set to zero, which will reduce the number of the non-zero values. This can be achieved with quantization (shown later). The zeroed elements can be eliminated with run-length encoding (see later), thus reducing the number of bits to be transferred, i.e. the required transmission bit rate.

The following four slides illustrate the effect of removing the small coefficients from the DCT matrix. The values of the reduction factor is defined as follows:

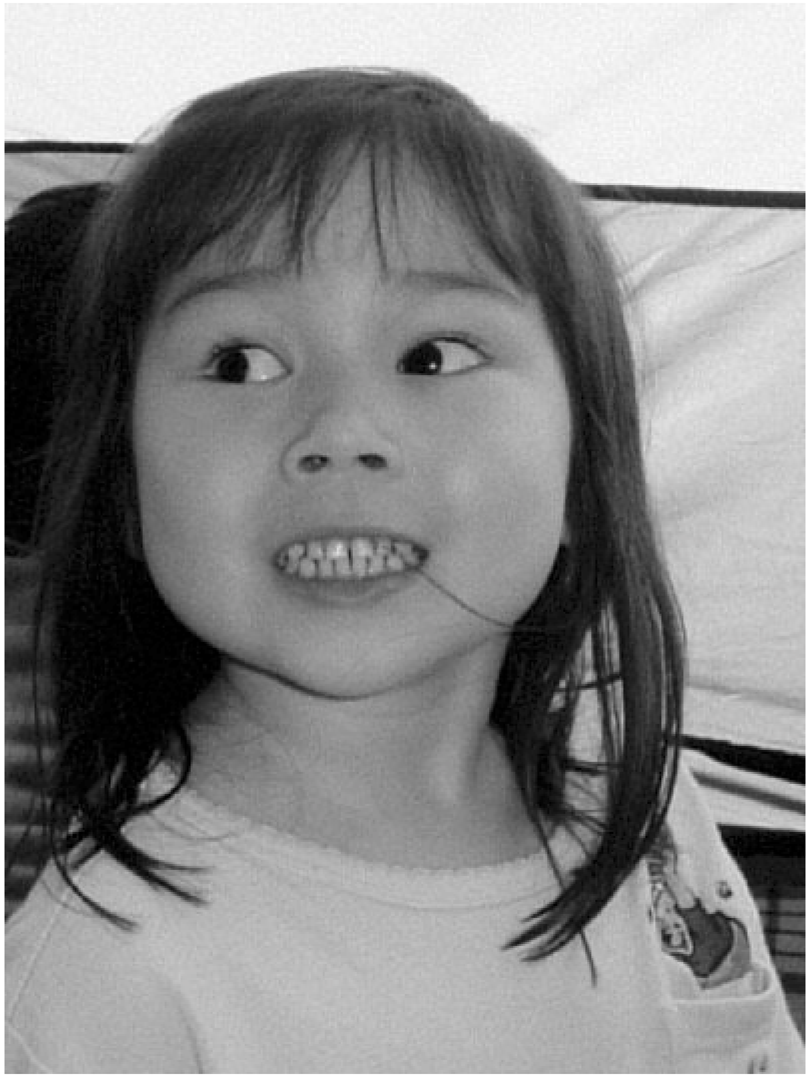
$$rf = \frac{\text{Number of pixels in original image } (N \times M)}{\text{Number of non-zero coefficients in DCT matrix}}$$

DCT (cont.)

A



IDCT(reduced(DCT(A)))

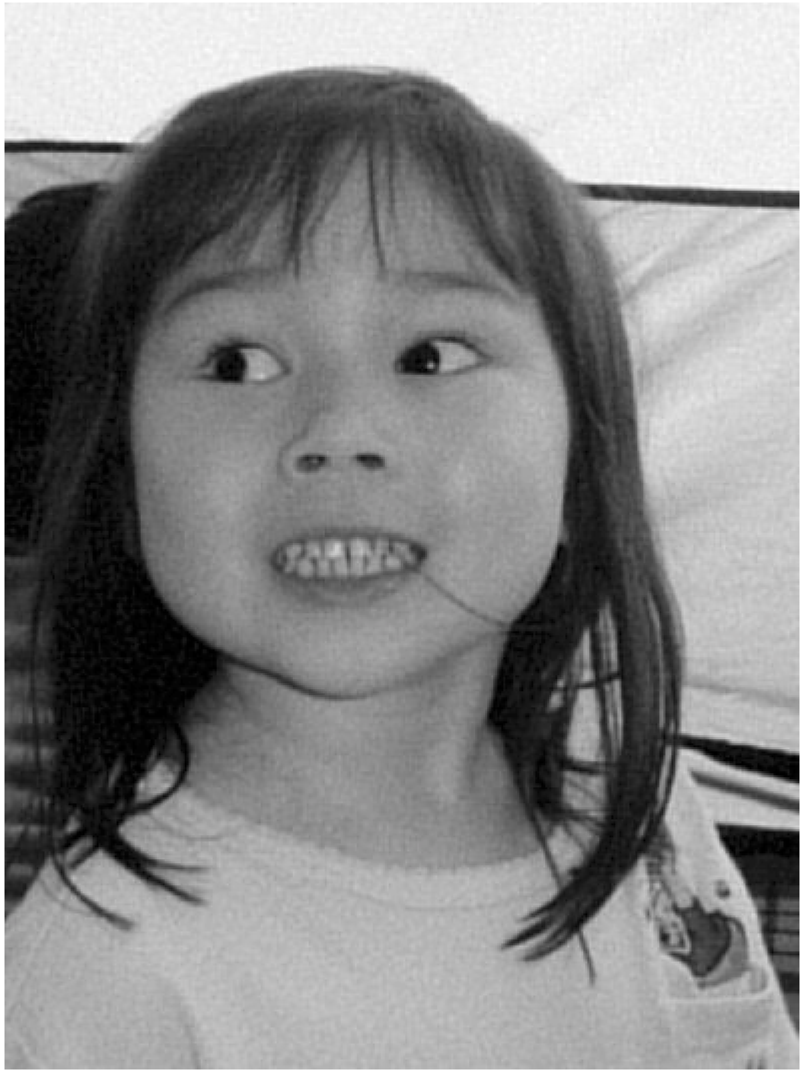


rf=37

DCT (cont.)

A

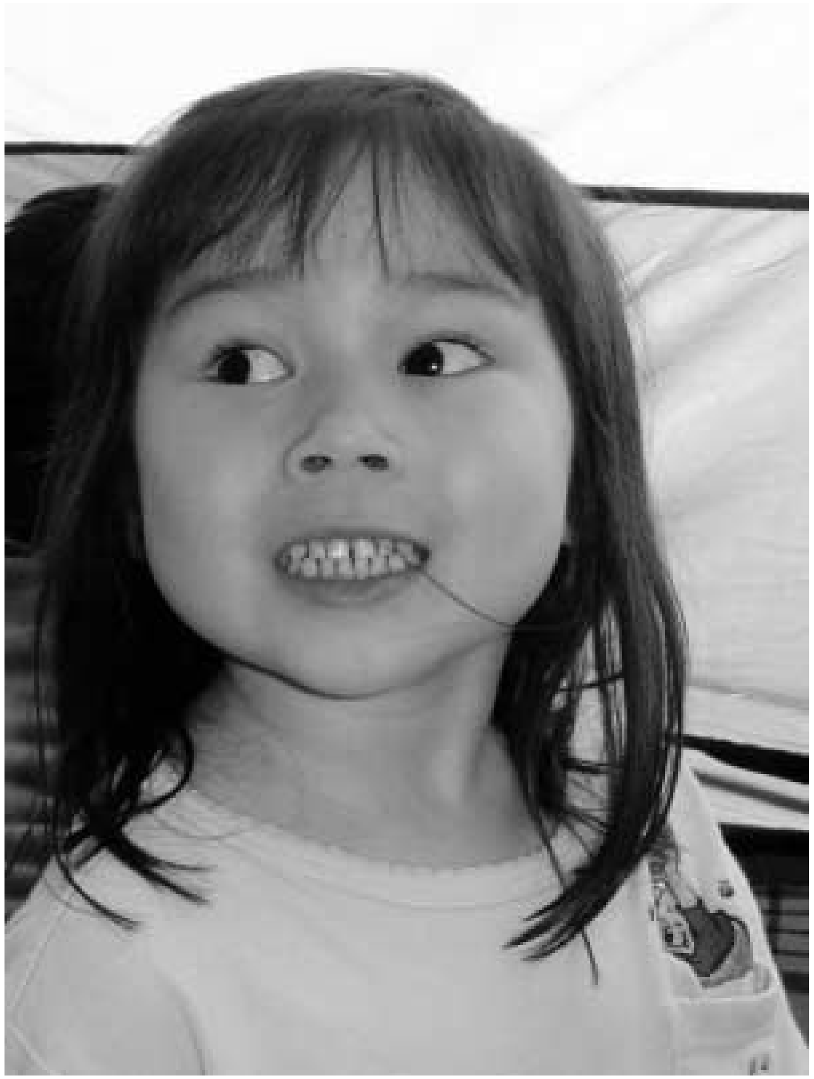
$IDCT(reduced(DCT(A)))$



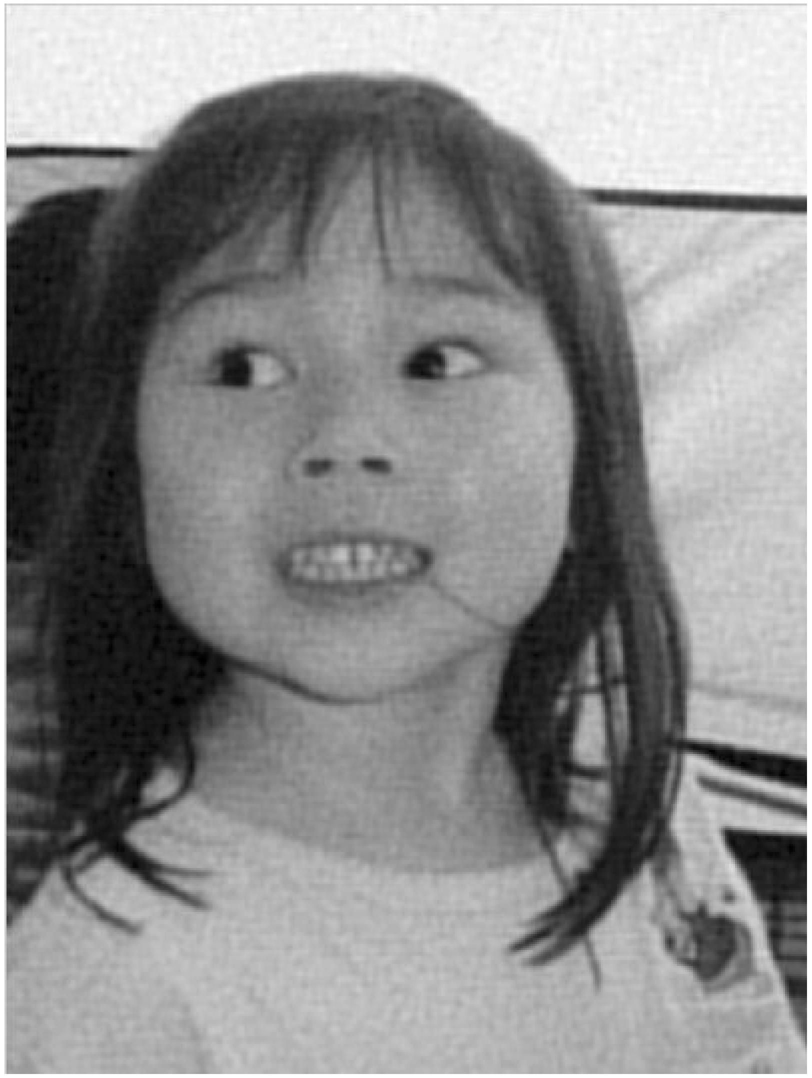
cf=74

DCT (cont.)

A



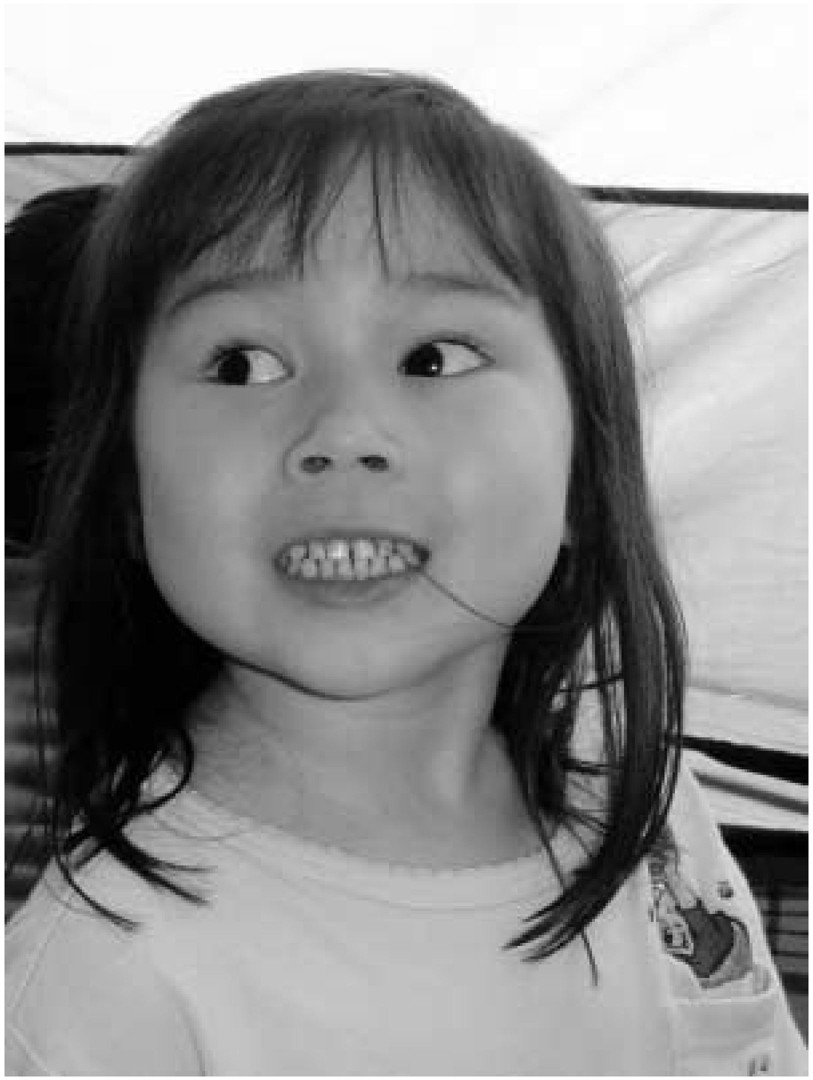
IDCT(red(DCT(A)))



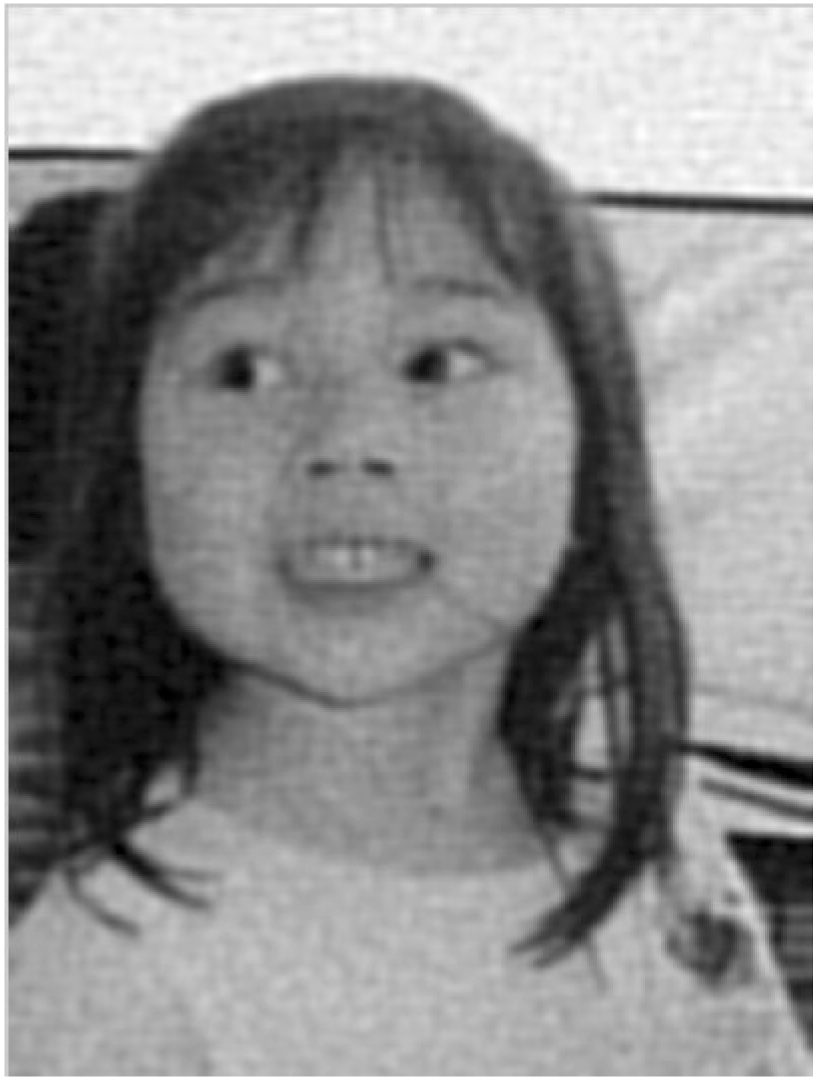
rf=191

DCT (cont.)

A



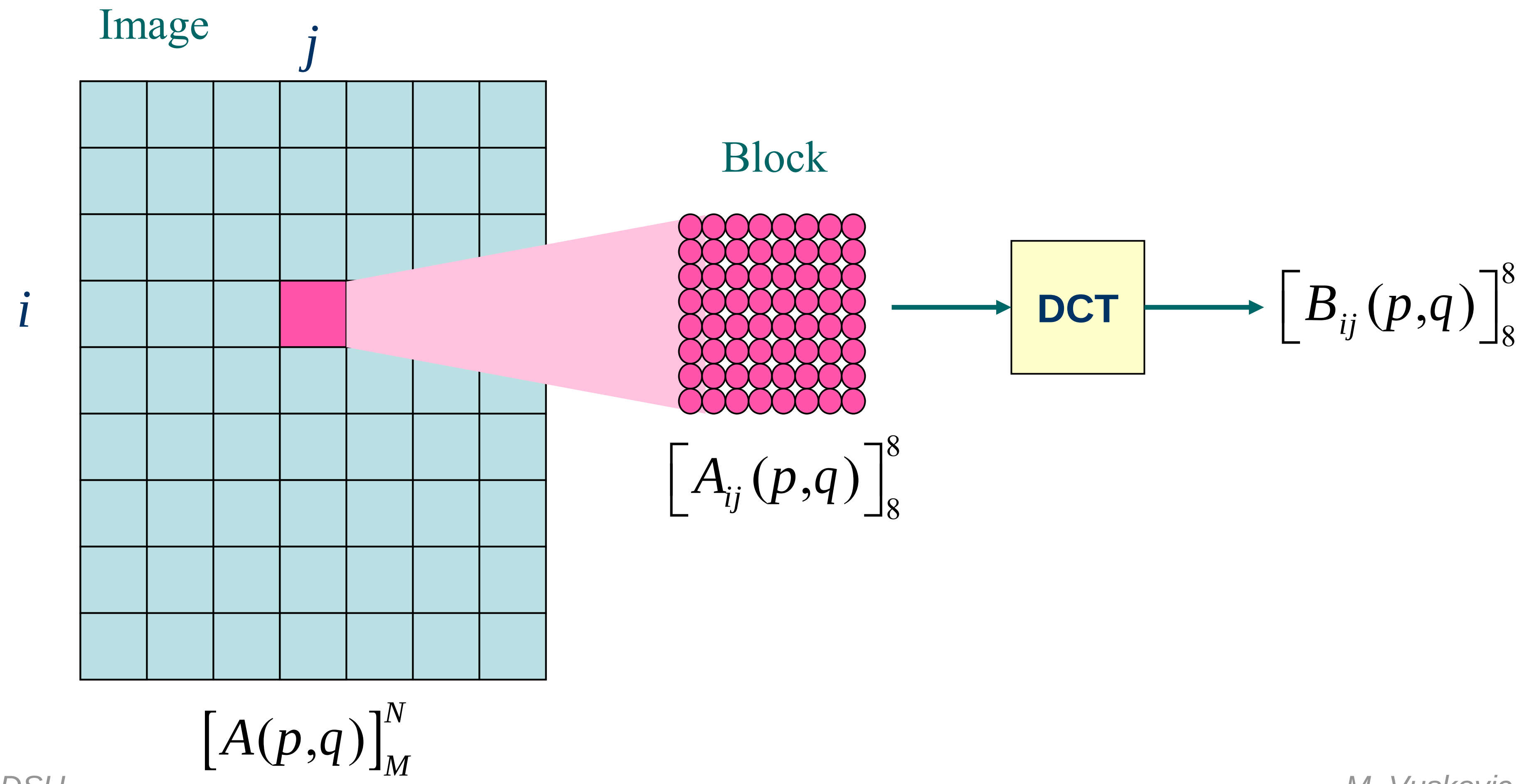
IDCT(reduced(DCT(A)))



rf=473

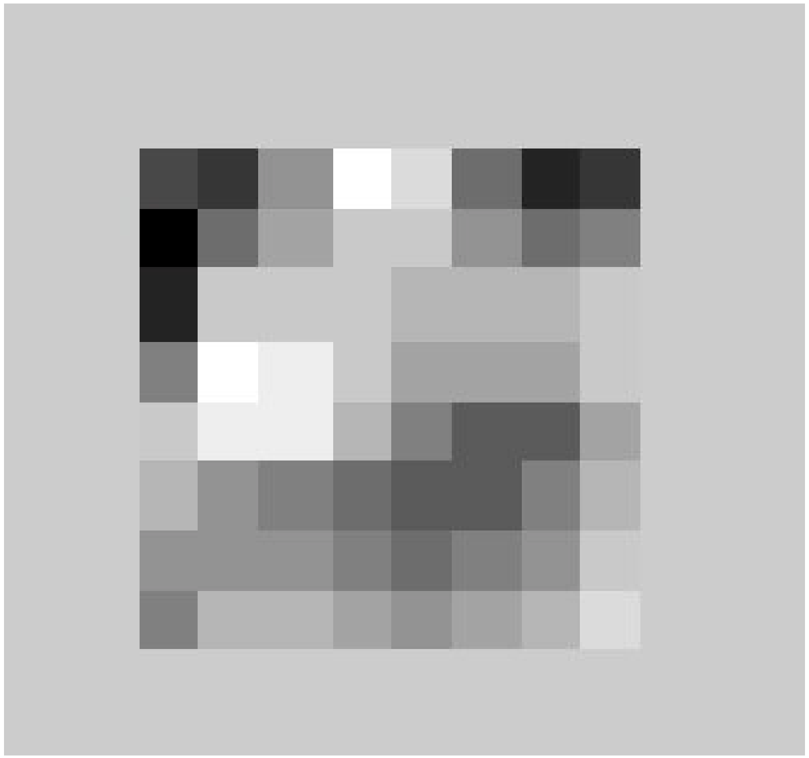
DCT (cont.)

In standards to be discussed later, the DCT is not applied to the entire image, it is rather applied to smaller blocks of the image. The standard block size is 8 by 8 pixels. For that purpose the image is divided into non overlapping 8 x 8 blocks, and the DCT is applied to each block independently.

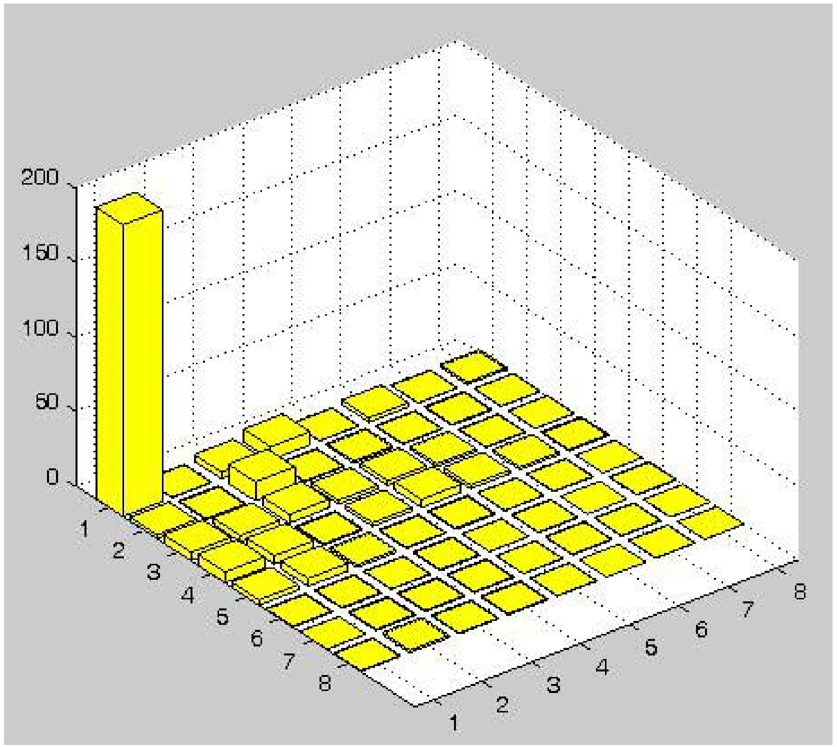


DCT (cont.)

Block A(300:308, 200:208)



DCT of A(300:308, 200:208)



8 x 8 DCT

Quantization

Quantization is a procedure used to encode an analog value into a binary number. The number of bits required to represent the binary encoded value depends on the number of the quantization levels.

Continuous signal

L-level quantizer

Quantized signal

$$\begin{tikzcd} s \in [0, s_{\max}] \arrow[r] & \boxed{Q(s)} \arrow[r] & s' \in \{0,1,2,\ldots,L-1\} \end{tikzcd}$$

log₂(L) bits required for binary representation

$$s' = Q(s) = \begin{cases} 0 & 0 \le s < h \\ 1 & h \le s < 2h \\ \ldots & \ldots\ldots\ldots \\ L-1 & (L-1)h \le s < s_{\max} \end{cases} = round\left(\frac{s}{h}\right), \quad h = \frac{s_{\max}}{L}$$

Quantization step size (quantum)

This quantizer is called uniform quantizer because it has constant step size independent of the magnitude of the input signal.

Quantization (cont.)

Reverse of quantization is called dequantization.

$$\begin{tikzcd}
\text{Quantized signal} & L\text{-level dequantizer} & \text{Reconstructed signal} \\
s' \in \{0,1,2,\ldots,L-1\} \arrow[r] & \boxed{Q^{-1}(s)} \arrow[r] & \hat{s} \in \{0,h,2h,\ldots,(L-1)h\}
\end{tikzcd}$$

$$\hat{s} = s'h + \frac{h}{2}$$

Quantization (cont.)

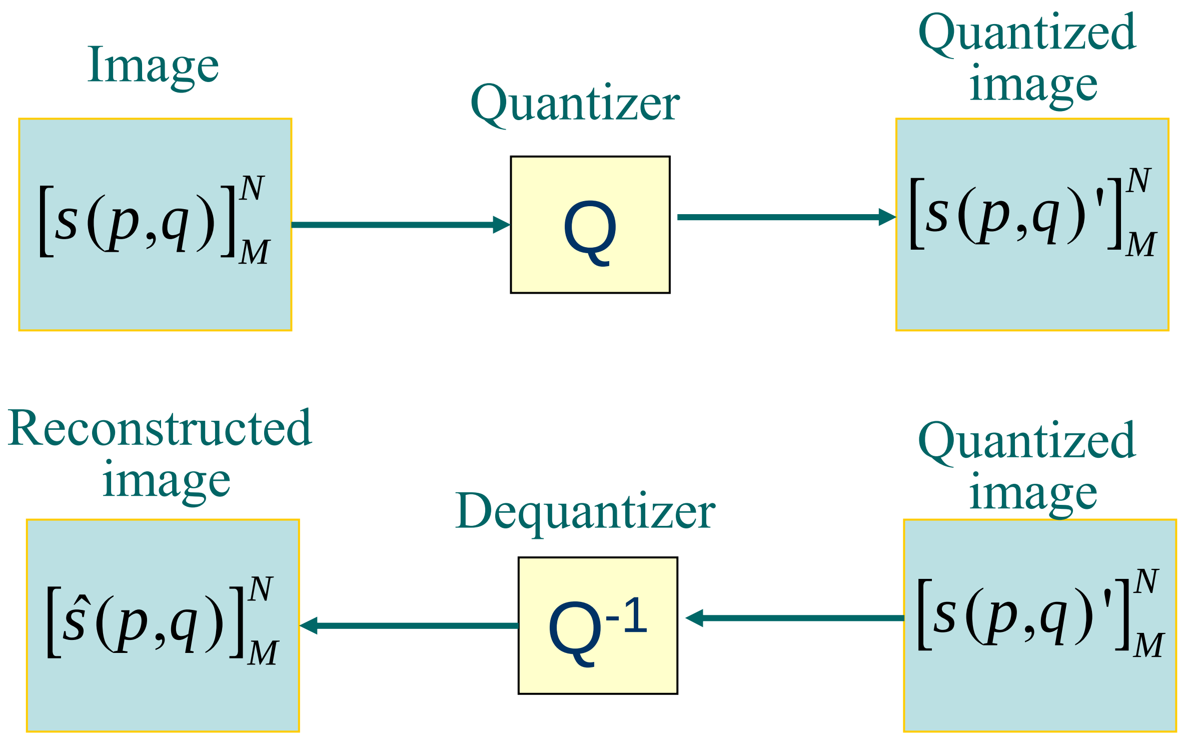
Comments:

- The quantization can be applied to a discrete signal to.
  For example signal $\{0, \Delta x, 2\Delta x, \ldots, (N\text{-}1)\Delta x\}$ can be quantized to $\{0, h, 2h, \ldots, (L\text{-}1)h\}$ where $\Delta x = S_{max}/N$ and $L<N$.
- Quantization can be applied to signals that have negative values.
- Quantization reduces the number of bits by which the signal is encoded. Therefore the quantization compresses the signal. This kind of compression is called lossy compression meaning that some amount of information is lost, i.e. the reconstructed signal is not equal to the original signal.
- The ratio of the number of bits of the original signal and the number of bits used in transmission (or storage) after the compression is called compression factor.
- The compression factor and the quality of the signal are in inverse relationship: generally the higher compression factor leads to lover signal quality.

Quantization (cont.)

Quantization of Images

In an image block every pixel is quantized independently



Quantization (cont.)

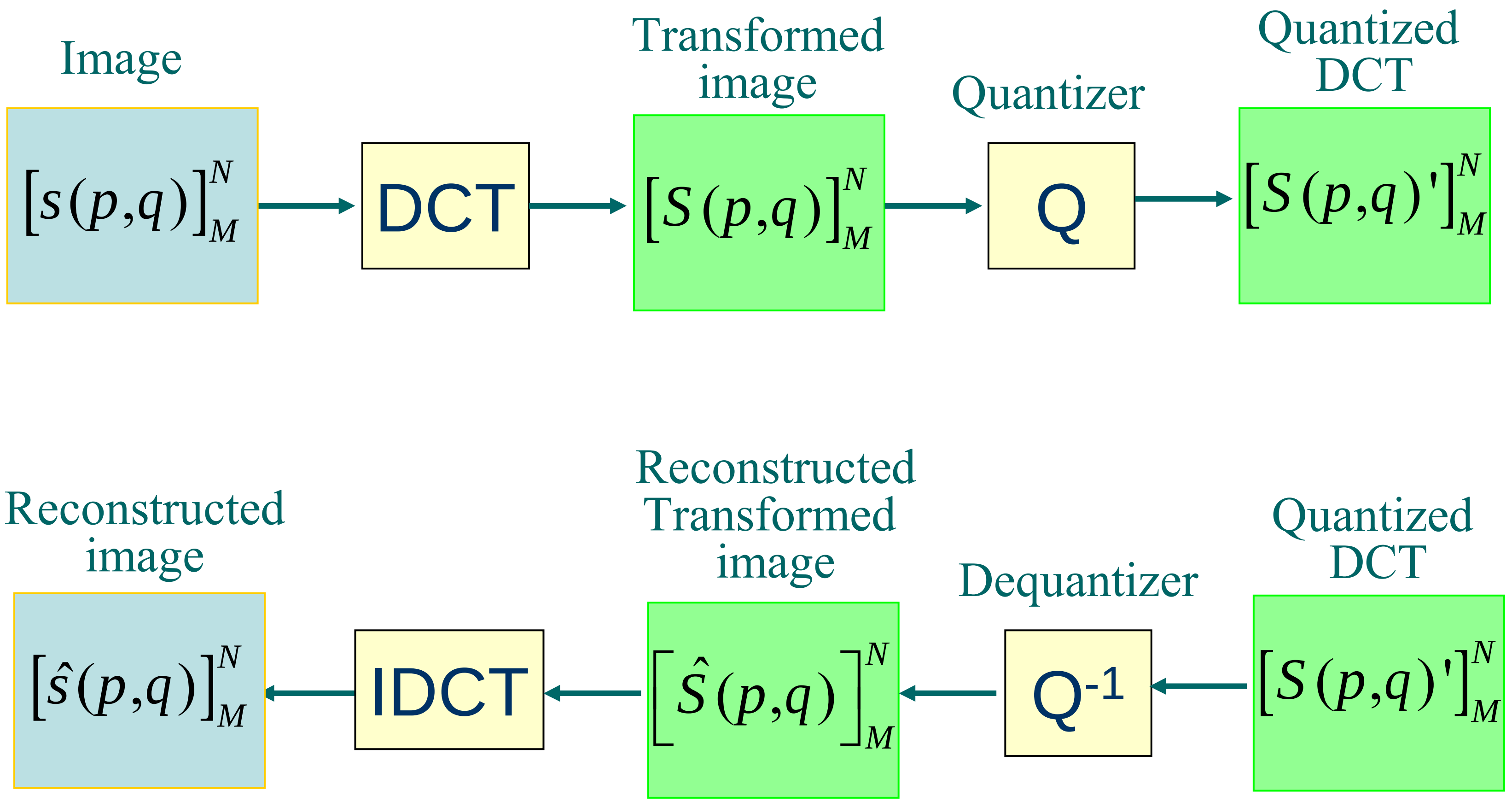
Comments:

- Quantization of an image does not reduce the number of pixels in the image, however the number of bits needed to encode each pixel is decreased. (Decreasing of number of pixels in an image is called down sampling, will be discussed later)
- Quantization of an image increases the number of pixels that have identical value. This gives a chance for further lossless compression by entropy coding as shown later.
- If the pixels with equal value are adjacent another lossless compression technique can be applied, called Run-Length Encoding (see later).

Quantization (cont.)

Quantization of Images after DCT

The DCT transformed images can also be quantized



Quantization (cont.)

Comment:

Quantization of DCT transformed images takes a full advantage of the fact that DCT concentrates information in the upper left corner of the matrix. Consequently the insignificant coefficients of DCT are nullified after quantization. The nullified DCT coefficients are generally adjacent which contributes to higher compression factor after removing the coefficients with run-length encoding.

Quantization (cont.)

Quantization of DCT Blocks

The DCT transformed images can also be quantized:

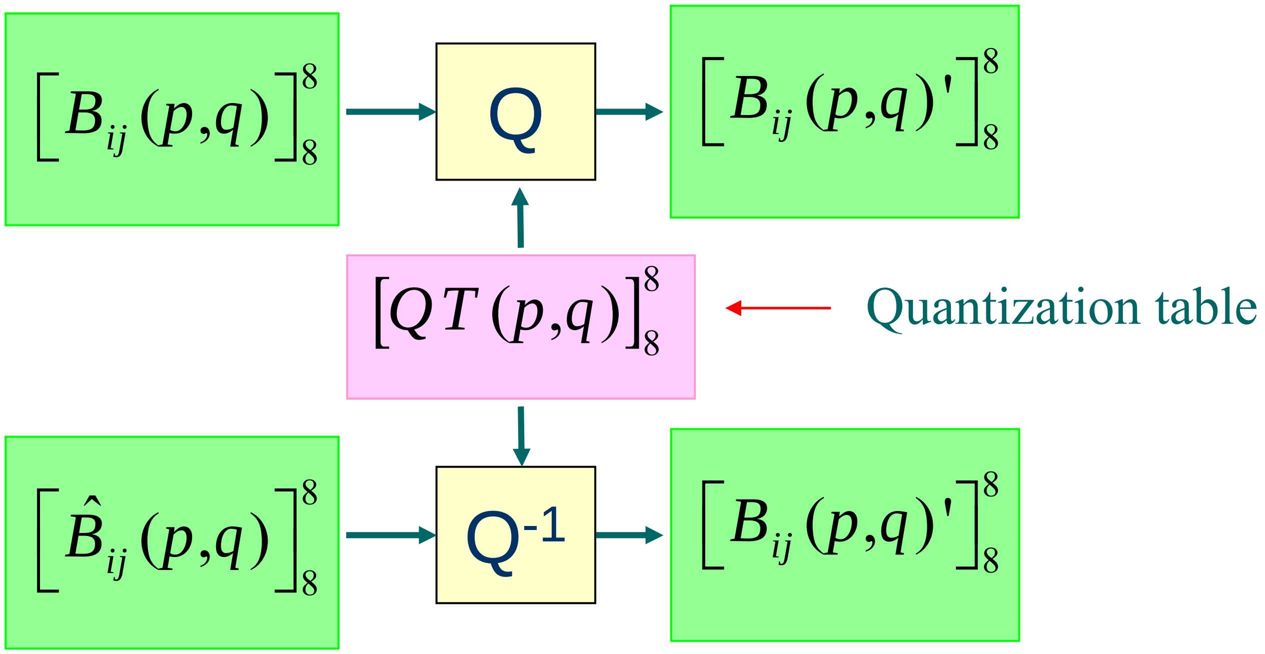
$$\begin{tikzcd}
\text{Image block} & & \text{Transformed image block} & & \text{Quantized DCT block} \\
\left[A_{ij}(p,q)\right]_8^8 \arrow[r] & \boxed{\text{DCT}} \arrow[r] & \left[B_{ij}(p,q)\right]_8^8 \arrow[r] & \boxed{\text{Q}} \arrow[r] & \left[B_{ij}(p,q)'\right]_8^8
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Reconstructed image block} & & \text{Reconstructed transformed image block} & & \text{Quantized DCT block} \\
\left[\hat{A}_{ij}(p,q)\right]_8^8 & \boxed{\text{IDCT}} \arrow[l] & \left[\hat{B}_{ij}(p,q)\right]_8^8 \arrow[l] & \boxed{\text{Q}^{-1}} \arrow[l] & \left[B_{ij}(p,q)'\right]_8^8 \arrow[l]
\end{tikzcd}$$

Quantization (cont.)

Quantization Tables

Since different coefficients in DCT have different weight in the amount of information they carry, they can be quantized with different quantization step sizes. The step sizes can be stored in a quantization table, or quantization matrix.

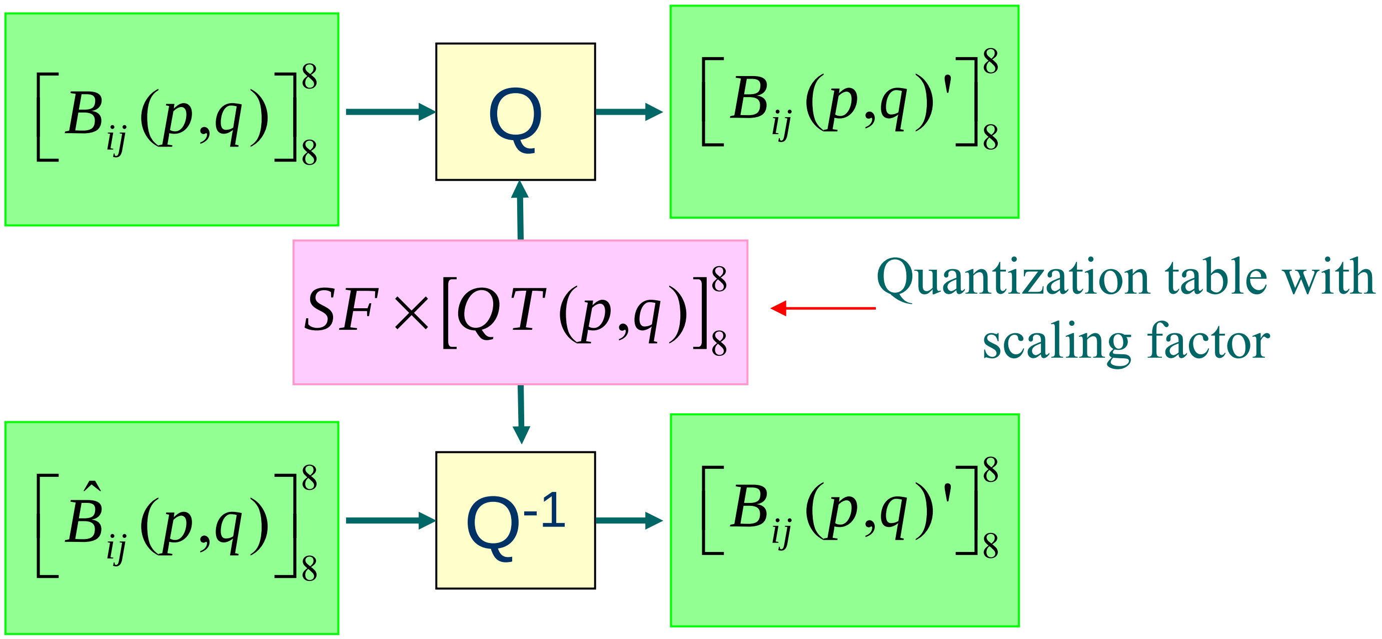


$$B_{ij}(p,q)' = round\left(\frac{B_{ij}(p,q)}{DT(p,q)}\right) \qquad \hat{B}_{ij}(p,q) = B_{ij}(p,q)'QT(p,q) + \frac{QT(p,q)}{2}$$

Quantization (cont.)

Quantization Table with Scaling Factor

Elements of a quantization table can be multiplied with a scaling factor (*CF*) in order to control the degree of compression and the resulting output bit rate.


$[B_{ij}(p,q)]_8^8$
Q
$[B_{ij}(p,q)']_8^8$
$SF \times [QT(p,q)]_8^8$
Quantization table with scaling factor
$[\hat{B}_{ij}(p,q)]_8^8$
Q^{-1}
$[B_{ij}(p,q)']_8^8$

$$B_{ij}(p,q)' = round\left(\frac{B_{ij}(p,q)}{SF \times DT(p,q)}\right)$$

Quantization (cont.)

Quantization Tables (cont.)

Quantization tables are standardised. JPEG defines the following quantization table:

DC component

AC components

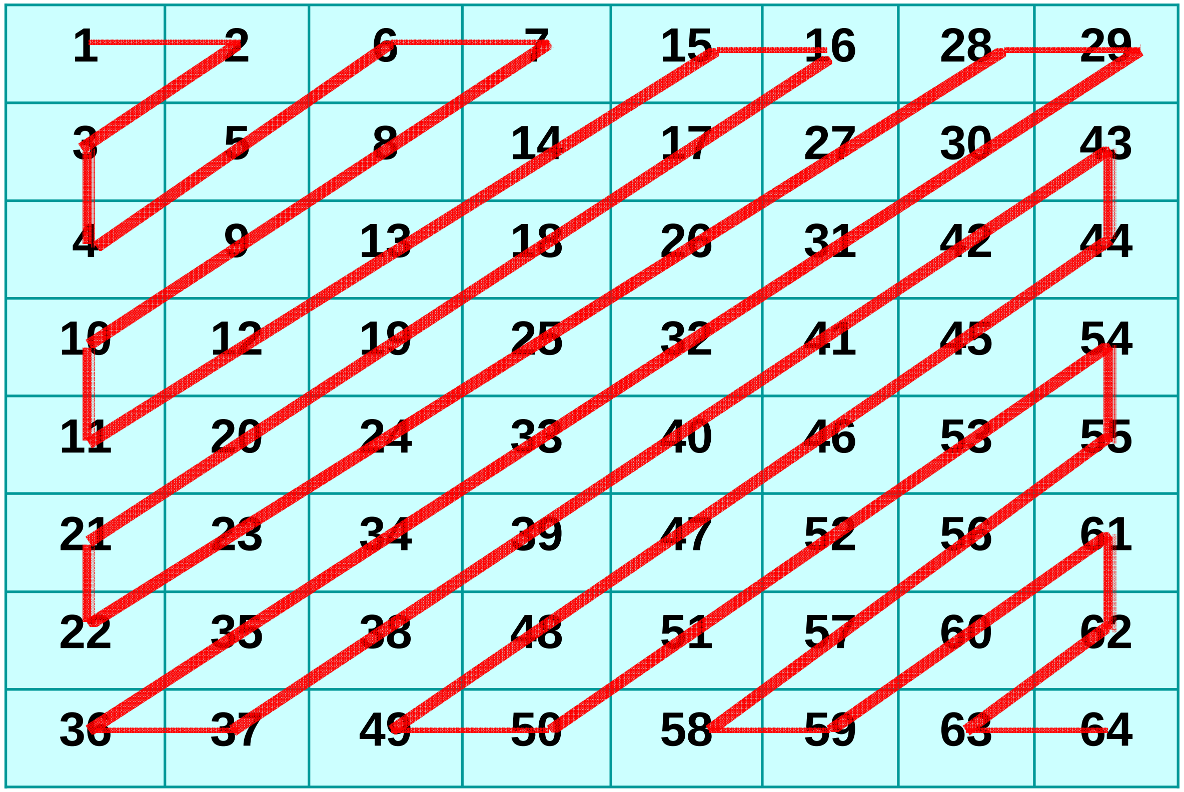
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	139	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table contains quantization step sizes. It is designed based on the visual sensitivity of human eye to different frequency components. Typically low-frequency coefficients are assigned smaller step sizes because the human eye is more sensitive to changes in low-frequency components.

The table can be scaled to achieve desired bit rate (see later).

Zig-Zag Scan

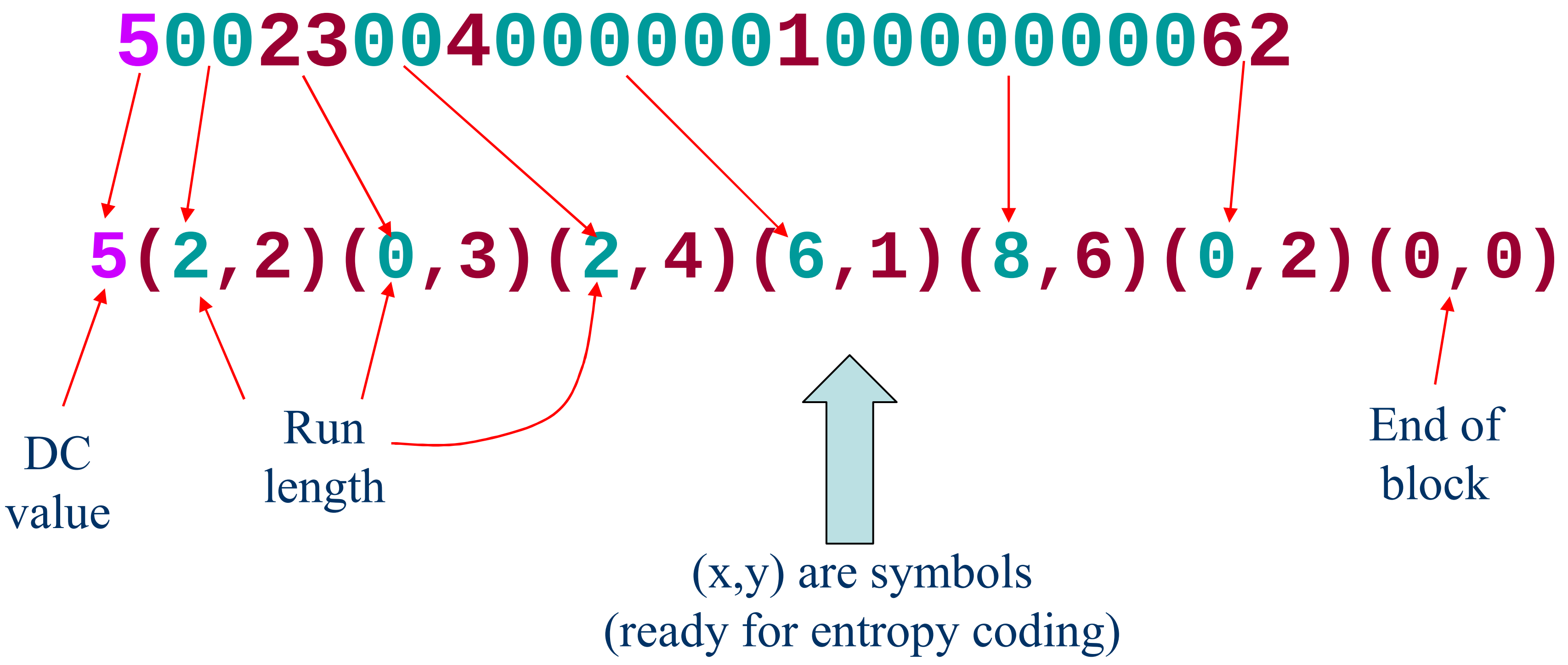
The result of quantization is an 8 x 8 matrix. In order to obtain a bit stream suitable for run-length encoding and entropy coding, and later for transmission, the 2D matrix needs to be mapped into a 1D array. This is done by scanning the matrix elements in some order. JPEG uses the zig-zag order, which tends to group low-frequency coefficients towards the beginning of the resulting 1D array.



1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

Run-Length Encoding (RLE)

After quantization, specially after quantization of DCT coefficients, there are many repeated zeros in the zig-zag bit stream. The non-zero elements can be paired with the number of preceeding zeros. The DC value goes first unpaired. The pairs *(run-length, non-zero element)* are called symbols. The symbols are fed to the entropy coder.



Entropy Coding

A message can be considered as a string of symbols $c_1\, c_2\, c_3\, c_4 \ldots$ where each symbol c_k takes values from a finite set of symbols $\{S_1, S_2, \ldots, S_M\}$.

Examples:

Textual message (string of printable ASCII characters) takes values from the set of 95 symbols:

{Space, !, “, . . . 0, 1, 2, . . . , A, B, C, . . .,|}, ~}

Audio or video message after RLE:

5 (2,2) (0,3) (2,4) (6,1) (8,6) (0,2) (0,0)

Entropy Coding (cont.)

The question is how many bits we need to encode a message?
Theoretically (Shannon) the minimal required number of bits is equal to:

$$b_i = \log_2\left(\frac{1}{p_i}\right)$$

where:

b_i – number of bits to encode the symbol S_i
p_i – probability that a symbol c_k has value S_i, i.e. $Prob(c_k = S_i)$

The probability can be estimated from the symbol frequency count:

$$p_i \approx \frac{n_i}{\sum_j n_j}$$

where: n_i is the number of occurrences of symbol S_i in the message, or in the relevant class of messages. This estimate is more accurate if the message is longer.

Entropy Coding (cont.)

The name “entropy encoding” comes from the fact that the information entropy of a source that generates messages is defined as:

$$H(S) = \sum_j p_j \log_2 \left(\frac{1}{p_j} \right)$$

One popular entropy encoding technique is Huffman coding which uses symbol codes with variable bit lengths.

Huffman coding represents a lossless compression algorithm (i.e. the decoded message is identical to the original message).

Huffman Coding

Suppose we generate messages (character strings) which use symbols from the set of 7 characters:

`{A,B,C,D,E,F,G}`

Since the set has 7 characters we can encode each character wit 3 bits:

A	–	001	E	–	101
B	–	010	F	–	110
C	–	011	G	–	111
D	-	100			

Suppose a specific message:

BABBAGE

Then the binary encoded message is:

$$\underbrace{010}_{B}\underbrace{001}_{A}\underbrace{010}_{B}\underbrace{010}_{B}\underbrace{001}_{A}\underbrace{111}_{G}\underbrace{101}_{E}$$

The string contains 21 bits

Huffman Coding (cont.)

Now, instead of straight forward fixed-length binary codes we use Huffman codes for the characters that appear in the message:

$$\left.\begin{array}{l} \texttt{A - 00} \\ \texttt{B - 1} \\ \texttt{E - 010} \\ \texttt{G - 011} \end{array}\right\} \text{Codebook}$$

The character B is the most frequent character in this message, therefore it uses the smallest numbers of bits (just one bit = 1)

The message is now:

$$\underbrace{\texttt{1}}_{\texttt{B}}\ \underbrace{\texttt{00}}_{\texttt{A}}\ \underbrace{\texttt{1}}_{\texttt{B}}\ \underbrace{\texttt{1}}_{\texttt{B}}\ \underbrace{\texttt{00}}_{\texttt{A}}\ \underbrace{\texttt{011}}_{\texttt{G}}\ \underbrace{\texttt{010}}_{\texttt{E}}$$

The string contains 13 bits, which is only 62% of the previously encoded message.

This encoding technique provides unambiguous and easy decoding, because the encoding has unique prefix property: no code is a prefix to any other code.

Huffman Coding (cont.)

Deriving the Huffman codebook

1. Find the frequency count for all characters that appear in the message:

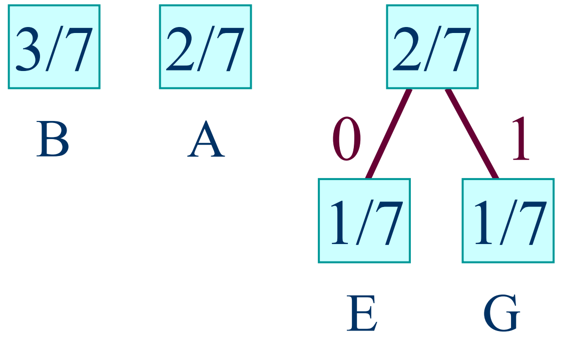
Symbol	*Count*	*Probability*
A	2	2/7
B	3	3/7
E	1	1/7
G	1	1/7
Total	7	1

2. Sort the symbols according to their probabilities:

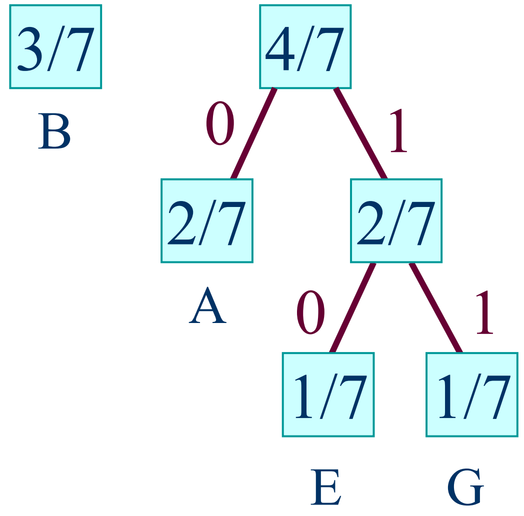
3/7	2/7	1/7	1/7
B	A	E	G

Huffman Coding (cont.)

3. Join the two rightmost nodes and form a new node. Label arks with 0 and 1:

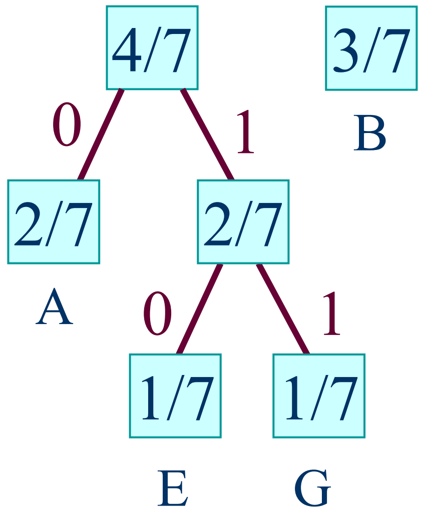


4. repeat step 3:

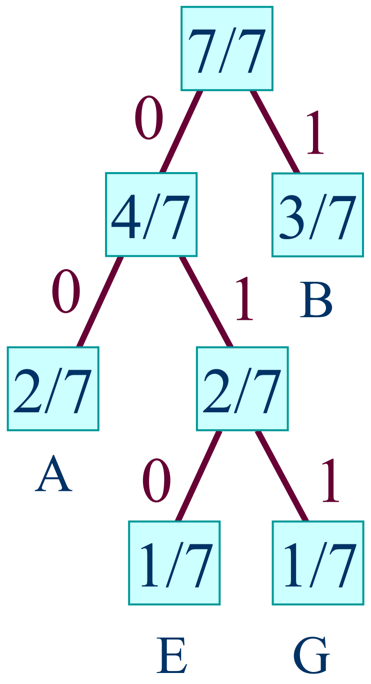


Huffman Coding (cont.)

5. Resort the new nodes:



6. Repeat step 3:



7. Read leafs in terms of 0s and 1s:

A = 00

E = 010

G = 011

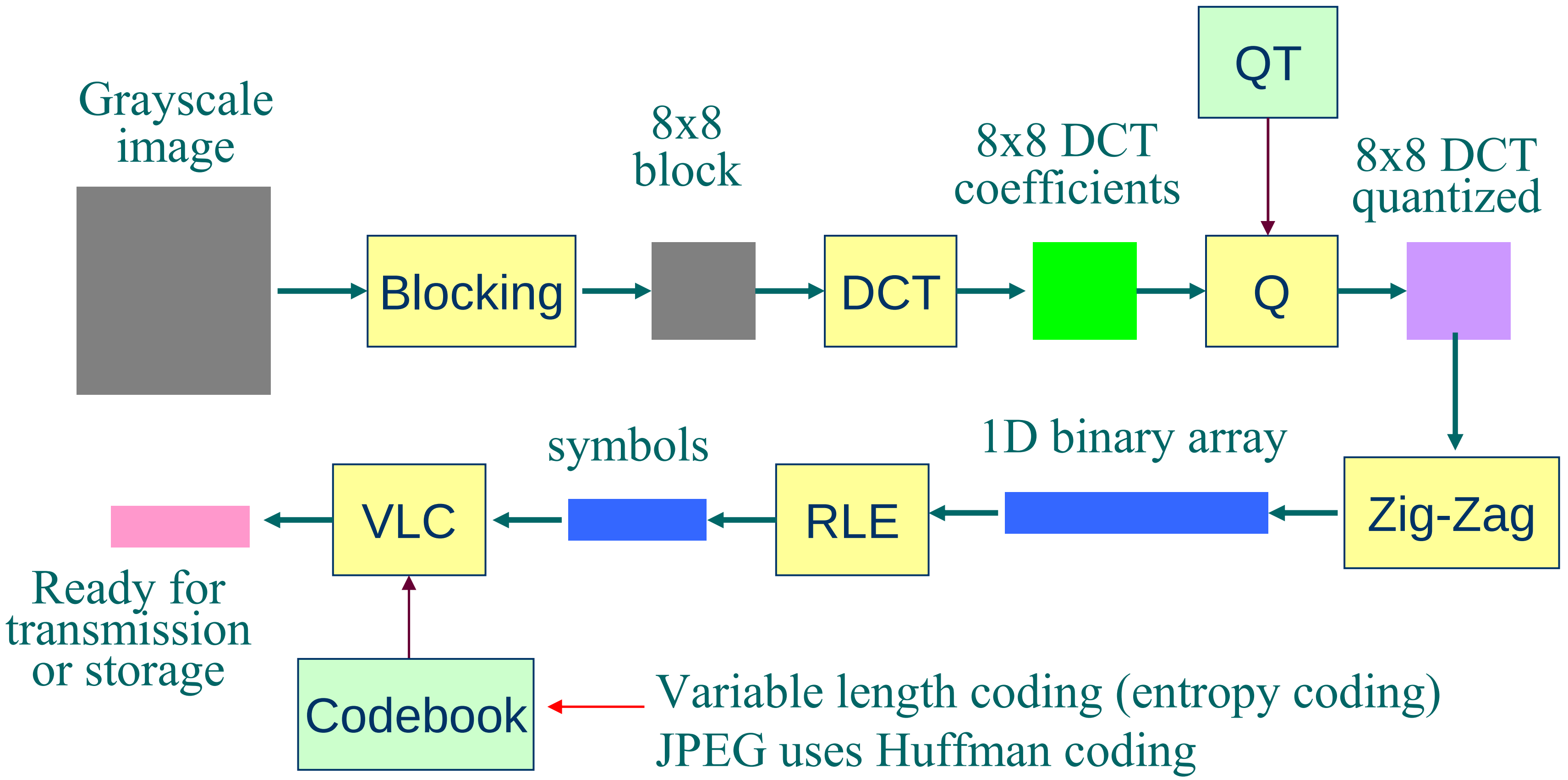
B = 1

Huffman Coding (cont.)

In summary, the Huffman compression consists of the following:

1. Generate string of symbols (the longer the better)
2. Derive the symbol statistics (frequency counts)
3. Derive the Huffman codebook for the statistics
4. Encode the message
5. Send the codebook to the receiver
6. Send the message to the receiver (can be done in several packets)

JPEG Grayscale Images



JPEG = Joint Photographic Expert Group
Voted as international standard in 1992
Works for color images (see later)

Color Images

Color images have three color components (planes). Each component is treated in a similar way as grayscale images. Two major color spaces are used in digital video: RGB and YUV

RGB

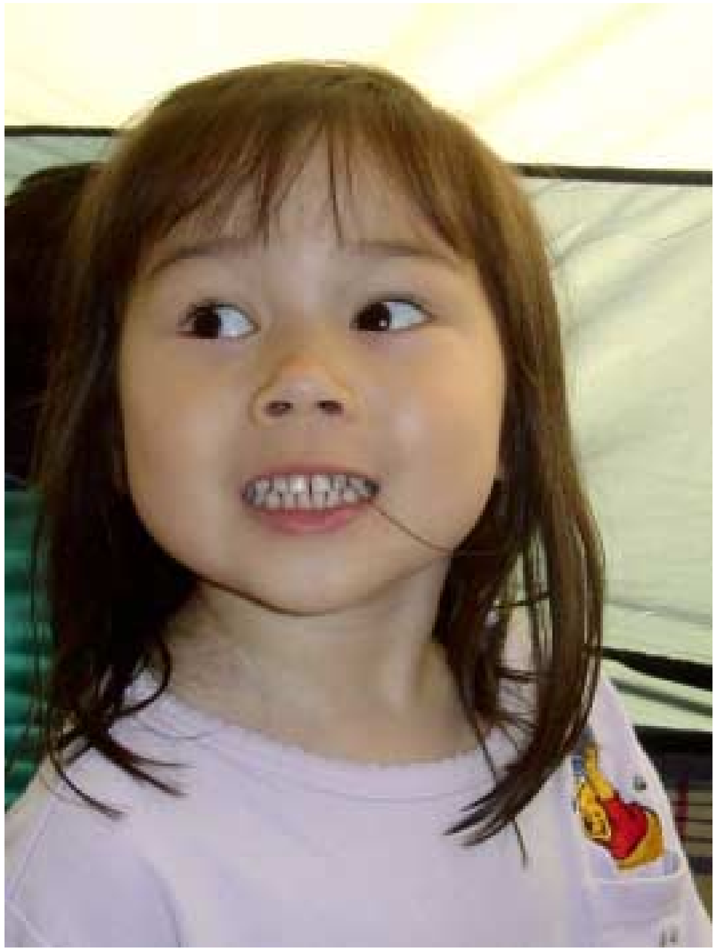
The picture has three components: R (red), G (green) and B (blue).
Each component has 256 levels (this is how much human eye can see), therefore each component is encoded with 8 bits, total for picture 24 bits.
RGB is used in image sensors and image display (like CCD, CRT and LCD)

YUV

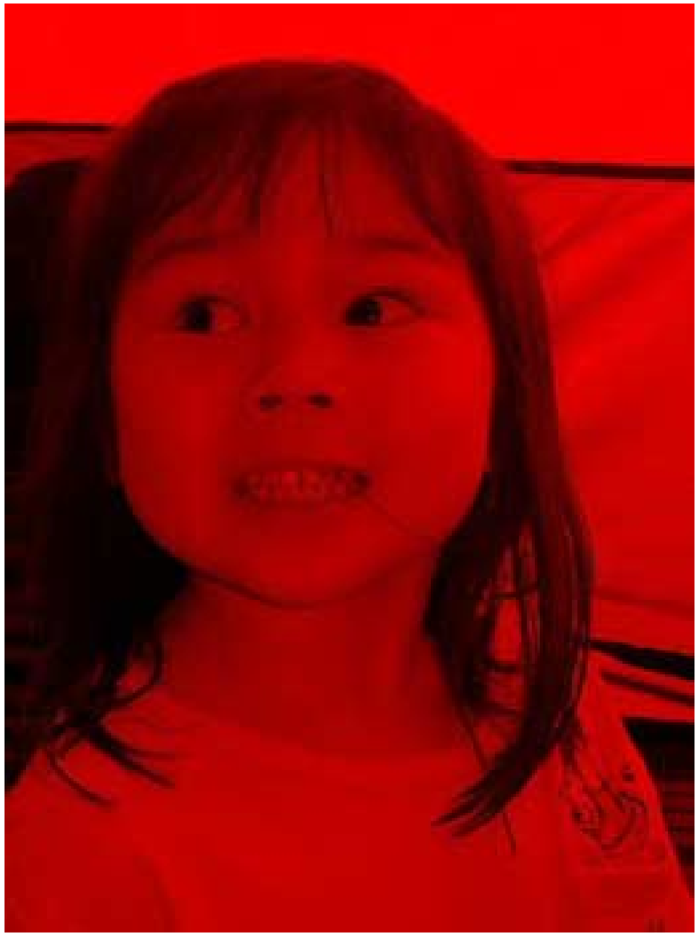
The picture has three components: Y (luminance, intensity, brightness), U and V (chrominance, color content). YUV color space is better suited to human perception, as the humans deal with brightness and color. YUV is used in JPEG, MPEG-1 and MPEG-2. There are variances of YUV like YIQ and YCbCr. Encoding of YUV requires less bits (see later)

RGB

Color image



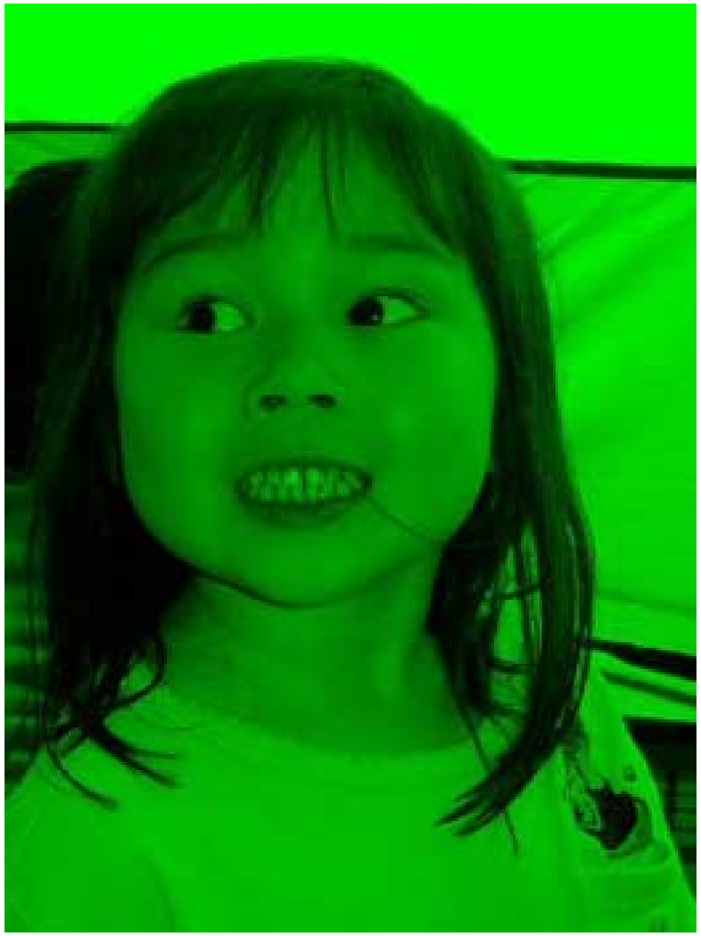
Red component



RGB (cont.)

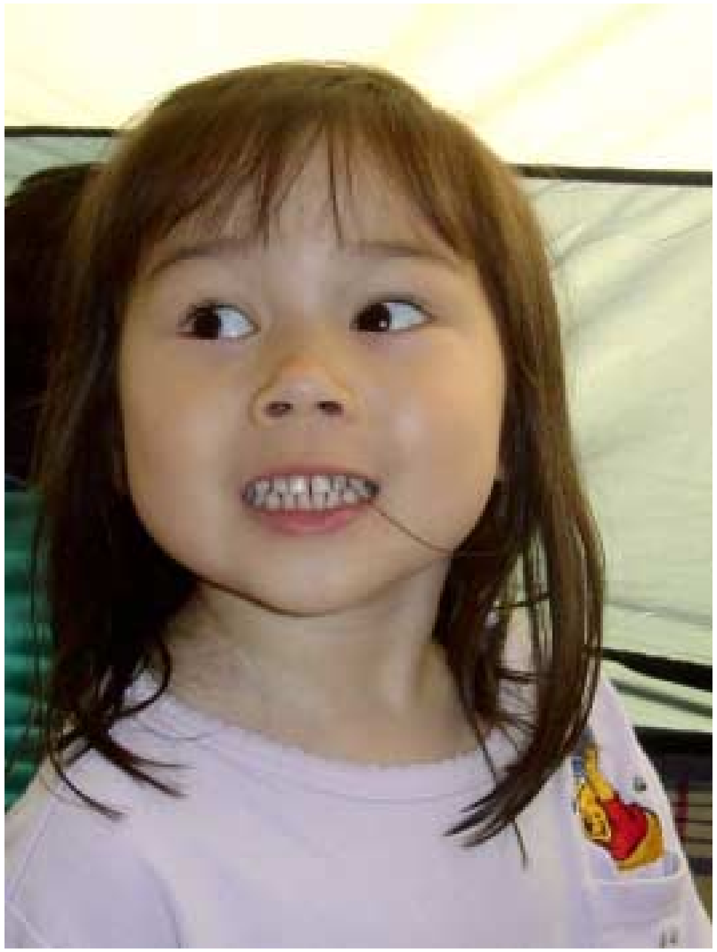
Color image

Green component

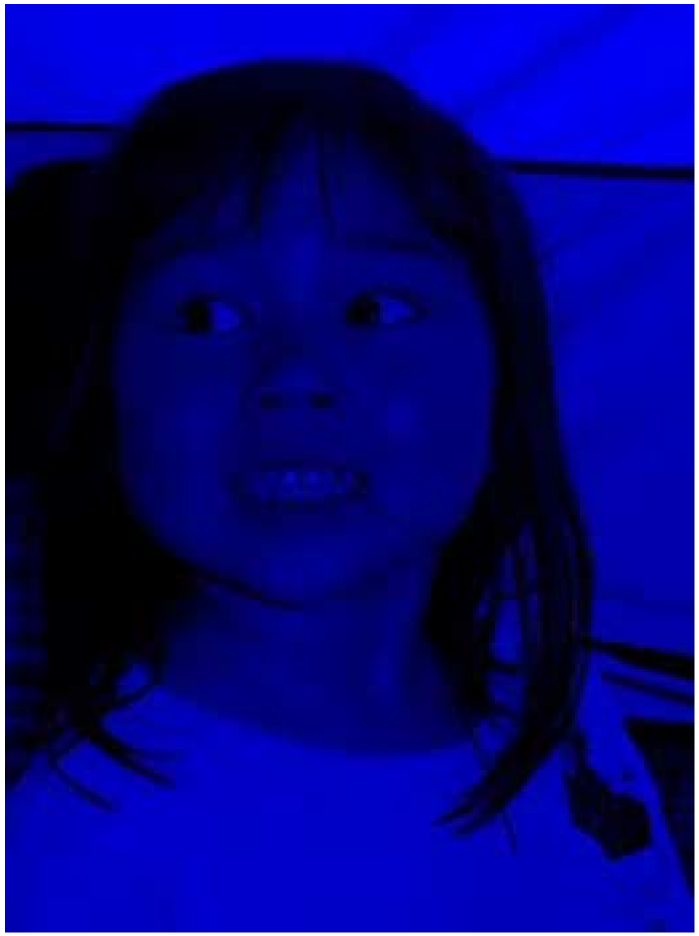


RGB (cont.)

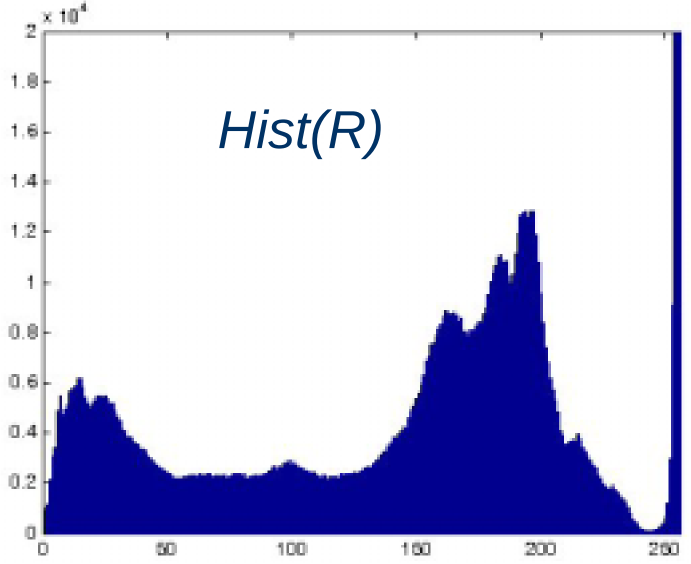
Color image

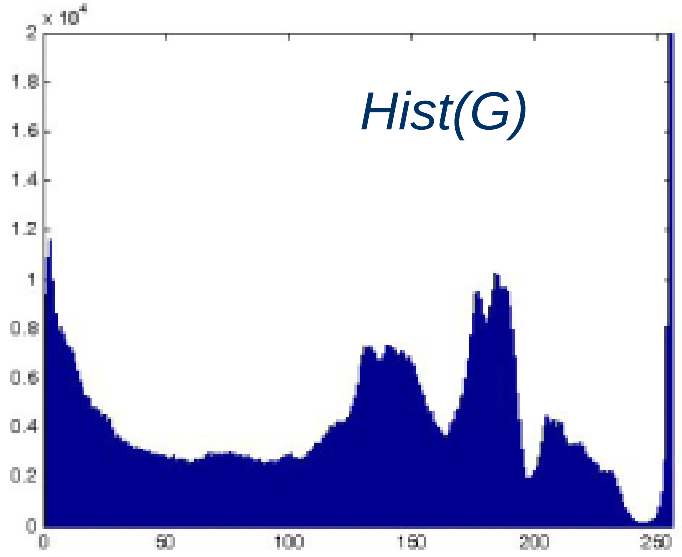


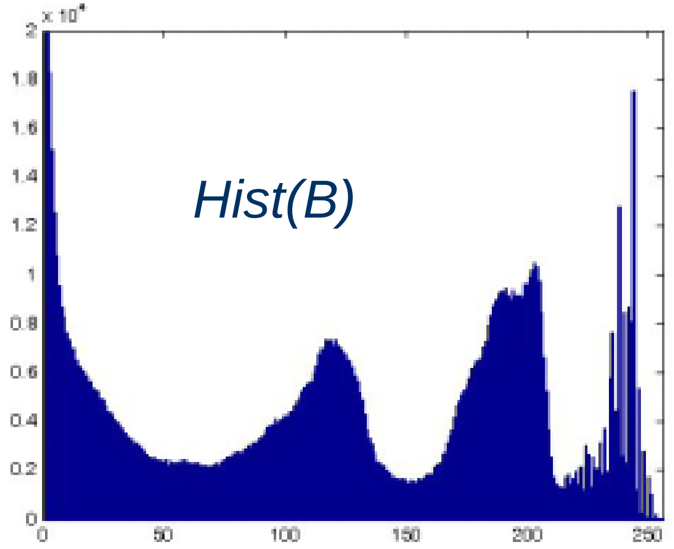
Blue component



RGB (cont.)







This slide shows histograms for the three RGB color components. As seen all intensities (0..255) are significant in all components.

Color Translation

YUV values can be derived from RGB values by simple mapping:

$$Y = 0.30R + 0.59G + 0.11B$$
$$U = 0.493\,(B - Y)$$
$$V = 0.877\,(R - Y)$$

or

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The Y component contains the information about the brightness (intensity) of the image, while the U and V components contain the information abou the color.

Color Translation (cont.)

Mapping is reversible, RGB components can be derived from the corresponding YUV components:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.395 & -0.581 \\ 1.000 & 2.032 & 0.001 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

YIQ color space is similar to YUV color space. It is used in NTSC (National Television System Committee):

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Color Translation (cont.)

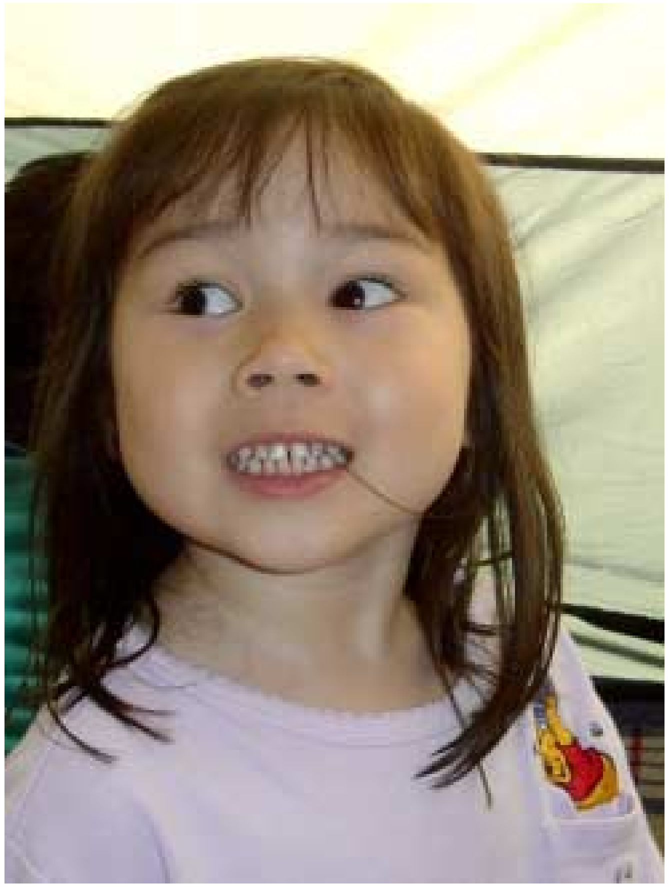
While RBG values are all in the range (0..255), the YUV and YIQ are not. In order to force the luminance and chrominances to the same range, the YUV components have to be scaled and shifted. The result is YC_bC_r color space:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

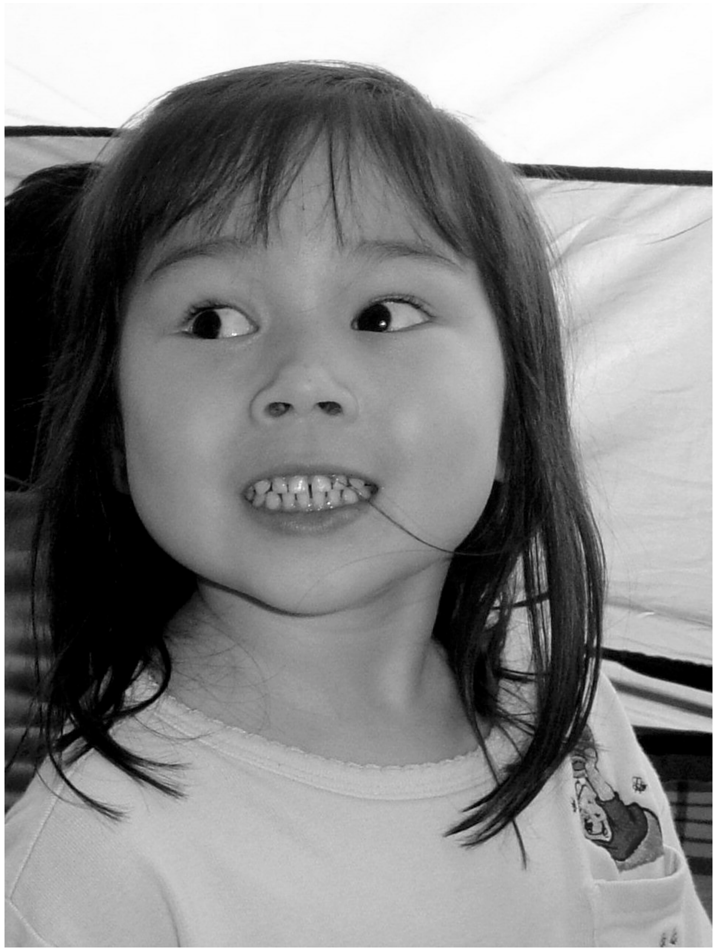
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0.000 & 1.596 \\ 1.164 & -0.392 & -0.813 \\ 1.164 & 2.017 & 0.000 \end{bmatrix} \begin{bmatrix} Y - 16 \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

YUV

Color image

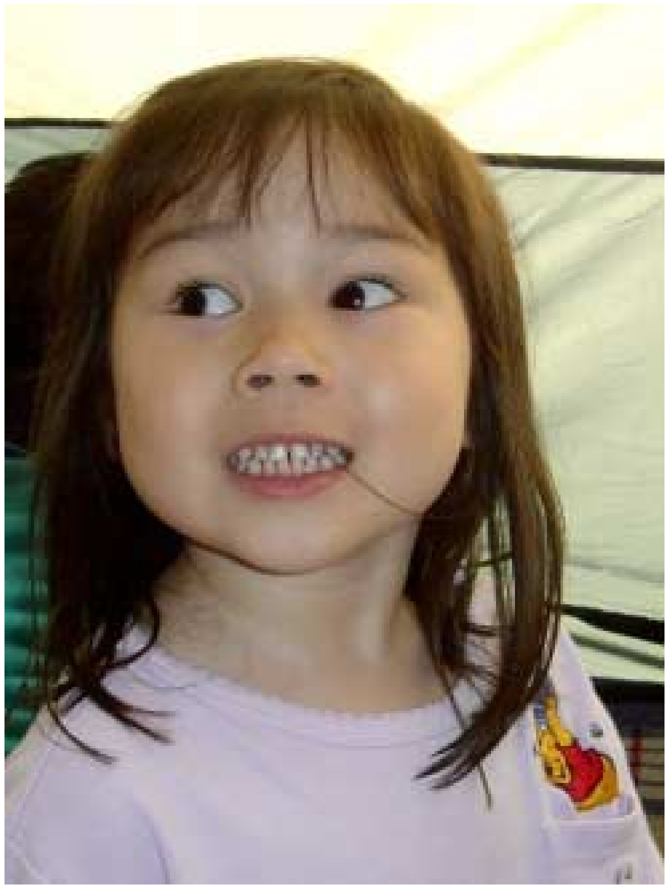


Y component

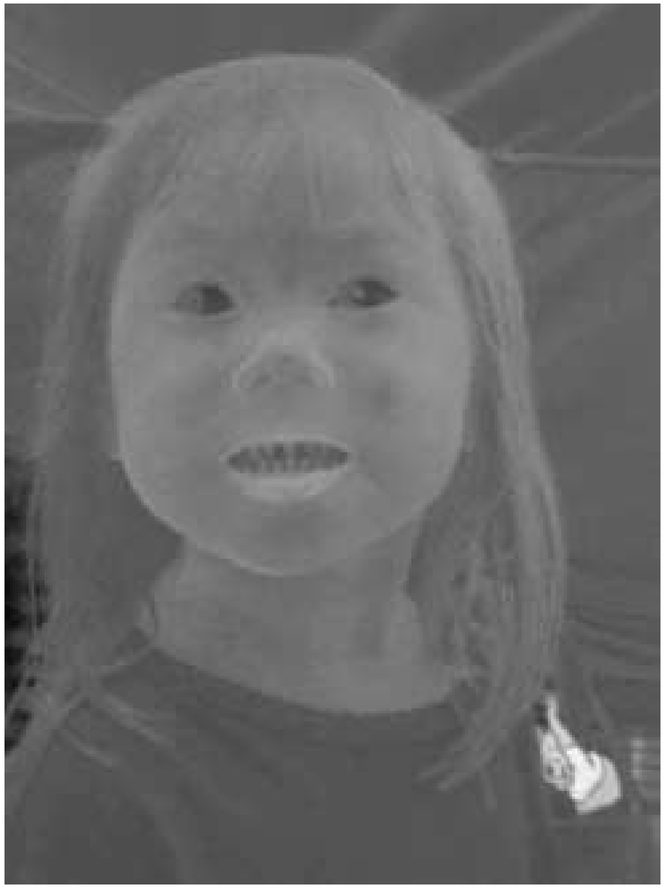


YUV (cont.)

Color image



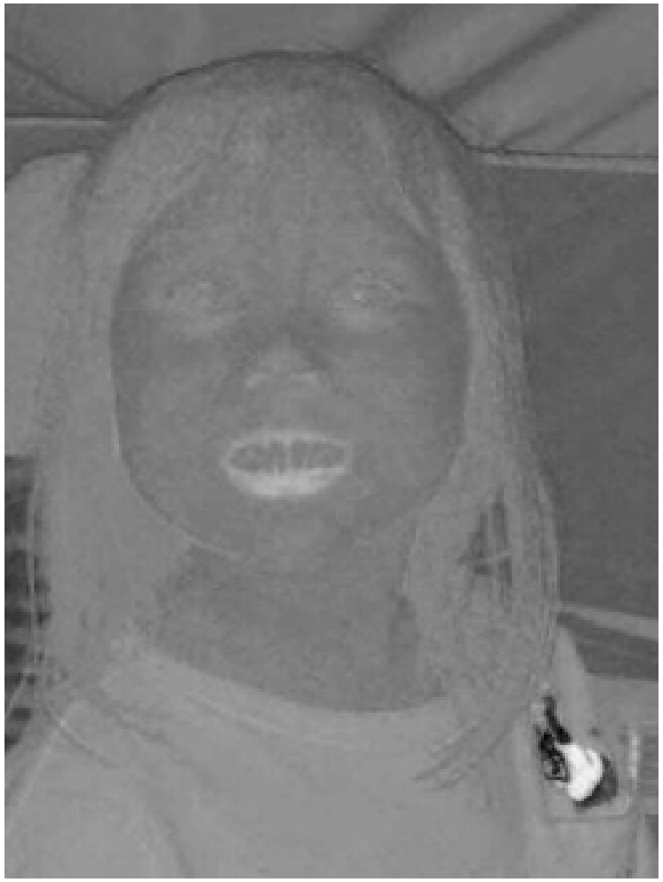
U component



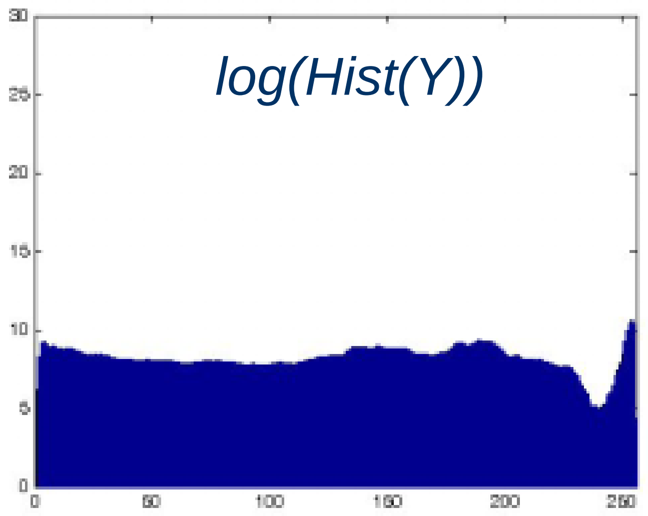
YUV (cont.)

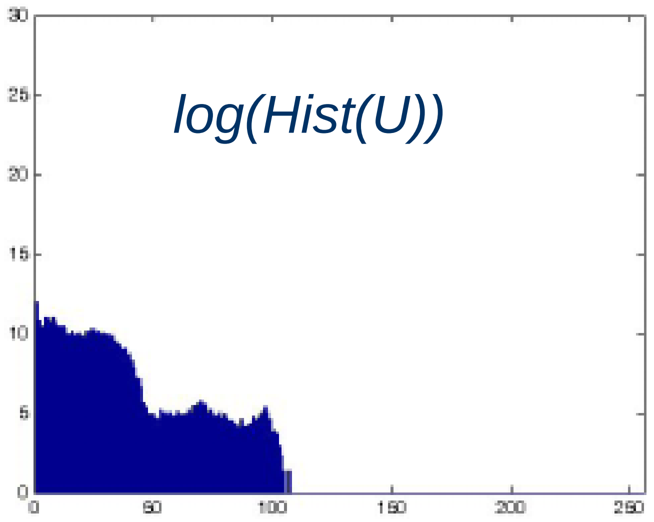
Color image

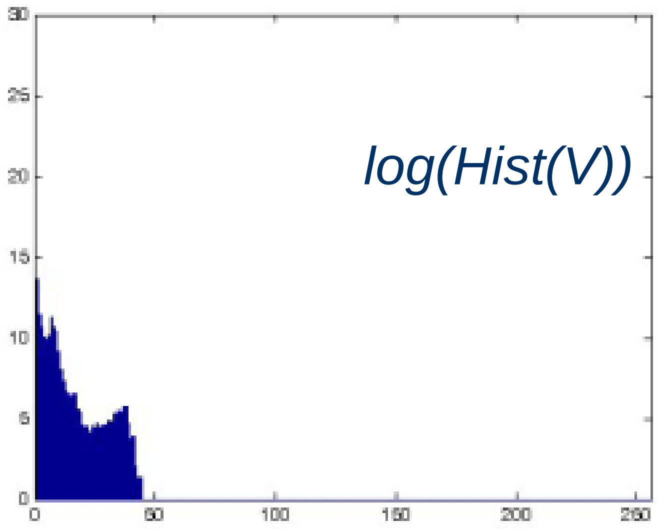
V component



YUV (cont.)





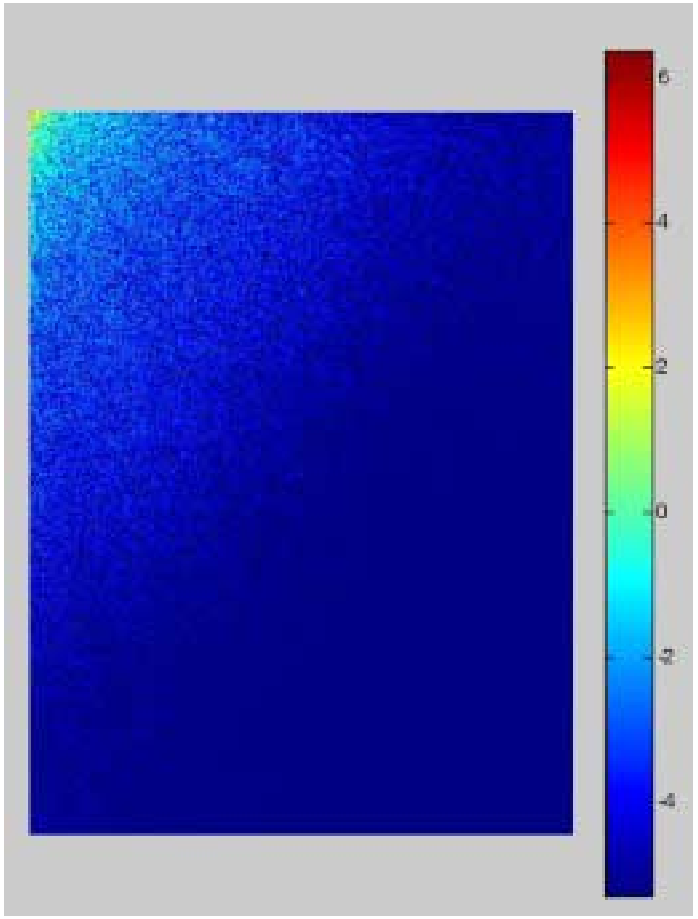


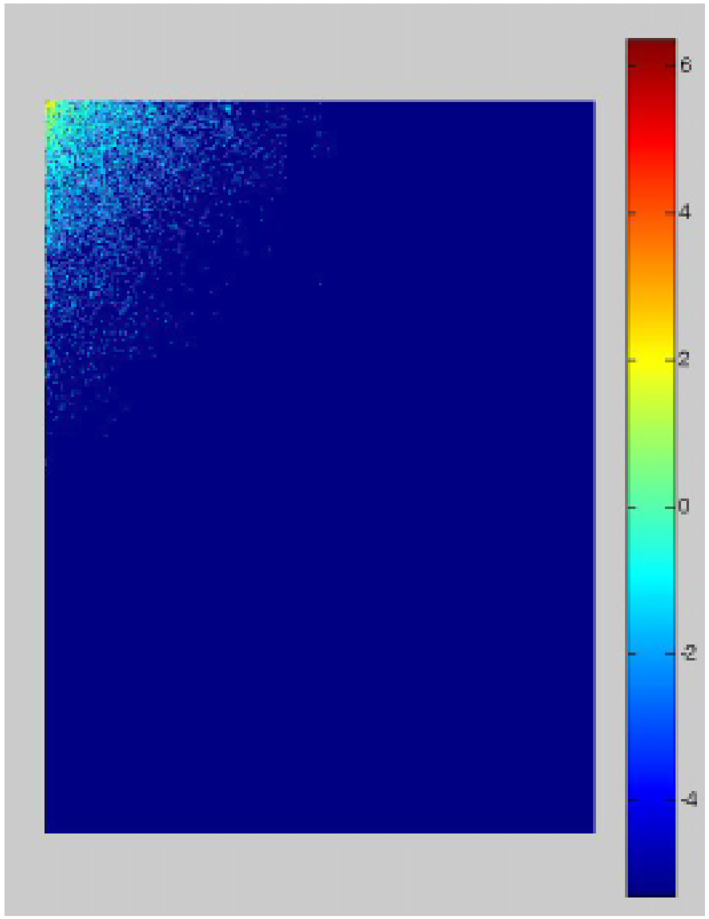
This slide shows histograms for the three YUV color components. All intensities (0..255) are significant in Y component, however the U and V components uses only intensities (0..120) and (0..45) respectively. This indicates that U and V components can use higher compression ratio than Y component

YUV (cont.)

DCT(Y)

Reduced(DCT(Y))

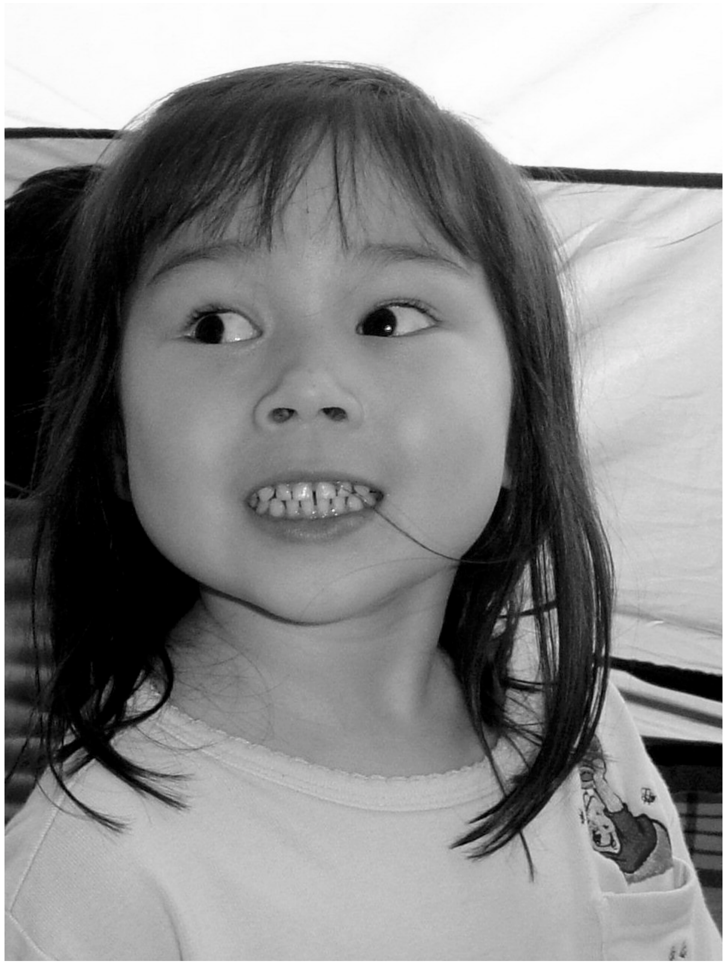




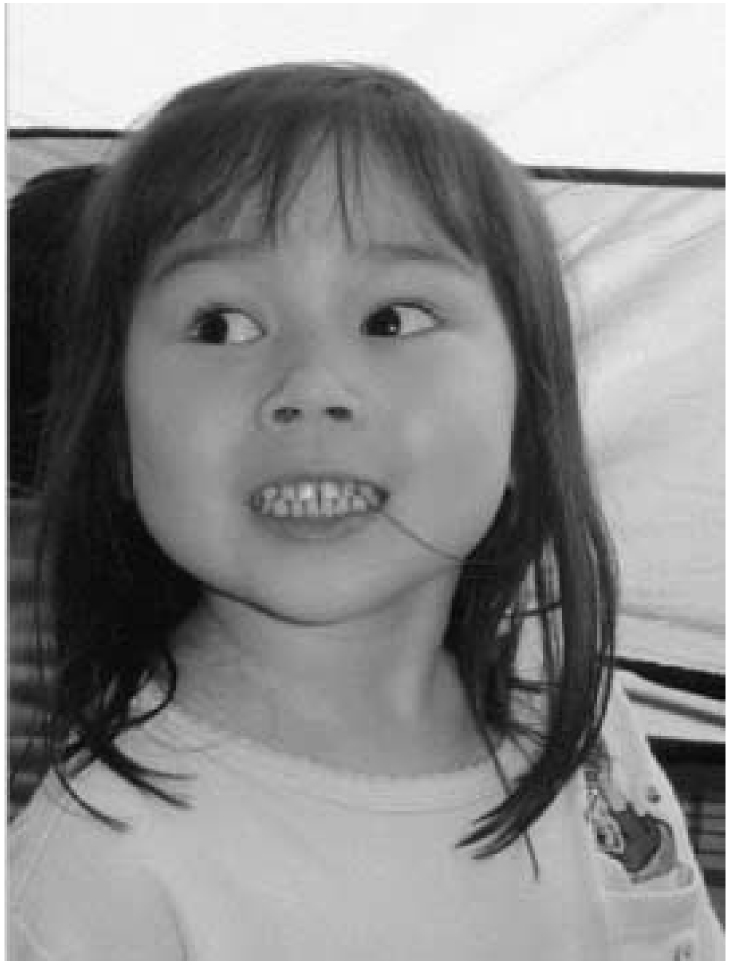
rf = 30

YUV (cont.)

Y



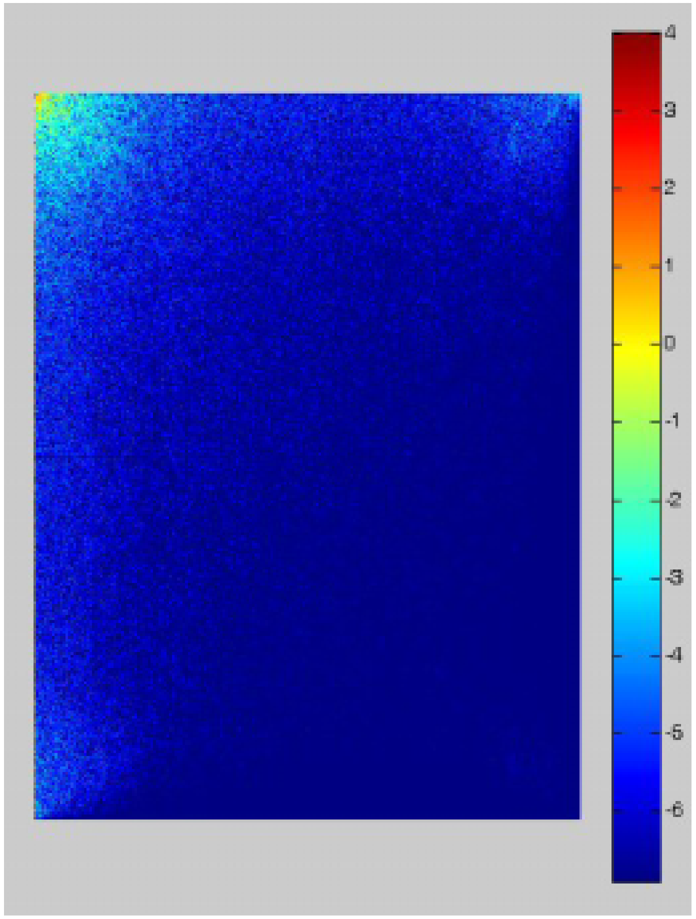
IDCT(reduced(DCT(Y)))



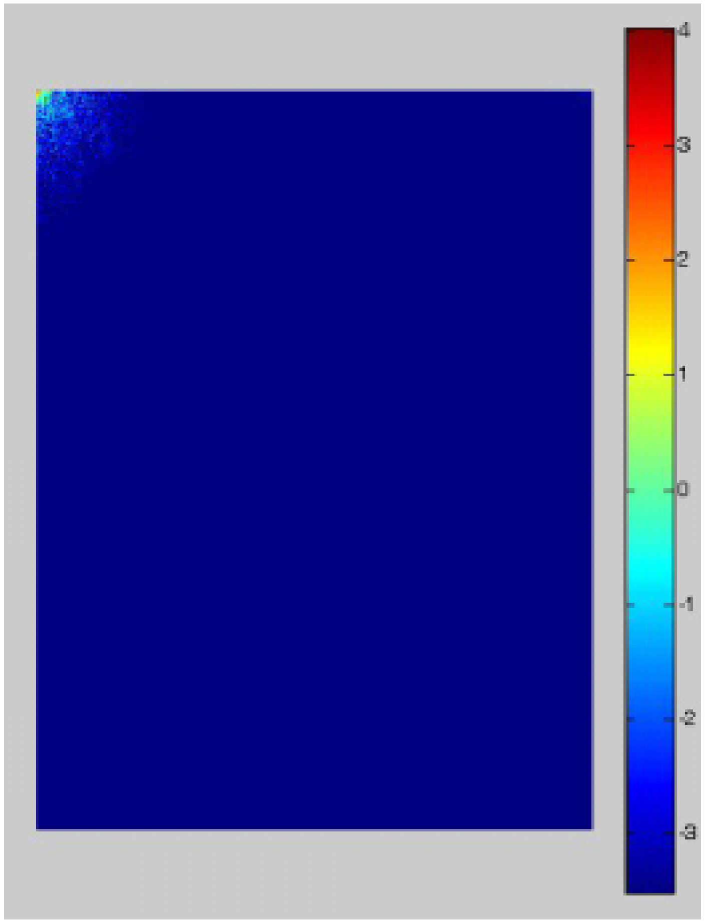
rf=30

YUV (cont.)

DCT(U)



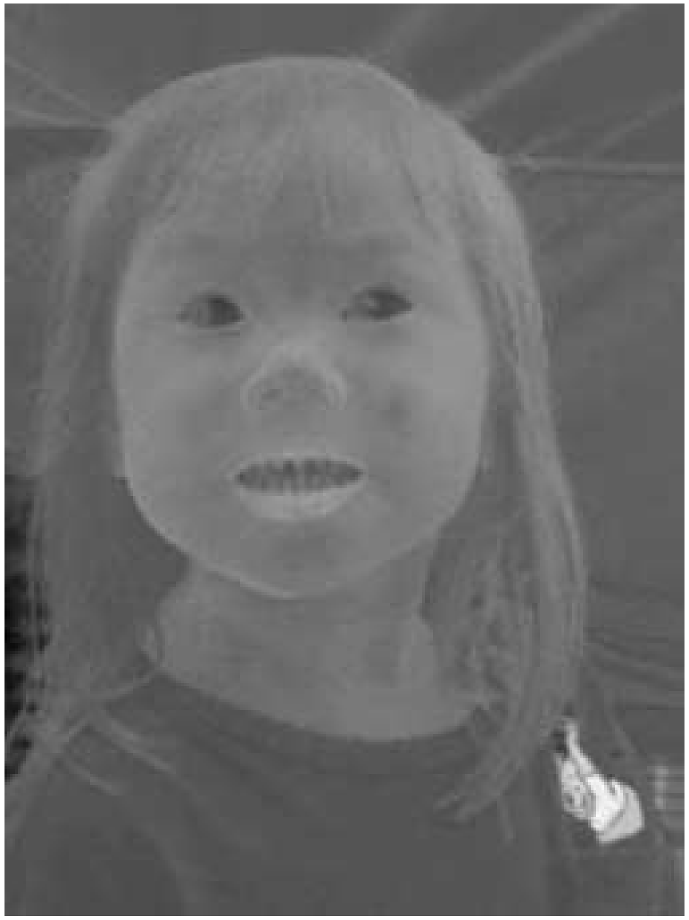
Reduced(DCT(U))



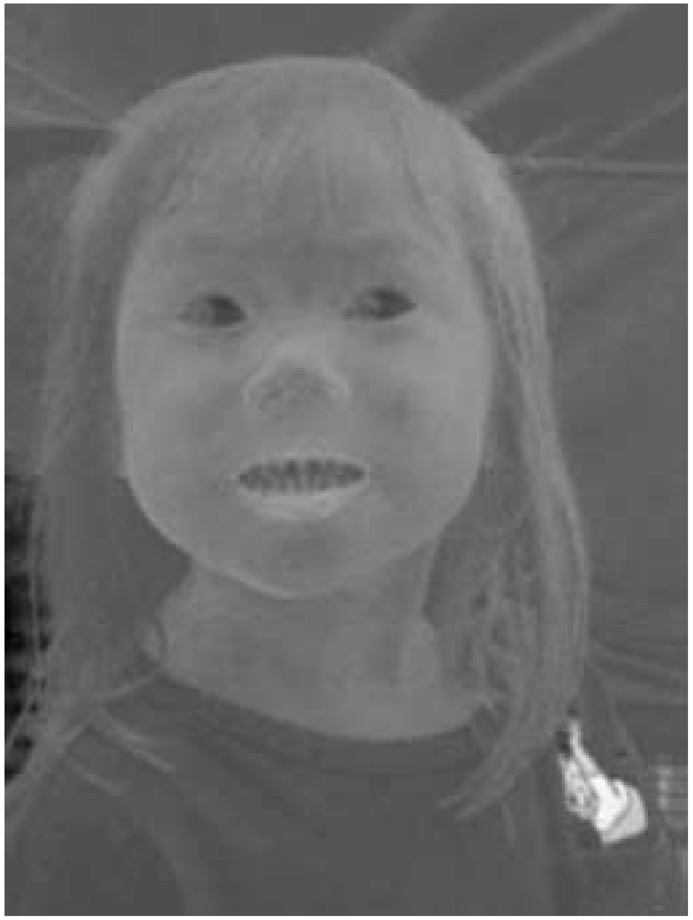
cf = 72

YUV (cont.)

U



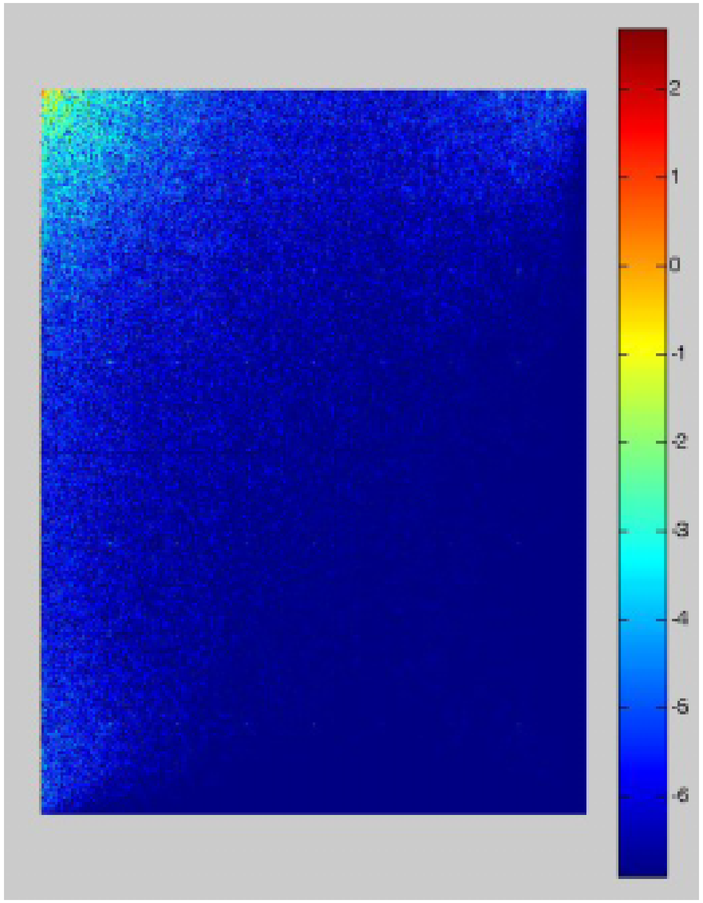
IDCT(reduced(DCT(U)))



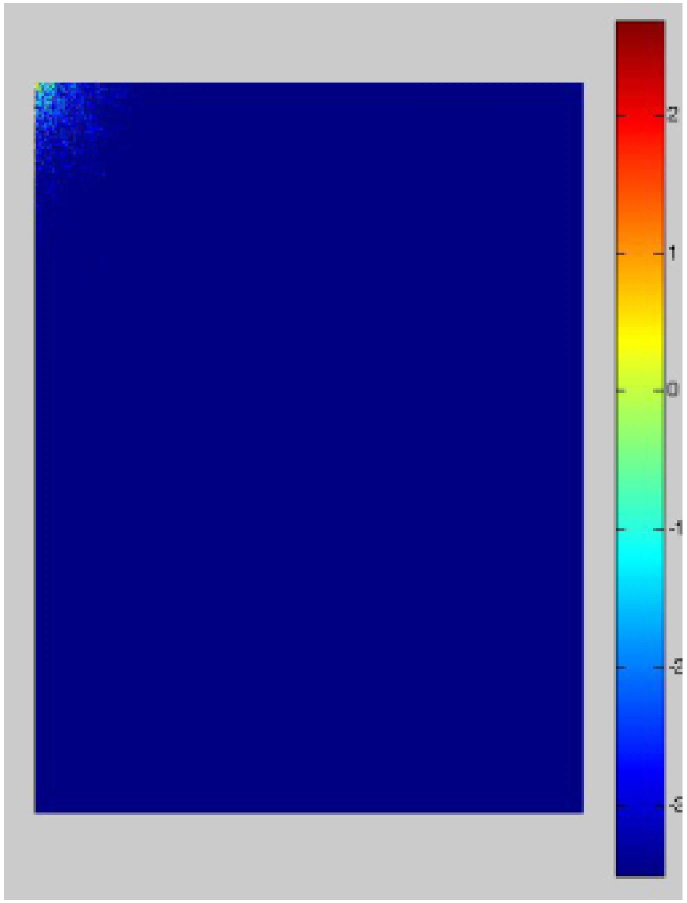
rf=70

YUV (cont.)

DCT(V)



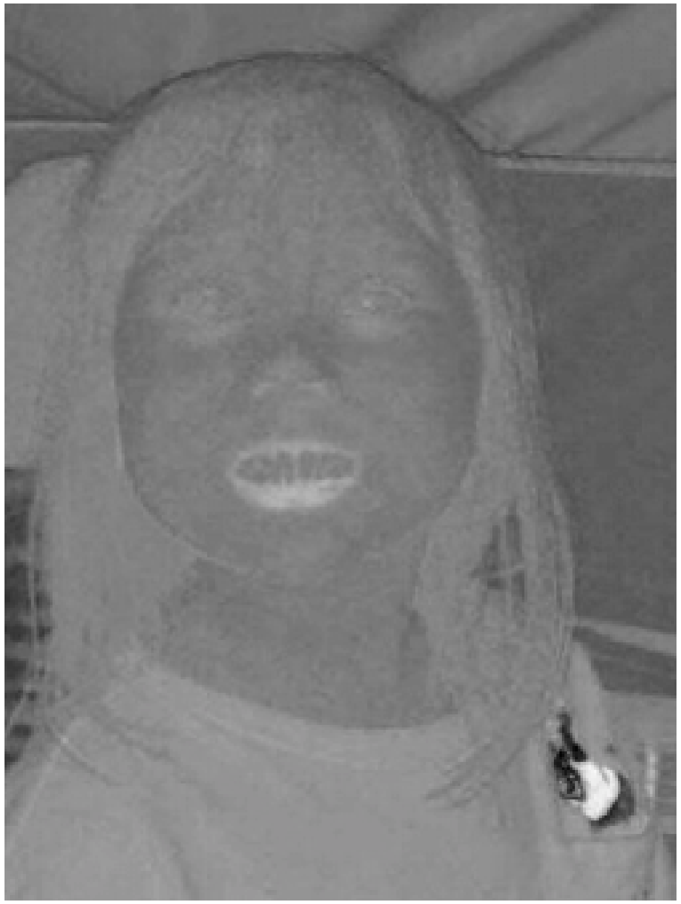
Reduced(DCT(V))



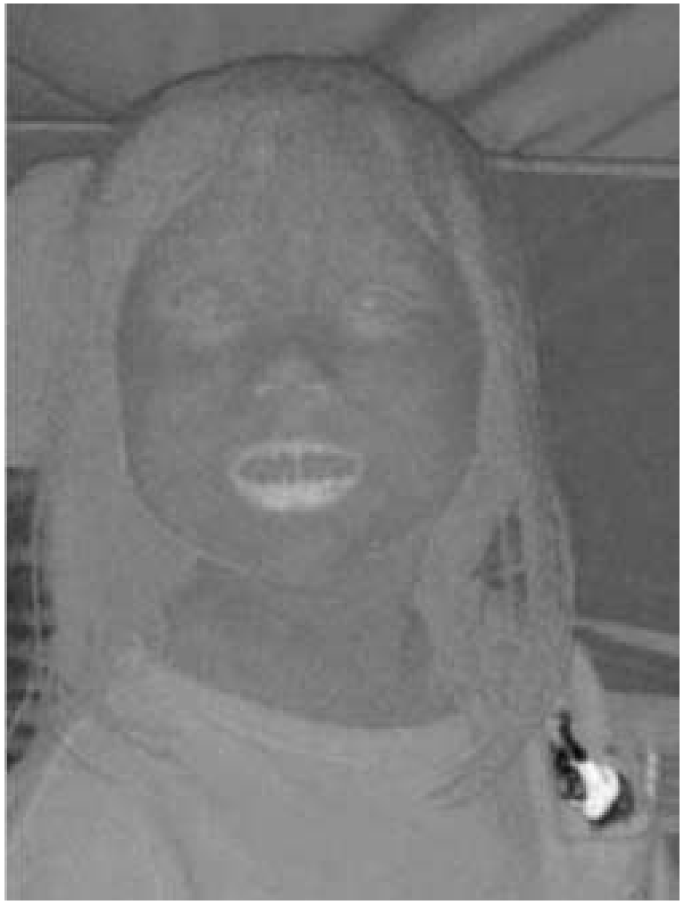
rf = 99

YUV (cont.)

V



IDCT(reduced(DCT(V)))



rf=99

YUV (cont.)

Quantization tables used in JPEG

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
12	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

Luminance Quantization Table (Y)

By examining previous slides we can conclude that grayscale images and the Y component of YUV color space can have reduction factor around 30, while U and V components can have reduction factors 70 and 99 respectively, without diminishing the quality. This means that U and V components can use larger step sizes in quantization. For this reason JPEG defines two quantization tables, one for luminance another for chromonances

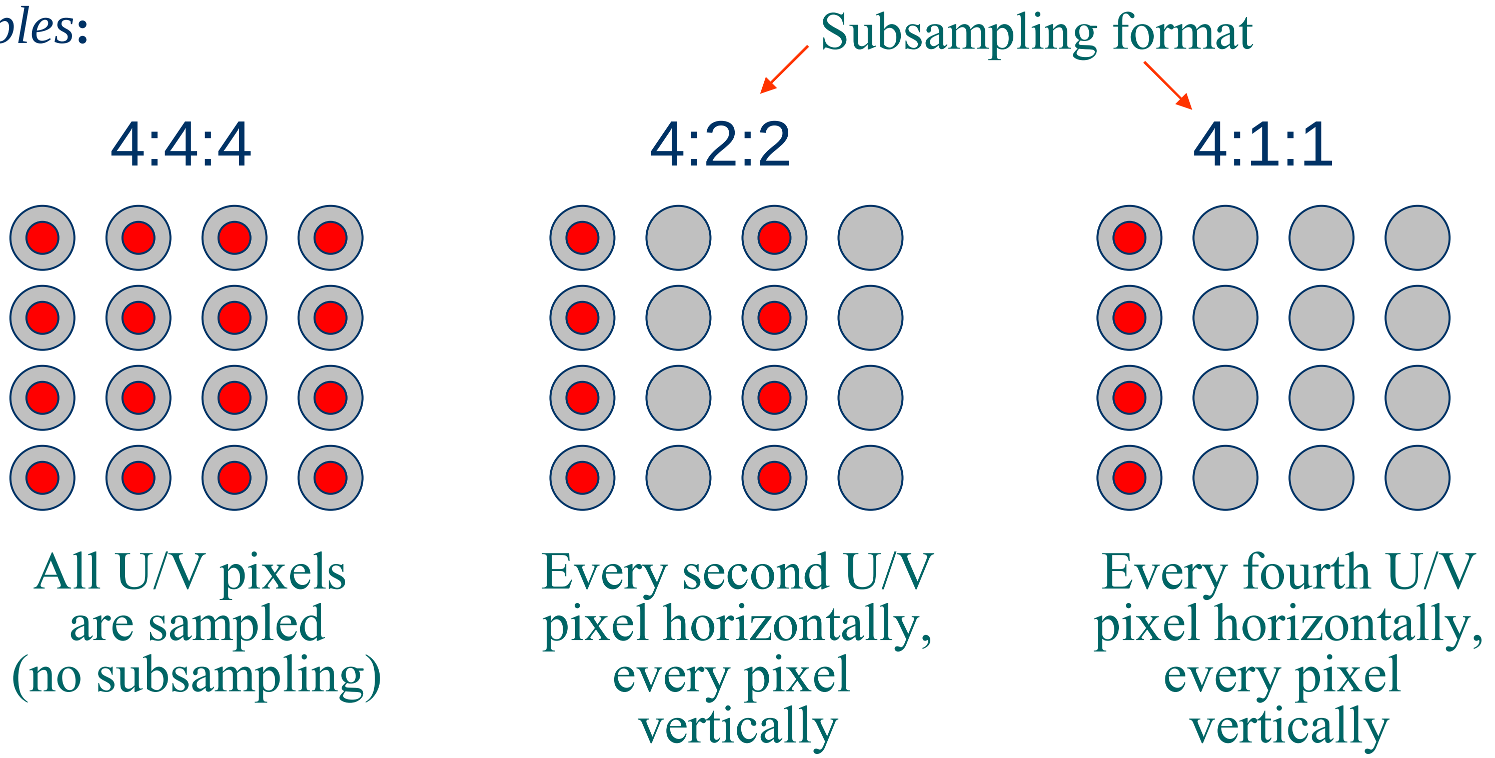
Chrominance Quantization Table (U,V/I,Q/Cb,Cr)

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

Chrominance Subsampling

Besides the smaller resolution in terms of quantization levels, the UV components require smaller spatial resolution. This means that U and V components (or Q and I, or Cb and Cr) require smaller number of pixels than the Y component. Therefore sampling of chrominance components can be done with smaller rates.

Examples:



All U/V pixels are sampled (no subsampling)

Every second U/V pixel horizontally, every pixel vertically

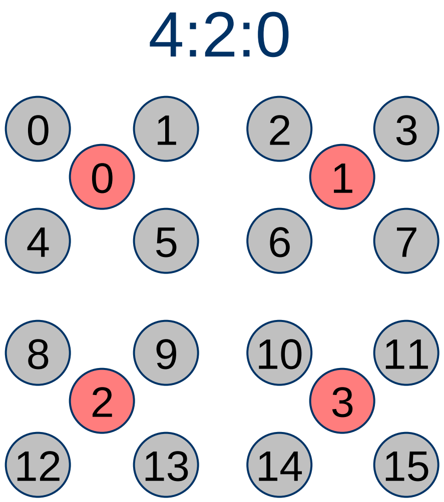
Every fourth U/V pixel horizontally, every pixel vertically

● - Luminance or original chrominance pixel

● - Subsampled chrominance pixel

Chrominance Subsampling (cont)

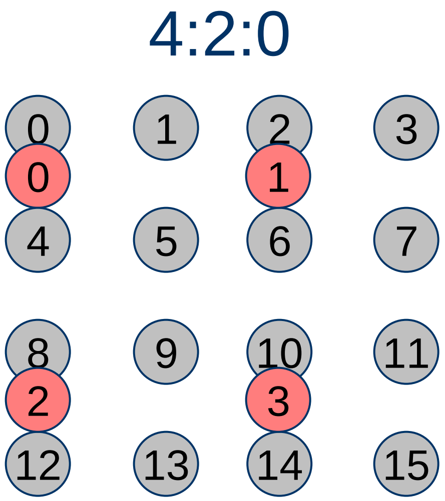
JPEG/MPEG use the lowest subsampling rate (format 4:2:0) which requires the smallest bandwidth. There are two variants of 4:2:0 subsampling with different grid offsets:



$$U_0' = \frac{U_0 + U_1 + U_4 + U_5}{4}$$

$$V_0' = \frac{V_0 + V_1 + V_4 + V_5}{4}$$

(JPEG/MPEG-1/MJPEG)



$$U_0' = \frac{U_0 + U_4}{2}$$

$$V_0' = \frac{V_0 + V_4}{2}$$

(MPEG-2)

Chrominance Subsampling (cont.)

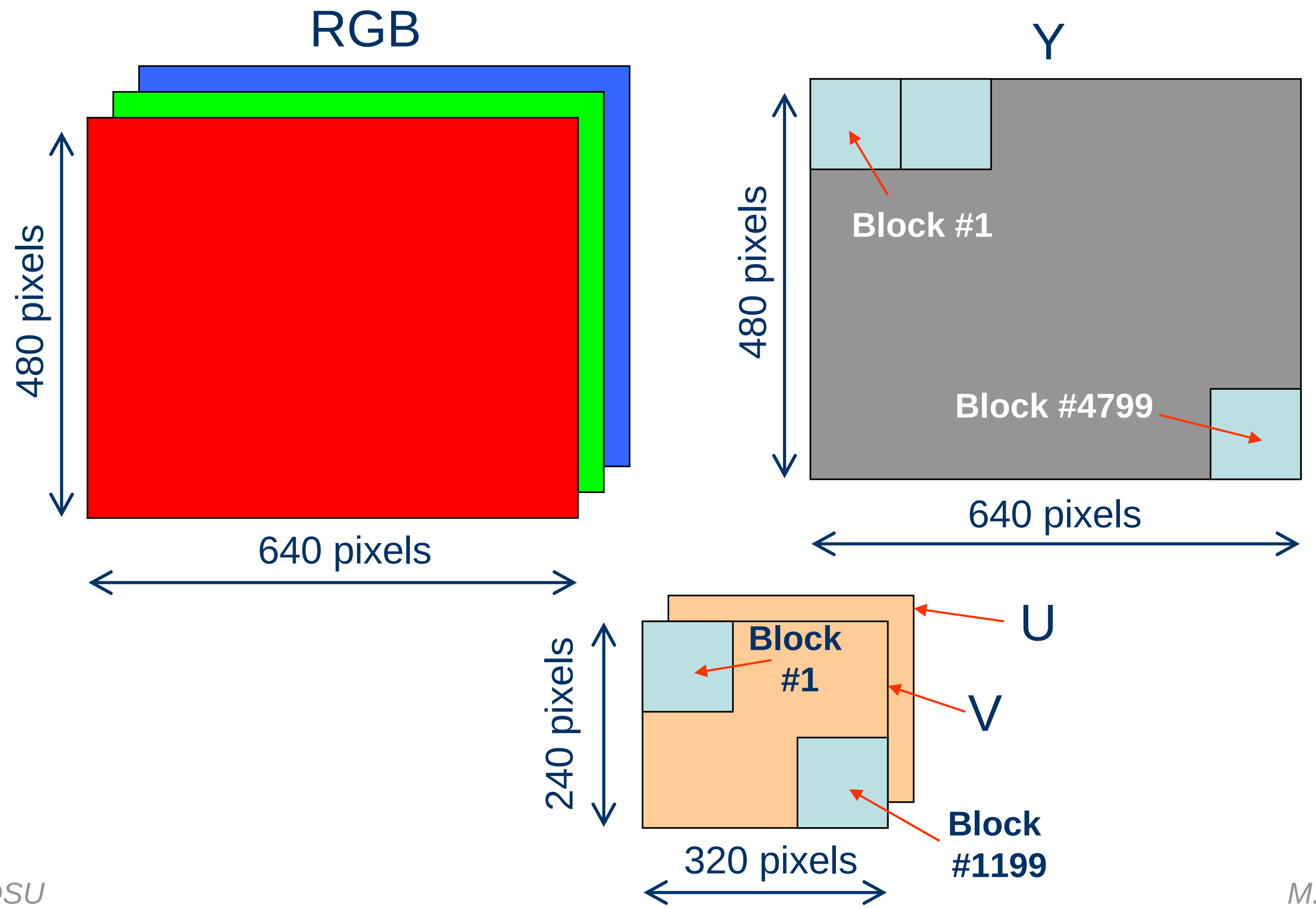
Summary:

Subsampling Format		*Subsampling Ratio*	
		Hor	Ver
4:4:4		1/1	1/1
4:2:2	SMPTE	1/2	1/1
4:2:0	JPEG/MPEG-1/MJPEG	1/2	1/2
	MPEG-2		
	PAL-DV		
4:1:1	NTSC-DV	1/4	1/1

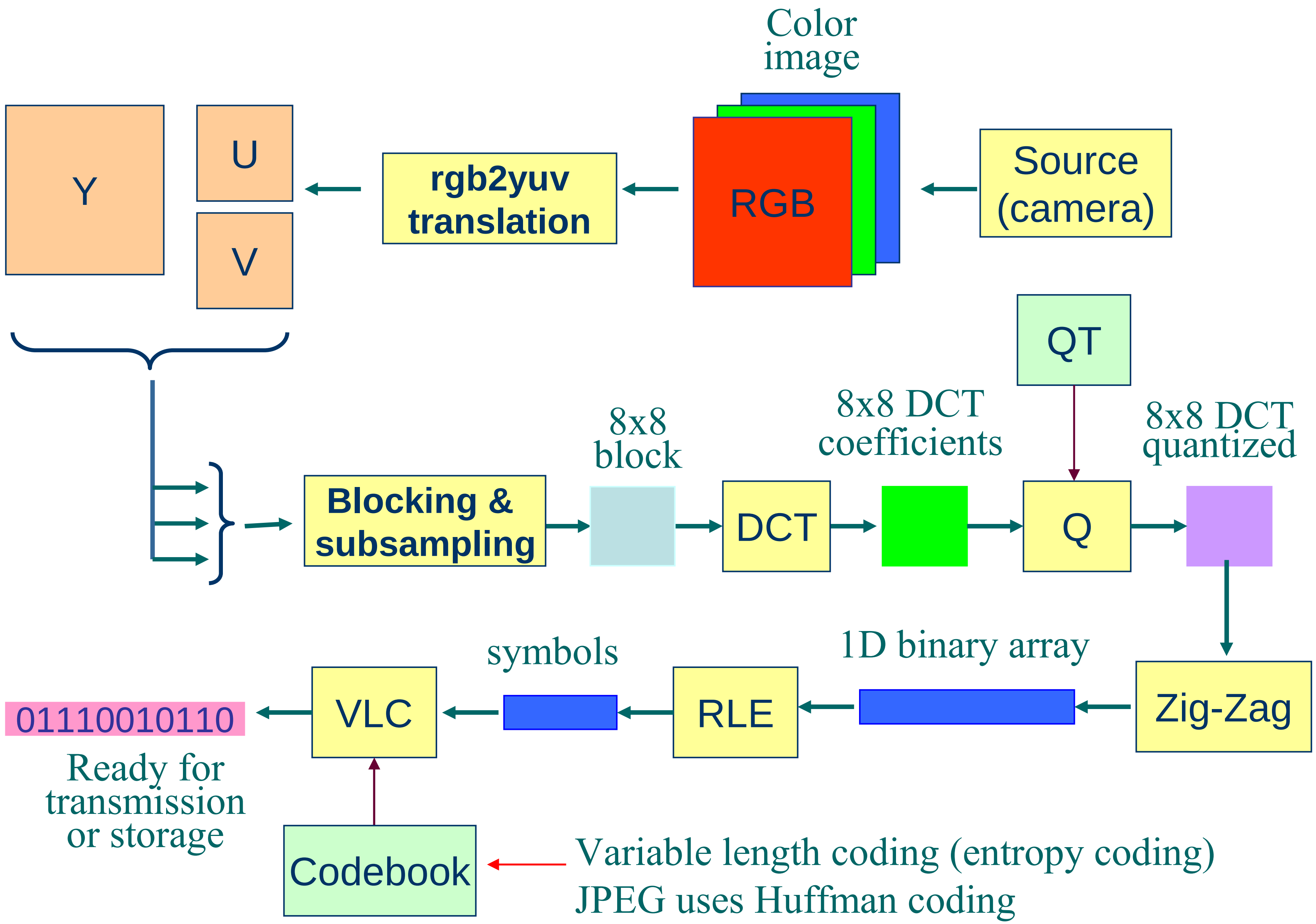
SMPTE = The Society of Motion Picture and Television Engineers

Chrominance Subsampling (cont.)

JPEG components and blocks



JPEG Color Images



Variable length coding (entropy coding)
JPEG uses Huffman coding

Compression of Moving Pictures

The compression techniques considered so far are applied to still pictures. This kind of compression is called spatial compression or intra-frame compression.

In case of moving pictures the source generates a sequence of pictures called frames. A typical frame rate is 30 frames per second. It is reasonable to expect that two adjacent frames are very similar since the picture doesn't change very rapidly in 1/30 seconds. In most of cases most of a picture is unchanged, while only parts of the picture are moving. There is an obvious redundancy between frames, which is called temporal redundancy or inter-frame redundancy.

The rest of this chapter discusses compression techniques that exploit the temporal redundancy in moving pictures. The core of the temporal compression are predictive coders and predictive decoders.

Predictive Coding

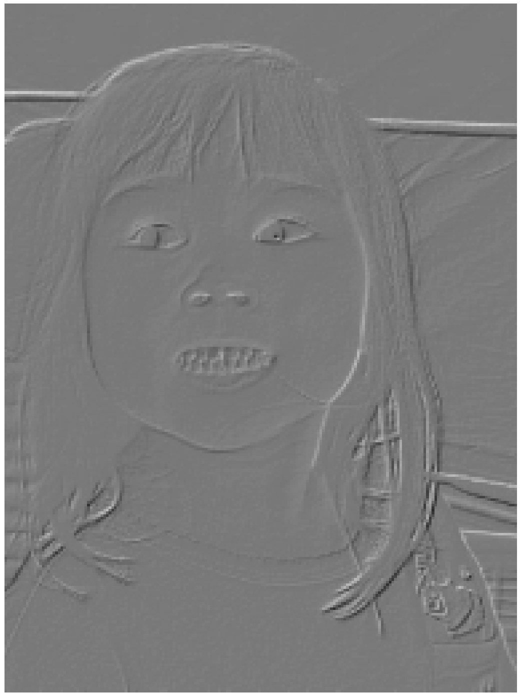
This slide shows the essence of temporal compression:

Frame #1

-

Frame #2 (shifted 3 pixels down and 3 pixels right)

=



Difference

Many pixels have equal value - very high spatial compression ratio can be achieved

Send the frame #1 and the difference, frame #2 can be constructed by adding the difference to the frame #1.

Predictive Coding (cont.)

Following the general idea of sending the differences instead of actual frames we can first consider a single pixel:

Sender sends:

$$s(1),$$
$$e(2)=s(2)-s(1),$$
$$e(3)=s(3)-s(2),....$$

Receiver reconstructs:

$$s(1),$$
$$s(2)=s(1)+e(2),$$
$$s(3)=s(2)+e(3),....$$

However, if sender sends the quantized errors:

Quantization error

$$e(2)'=Q[s(2)-s(1)]=e(2)+q(2),$$
$$e(3)'=Q[s(3)-s(2)]=e(3)+q(3),....$$

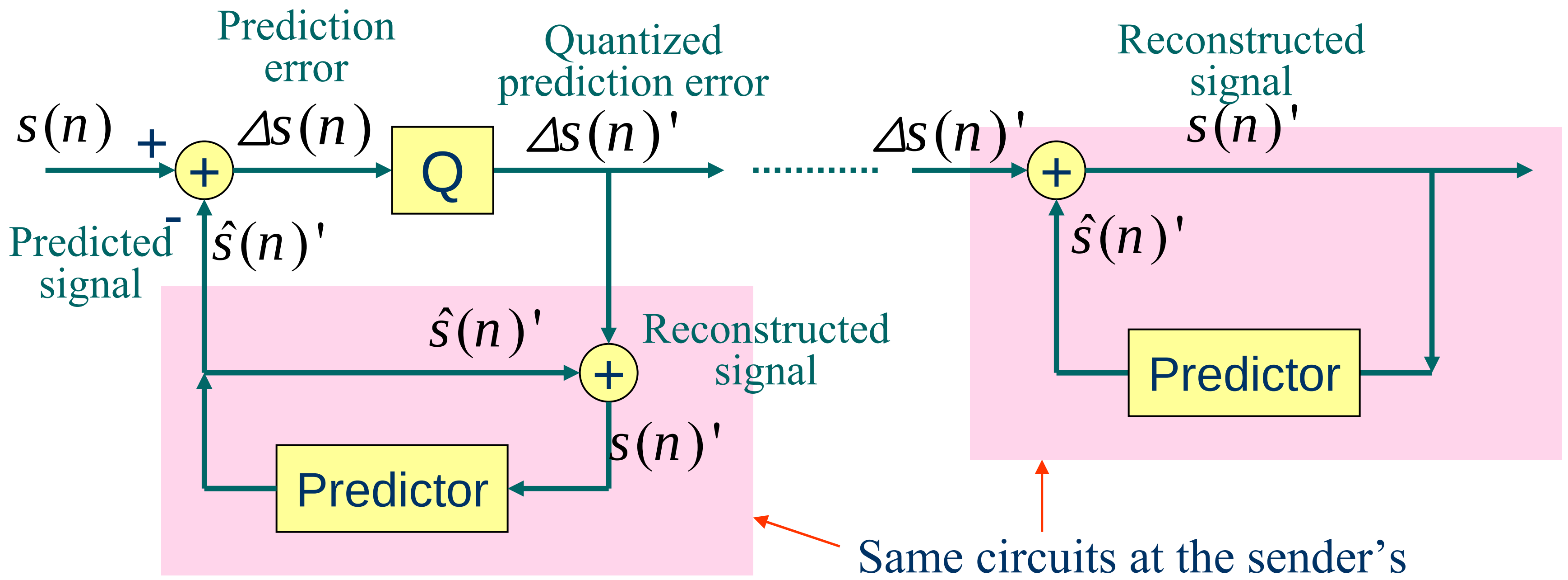
The receiver is forced to reconstruct the signal using *e(n)'* **instead of** *e(n)*:

$$\tilde{s}(1)=s(1),$$
$$\tilde{s}(2)=\tilde{s}(1)+e(1)'=s(1)+e(1)+q(1)=s(2)+q(1),$$
$$\tilde{s}(3)=\tilde{s}(2)+e(2)'=s(2)+q(1)+e(2)+q(2)=s(3)+q(1)+q(2),$$
$$. \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad .$$
$$\tilde{s}(n)=s(n)+q(1)+q(2)+q(3)+...+q(n-1)$$

Huge cumulative quantization error possible!

Predictive Coding (cont.)

An approach to avoid cumulative quantization errors is to send the quantized prediction error instead of quantized difference between two adjacent samples:



Same circuits at the sender's and the receiver's side

Prediction error: $\Delta s(n) = s(n) - \hat{s}(n)'$

Quantized pred. error: $\Delta s(n)' = Q[\Delta s(n)] = \Delta s(n) + q(n)$

Reconstructed signal: $s(n)' = \hat{s}(n)' + \Delta s(n)' = [s(n) - \Delta s(n)] + [\Delta s(n) + q(n)]$

$= s(n) + q(n)$ ← No cumulative quantization error!

Predictive Coding (cont.)

Predictor

The simplest predictor is <u>linear predictor</u> which is based on Taylor expansion of continuous functions. Suppose the continuous version of the audio/video signal *f(t)* and sampling time *Δt*, then:

$$f(t+\Delta t)=f(t)+\sum_{k=1}^{\infty}\frac{d^k f(t)}{dt^k}\frac{\Delta t^k}{k!}=f(t)+\frac{df(t)}{dt}\Delta t+O(\Delta t^2)$$

If *Δt* is small enough, then

$$\frac{df(t)}{dt}\Delta t\approx f(t)-f(t-\Delta t)$$

$$f(t+\Delta t)\approx f(t)+[f(t)-f(t-\Delta t)]=2f(t)-f(t-\Delta t)$$

If *s(n+1)*, *s(n)* and *s(n-1)* represent samples of *f(t+Δt)*, *f(t)* and *f(t-Δt)*, then:

$$s(n+1)\approx 2s(n)-s(n-1) \qquad \textbf{or} \qquad s(n)\approx 2s(n-1)-s(n-2)$$

Predictive Coding (cont.)

Predictor (cont.)

In general case the prediction of *s(n)* can be based on several past samples of *s*:

$$\hat{s}(n) = \alpha_1 s(n-1) + \alpha_2 s(n-2) + \alpha_3 s(n-3) + \ldots + \alpha_K s(n-K)$$

This can be applied to the quantized samples as well:

$$\hat{s}(n)' = \alpha_1 s(n-1)' + \alpha_2 s(n-2)' + \alpha_3 s(n-3)' + \ldots + \alpha_K s(n-K)'$$

In order to simplify and to reduce the memory requirements, the predictors can be based on only one past sample:

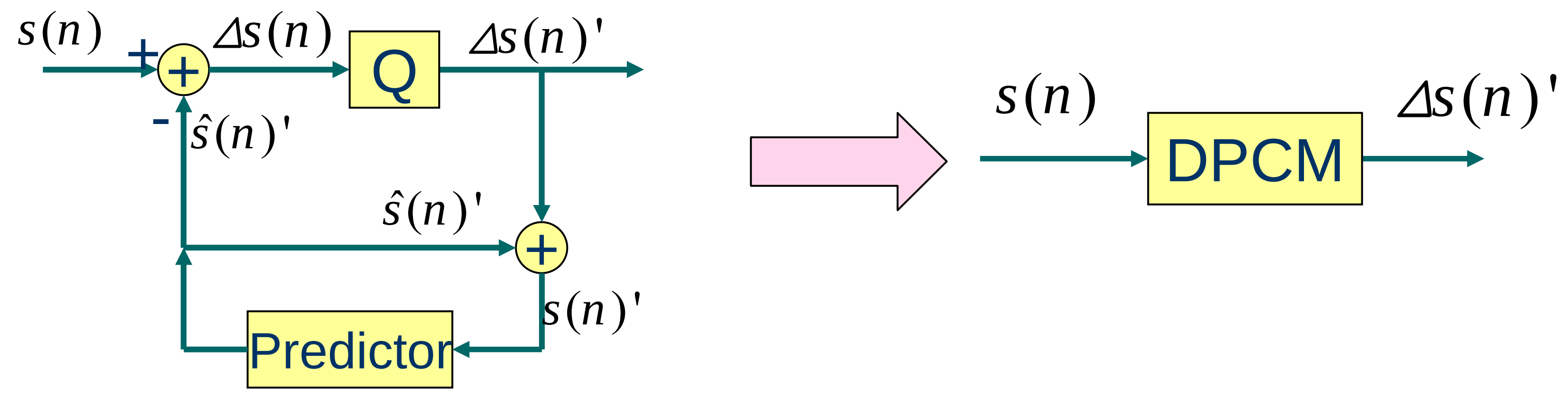
$$\hat{s}(n)' = s(n-1)'$$

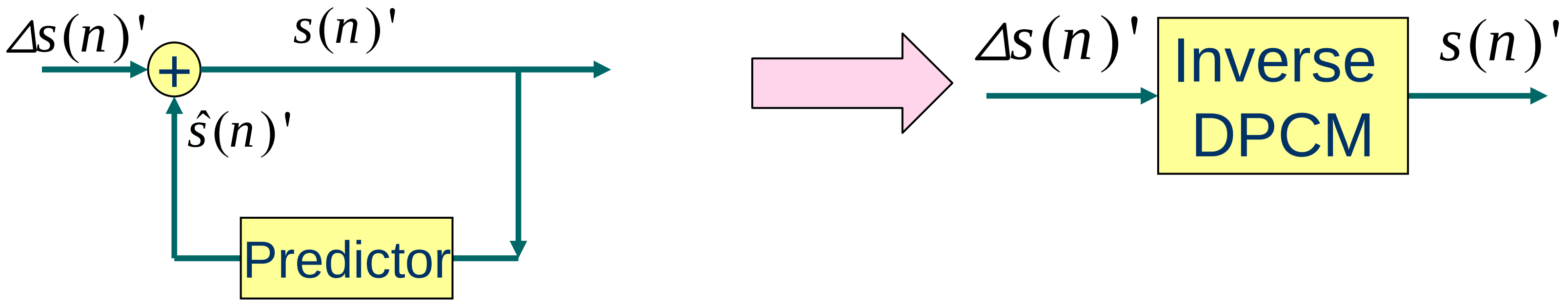
Such simple predictor is good enough with stationary pictures, i.e. pictures in which details are not moving, but may change the YUV values.

Predictive Coding (cont.)

DPCM

The encoding method described in previous slides is called Differential Pulse Code Modulation (DPCM).



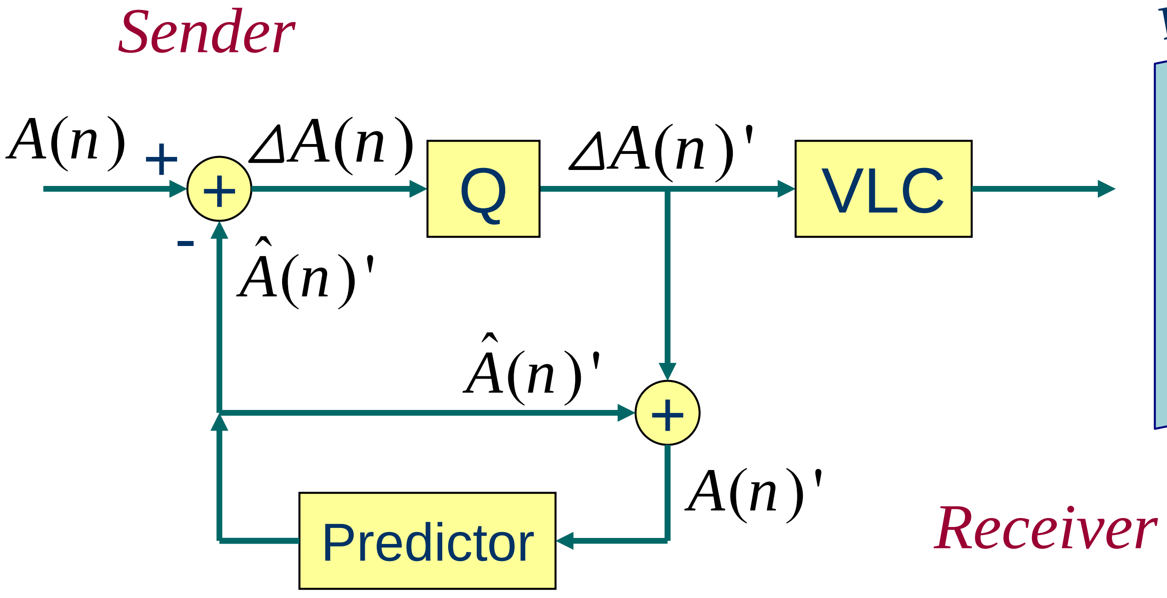


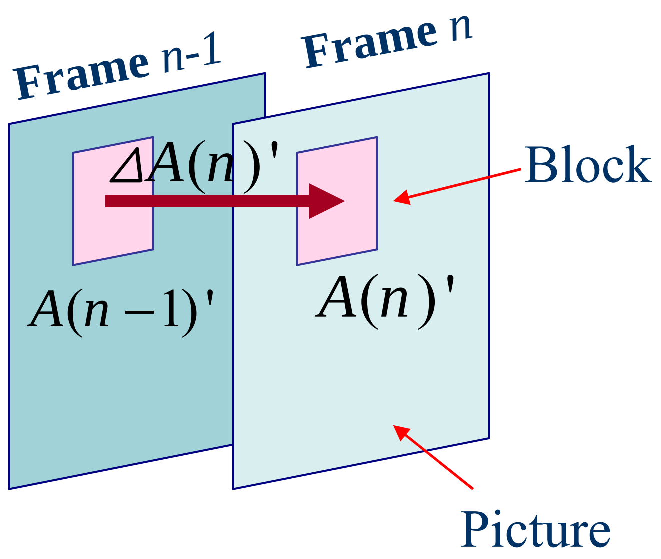
DCPM is often used to encode audio signals and DC values after block DCT

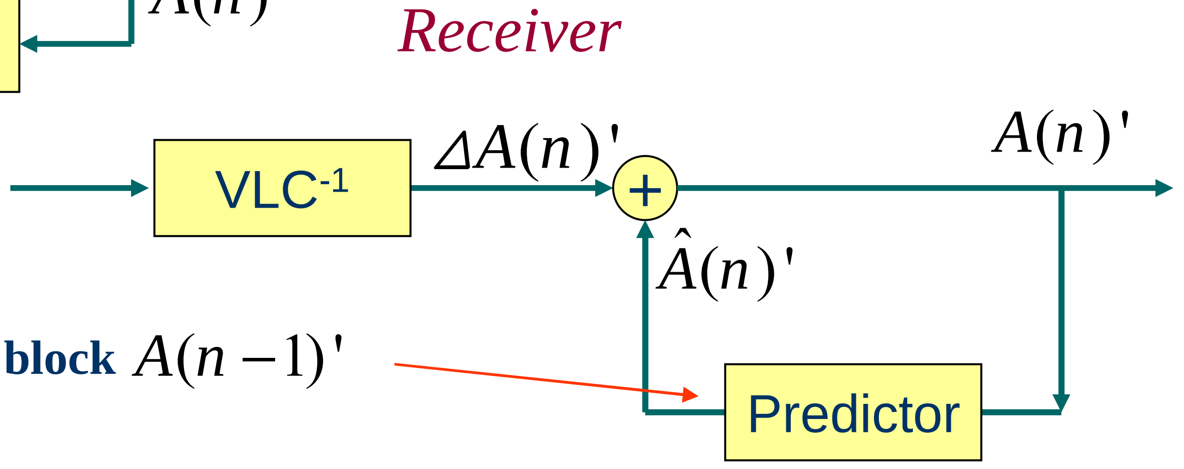
Predictive Coding (cont.)

The predictive encoder can also be applied to blocks of pixels.

Let $A(n)$ **be the block** $[A(p,q)]_8^8$ **in** n**-th frame. Then:**



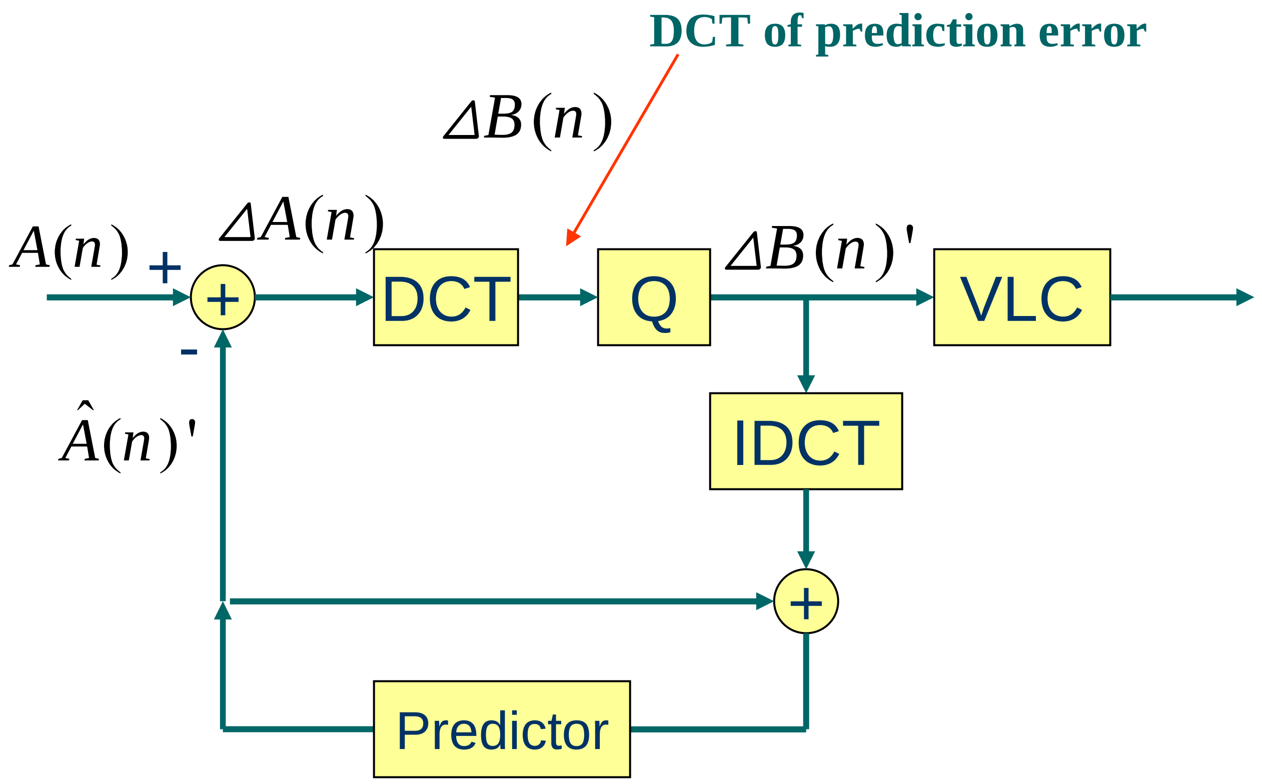




Predictor's memory contains block $A(n-1)'$

Predictive Coding (cont.)

Higher compression ratio and lower bandwidth can be achieved if we differentially encode DCT of blocks instead of blocks themselves:



Higher compression ratio and lower bandwidth can be achieved if we differentially encode DCT of blocks instead of blocks themselves:

Interframe Prediction with Motion Compensation

The predictive encoder is the simplest interframe prediction algorithm. It works well in cases of stationary pictures. In case where the picture details are moving a more sophisticated predictor is required.

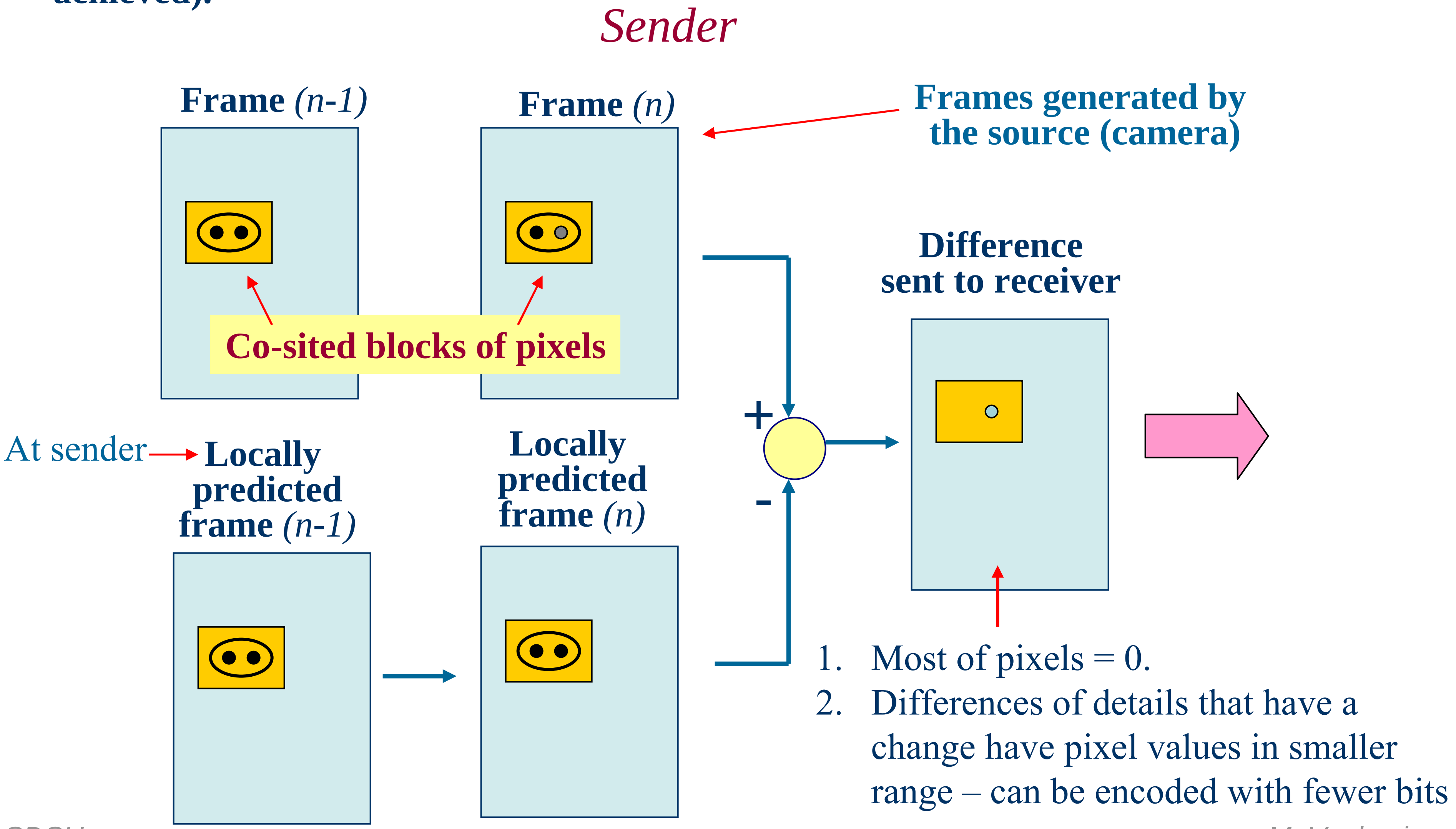
The more sophisticated predictors use techniques based on motion estimation and motion compensation.

There are two basic approaches in prediction with motion compensation:

- Forward interframe prediction
- B-directional interframe prediction

Forward Interframe Prediction

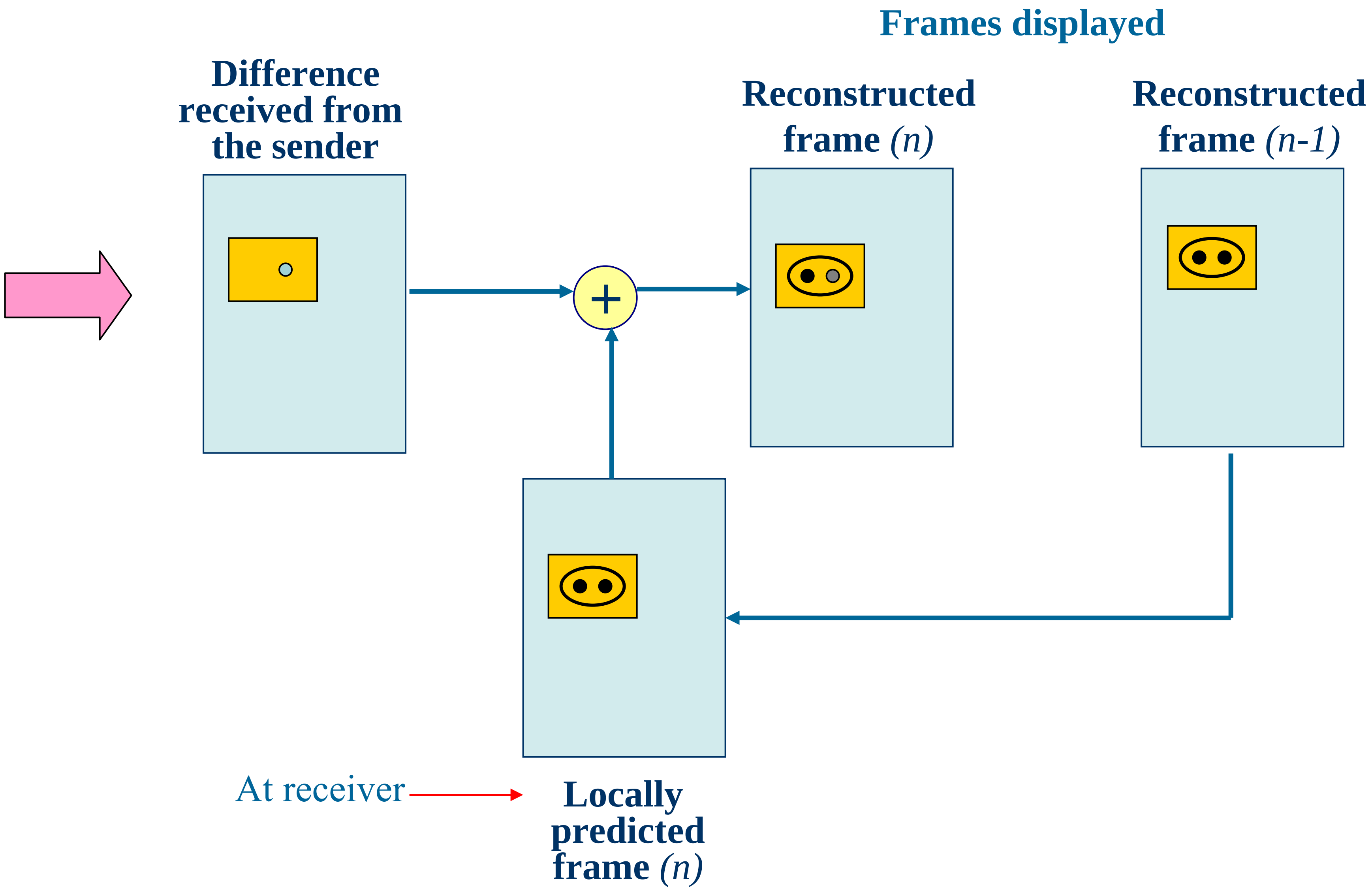
The simplest case: stationary picture. Details of the picture are not moving, only pixel values are changing from frame to frame. The illustration below ignores DCT for the sake of simplicity (with DCT a much better spatial compression can be achieved).


Sender
Frame *(n-1)*
Frame *(n)*
Frames generated by the source (camera)
Co-sited blocks of pixels
Difference sent to receiver
+
-
At sender
Locally predicted frame *(n-1)*
Locally predicted frame *(n)*

1. Most of pixels = 0.
2. Differences of details that have a change have pixel values in smaller range – can be encoded with fewer bits

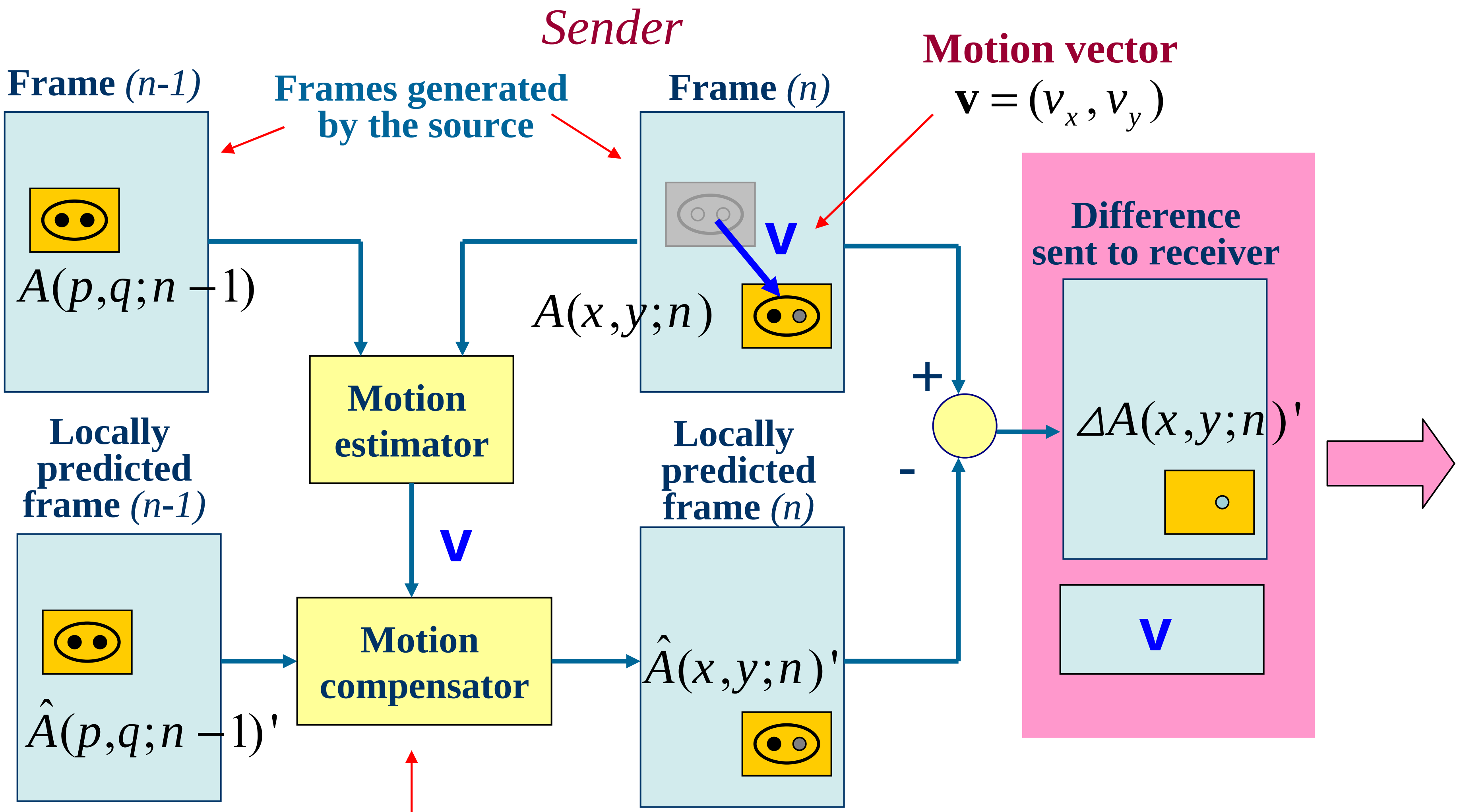
Forward Interframe Prediction (cont.)

Receiver


Frames displayed
Difference received from the sender
Reconstructed frame (n)
Reconstructed frame (n-1)
+
At receiver
Locally predicted frame (n)

Forward Interframe Prediction (cont.)

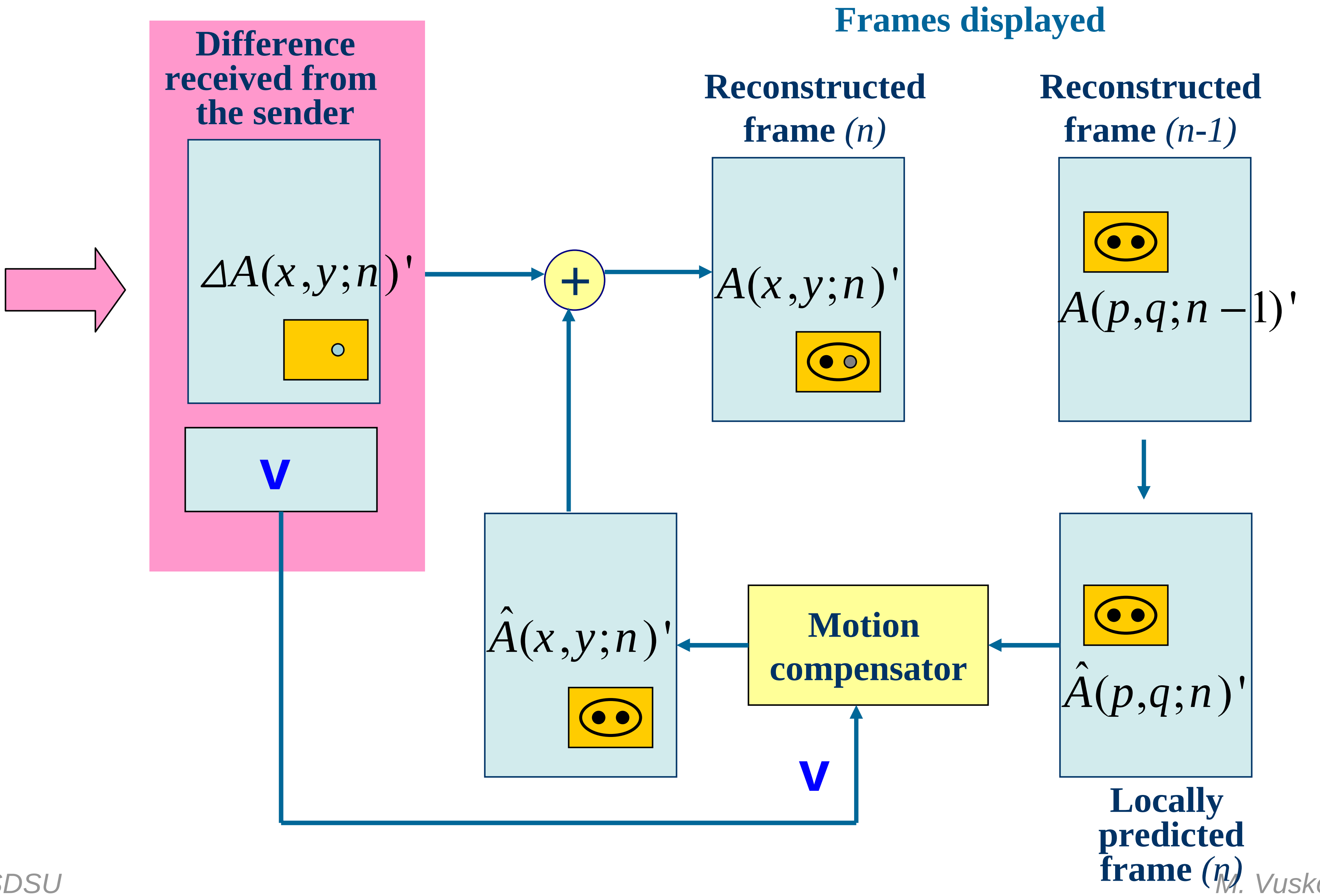
Less simple case: picture detail moves by translation, blocks are no longer co-sited.



Motion compensation: $\hat{A}(x,y;n)' = \hat{A}(x-v_x, y-v_y; n-1)'$

Forward Interframe Prediction (cont.)

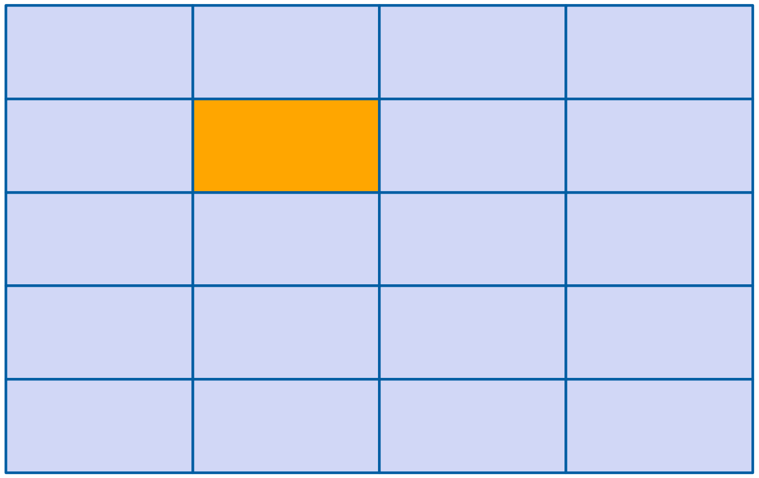
Receiver


Frames displayed
Difference received from the sender
$\Delta A(x,y;n)'$
V
+
Reconstructed frame *(n)*
$A(x,y;n)'$
Reconstructed frame *(n-1)*
$A(p,q;n-1)'$
$\hat{A}(x,y;n)'$
Motion compensator
V
$\hat{A}(p,q;n)'$
Locally predicted frame *(n)*

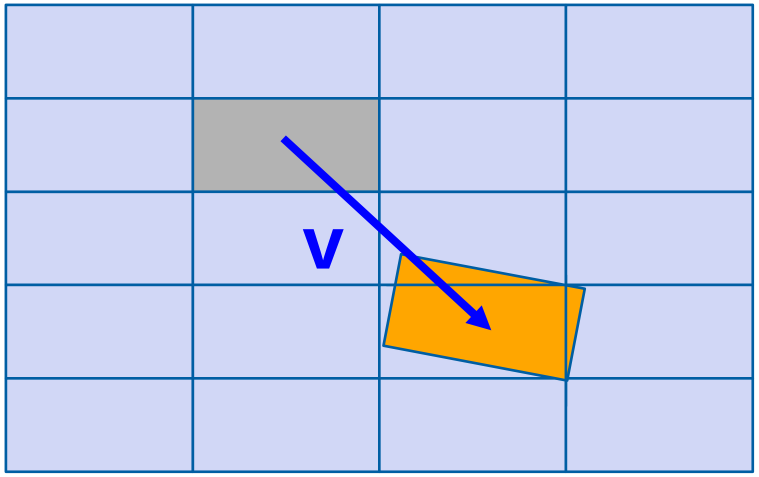
Forward Interframe Prediction (cont.)

Even less simple case: <u>picture detail moves by translation and rotation</u>.

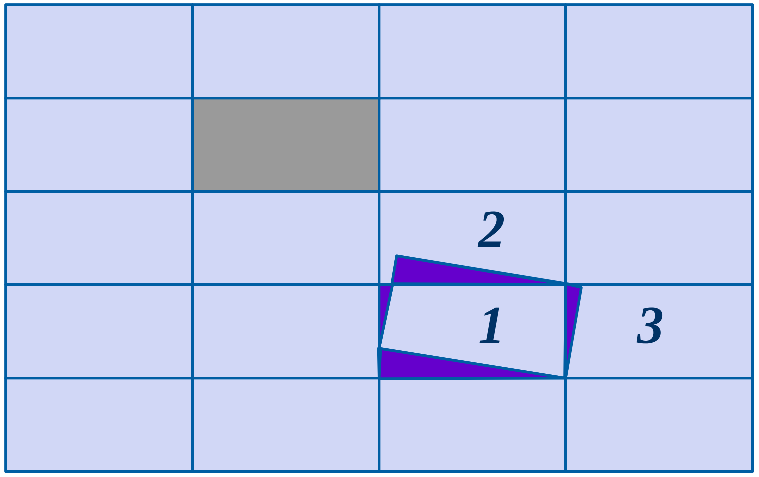
Frame ***(n-1)***



Frame ***(n)***



Difference ***(n)***



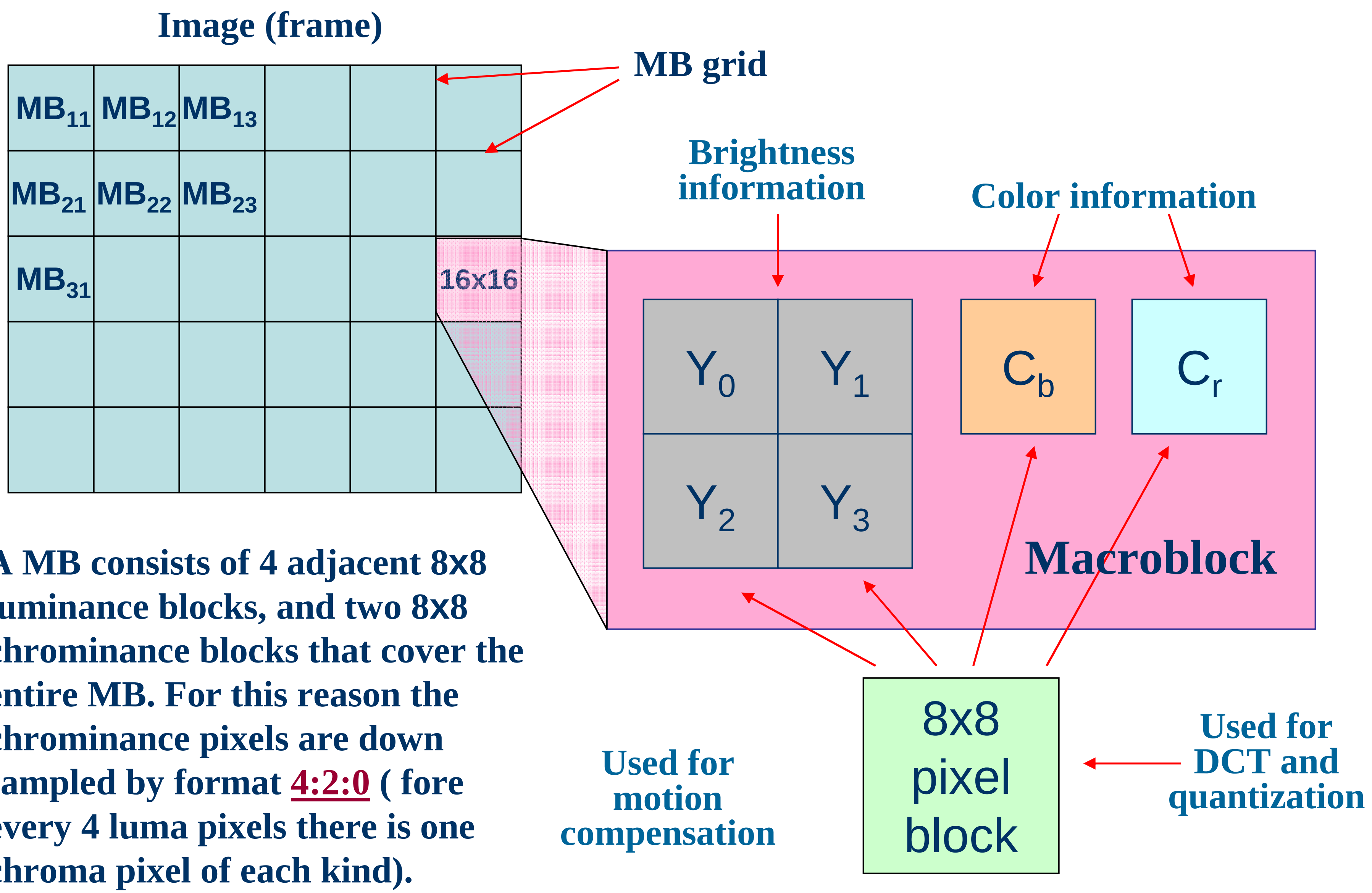
That is OK. There will be more error details. In addition, more blocks will be affected. In this example blocks marked *1*, *2* and *3* will have some error, which will be taken care by the motion compensated prediction algorithm.

Macroblocks

- What happens if the errors mentioned in the previous slide become too big? Then the sending of difference images could require more information, i.e. more bits, than sending the image itself (i.e. its quantized DCT).
- What if different parts of the image move in different directions and by different amounts?

Both problems are addressed by introducing macrobloks (MB). Entire image is divided into non-overlapping blocks, whereby each MB has its own motion vector. The size of the MB is a compromise between decreasing errors and decreasing the amount of additional information that needs to be sent along with the difference images, that is the motion vectors. MPEG has chosen 16 x 16 pixel MBs.

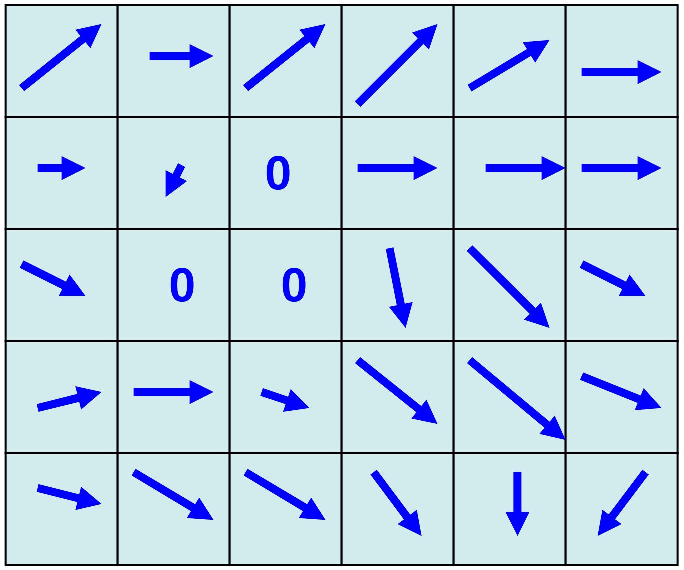
Macroblocks (cont.)



A MB consists of 4 adjacent 8x8 luminance blocks, and two 8x8 chrominance blocks that cover the entire MB. For this reason the chrominance pixels are down sampled by format 4:2:0 (fore every 4 luma pixels there is one chroma pixel of each kind).

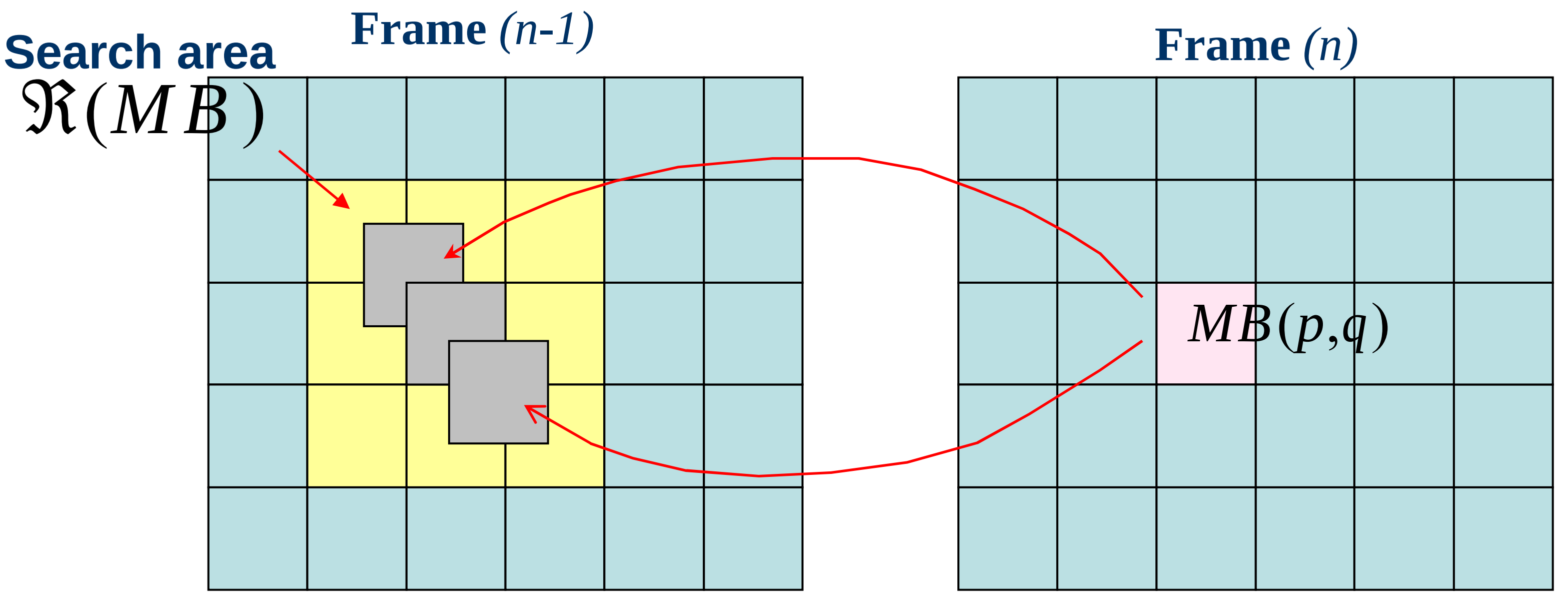
Macroblocks (cont.)

Motion vectors in a frame



Motion Estimation

The motion estimation algorithm is not standardized by MPEG or other protocols, they are left to the implementer (vendor). A straightforward approach would be the exhaustive search within some neighborhood of the MB, called search area.



For a given MB in frame *(n)* find the best matching block in frame *(n-1)*. The best matching block has the size of MB, but doesn't have to be aligned with the MB grid.

Assumption: Sub pixel motions can be ignored

Motion Estimation (cont)

In order to define what is the best matching block, a matching criteria has to be established. A commonly used measure of similarity is the mean-square error:

Mean-square error for a single vector v = *(x,y)*

Current image

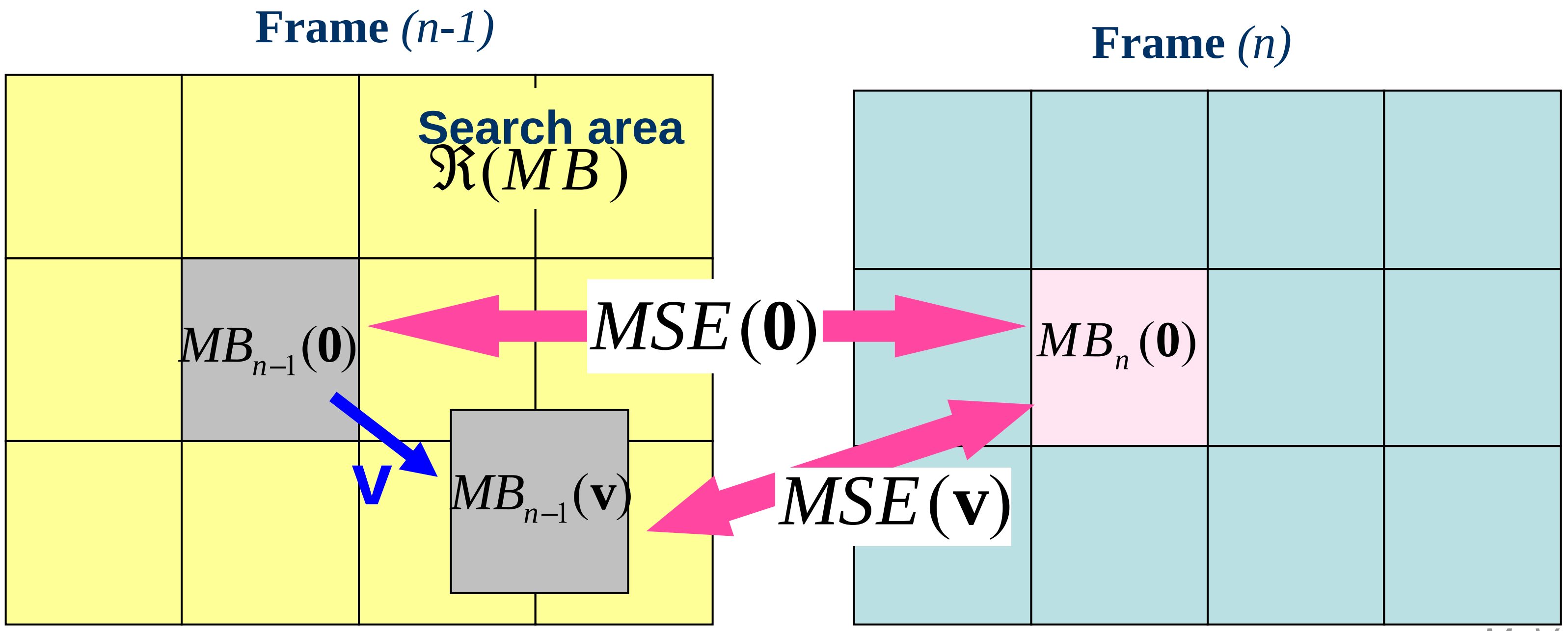
Previous image

Only the luminance pixel values (Y component) are used in this equation.

$$MSE(\mathbf{v}) = \sum_{(i,j)\in MB} \left[S_n(i,j) - S_{n-1}(i+x, j+y)\right]^2, \quad \mathbf{v} = (x,y) \in \mathrm{V}(MB)$$

Image indices must fall into MB

Values of v = *(x,y)* are chosen such to stay in the search area



Motion Estimation (cont.)

The best match is the one with the minimal MSE:

$$\mathbf{v}_o = (x_o, y_o) = \underset{\mathbf{v} \in V(MB)}{\arg\min} \; MSE(\mathbf{v})$$

$$MSE(\mathbf{v}_o) = \min_{\mathbf{v} \in V(MB)} MSE(\mathbf{v})$$

where $\mathbf{v}_o = (x_o, y_o)$ is the optimal value of v, the estimated motion vector.

There are several cases:

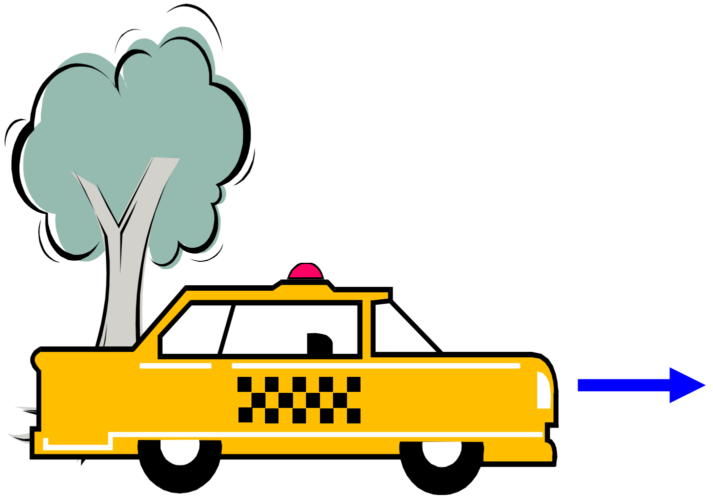
$MSE(\mathbf{0}) = 0$	- MB didn't move and didn't change the luminance
$MSE(\mathbf{v}_o) = 0$	- MB moved by $\mathbf{v}_o$ but didn't change the luminance
$MSE(\mathbf{v}_o) \le \text{maxerr}$	- MB moved by $\mathbf{v}_o$ and changed the luminance
$MSE(\mathbf{v}_o) > \text{maxerr}$	- error to big, couldn't find the matching block, use intra-coding for this MB

**The best search strategy is left to the implementer and is still a matter of research.
The algorithm above is called exhaustive block-matching algorithm.
There are rapid ("intelligent") search strategies, which are not considered here.**

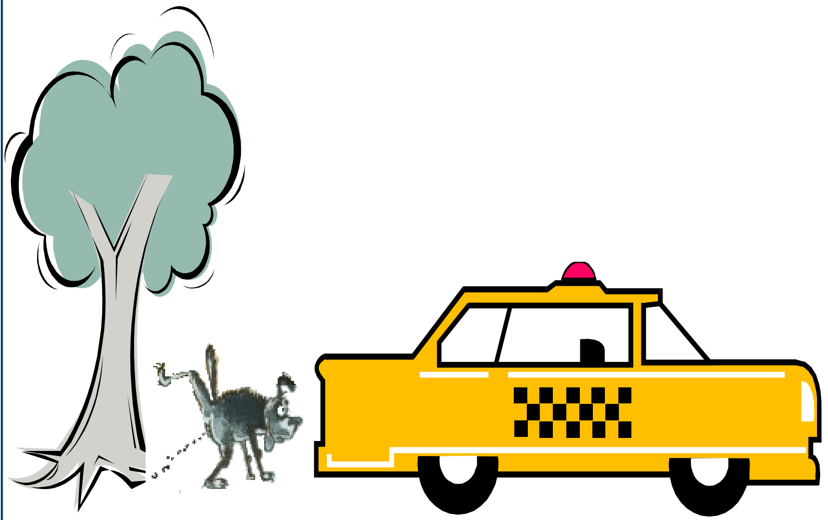
Bi-Directional Interframe Prediction

What happens if the moving object <u>covers or uncovers</u> new objects?
In the case below the forward interframe prediction would cause to much information in the difference frame. Better result would have been achieved if the prediction was based on the future frame *(n)* instead of the past frame *(n-1)*.

Frame *(n-1)*



Frame *(n)*



Bi-Directional Interframe Prediction (cont.)

For that reason MPEG specifies bi-directional motion compensation which allows a choice between forward or backward prediction. This much better handles covered or uncovered regions.

Current frame **Previous frame** **Future frame**

$$\hat{A}(x,y;n)' = \alpha_b \hat{A}(x - v_{bx}, y - v_{by}; n-1)' + \alpha_f \hat{A}(x - v_{fx}, y - v_{fy}; n+1)'$$

Motion vector for frame *(n-1)*

Motion vector for frame *(n+1)*

α_b	α_f	
0	1	Forward prediction
1	0	Backward prediction
0.5	0.5	Combined

Weighting coefficients α_b and α_f are side information that has to be sent, along with the motion vectors, to the receiver.

Bi-Directional Interframe Prediction (cont.)

How can we make a prediction based on future frames?

Send the future frame before the actual frame. This requires reordering of frames that are naturally generated in display order, into rearranged transmit order.
For that purpose MPEG defines three kinds of frames:

- **I-frames (intra frames)**
  Frames encoded entirely in intra-mode. Only spatial compression, no temporal compression (like JPEG). More information transmitted. This frame is used as a reference for predictive and bi-directional coding. Used periodically to stop error propagation (video conferencing however ca not afford periodic refresh).

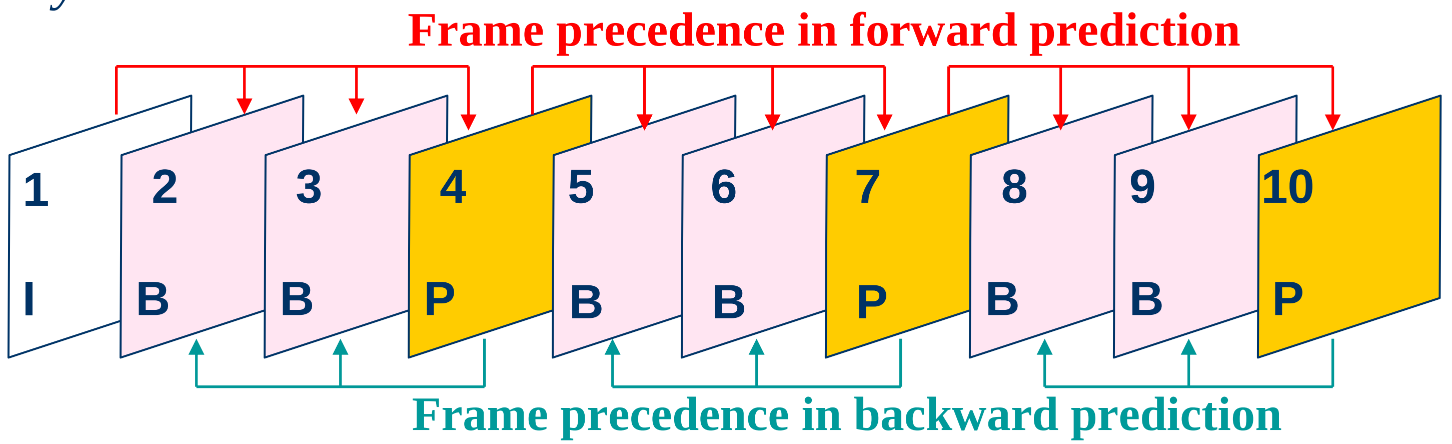
- **P-frames (predictive frames)**
  Frames used for forward prediction. Enable both, spatial and temporal compression. Less information to transfer, higher compression ratio. In order to decode a P-frame the previous I-frame or P-frame needs to be used. Along with P-frame the corresponding motion vectors have to be transmitted. A P-frame has three to five times less bits than an I-frame.

- **B-frames (bi-directional frames)**
  Used for bi-directional decoding. Contain picture difference between the previous, or the future P-frame or I-frame (whichever needs less bits for transfer). Contains picture difference, motion vectors and side information. Has number of bits about 50% of P-frames. Provides better quality of decoded pictures, but requires reordering in transmission, which introduces delay. Can’t be used in real-time streams (interactive video, video conferencing)

Group of Pictures

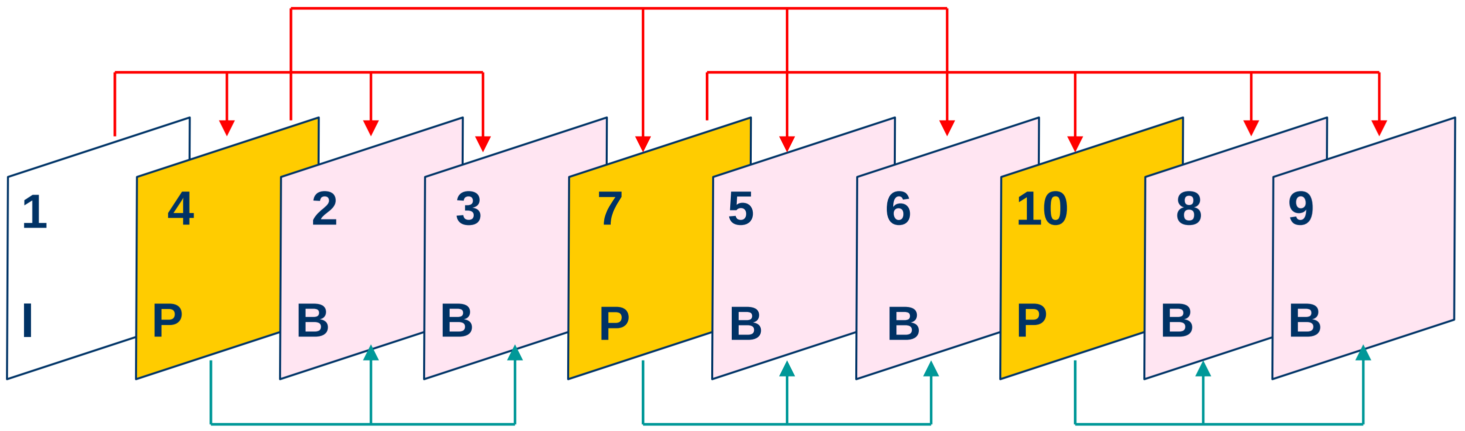
The three types of frames are typically arranged in a sequence called group of pictures (GOP):

Display order:


Frame precedence in forward prediction
1
I
2
B
3
B
4
P
5
B
6
B
7
P
8
B
9
B
10
P
Frame precedence in backward prediction

Parameters of this GOP structure:
N = 10 **(number of pictures in GOP)**
M = 3 **(spacing of P-frames)**

Transmit order:



Group of Pictures (cont.)

Comments:

- **MPEG-2 doesn’t insist on a fixed GOP structure. For example, after a shot change a new GOP sequence can be started, which requires an I-frame to start with.**
- **Transmitted I/P/B frames are sometimes called anchor frames. They carry the new information needed for the reconstruction of true frames. The previously reconstructed I and P frames are called reference frames. Frames to be reconstructed are called target frames.**
- **I-frames provide random access point into the bit stream of a GOP.**
- **A frame can contain different (and contradictory) motions. Therefore not all MBs of a P-frame are predicted. If the prediction error is too big the coder can decide to intra-code that MB. Similarly, the MBs of a B-frame can be forward, backward or combined predicted, or intra-coded.**
- **B-frames can be discarded without error propagation (temporal scalability)**

Group of Pictures (cont.)

Compression Ratio of GOP

Average compression factor of a single picture:

$$CF = \frac{b}{\mathrm{c}}$$

b – number of bits for uncompressed image
c – number of bits for compressed image

Average compression factor of a GOP:

$$CF = \frac{b\,(n_I + n_P + n_B)}{n_I c_I + n_P c_P + n_B c_B} = \frac{n_I + n_P + n_B}{n_I \dfrac{1}{CF_I} + n_P \dfrac{1}{CF_P} + n_B \dfrac{1}{CF_B}}$$

n_i – number of pictures of type *i* (*i* = I, P, B)
c_i – number of bits for compressed image of tipe *i*
CF_c – average compression factor for picturs of type *i*

Example:

Given is GOP: `IPBBPBBPBBPBBP`. Average compression ratios for picture types are typically $CF_I = 8$, $CF_P = 25$, $CF_B = 100$. The average compression factor for a GOP is:

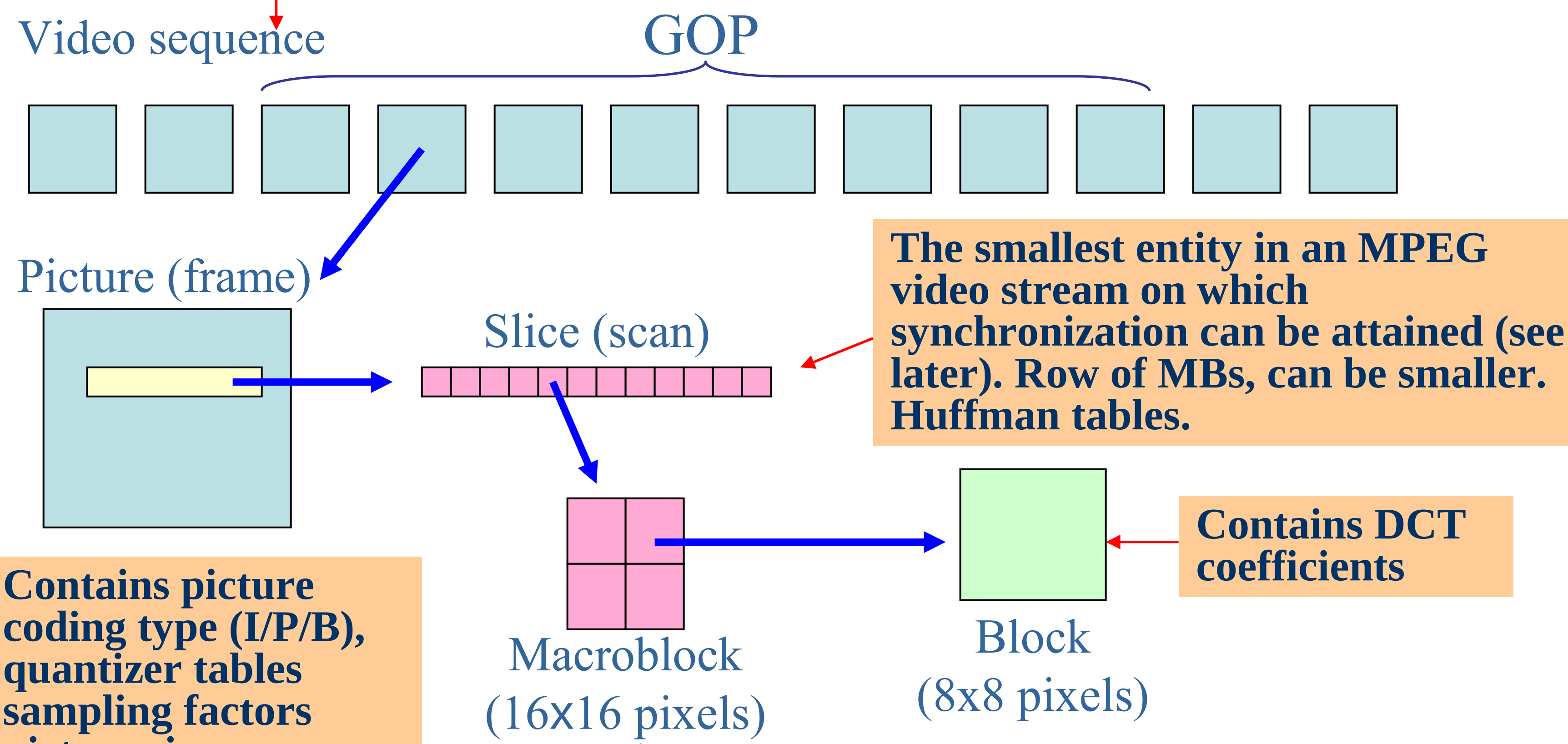
$$CF_{GOP} = \frac{1+5+8}{1\frac{1}{8} + 5\frac{1}{25} + 8\frac{1}{100}} = \frac{14}{0.405} = 34.6$$

Picture Syntax

A video stream can be viewed as a hierarchy of structures:

Video sequence contains all necessary information to produce decoded video signal. Header: picture size, frame rate,…Has 100s and 1000s of frames.

Small number of frames (typically 10 to 12)



The smallest entity in an MPEG video stream on which synchronization can be attained (see later). Row of MBs, can be smaller. Huffman tables.

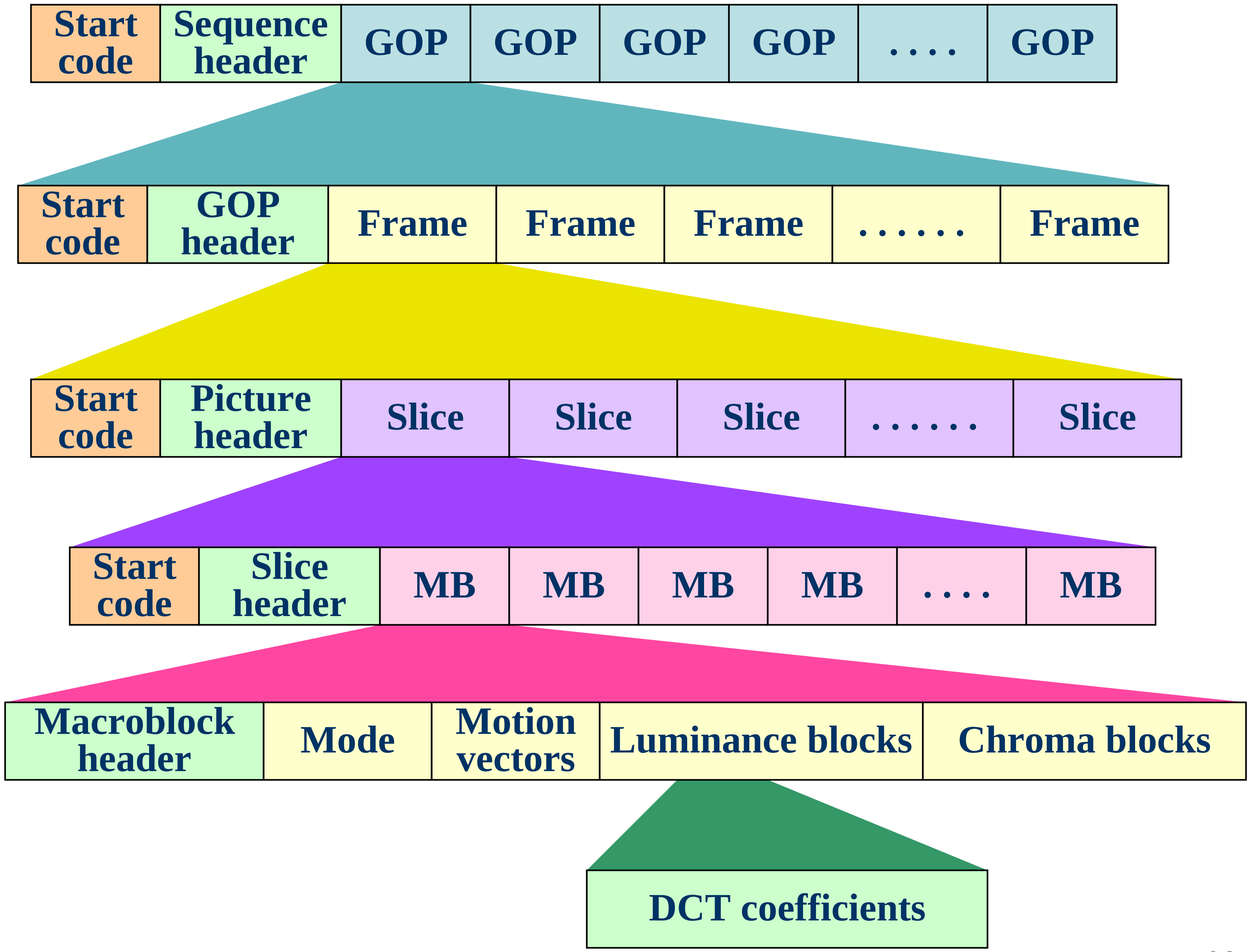
Contains DCT coefficients

Contains picture coding type (I/P/B), quantizer tables sampling factors picture size

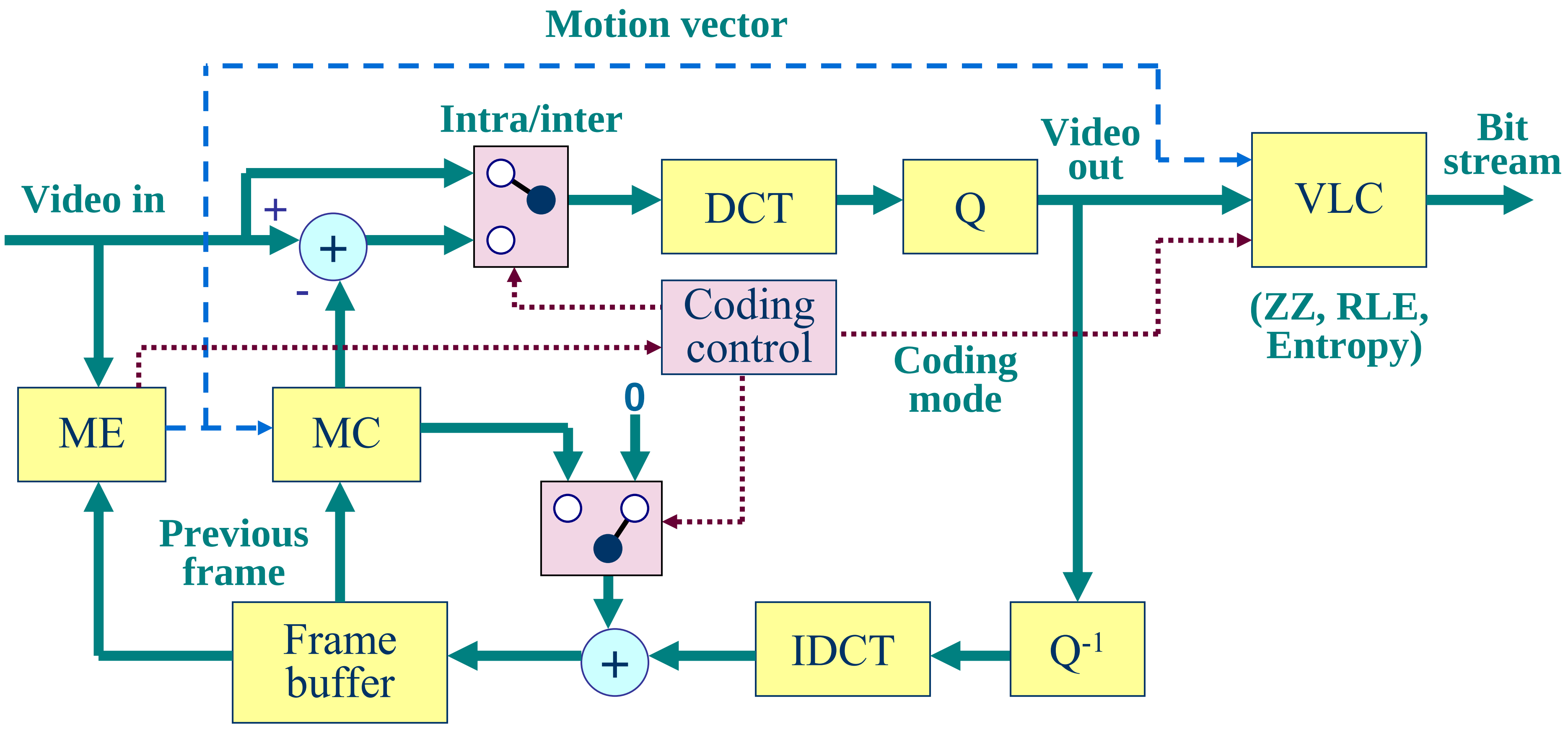
Contains spatial address, motion vectors, prediction modes, quantizer step size

MPEG Bit Stream

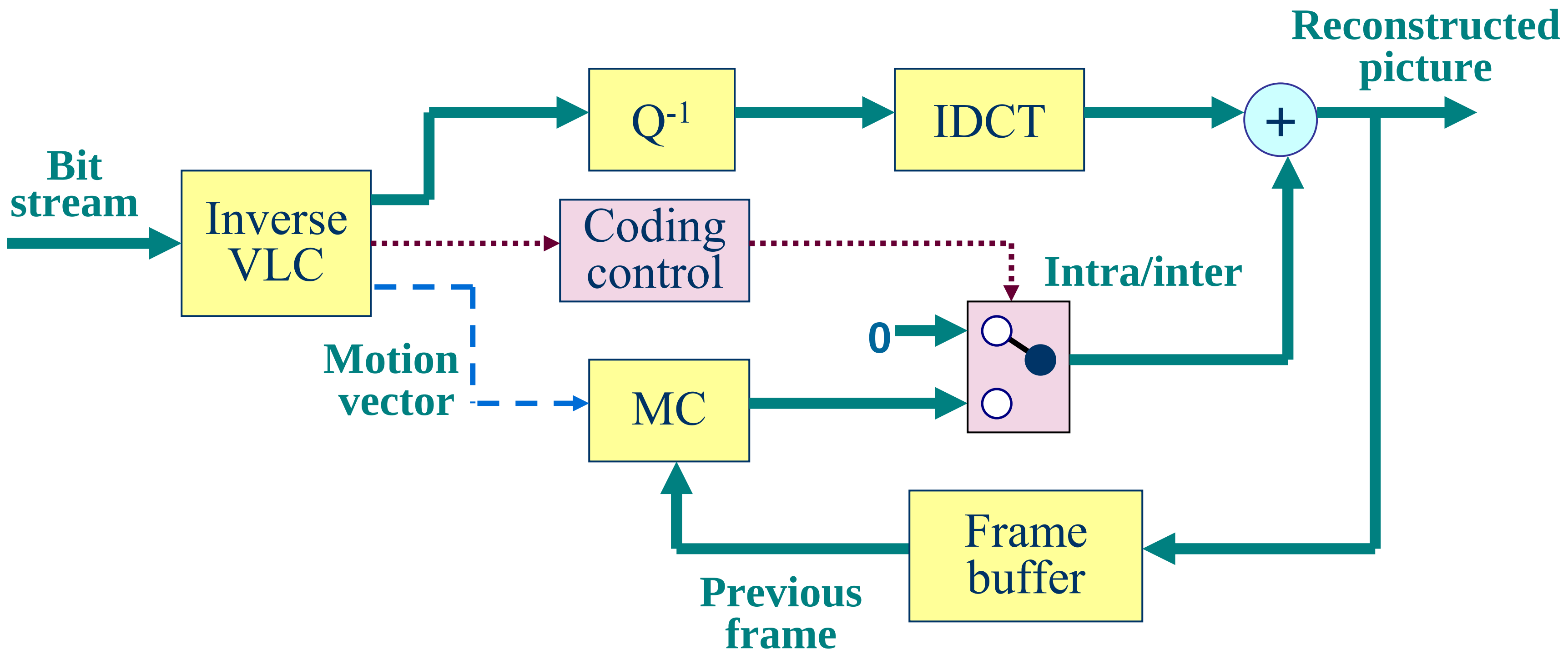
Encoded bits of a video sequence are framed through several layers:



MPEG Video Coder



MPEG Video Decoder

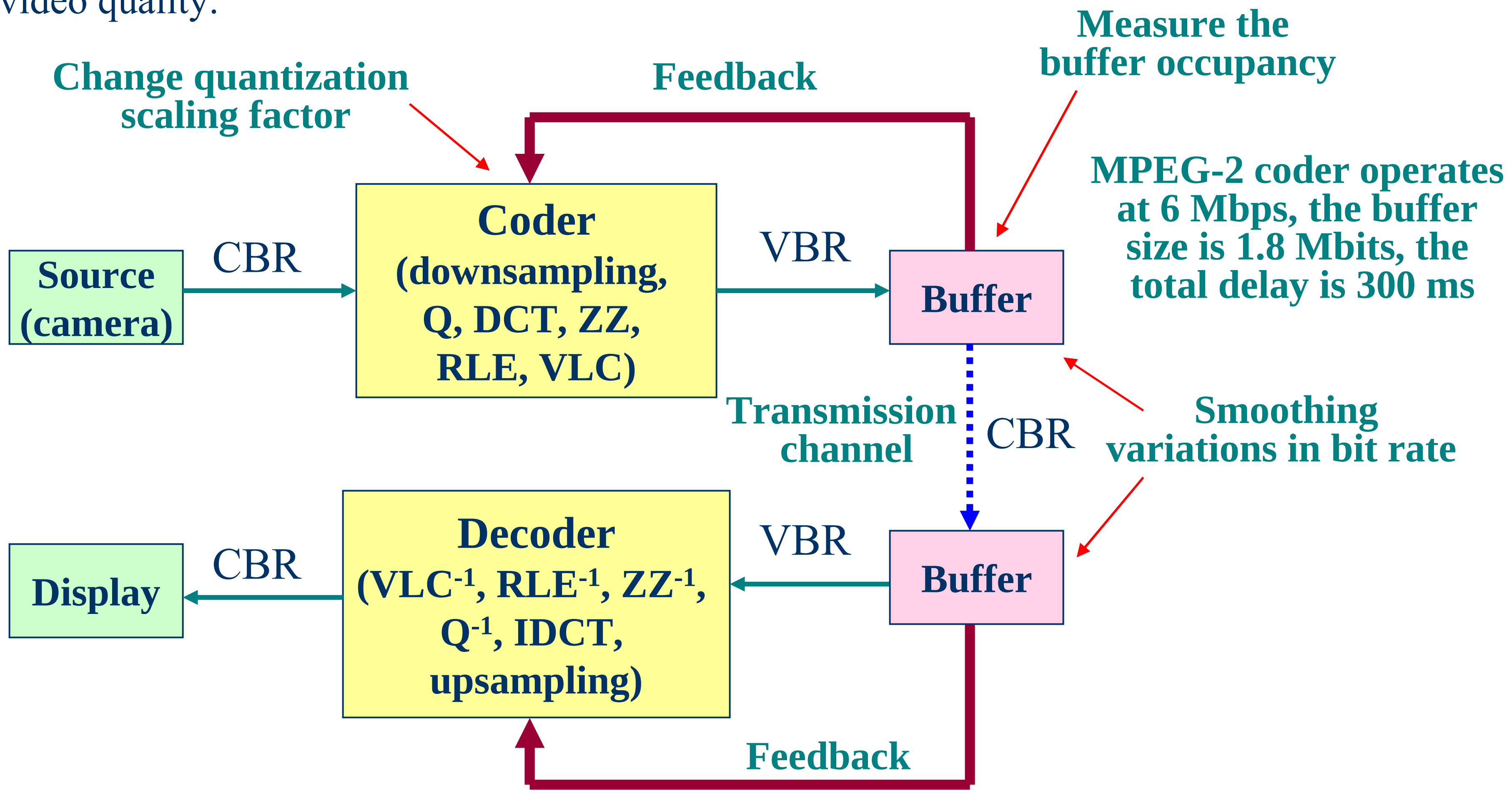


Comment:

Decoder is much simpler (no motion estimation).
Typically encoder is implemented in hardware,
while decoder is implemented in software

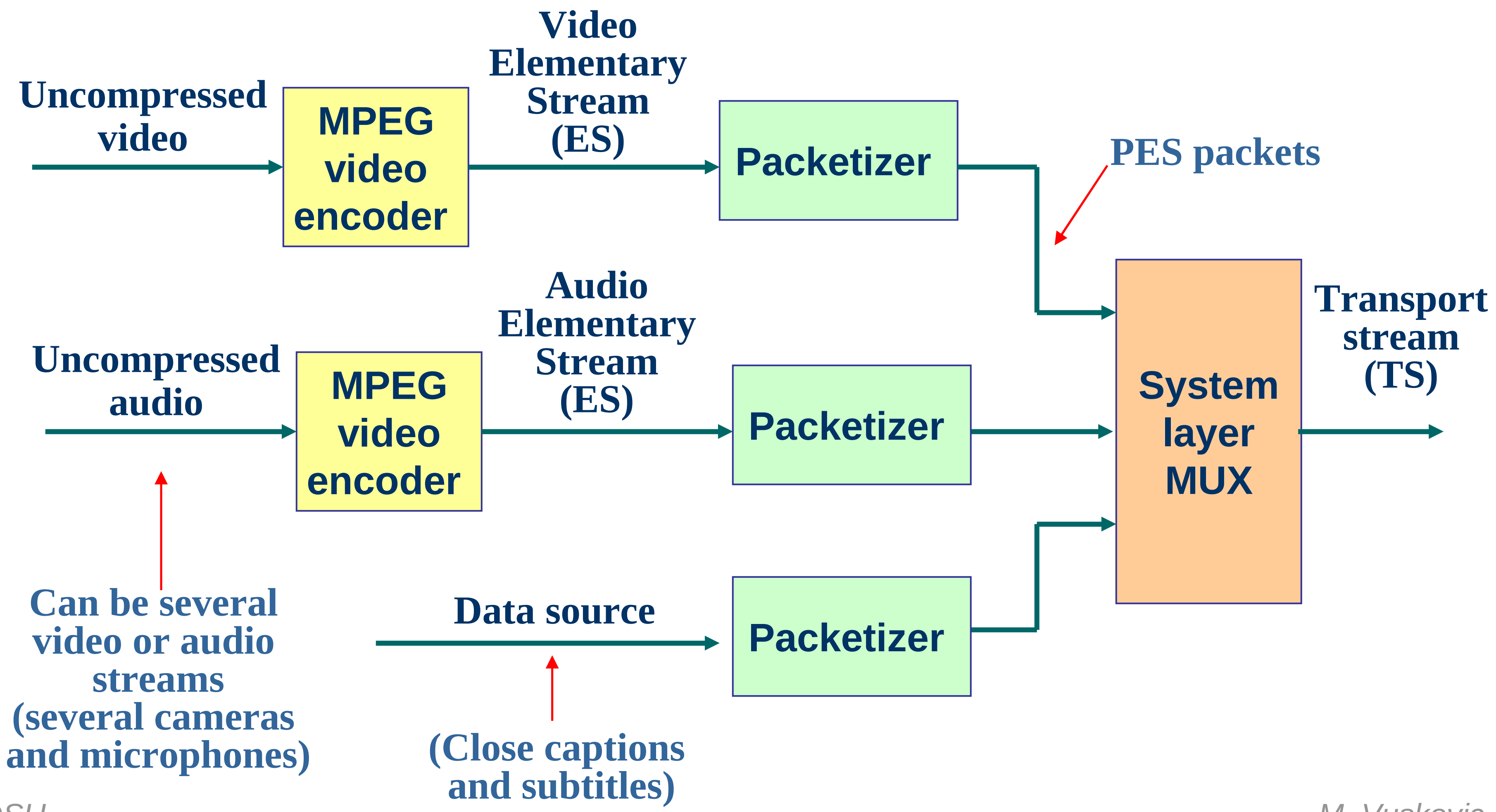
Bit Rate Control

Removing the spatial and temporal redundancy produces a variable bit rate (VBR) bit stream. The variations depend on the picture complexity and on the amount and type of motion. Since many applications (including the transmission) require constant bit stream (CBR) a smoothing buffer is needed. The buffer feedback makes sure that the buffer won't under- or overflow. The bit rate can be controlled by changing the quantization step size through the scaling factor (SF). Larger buffers introduce more delay, while smaller buffers cause lower video quality.


Change quantization scaling factor
Feedback
Measure the buffer occupancy
Source (camera)
CBR
Coder (downsampling, Q, DCT, ZZ, RLE, VLC)
VBR
Buffer
MPEG-2 coder operates at 6 Mbps, the buffer size is 1.8 Mbits, the total delay is 300 ms
Transmission channel
CBR
Smoothing variations in bit rate
Display
CBR
Decoder (VLC^{-1}, RLE^{-1}, ZZ^{-1}, Q^{-1}, IDCT, upsampling)
VBR
Buffer
Feedback

MPEG Transport Stream

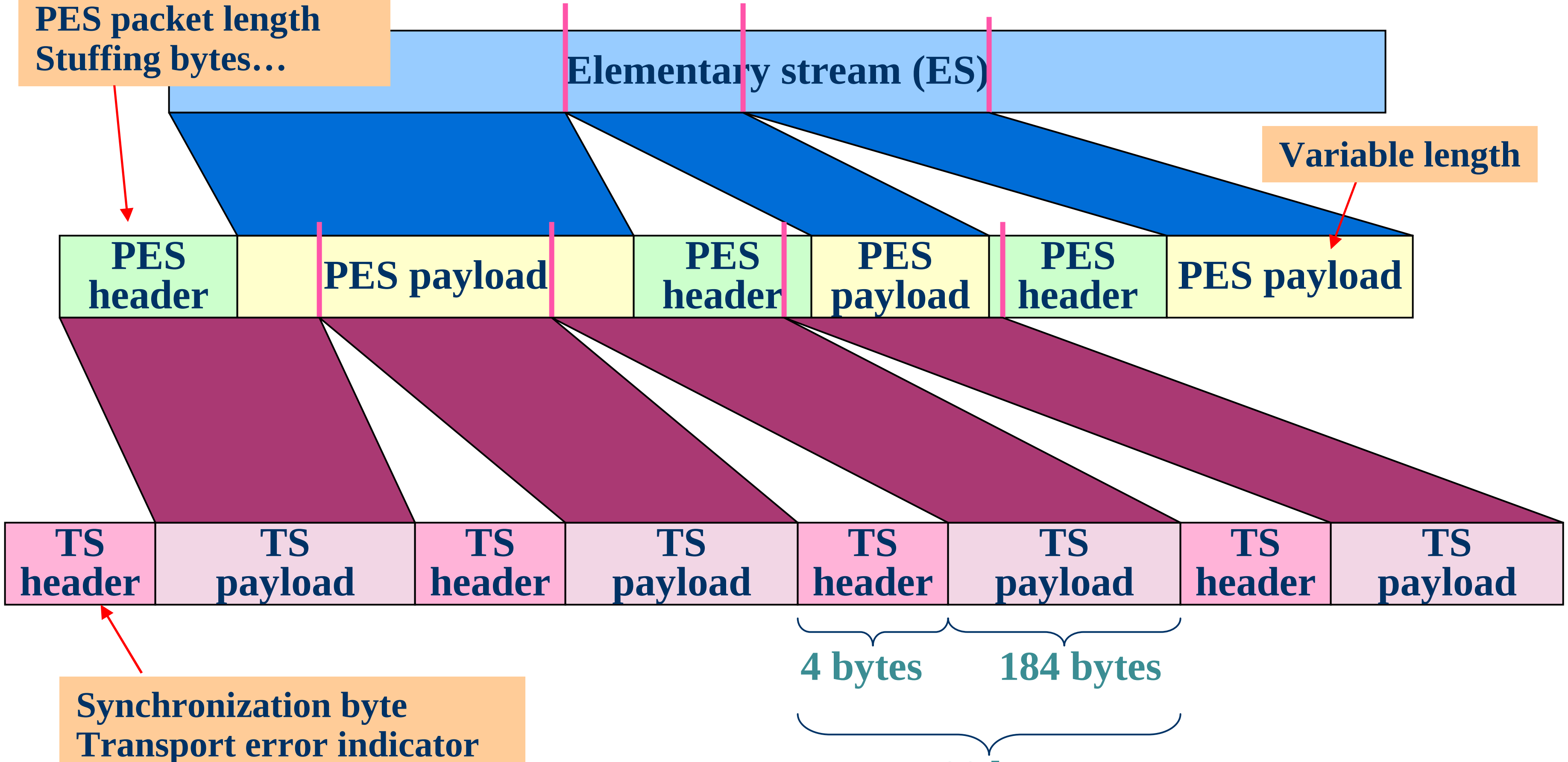
Audio, video streams and associated user data streams are usually combined together thus forming transport streams by using a system layer multiplexer. Without this, we would watch a silent movie. The receiver has similar diagram that contains a demultiplexer, depacetizers and decoders.



MPEG Transport Stream (cont.)

Format of transport stream

Packet start indicator
Stream identifier
PES packet length
Stuffing bytes…



Synchronization byte
Transport error indicator
PES payload start indicator
PES ID (PID – 13 bits)…

TS packets have small and constant size to limit the packetization delay

MPEG Transport Stream (cont.)

Comments:

- Size of TS packets (188 bytes) is decided so that is can comply with ATM transport: ATM cells have length of 53 bytes, with a payload of 48 bytes. One byte is used for ATM adaptation layer header leaving 47 byes. A TS packet contains four adaptation payloads: 4 x 47 = 188.
- TS packets can be transported with ATM, PDH, SDH, Ethernet, TCP/IP, UDP/IP or RTP/UDP/IP (RTP is real-time protocol, discussed in the next chapter)
- As shown in next chapter, RTP can encapsulate TS packets, and video or audio ES as well.

PDH – Plesiohronous Digital Hierarchy (like T1, T3, …)
SDH – Synchronous Digital Hierarchy (like SONET)

Compression Standards

Standard	*Compressed rate*	*Comment*
H.261	x 64 kbps	Video conferencing over low quality links (uses only I and P frames)
H.263	< 64 kbps	VC over low bits rate channels
MPEG 1	< 1.5 Mbps	VHS quality video storage (uses I, P and B frames)
MPEG 2 (H.262)	< 4 .. 100 Mbps	Digital TV broadcasting HDTV (uses I, P and B frames)
MPEG 4	5 kbps … x Mbps	Versatile multimedia coding (not discussed in this chapter)