

Video Compression

Djordje Mitrovic – University of Edinburgh

This document deals with the issues of video compression. The algorithm, which is used by the MPEG standards, will be elucidated upon in order to explain video compression. Only visual compression will be discussed (no audio compression). References and links to further readings will be provided in the text.

What is compression?

Compression is a reversible conversion (*encoding*) of data that contains fewer bits. This allows a more efficient storage and transmission of the data. The inverse process is called *decompression* (*decoding*). Software and hardware that can encode and decode are called *decoders*. Both combined form a *codec* and should not be confused with the terms *data container* or *compression algorithms*.

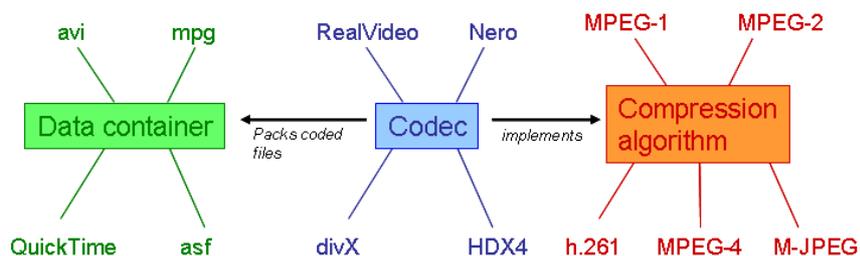


Figure 1: Relation between codec, data containers and compression algorithms.

Lossless compression allows a 100% recovery of the original data. It is usually used for text or executable files, where a loss of information is a major damage. These compression algorithms often use statistical information to reduce redundancies. *Huffman-Coding* [1] and *Run Length Encoding* [2] are two popular examples allowing high compression ratios depending on the data.

Using *lossy compression* does not allow an exact recovery of the original data. Nevertheless it can be used for data, which is not very sensitive to losses and which contains a lot of redundancies, such as images, video or sound. Lossy compression allows higher compression ratios than lossless compression.

Why is video compression used?

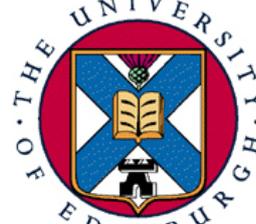
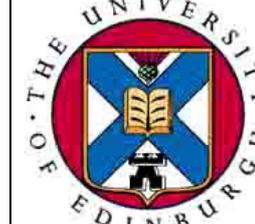
A simple calculation shows that an uncompressed video produces an enormous amount of data: a resolution of 720x576 pixels (PAL), with a refresh rate of 25 fps and 8-bit colour depth, would require the following bandwidth:

$$720 \times 576 \times 25 \times 8 + 2 \times (360 \times 576 \times 25 \times 8) = 1.66 \text{ Mb/s (luminance + chrominance)}$$

For *High Definition Television (HDTV)*:

$$1920 \times 1080 \times 60 \times 8 + 2 \times (960 \times 1080 \times 60 \times 8) = 1.99 \text{ Gb/s}$$

Even with powerful computer systems (storage, processor power, network bandwidth), such data amount cause extreme high computational demands for managing the data. Fortunately, digital video contains a great deal of *redundancy*. Thus it is suitable for compression, which can reduce these problems significantly. Especially lossy compression techniques deliver high compression ratios for video data. However, one must keep in mind that there is always a trade-off between data size (therefore computational time) and quality. The higher the compression ratio, the lower the size and the lower the quality. The encoding and decoding process itself also needs computational

	
Uncompressed Image 72KB	Compressed Image has only 28 KB, but its has worse quality

resources, which have to be taken into consideration. It makes no sense, for example for a real-time application with low bandwidth requirements, to compress the video with a computational expensive algorithm which takes too long to encode and decode the data.

Image and Video Compression Standards

The following compression standards are the most known nowadays. Each of them is suited for specific applications. Top entry is the lowest and last row is the most recent standard. The MPEG standards are the most widely used ones, which will be explained in more details in the following sections.

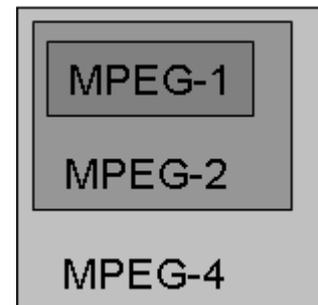
Standard	Application	Bit Rate
JPEG	Still image compression	Variable
H.261	Video conferencing over ISDN	P x 64 kb/s
MPEG-1	Video on digital storage media (CD-ROM)	1.5Mb/s
MPEG-2	Digital Television	2-20 Mb/s
H.263	Video telephony over PSTN	33.6-? kb/s
MPEG-4	Object-based coding, synthetic content, interactivity	Variable
JPEG-2000	Improved still image compression	Variable
H.264/ MPEG-4 AVC	Improved video compression	10's to 100's kb/s

Adapted from [3]

The MPEG standards

MPEG stands for *Moving Picture Coding Experts Group* [4]. At the same time it describes a whole family of international standards for the compression of audio-visual digital data. The most known are *MPEG-1*, *MPEG-2* and *MPEG-4*, which are also formally known as ISO/IEC-11172, ISO/IEC-13818 and ISO/IEC-14496. More details about the MPEG standards can be found in [4],[5],[6]. The most important aspects are summarised as follows:

The MPEG-1 Standard was published 1992 and its aim was it to provide VHS quality with a bandwidth of 1,5 Mb/s, which allowed to play a video in real time from a 1x CD-ROM. The frame rate in MPEG-1 is locked at 25 (PAL) fps and 30 (NTSC) fps respectively. Further MPEG-1 was designed to allow a fast forward and backward search and a synchronisation of audio and video. A stable behaviour, in cases of data loss, as well as low computation times for encoding and decoding was reached, which is important for symmetric applications, like video telephony.



In 1994 MPEG-2 was released, which allowed a higher quality with a slightly higher bandwidth. MPEG-2 is compatible to MPEG-1. Later it was also used for *High Definition Television (HDTV)* and *DVD*, which made the MPEG-3 standard disappear completely. The frame rate is locked at 25 (PAL) fps and 30 (NTSC) fps respectively, just as in MPEG-1. MPEG-2 is more scalable than MPEG-1 and is able to play the same video in different resolutions and frame rates.

MPEG-4 was released 1998 and it provided lower bit rates (10Kb/s to 1Mb/s) with a good quality. It was a major development from MPEG-2 and was designed for the use in interactive environments, such as multimedia applications and video communication. It enhances the MPEG family with tools to lower the bit-rate individually for certain applications. It is therefore more adaptive to the specific area of the video usage. For multimedia producers, MPEG-4 offers a better reusability of the contents as well as a copyright protection. The content of a frame can be grouped into object, which can be accessed individually via the *MPEG-4 Syntactic Description Language (MSDL)*. Most of the tools require immense computational power (for encoding and decoding), which makes them impractical for most "normal, non-professional user" applications or real time applications. The real-time tools in MPEG-4 are already included in MPEG-1 and MPEG-2. More details about the MPEG-4 standard and its tool can be found in [7].

The MPEG Compression

The MPEG compression algorithm encodes the data in 5 steps [6], [8]:

First a *reduction of the resolution* is done, which is followed by a *motion compensation* in order to reduce temporal redundancy. The next steps are the *Discrete Cosine Transformation (DCT)* and a *quantization* as it is used for the JPEG compression; this reduces the spatial redundancy (referring to human visual perception). The final step is an *entropy coding* using the *Run Length Encoding* and the *Huffman coding* algorithm.

Step 1: Reduction of the Resolution

The human eye has a lower sensibility to colour information than to dark-bright contrasts. A conversion from RGB-colour-space into YUV colour components help to use this effect for compression. The chrominance components U and V can be reduced (subsampling) to half of the pixels in horizontal direction (4:2:2), or a half of the pixels in both the horizontal and vertical (4:2:0).

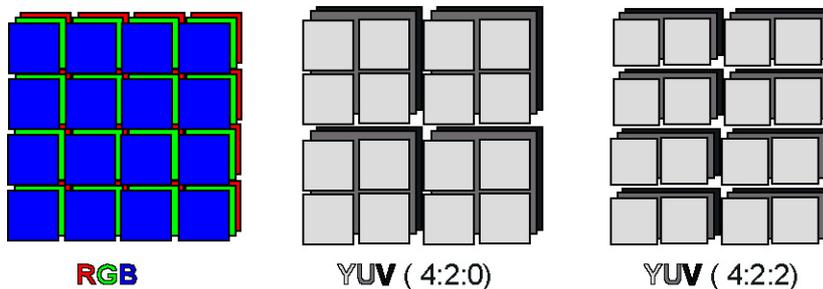


Figure 2: Depending on the subsampling, 2 or 4 pixel values of the chrominance channel can be grouped together.

The subsampling reduces the data volume by 50% for the 4:2:0 and by 33% for the 4:2:2 subsampling:

$$|Y| = |U| = |V|$$

$$4:2:0 = \frac{|Y| + \frac{1}{4}|U| + \frac{1}{4}|V|}{|Y| + |U| + |V|} = \frac{1}{2}$$

$$4:2:2 = \frac{|Y| + \frac{1}{2}|U| + \frac{1}{2}|V|}{|Y| + |U| + |V|} = \frac{2}{3}$$

MPEG uses similar effects for the audio compression, which are not discussed at this point.

Step 2: Motion Estimation

An MPEG video can be understood as a sequence of frames. Because two successive frames of a video sequence often have small differences (except in scene changes), the MPEG-standard offers a way of reducing this temporal redundancy. It uses three types of frames:

I-frames (intra), *P-frames (predicted)* and *B-frames (bidirectional)*.

The I-frames are “key-frames”, which have no reference to other frames and their compression is not that high. The P-frames can be predicted from an earlier I-frame or P-frame. P-frames cannot be reconstructed without their referencing frame, but they need less space than the I-frames, because only the differences are stored. The B-frames are a two directional version of the P-frame, referring to both directions (one forward frame and one backward frame). B-frames cannot be referenced by other P- or B-frames, because they are interpolated from forward and backward frames. P-frames and B-frames are called *inter coded frames*, whereas I-frames are known as *intra coded frames*.

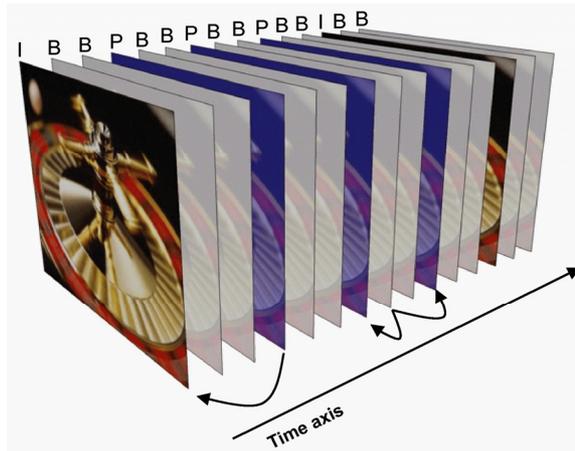


Figure 3: An MPEG frame sequence with two possible references: a P-frame referring to a I-frame and a B-frame referring to two P-frames.

The usage of the particular frame type defines the quality and the compression ratio of the compressed video. I-frames increase the quality (and size), whereas the usage of B-frames compresses better but also produces poorer quality. The distance between two I-frames can be seen as a measure for the quality of an MPEG-video. In practise following sequence showed to give good results for quality and compression level: **IBBPBBPBBPBBIBBP**.

The references between the different types of frames are realised by a process called *motion estimation or motion compensation*. The correlation between two frames in terms of motion is represented by a *motion vector*. The resulting frame correlation, and therefore the pixel arithmetic difference, strongly depends on how good the motion estimation algorithm is implemented. Good estimation results in higher compression ratios and better quality of the coded video sequence. However, motion estimation is a computational intensive operation, which is often not well suited for real time applications. Figure 4 shows the steps involved in motion estimation, which will be explained as follows:

Frame Segmentation - The Actual frame is divided into non-overlapping blocks (*macro blocks*) usually 8x8 or 16x16 pixels. The smaller the block sizes are chosen, the more vectors need to be calculated; the block size therefore is a critical factor in terms of time performance, but also in terms of quality: if the blocks are too large, the motion matching is most likely less correlated. If the blocks are too small, it is probably, that the algorithm will try to match noise. MPEG uses usually block sizes of 16x16 pixels.

Search Threshold - In order to minimise the number of expensive motion estimation calculations, they are only calculated if the difference between two blocks at the same position is higher than a threshold, otherwise the whole block is transmitted.

Block Matching - In general block matching tries, to “stitch together” an actual predicted frame by using snippets (blocks) from previous frames. The process of block matching is the most time consuming one during encoding. In order to find a matching block, each block of the current frame is compared with a past frame within a search area. Only the luminance information is used to compare the blocks, but obviously the colour information will be included in the encoding. The search area is a critical factor for the quality of the matching. It is more likely that the algorithm finds a matching block, if it searches a larger area. Obviously the number of search operations increases quadratically, when extending the search area. Therefore too large search areas slow down the encoding process dramatically. To reduce these problems often rectangular search areas are used, which take into

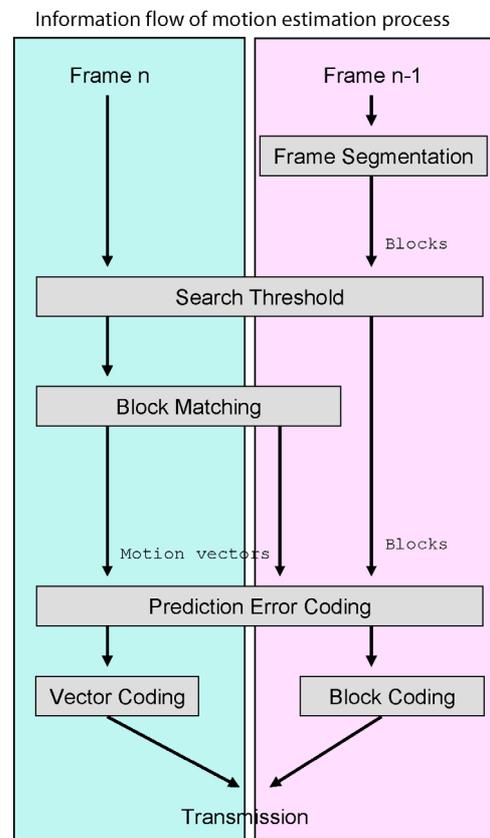


Figure 4: Schematic process of motion estimation. Adapted from [8]

account, that horizontal movements are more likely than vertical ones. More details about block matching algorithms can be found in [9], [10].

Prediction Error Coding - Video motions are often more complex, and a simple “shifting in 2D” is not a perfectly suitable description of the motion in the actual scene, causing so called *prediction errors* [13]. The MPEG stream contains a matrix for compensating this error. After prediction the, the predicted and the original frame are compared, and their differences are coded. Obviously less data is needed to store only the differences (yellow and black regions in Figure 5).

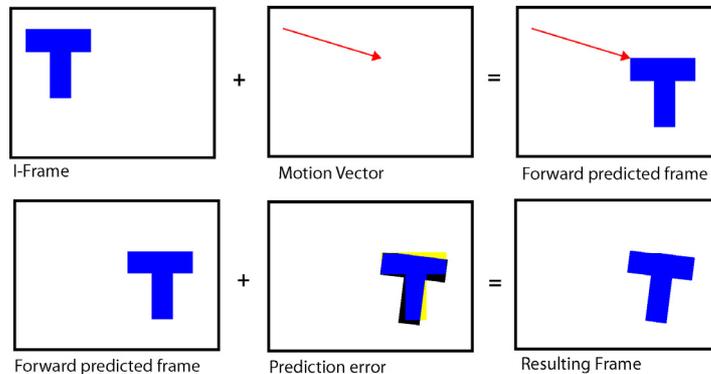


Figure 5

Vector Coding - After determining the motion vectors and evaluating the correction, these can be compressed. Large parts of MPEG videos consist of B- and P-frames as seen before, and most of them have mainly stored motion vectors. Therefore an efficient compression of motion vector data, which has usually high correlation, is desired. Details about motion vector compression can be found in [11].

Block Coding - see Discrete Cosine Transform (DCT) below.

Step 3: Discrete Cosine Transform (DCT)

DCT allows, similar to the *Fast Fourier Transform (FFT)*, a representation of image data in terms of frequency components. So the frame-blocks (8x8 or 16x16 pixels) can be represented as frequency components. The transformation into the frequency domain is described by the following formula:

$$F(u, v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cdot \cos \frac{(2x+1)u\pi}{2N} \cdot \cos \frac{(2y+1)v\pi}{2N}$$

$$C(u), C(v) = 1/\sqrt{2} \text{ for } u, v = 0$$

$$C(u), C(v) = 1, \text{ else}$$

$$N = \text{block size}$$

The inverse DCT is defined as:

$$f(x, y) = \frac{1}{4} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \cos \frac{(2y+1)u\pi}{16} \cos \frac{(2x+1)v\pi}{16}$$

The DCT is unfortunately computational very expensive and its complexity increases disproportionately ($O(N^2)$). That is the reason why images compressed using DCT are divided into blocks. Another disadvantage of DCT is its inability to decompose a broad signal into high and low frequencies at the same time. Therefore the use of small blocks allows a description of high frequencies with less cosine-terms.

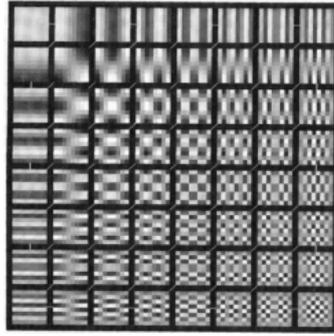


Figure 6: Visualisation of 64 basis functions (cosine frequencies) of a DCT. Reproduced from [12]

The first entry (top left in Figure 6) is called the *direct current-term*, which is constant and describes the *average grey level of the block*. The 63 remaining terms are called *alternating-current terms*. Up to this point no compression of the block data has occurred. The data was only well-conditioned for a compression, which is done by the next two steps.

Step 4: Quantization

During quantization, which is the primary source of data loss, the DCT terms are divided by a quantization matrix, which takes into account human visual perception. The human eyes are more reactive to low frequencies than to high ones. Higher frequencies end up with a zero entry after quantization and the domain was reduced significantly.

$$F_{QUANTISED} = F(u, v) \text{ DIV } Q(u, v)$$

Where Q is the quantisation Matrix of dimension N. The way Q is chosen defines the final compression level and therefore the quality. After Quantization the DC- and AC- terms are treated separately. As the correlation between the adjacent blocks is high, only the differences between the DC-terms are stored, instead of storing all values independently. The AC-terms are then stored in a zig-zag-path with increasing frequency values. This representation is optimal for the next coding step, because same values are stored next to each other; as mentioned most of the higher frequencies are zero after division with Q.

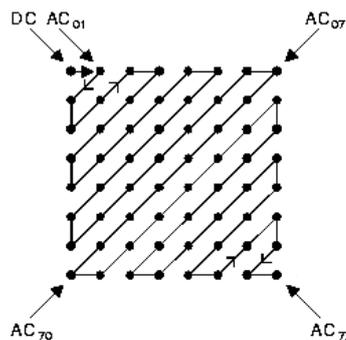


Figure 7: Zig-zag-path for storing the frequencies. Reproduced from [13].

If the compression is too high, which means there are more zeros after quantization, artefacts are visible (Figure 8). This happens because the blocks are compressed individually with no correlation to each other. When dealing with video, this effect is even more visible, as the blocks are changing (over time) individually in the worst case.



Figure 8: Block artefacts after DCT.

Step 5: Entropy Coding

The entropy coding takes two steps: *Run Length Encoding (RLE)* [2] and *Huffman coding* [1]. These are well known lossless compression methods, which can compress data, depending on its redundancy, by an additional factor of 3 to 4.

All five Steps together

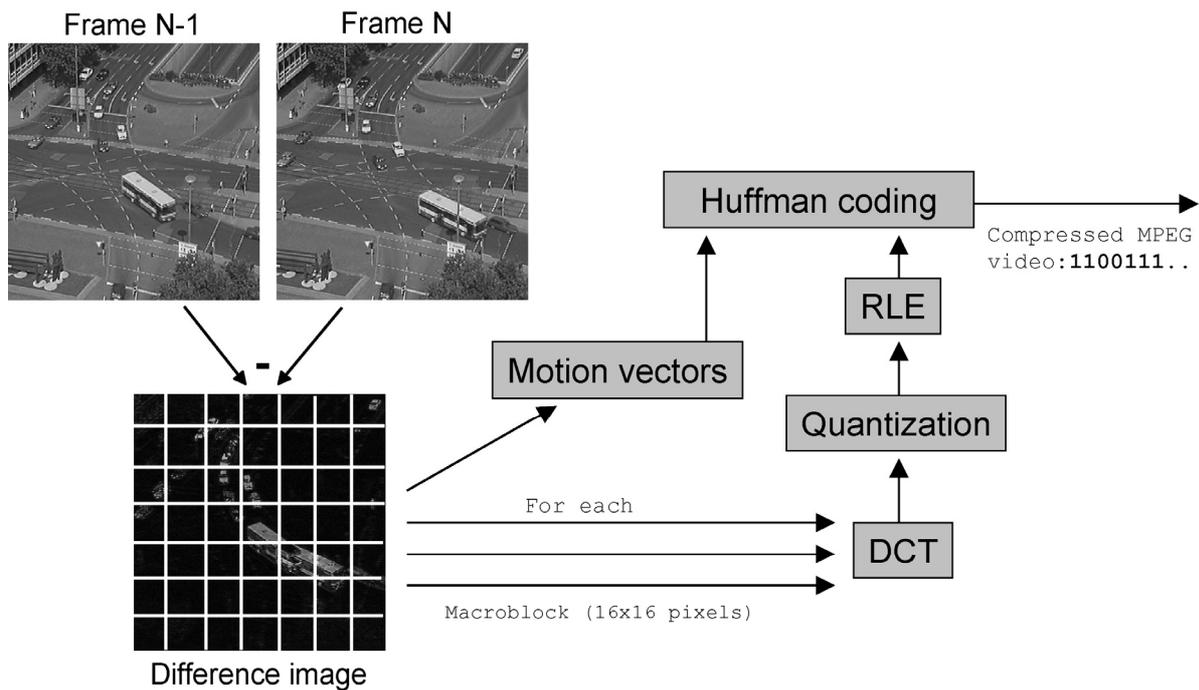


Figure 9: Illustration of the discussed 5 steps for a standard MPEG encoding.

As seen, MPEG video compression consists of multiple conversion and compression algorithms. At every step other critical compression issues occur and always form a trade-off between quality, data volume and computational complexity. However, the area of use of the video will finally decide which compression standard will be used. Most of the other compression standards use similar methods to achieve an optimal compression with best possible quality.

References

- [1] HUFFMAN, D. A. (1951). *A method for the construction of minimum redundancy codes*. In the Proceedings of the Institute of Radio Engineers 40, pp. 1098-1101.
- [2] CAPON, J. (1959). *A probabilistic model for run-length coding of pictures*. IRE Trans. On Information Theory, IT-5, (4), pp. 157-163.
- [3] APOSTOLOPOULOS, J. G. (2004). *Video Compression*. Streaming Media Systems Group. http://www.mit.edu/~6.344/Spring2004/video_compression_2004.pdf (3. Feb. 2006)
- [4] The Moving Picture Experts Group home page. <http://www.chiariglione.org/mpeg/> (3. Feb. 2006)
- [5] CLARKE, R. J. *Digital compression of still images and video*. London: Academic press. 1995, pp. 285-299
- [6] Institut für Informatik – Universität Karlsruhe. <http://www.irf.uka.de/seminare/redundanz/vortrag15/> (3. Feb. 2006)
- [7] PEREIRA, F. *The MPEG4 Standard: Evolution or Revolution* www.img.lx.it.pt/~fp/artigos/VLVB96.doc (3. Feb. 2006)
- [8] MANNING, C. The digital video site. <http://www.newmediarepublic.com/dvideo/compression/adv08.html> (3. Feb. 2006)
- [9] SEFERIDIS, V. E. GHANBARI, M. (1993). *General approach to block-matching motion estimation*. Optical Engineering, (32), pp. 1464-1474.
- [10] GHARAVI, H. MILLIS, M. (1990). *Block matching motion estimation algorithms-new results*. IEEE Transactions on Circuits and Systems, (37), pp. 649-651.
- [11] CHOI, W. Y. PARK R. H. (1989). *Motion vector coding with conditional transmission*. Signal Processing, (18), pp. 259-267.
- [12] Institut für Informatik – Universität Karlsruhe. <http://goethe.ira.uka.de/seminare/redundanz/vortrag11/#DCT> (3. Feb. 2006)
- [13] Technische Universität Chemnitz – Fakultät für Informatik. http://rnvs.informatik.tu-chemnitz.de/~jan/MPEG/HTML/mpeg_tech.html (3. Feb. 2006)