CHAPTER 7

Data Compression

In comparison to the text medium, video frames have high storage requirements. Audio and particularly video pose even greater demands in this regard. The data rates needed to process and send continuous media are also considerable. This chapter thus covers the efficient compression of audio and video, which will be explained first in theory and subsequently in terms of compression standards.

7.1 Storage Space

Uncompressed graphics, audio, and video data require substantial storage capacity, which is not possible in the case of uncompressed video data, even given today's CD and DVD technology. The same is true for multimedia communications. Data transfer of uncompressed video data over digital networks requires that very high bandwidth be provided for a single point-to-point communication. To be cost-effective and feasible, multimedia systems must use compressed video and audio streams.

Most compression methods address the same problems, one at a time or in combination. Most are already available as products. Others are currently under development or are only partially completed (see also [SPI94]). While fractal image compression [BH93] may be important in the future, the most important compression techniques in use today are JPEG [Org93, PM93, Wal91] for single pictures, H.263 ($p\times64$) [Le 91, ISO93b] for video, MPEG [Lio91, ITUC90] for video and audio, as well as proprietary techniques such as QuickTime from Apple and Video for Windows from Microsoft.

In their daily work, developers and multimedia experts often need a good understanding of the most popular techniques. Most of today's literature, however, is either

too comprehensive or is dedicated to just one of the above-mentioned compression techniques, which is then described from a very narrow point of view. In this chapter, we compare the most important techniques—JPEG, H.263, and MPEG—in order to show their advantages, disadvantages, their similarities and differences, as well as their suitability for today's multimedia systems (for further analysis, see [ES98]). First, the motivation for the use of compression techniques will be illustrated. Subsequently, requirements for compression techniques will be derived. Section 7.3 covers source, entropy, and hybrid coding. Sections 7.5 through 7.7 provide details about JPEG, H.263, and MPEG. Section 7.8 explains the basics of fractal compression.

7.2 Coding Requirements

Images have considerably higher storage requirements than text, and audio and video have still more demanding properties for data storage. Moreover, transmitting continuous media also requires substantial communication data rates. The figures cited below clarify the qualitative transition from simple text to full-motion video data and demonstrate the need for compression. In order to be able to compare the different data storage and bandwidth requirements of various visual media (text, graphics, images, and video), the following specifications are based on a small window of 640×480 pixels on a display. The following holds always:

1kbit = 1000bit

1 Kbit = 1024 bit

1 Mbit = 1024×1024 bit

1. For the representation of the text medium, two bytes are used for every 8×8 pixel character.

   $$\text{Character per screen page} = \frac{640 \times 480}{8 \times 8} = 4{,}800$$

   $$\text{Storage required per screen page} = 4{,}800 \times 2 \text{ byte} = 9{,}600 \text{ byte} = 9.4\,\text{Kbyte}$$

2. For the representation of vector images, we assume that a typical image consists of 500 lines [BHS91]. Each line is defined by its coordinates in the x direction and the y direction, and by an 8-bit attribute field. Coordinates in the x direction require 10 bits ($\log_2(640)$), while coordinates in the y direction require 9 bits ($\log_2(480)$).

   Bits per line = 9bits + 10bits + 9bits + 10bits + 8bits = 46bits

   $$\text{Storage required per screen page} = 500 \times \frac{46}{8} = 2{,}875 \text{ byte} = 2.8\,\text{Kbyte}$$

3. Individual pixels of a bitmap can be coded using 256 different colors, requiring a single byte per pixel.

   Storage required per screen page = 640×480×1 byte = 307,200 byte = 300 Kbyte

The next examples specify continuous media and derive the storage required for one second of playback.

1. Uncompressed speech of telephone quality is sampled at 8 kHz and quantized using 8 bit per sample, yielding a data stream of 64 Kbit/s.

   $$\text{Required storage space/s} = \frac{64\ \text{Kbit/s}}{8\ \text{bit/byte}} \times \frac{1\,\text{s}}{1{,}024\ \text{byte/Kbyte}} = 8\ \text{Kbyte}$$

2. An uncompressed stereo audio signal of CD quality is sampled at 44.1 kHZ and quantized using 16 bits.

   $$\text{Data rate} = 2 \times \frac{44{,}100}{\text{s}} \times \frac{16\ \text{bit}}{8\ \text{bit/byte}} = 176{,}400\ \text{byte/s}$$

   $$\text{Required storage space/s} = 176{,}400\ \text{byte/s} \times \frac{1\,\text{s}}{1{,}024\ \text{byte/Kbyte}} = 172\ \text{Kbyte}$$

3. A video sequence consists of 25 full frames per second. The luminance and chrominance of each pixel are coded using a total of 3 bytes.

   According to the European PAL standard, each frame consists of 625 lines and has a horizontal resolution of more than 833 pixels. The luminance and color difference signals are encoded separately and transmitted together using a multiplexing technique (4:2:2).

   According to CCIR 601 (studio standard for digital video), the luminance (Y) is sampled at 13.5 MHz, while chrominance ($R–Y$ and $B–Y$) is sampled using 6.75 MHz. Samples are coded uniformly using 8 bits.

   Bandwidth = (13.5 MHz + 6.75 MHz + 6.75 MHz)×8 bit = 216×10^6 bit/s.

   $$\text{Data rate} = 640 \times 480 \times 25 \times 3\ \text{byte/s} = 23{,}040{,}000\ \text{byte/s}$$

   $$\text{Required storage sapce/s} = 2{,}304 \times 10^4\ \text{byte/s} \times \frac{1\,\text{s}}{1{,}024\ \text{byte/Kbyte}} = 22{,}500\ \text{Kbyte}$$

   High-resolution television uses twice as many lines and an aspect ratio of 16:9, yielding a data rate 5.33 times that of current televisions.

These examples briefly illustrate the increased demands on a computer system in terms of required storage space and data throughput if still images and in particular continuous media are to be processed. Processing uncompressed video data streams in an integrated multimedia system requires storage space in the gigabyte range and buffer

space in the megabyte range. The throughput in such a system can be as high as 140 Mbit/s, which must also be transmitted over networks connecting systems (per unidirectional connection). This kind of data transfer rate is not realizable with today's technology, or in the near future with reasonably priced hardware.

However, these rates can be considerably reduced using suitable compression techniques [NH88, RJ91], and research, development, and standardization in this area have progressed rapidly in recent years [ACM89, GW93]. These techniques are thus an essential component of multimedia systems.

Several compression techniques for different media are often mentioned in the literature and in product descriptions:

- JPEG (Joint Photographic Experts Group) is intended for still images.
- H.263 (H.261 $p \times 64$) addresses low-resolution video sequences. This can be complemented with audio coding techniques developed for ISDN and mobile communications, which have also been standardized within CCITT.
- MPEG (Moving Picture Experts Group) is used for video and audio compression.

Compression techniques used in multimedia systems are subject to heavy demands. The quality of the compressed, and subsequently decompressed, data should be as good as possible. To make a cost-effective implementation possible, the complexity of the technique used should be minimal. The processing time required for the decompression algorithms, and sometimes also the compression algorithms, must not exceed certain time spans. Different techniques address requirements differently (see, for example, the requirements of [Org93]).

One can distinguish between requirements of "dialogue" mode applications (e.g. videoconferencing or picture transmission) and "retrieval" mode applications (e.g. audiovisual information systems, where a human user retrieves information from a multimedia database).

Compression techniques like $p \times 64$, with its symmetric processing expenditure for compression and decompression and its strict delay limits, are better suited to dialogue applications. Other techniques, such as MPEG-1, are optimized for use in retrieval applications at the expense of considerable effort during compression.

The following requirement applies to compression techniques used for dialogue mode applications:

- The end-to-end delay for a technique used in a dialogue system should not exceed 150 ms for compression and decompression alone. To support an easy, natural dialogue, ideally compression or decompression by itself should introduce an additional delay of no more than 50 ms. The overall end-to-end delay additionally comprises delay in the network, communications protocol processing in the end system, and data transfer to and from the respective input and output devices.

The following requirements apply to compression techniques used for *r*etrieval mode applications:

- Fast forward and fast rewind with simultaneous display (or playback) of the data should be possible, so that individual passages can be sought and found quicker.
- Random access to single images or audio passages in a data stream should be possible in less than half a second. This access should be faster than a CD-DA system in order to maintain the interactive character of the retrieval application.
- Decompression of images, video, or audio passages should be possible without interpreting all preceding data. This allows random access and editing.

The following requirements apply to compression techniques used for either dialogue or retrieval mode applications:

- To support display of the same data in different systems, it is necessary to define a format that is independent of frame size and video frame rate.
- Audio and video compression should support different data rates (at different quality) so that the data rate can be adjusted to specific system conditions.
- It should be possible to precisely synchronize audio and video. It should also be possible for a system program to synchronize with other media.
- It should be possible to implement an economical solution today either in software or using at most a few VLSI chips.
- The technique should enable cooperation among different systems. It should be possible for data generated on a local multimedia system to be reproduced on other systems. This is relevant, for example, in the case of course materials distributed on CDs. This allows many participants to read the data on their own systems, which may be made by different manufacturers. Also, since many applications exchange multimedia data between systems over communications networks, the compression techniques must be compatible. This can be assured with *de jure* (e.g. ITU, ISO, or ECMA) and/or *de facto* standards.

Different compression schemes take these requirements into account to a varying extent.

Coding Type	Basis	Technique
Entropy Coding	Run-length Coding	
	Huffman Coding	
	Arithmetic Coding	
Source Coding	Prediction	DPCM
		DM
	Transformation	FFT
		DCT
	Layered Coding (according to importance)	Bit Position
		Subsampling
		Subband Coding
	Vector Quantization	
Hybrid Coding	JPEG	
	MPEG	
	H.263	
	Many proprietary systems	

Table 7-1 Overview of some coding and compression techniques.

7.3 Source, Entropy, and Hybrid Coding

Compression techniques can be categorized as shown in Table 7-1. We distinguish among three types of coding: entropy, source, and hybrid coding. Entropy coding is a lossless process, while source coding is often lossy. Most multimedia systems use hybrid techniques; most are only combinations of entropy and source coding, without any new processing algorithms.

7.3.1 Entropy Coding

Entropy coding can be used for different media regardless of the medium's specific characteristics. The data to be compressed are viewed as a sequence of digital data values, and their semantics are ignored. It is lossless because the data prior to encoding is identical to the data after decoding; no information is lost. Thus run-length encoding, for example, can be used for compression of any type of data in a file system, for example, text, still images for facsimile, or as part of motion picture or audio coding.

7.3.2 Source Coding

Source coding takes into account the semantics of the information to be encoded [SGC90]. The degree of compression attainable with this often lossy technique depends on the medium. In the case of lossy compression, a relation exists between the uncoded data and the decoded data; the data streams are similar but not identical. The characteristics of the medium can be exploited. In the case of speech, a considerable reduction of the amount of data can be achieved by transforming the time-dependent signal into the frequency domain, followed by an encoding of the formants (see Chapter 3 regarding audio).

Formants are defined as the maxima in the voice spectrum. Generally, five formants along with the voice fundamental frequency suffice for a very good reconstruction of the original signal, although formant tracking has been shown to be a problem with this form of speech analysis (see Chapter 3).

In the case of still images, spatial redundancies can be used for compression through a content prediction technique. Other techniques perform a transformation of the spatial domain into the two-dimensional frequency domain by using the cosine transform. Low frequencies define the average color, and the information of higher frequencies contains the sharp edges. Hence, low frequencies are much more important than the higher frequencies, a feature that can be used for compression.

Table 7-1 shows only a sampling of all coding and compression techniques. The emphasis is on the algorithms that are most important for multimedia systems and on their properties. To better understand the hybrid schemes, we consider a set of typical processing steps common to all techniques (entropy, source, and hybrid).

7.3.3 Major Steps of Data Compression

Figure 7-1 shows the typical sequence of operations performed in the compression of still images and video and audio data streams. The following example describes the compression of one image:

1. The preparation step (here picture preparation) generates an appropriate digital representation of the information in the medium being compressed. For example, a picture might be divided into blocks of 8×8 pixels with a fixed number of bits per pixel.
2. The processing step (here picture processing) is the first step that makes use of the various compression algorithms. For example, a transformation from the time domain to the frequency domain can be performed using the Discrete Cosine Transform (DCT). In the case of interframe coding, motion vectors can be determined here for each 8×8 pixel block.
3. Quantization takes place after the mathematically exact picture processing step. Values determined in the previous step cannot and should not be processed with

full exactness; instead they are quantized according to a specific resolution and characteristic curve. This can also be considered equivalent to the μ-law and *A*-law, which are used for audio data [JN84]. In the transformed domain, the results can be treated differently depending on their importance (e.g., quantized with different numbers of bits).

4. Entropy coding starts with a sequential data stream of individual bits and bytes. Different techniques can be used here to perform a final, lossless compression. For example, frequently occurring long sequences of zeroes can be compressed by specifying the number of occurrences followed by the zero itself.

Picture processing and quantization can be repeated iteratively, such as in the case of Adaptive Differential Pulse Code Modulation (ADPCM). There can either be "feedback" (as occurs during delta modulation), or multiple techniques can be applied to the data one after the other (like interframe and intraframe coding in the case of MPEG). After these four compression steps, the digital data are placed in a data stream having a defined format, which may also integrate the image starting point and type of compression. An error correction code can also be added at this point.

Figure 7-1 shows the compression process applied to a still image; the same principles can also be applied to video and audio data.

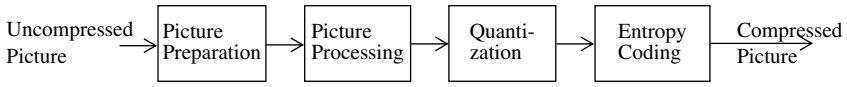


Figure 7-1 Major steps of data compression.

Decompression is the inverse process of compression. Specific coders and decoders can be implemented very differently. Symmetric coding is characterized by comparable costs for encoding and decoding, which is especially desirable for dialogue applications. In an asymmetric technique, the decoding process is considerably less costly than the coding process. This is intended for applications where compression is performed once and decompression takes place very frequently, or if the decompression must take place very quickly. For example, an audio-visual course module is produced once, but subsequently decoded by the many students who use it. The main requirement is real-time decompression. An asymmetric technique can be used to increase the quality of the compressed images.

The following section discusses some basic compression techniques. Subsequent sections describe hybrid techniques frequently used in the multimedia field.

7.4 Basic Compression Techniques

The hybrid compression techniques often used on audio and video data in multimedia systems are themselves composed of several different techniques. For example, each technique listed in Table 7-1 employs entropy coding (in the form of variations of run-length coding and/or a statistical compression technique).

The simplest techniques are based on interpolation, whereby it is possible to make use of properties of the human eye or ear. For example, the eye is more sensitive to changes in brightness than to color changes. Therefore, instead of dividing an image into RGB (red, green, blue) components, a YUV representation can be used (see Chapter 5). The *U* and *V* components can then be sampled with lower horizontal and vertical resolution, a technique called subsampling.

7.4.1 Run-Length Coding

Data often contains sequences of identical bytes. By replacing these repeated byte sequences with the number of occurrences, a substantial reduction of data can be achieved. This is known as run-length coding. A special marker *M* is needed in the data that does not occur as part of the data stream itself. This M-byte can also be realized if all 256 possible bytes can occur in the data stream by using byte stuffing. To illustrate this, we define the exclamation mark to be the M-byte. A single occurrence of an exclamation mark is interpreted as the M-byte during decompression. Two consecutive exclamation marks are interpreted as an exclamation mark occurring within the data.

The M-byte can thus be used to mark the beginning of a run-length coding. The technique can be described precisely as follows: if a byte occurs at least four times in a row, then the number of occurrences is counted. The compressed data contain the byte, followed by the M-byte and the number of occurrences of the byte. Remembering that we are compressing at least four consecutive bytes, the number of occurrences can be offset by –4. This allows the compression of between four and 258 bytes into only three bytes. (Depending on the algorithm, one or more bytes can be used to indicate the length; the same convention must be used during coding and decoding.)

In the following example, the character `C` occurs eight times in a row and is compressed to the three characters `C!4`:

Uncompressed data: `ABCCCCCCCCDEFGGG`

Run-length coded: `ABC!4DEFGGG`

7.4.2 Zero Suppression

Run-length coding is a generalization of zero suppression, which assumes that just one symbol appears particularly often in sequences. The blank (space) character in text is such a symbol; single blanks or pairs of blanks are ignored. Starting with sequences of three bytes, they are replaced by an M-byte and a byte specifying the number of

blanks in the sequence. The number of occurrences can again be offset (by −3). Sequences of between three and 257 bytes can thus be reduced to two bytes. Further variations are tabulators used to replace a specific number of null bytes and the definition of different M-bytes to specify different numbers of null bytes. For example, an M5-byte could replace 16 null bytes, while an M4-byte could replace 8 null bytes. An M5-byte followed by an M4-byte would then represent 24 null bytes.

7.4.3 Vector Quantization

In the case of vector quantization, a data stream is divided into blocks of n bytes each ($n>1$). A predefined table contains a set of patterns. For each block, the table is consulted to find the most similar pattern (according to a fixed criterion). Each pattern in the table is associated with an index. Thus, each block can be assigned an index. Such a table can also be multidimensional, in which case the index will be a vector. The corresponding decoder has the same table and uses the vector to generate an approximation of the original data stream. For further details see [Gra84] for example.

7.4.4 Pattern Substitution

A technique that can be used for text compression substitutes single bytes for patterns that occur frequently. This pattern substitution can be used to code, for example, the terminal symbols of high-level languages (begin, end, if). By using an M-byte, a larger number of words can be encoded–the M-byte indicates that the next byte is an index representing one of 256 words. The same technique can be applied to still images, video, and audio. In these media, it is not easy to identify small sets of frequently occurring patterns. It is thus better to perform an approximation that looks for the most similar (instead of the same) pattern. This is the above described vector quantization.

7.4.5 Diatomic Encoding

Diatomic encoding is a variation based on combinations of two data bytes. This technique determines the most frequently occurring pairs of bytes. Studies have shown that the eight most frequently occurring pairs in the English language are “E,” “T,” “TH,” “A,” “S,” “RE,” “IN,” and “HE.” Replacing these pairs by special single bytes that otherwise do not occur in the text leads to a data reduction of more than ten percent.

7.4.6 Statistical Coding

There is no fundamental reason that different characters need to be coded with a fixed number of bits. Morse code is based on this: frequently occurring characters are coded with short strings, while seldom-occurring characters are coded with longer strings. Such statistical coding depends how frequently individual characters or

sequences of data bytes occur. There are different techniques based on such statistical criteria, the most prominent of which are Huffman coding and arithmetic coding.

7.4.7 Huffman Coding

Given the characters that must be encoded, together with their probabilities of occurrence, the Huffman coding algorithm determines the optimal coding using the minimum number of bits [Huf52]. Hence, the length (number of bits) of the coded characters will differ. The most frequently occurring characters are assigned to the shortest code words. A Huffman code can be determined by successively constructing a binary tree, whereby the leaves represent the characters that are to be encoded. Every node contains the relative probability of occurrence of the characters belonging to the subtree beneath the node. The edges are labeled with the bits 0 and 1.

The following brief example illustrates this process:

1. The letters *A, B, C, D,* and *E* are to be encoded and have relative probabilities of occurrence as follows:

   $p(A)=0.16,\ p(B)=0.51,\ p(C)=0.09,\ p(D)=0.13,\ p(E)=0.11$

2. The two characters with the lowest probabilities, *C* and *E,* are combined in the first binary tree, which has the characters as leaves. The combined probability of their root node *CE* is 0.20. The edge from node *CE* to *C* is assigned a 1 and the edge from *CE* to *C* is assigned a 0. This assignment is arbitrary; thus, different Huffman codes can result from the same data.

3. Nodes with the following relative probabilities remain:

   $p(A)=0.16,\ p(B)=0.51,\ p(CE)=0.20,\ p(D)=0.13$

   The two nodes with the lowest probabilities are *D* and *A*. These nodes are combined to form the leaves of a new binary tree. The combined probability of the root node *AD* is 0.29. The edge from *AD* to *A* is assigned a 1 and the edge from *AD* to *D* is assigned a 0.

   If root nodes of different trees have the same probability, then trees having the shortest maximal path between their root and their nodes should be combined first. This keeps the length of the code words roughly constant.

4. Nodes with the following relative probabilities remain:

   $p(AD)=0.29,\ p(B)=0.51,\ p(CE)=0.20$

   The two nodes with the lowest probabilities are *AD* and *CE.* These are combined into a binary tree. The combined probability of their root node *ADCE* is 0.49. The edge from *ADCE* to *AD* is assigned a 0 and the edge from *ADCE* to *CE* is assigned a 1.

5. Two nodes remain with the following relative probabilities:

   $p(ADCE)=0.49,\ p(B)=0.51$

These are combined to a final binary tree with the root node *ADCEB*. The edge from *ADCEB* to *B* is assigned a 1, and the edge from *ADCEB* to *ADCE* is assigned a 0.

6. Figure 7-2 shows the resulting Huffman code as a binary tree. The result is the following code words, which are stored in a table:

   $w(A)=001,\ w(B)=1,\ w(C)=011,\ w(D)=000,\ w(E)=010$

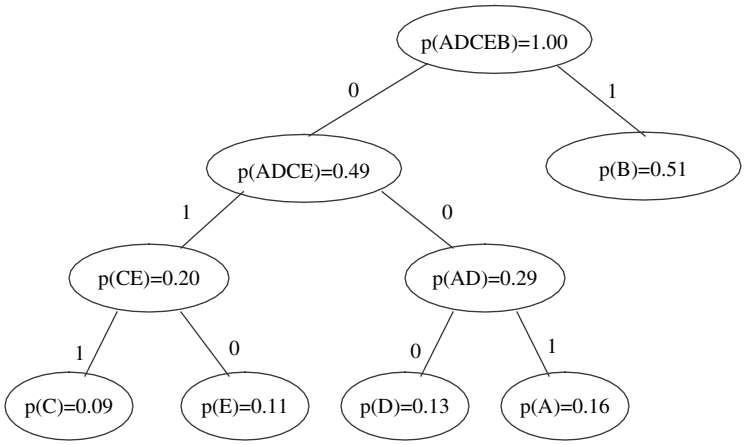


Figure 7-2 Example of a Huffman code represented as a binary tree.

Such a table could be generated for a single image or for multiple images together. In the case of motion pictures, a Huffman table can be generated for each sequence or for a set of sequences. The same table must be available for both encoding and decoding. If the information of an image can be transformed into a bit stream, then a Huffman table can be used to compress the data without any loss. The simplest way to generate such a bit stream is to code the pixels individually and read them line by line. Note that usually more sophisticated methods are applied, as described in the remainder of this chapter.

If one considers run-length coding and all the other methods described so far, which produce the same consecutive symbols (bytes) quite often, it is certainly a major objective to transform images and videos into a bit stream. However, these techniques also have disadvantages in that they do not perform efficiently, as will be explained in the next step.

7.4.8 Arithmetic Coding

Like Huffman coding, arithmetic coding is optimal from an information theoretical point of view (see [Lan84, PMJA88]). Therefore, the length of the encoded data is

also minimal. Unlike Huffman coding, arithmetic coding does not code each symbol separately. Each symbol is instead coded by considering all prior data. Thus a data stream encoded in this fashion must always be read from the beginning. Consequently, random access is not possible. In practice, the average compression rates achieved by arithmetic and Huffman coding are similar [Sto88].

Differences in compression efficiency arise in special cases, for example in digitized graphics consisting primarily of a single (background) color. Arithmetic coding is better suited than Huffman coding in this case. This is always the case if symbols occur in the input data stream with such a frequency that they have a very small information content. These can be encoded using less than one bit, whereas in Huffman coding each symbol requires at least one bit.

7.4.9 Transformation Coding

Transformation coding pursues a different approach. Data is transformed into another mathematical domain that is more suitable for compression. An inverse transformation must exist. The simplest example is the Fourier transform, which transforms data from the time domain into the frequency domain. Other examples include the Walsh, Hadamard, Haar, and Slant transforms. However, the transformed data have no major advantages with respect to a subsequent compression. The most effective transformations for data reduction are the Discrete Cosine Transform (DCT), described in Section 7.5.2, and the Fast Fourier Transform (FFT).

7.4.10 Subband Coding

Unlike transformation coding, which transforms all data into another domain, selective frequency transformation (subband coding) considers the signal only in predefined regions of the spectrum, such as frequency bands. The number of bands is a crucial quality criterion. This technique is well suited for the compression of speech.

7.4.11 Prediction or Relative Coding

Instead of compressing single bytes or sequences of bytes, differential encoding can be used. This is also known as prediction or relative coding. For example, if characters in a sequence are clearly different from zero, but do not differ much amongst themselves, then calculating differences from the respective previous value could be profitable for compression. The following examples explain this technique for different media:

- For still images, edges yield large difference values, while areas with similar luminance and chrominance yield small values. A homogeneous area is characterized by a large number of zeroes, which could be further compressed using run-length coding.

- For video, performing relative coding in the time domain leads to an encoding of differences from the previous image only. In a newscast or a video telephony application, there will be a large number of zero bytes, because the background of successive images does not change very often. Motion compensation can also be performed here (see e.g. [PA91]). Blocks of, for example, 16×16 pixels are compared with each other in successive pictures. In the case of a car moving from left to right, an area in the current image would be most similar to an area lying further to the left in the previous image. This motion can be coded as a vector.
- Audio techniques often apply Differential Pulse Code Modulation (DPCM) to a sequence of PCM-coded samples (see e.g. [JN84]). This technique requires a linear characteristic curve for quantization. It is thus not necessary to store the whole number of bits for each sample. It is sufficient to represent the first PCM-coded sample as a whole and all following samples as the difference from the previous one.

7.4.12 Delta Modulation

Delta Modulation is a modification of DPCM where difference values are encoded with exactly one bit, which indicates whether the signal increases or decreases. This leads to an inaccurate coding of steep edges. This technique is particularly profitable if the coding does not depend on 8-bit grid units. If the differences are small, then a much smaller number of bits is sufficient. Difference encoding is an important feature of all techniques used in multimedia systems. Section 7.5.2 describes other "delta" methods that can be applied to images.

7.4.13 Adaptive Compression Techniques

Most of the compression techniques described so far take advantage of already known characteristics of the data to be compressed (e.g., frequently occurring sequences of bytes or the probability of occurrence of individual bytes). An atypical sequence of characters results in a poor compression. However, there are also adaptive compression techniques, which can adjust themselves to the data being compressed. This adaptation can be implemented in different ways:

- We illustrate the first technique with the following example, which assumes a coding table has been generated in advance (e.g., as per Huffman). For each symbol to be encoded, the table contains the corresponding code word and, in an additional column, a counter. All entries' counters are initialized to zero at the beginning. For the first symbol to be encoded, the coder determines the code word according to the table. Additionally, the coder increments the counter by one. The symbols and their respective counters are then sorted in decreasing order by counter value. The order of the code words is not changed. The most frequently

occurring symbols are now at the beginning of the table, since they have the highest counter values. Symbols with the highest frequency count are thus always encoded with the shortest code words.

- Another adaptive compression technique is Adaptive DPCM (ADPCM), a generalization of DPCM. For simplicity, this is often called DPCM.

  Here, difference values are encoded using only a small number of bits. With DPCM, either coarse transitions would be coded correctly (the DPCM-encoded bits are used represent bits with a higher significance), or fine transitions are coded exactly (if the bits are used to represent bits with less significance). In the first case, the resolution of low audio signals would not be sufficient, and in the second case a loss of high frequencies would occur.

  ADPCM adapts to the significance of the data stream. The coder divides the DPCM sample values by an appropriate coefficient and the decoder multiplies the compressed data by the same coefficient, thus changing the step size of the signal. The coder of the DPCM-encoded signal adjusts the value of the coefficient.

  A signal with a large portion of high frequencies will result in frequent, very high DPCM values. The coder will select a high value for the coefficient. The result is a very rough quantization of the DPCM signal in passages with steep edges. Low-frequency portions of such passages are hardly considered at all.

  In the case of a signal with steady, relatively low DPCM values, that is, with a small portion of high frequencies, the coder will choose a small coefficient. This ensures good resolution for the dominant, low frequency portion of the signal. If high frequencies suddenly occur in such a passage, a signal distortion in the form of a slope overload occurs. The greatest possible change that can be represented by an ADPCM value using the available number of bits and the current step size is not enough to represent the new DPCM value. The jump in the PCM signal will be faded.

  Changes in the adaptively set coefficient can be explicitly inserted in the compressed data by the encoder. Alternatively, the decoder can calculate the coefficients itself from an ADPCM-coded data stream. This predictor is rated so as to minimize errors in the data stream. Note that the definition of an error and the associated predictor rating depends on the medium and is ideally trivial.

  An audio signal with frequently changing portions of extreme low or high frequencies is generally not suited for ADPCM coding. For telephone applications, the ITU has standardized a version of the ADPCM technique using 32 Kbit/s that is based on four bits per difference value and a sampling frequency of 8 kHz.

7.4.14 Other Basic Techniques

Apart from the compression techniques described earlier, some additional well known techniques are used today:

- Video compression techniques often use Color Look-Up Tables (see Section 5.1.2). This technique is used in distributed multimedia systems, for example in [LE91, LEM92].
- A simple technique for audio is silence suppression, whereby data is only encoded if the volume exceeds a certain threshold.

ITU incorporates some of the basic audio coding techniques in the G.700 series of standards: G.721 defines PCM coding for 3.4kHz quality over 64Kbit/s channels, and G.728 defines 3.4kHz quality over 16Kbit/s channels. See [ACG93] for a detailed description of various audio coding techniques.

The following sections describe the most important work in the standardization bodies concerning image and video coding. In the framework of ISO/IECJTC1/SC2/WG8, four subgroups were established in May 1988: JPEG (Joint Photographic Experts Group), working on coding of still images; JBIG (Joint Bi-Level Image Experts Group), working on the progressive processing of bi-level coding algorithms; CGEG (Computer Graphics Experts Group), working on coding principles; and MPEG (Moving Picture Experts Group), working on the coded representation of motion video. The next section presents the results of the JPEG activities.

7.5 JPEG

Since June 1982, Working Group 8 (WG8) of ISO has been working on standards for the compression and decompression of still images [HYS88]. In June 1987, ten different techniques for coding color and gray-scaled still images were presented. These ten were compared, and three were analyzed further. An adaptive transformation coding technique based on the Discrete Cosine Transform (DCT) achieved the best subjective results [LMY88, WVP88]. This technique was then further developed with consideration paid to the other two methods. The coding known as JPEG (Joint Photographic Experts Group) is a joint project of ISO/IECJTC1/SC2/WG10 and Commission Q.16 of CCITT SGVIII. Hence the "J" (from "Joint") in JPEG—ISO together with CCITT. In 1992, JPEG became an ISO International Standard (IS) [Org93].

JPEG applies to color and gray-scaled still images [LOW91, MP91, Wal91]. Video sequences can also be handled through a fast coding and decoding of still images, a technique known as Motion JPEG. Today, implementations of parts of JPEG are already available, either as software-only packages or using special-purpose hardware support. It should be noted that as yet, most products support only the absolutely necessary algorithms. The only part of JPEG currently commercially available is the

base mode with certain processing restrictions (limited number of image components and color coding).

7.5.0.1 Requirements

In order to ensure the widespread distribution and application of JPEG, the following requirements were established and fulfilled [Wal91]:

- The standard should be independent of image size.
- It should be applicable to any image aspect ratio and any pixel aspect ratio.
- The color space and the number of colors used should be independent of one another.
- Image content may be of any complexity, with any statistical characteristics.
- The JPEG standard should be start-of-the-art (or near) regarding the compression factor and image quality.
- The processing complexity should permit a software solution to run on a large number of available general-purpose processors, and should be drastically reduced with the use of special-purpose hardware.
- Sequential (line by line) and progressive (successive refinement of the whole image) decoding should both be possible. A lossless, hierarchical coding of the same image with different resolutions should also be possible.

The user can select parameters to trade off the quality of the reproduced image, the compression processing time, and the size of the compressed image.

7.5.0.2 JPEG Overview

Applications do not have to include both an encoder and a decoder. In many applications only one of them is needed. The encoded data stream has a fixed interchange format that includes encoded image data, as well as the chosen parameters and tables of the coding process, enabling decoding. If there is a common context between the coder and the decoder (e.g., if the coder and decoder are parts of the same application), then there can be an abbreviated interchange format. This format includes few if any of the required tables (Appendix A in [Org93] describes this format in detail). The interchange format includes all information required during decoding, if this information is not available as part of the common context. In the regular, non-abbreviated mode, the interchange format includes all information required to decode without prior knowledge of the coding process.

Figure 7-3 outlines the fundamental steps of JPEG compression in accordance with the general scheme illustrated in Figure 7-1. JPEG defines several image compression modes by selecting different combinations of these steps.

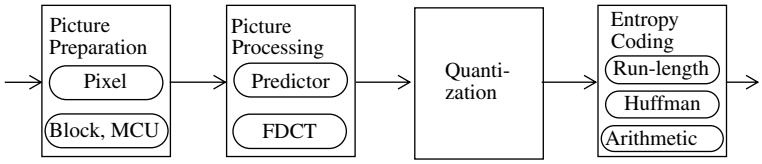


Figure 7-3 Steps of the JPEG compression technique: summary of the different modes.

7.5.0.3 JPEG Modes

JPEG defines four modes, which themselves include additional variations:

- The lossy, sequential DCT-based mode (baseline process, base mode) must be supported by every JPEG decoder.
- The expanded lossy, DCT-based mode provides a set of further enhancements to the base mode.
- The lossless mode has a low compression ratio and allows a perfect reconstruction of the original image.
- The hierarchical mode accommodates images of different resolutions by using algorithms defined for the other three modes.

The baseline process uses the following techniques: block, Minimum Coded Unit (MCU), FDCT, run-length, and Huffman, which are explained in this section together with the other modes. Image preparation is first presented for all modes. The picture processing, quantization, and entropy encoding steps used in each mode are then described separately for each mode.

7.5.1 Image Preparation

For image preparation, JPEG specifies a very general image model that can describe most commonly used still image representations. For instance, the model is not based on three image components with 9-bit YUV coding and a fixed numbers of lines and columns. The mapping between coded color values and the colors they represent is also not coded. This fulfills the JPEG requirement of independence from image parameters such as image size or the image and pixel aspect ratios.

An image consists of at least one and at most N=255 components or planes, as shown on the left side of Figure 7-4. These planes can be assigned to individual RGB (red, green, blue) colors, or to the YIQ or YUV signals, for example.

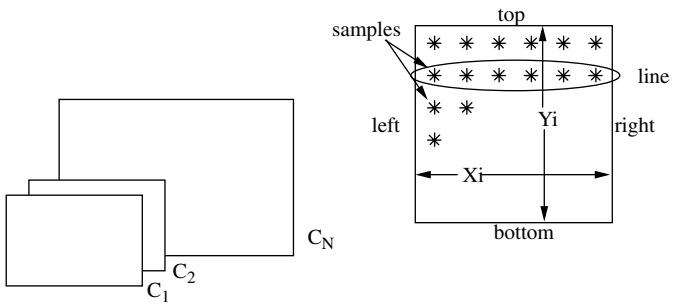


Figure 7-4 Uncompressed digital image as per [Org93].

Each component is a rectangular array $X_i \times Y_i$ of pixels (the samples). Figure 7-5 shows an image with three planes, each with the same resolution.

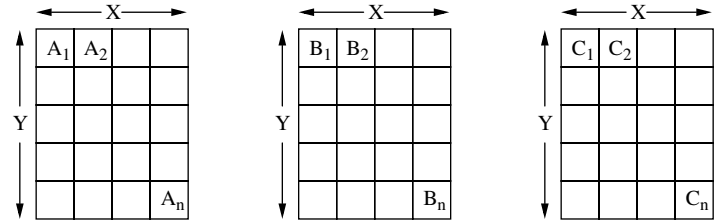


Figure 7-5 Example of JPEG image preparation with three components having the same resolution.

The resolution of the individual components may be different. Figure 7-6 shows an image with three planes where the second and third planes have half as many columns as the first plane. A gray-scale image will, in most cases, consist of a single component, while an RGB color image will have three components with the same resolution (same number of lines $Y_1 = Y_2 = Y_3$ and the same number of columns $X_1 = X_2 = X_3$). In JPEG image preparation, YUV color images with subsampling of the chrominance components use three planes with $Y_1 = 4\,Y_2 = 4\,Y_3$ and $X_1 = 4\,X_2 = 4\,X_3$.

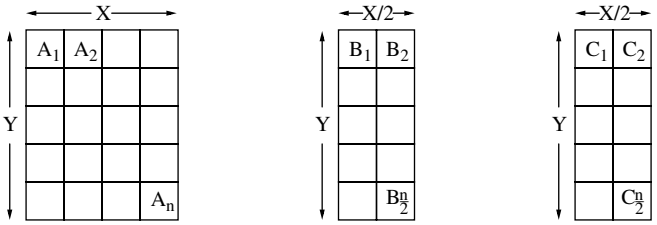


Figure 7-6 Example of JPEG image preparation with three components having different resolutions.

Each pixel is represented by p bits with values in the range from 0 to 2^p-1. All pixels of all components of an image must have the same number of bits. The lossy modes of JPEG use a precision of either 8 or 12 bits per pixel. The lossless modes can use between 2 and 12 bits per pixel. If a JPEG application uses any other number of bits, the application itself must suitably transform the image to conform to the number of bits defined by the JPEG standard.

Instead of the values X_i and Y_i, the compressed data includes the values X (maximum of all X_i) and Y (maximum of all Y_i), as well as factors H_i and V_i for each plane. H_i and V_i represent the relative horizontal and vertical resolutions with respect to the minimal horizontal and vertical resolutions.

Let us consider the following example from [Org93]. An image has a maximum resolution of 512 pixels in both the horizontal and vertical directions and consists of three planes. The following factors are given:

```
Plane 0: H0 = 4, V0 = 1
Plane 1: H1 = 2, V1 = 2
Plane 2: H2 = 1, V2 = 1
```

This leads to:

```
X = 512, Y = 512, Hmax = 4 and Vmax = 2
Plane 0: X0 = 512, Y0 = 256
Plane 1: X1 = 256, Y1 = 512
Plane 2: X2 = 128, Y2 = 256
```

H_i and V_i must be integers between 1 and 4. This awkward-looking definition is needed for the interleaving of components.

In the image preparation stage of compression, the image is divided into data units. The lossless mode uses one pixel as one data unit. The lossy mode uses blocks of 8×8 pixels. This is a consequence of DCT, which always transforms connected blocks.

Up to now, the data units are usually prepared component by component and passed on in order to the following image processing. Within each component, the data

units are processed from left to right, as shown in Figure 7-7. This is known as a non-interleaved data ordering.

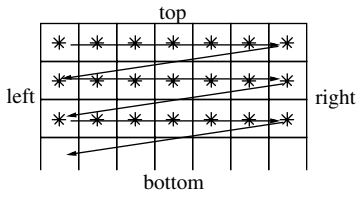


Figure 7-7 Noninterleaved processing order of data units when processing a single component as per [Org93].

Due to the finite processing speed of the decoder, processing of data units of different components may be interleaved. If the noninterleaved mode were used for a very high-resolution, RGB-encoded image, during rendering the display would first show only red, then red-green, and finally the correct colors.

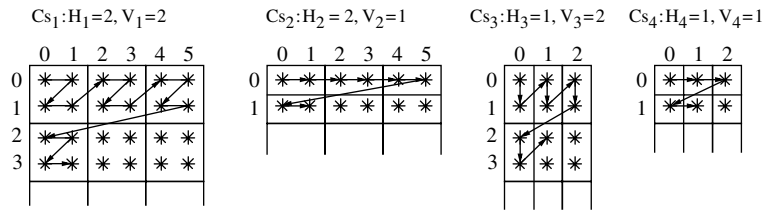


Figure 7-8 Interleaved processing order of data units.

Figure 7-8 shows an example with four components from [Org93]. Above each component, the corresponding values for H and V are shown. The first component has the highest resolution in both dimensions and the fourth component has the lowest resolution. The arrows within each component indicate the sampling order of individual data units.

Minimum Coded Units (MCUs) are built in the following order:

$$MCU_1 = d^1_{00}d^1_{01}d^1_{10}d^1_{11}d^2_{00}d^2_{01}d^3_{00}d^3_{10}d^4_{00}$$

$$MCU_2 = d^1_{02}d^1_{03}d^1_{12}d^1_{13}d^2_{02}d^2_{03}d^3_{01}d^3_{11}d^4_{01}$$

$$MCU_3 = d^1_{04}d^1_{05}d^1_{14}d^1_{15}d^2_{04}d^2_{05}d^3_{02}d^3_{12}d^4_{02}$$

$$MCU_4 = d^1_{20}d^1_{21}d^1_{30}d^1_{31}d^2_{10}d^2_{11}d^3_{20}d^3_{30}d^4_{10}$$

The data units of the first component are $C_{s1}: d_{00}^{1} \ldots d_{31}^{1}$

The data units of the second component are $C_{s2}: d_{00}^{2} \ldots d_{11}^{2}$

The data units of the third component are $C_{s3}: d_{00}^{3} \ldots d_{30}^{3}$

The data units of the fourth component are $C_{s4}: d_{00}^{4} \ldots d_{10}^{4}$

Interleaved data units of different components are combined into Minimum Coded Units. If all components have the same resolution ($X_i \times Y_i$), an MCU consists of exactly one data unit from each component. The decoder displays the image MCU by MCU. This allows for correct color presentation, even for partly decoded images.

In the case of different component resolutions, the construction of MCUs becomes more complex (see Figure 7-8). For each component, regions of data units, potentially with different numbers of data units, are constructed. Each component consists of the same number of regions. For example, in Figure 7-8 each component has six regions. MCUs are comprised of exactly one region from each component. The data units within a region are ordered from left to right and from top to bottom.

According to the JPEG standard, a maximum of four components can be encoded using the interleaved mode. This is not a limitation, since color images are generally represented using three components. Each MCU can contain at most ten data units. Within an image, some components can be encoded in the interleaved mode and others in the noninterleaved mode.

7.5.2 Lossy Sequential DCT-Based Mode

After image preparation, the uncompressed image samples are grouped into data units of 8×8 pixels, as shown in Figure 7-9; the order of these data units is defined by the MCUs. In this baseline mode, each sample is encoded using $p = 8$ bit. Each pixel is an integer in the range 0 to 255.

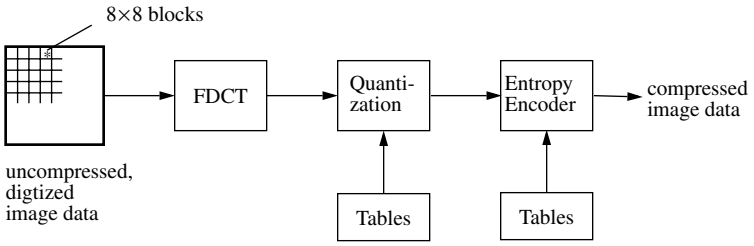


Figure 7-9 Steps of the lossy sequential DCT-based coding mode, starting with an uncompressed image after image preparation.

Figure 7-10 shows the compression and decompression process outlined here.

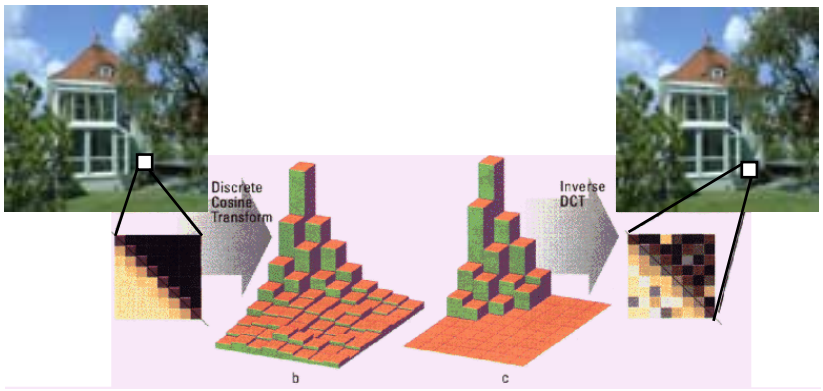


Figure 7-10 Example of DCT and IDCT.

7.5.2.1 Image Processing

The first step of image processing in the baseline mode (baseline process in [Org93]), as shown in Figure 7-9, is a transformation coding performed using the Discrete Cosine Transform (DCT) [ANR74, NP78]. The pixel values are shifted into the zero-centered interval (−128, 127). Data units of 8×8 shifted pixel values are defined by S_{yx}, where x and y are in the range of zero to seven.

The following FDCT (Forward DCT) is then applied to each transformed pixel value:

$$S_{vu} = \frac{1}{4} c_u c_v \sum_{x=0}^{7} \sum_{y=0}^{7} S_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

where: $c_u, c_v = \frac{1}{\sqrt{2}}$ for $u, v = 0$; otherwise $c_u, c_v = 1$

Altogether, this transformation must be carried out 64 times per data unit. The result is 64 coefficients S_{vu}. Due to the dependence of DCT on the Discrete Fourier Transform (DFT), which maps values from the time domain to the frequency domain, each coefficient can be regarded as a two-dimensional frequency.

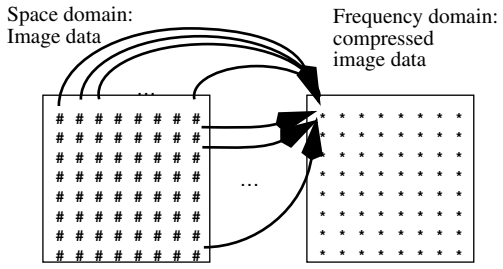


Figure 7-11 Relationship between the two-dimensional space and frequency domains.

The coefficient S_{00} corresponds to the portion where the frequency is zero in both dimensions. It is also known as the DC-coefficient (DC voltage portion) and determines the fundamental color of all 64 pixels of the data unit. The other coefficients are called AC-coefficients (analogous to the AC voltage portion). For instance, S_{70} represents the highest frequency that occurs in the horizontal direction, that is, the closest possible separation of vertical lines in the 8×8 data unit. S_{07} represents the highest frequency in the vertical dimension, that is, the closest possible separation of horizontal lines. S_{77} indicates the highest frequency appearing equally in both dimensions. Its absolute value is greatest if the original data unit contains as many squares as possible, that is, if it consists solely of 1×1 squares. Accordingly, for example, S_{33} will be maximal if a block consists of 16 squares of 4×4 pixels. Taking a closer look at the above formula, we recognize that the cosine expressions depend only upon *x* and *u,* or upon *y* and *v,* but not on S_{yx}. Therefore, these expressions represent constants that do not need to be recalculated over and over again. There are many effective DCT techniques and implementations [DG90, Fei90, Hou88, Lee84, LF91, SH86, VN84, Vet85].

For later reconstruction of the image, the decoder uses the Inverse DCT (IDCT). The coefficients S_{vu} must be used for the calculation:

$$s_{xy} = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} c_u c_v s_{vu} \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

where $c_u, c_v = \frac{1}{\sqrt{2}}$ for $u, v = 0$; otherwise $c_u, c_v = 1$

If the FDCT, as well as the IDCT, could be calculated with full precision, it would be possible to reproduce the original 64 pixels exactly. From a theoretical point of view, DCT would be lossless in this case. In practice, precision is limited and DCT is thus lossy. The JPEG standard does not define any specific precision. It is thus possible that

two different JPEG decoder implementations could generate different images as output of the same compressed data. JPEG merely defines the maximum allowable deviation.

Many images contain only a small portion of sharp edges; they consist mostly of areas of a single color. After applying DCT, such areas are represented by a small portion of high frequencies. Sharp edges, on the other hand, are represented as high frequencies. Images of average complexity thus consist of many AC-coefficients with a value of zero or almost zero. This means that subsequent entropy encoding can be used to achieve considerable data reduction.

7.5.2.2 Quantization

Image processing is followed by the quantization of all DCT coefficients; this is a lossy process. For this step, the JPEG application provides a table with 64 entries, one for each of the 64 DCT coefficients. This allows each of the 64 coefficients to be adjusted separately. The application can thus influence the relative significance of the different coefficients. Specific frequencies can be given more importance than others depending on the characteristics of the image material to be compressed. The possible compression is influenced at the expense of achievable image quality.

The table entries Q_{vu} are integer values coded with 8 bits. Quantization is performed according to the formula:

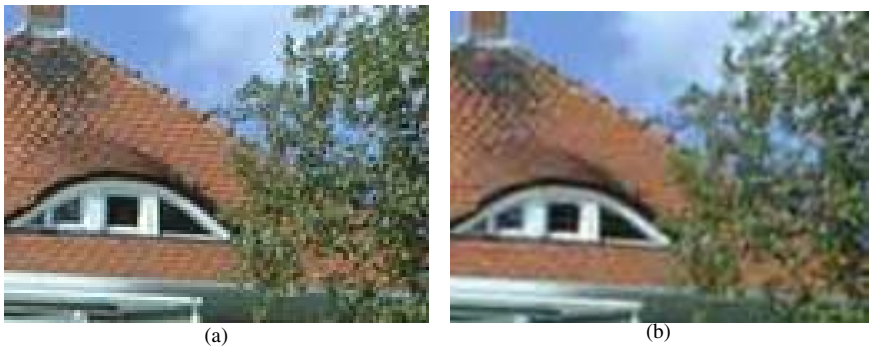
$$sq_{vu} = \text{round}\frac{S_{vu}}{Q_{vu}}$$

The greater the table entries, the coarser the quantization. Dequantization is performed prior to the IDCT according to the formula:

$$R_{vu} = Sq_{vu} \times Q_{vu}$$

Quantization and dequantization must use the same tables.

Figure 7-12 shows a greatly enlarged detail from Figure 7-16. The blocking and the effects of quantization are clearly visible. In Figure 7-12(b) a coarser quantization was performed to highlight the edges of the 8×8 blocks.



(a) (b)

Figure 7-12 Quantization effect.

7.5.2.3 Entropy Encoding

During the next step, either the initial step of entropy encoding or preparation for the coding process, the quantized DC-coefficients are treated differently than the quantized AC-coefficients. The processing order of all coefficients is specified by the zig-zag sequence.

- The DC-coefficients determine the fundamental color of the data units. Since this changes little between neighboring data units, the differences between successive DC-coefficients are very small values. Thus each DC-coefficient is encoded by subtracting the DC-coefficient of the previous data unit, as shown in Figure 7-13, and subsequently using only the difference.

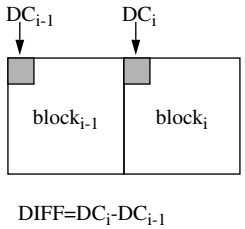


Figure 7-13 Preparation of DCT DC-coefficients for entropy encoding. Calculation of the difference between neighboring values.

- The DCT processing order of the AC-coefficients using the zig-zag sequence as shown in Figure 7-14 illustrates that coefficients with lower frequencies (typically with higher values) are encoded first, followed by the higher frequencies (typically zero or almost zero). The result is an extended sequence of similar data bytes, permitting efficient entropy encoding.

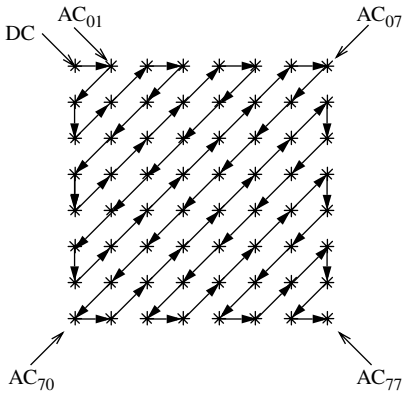


Figure 7-14 Preparation of DCT AC-coefficients for entropy encoding, in order of increasing frequency.

JPEG uses Huffman coding and arithmetic coding as entropy encoding methods. For the lossy sequential DCT-based base mode, discussed in this section, only Huffman encoding is allowed. In both methods, a run-length encoding of zero values is first applied to the quantized AC-coefficients. Additionally, non-zero AC-coefficients as well as the DC-coefficients are transformed into a spectral representation to further compress the data. The number of bits required depends on the value of each coefficient. Non-zero AC-coefficients are represented using between 1 and 10 bits. For the representation of DC-coefficients, a higher resolution of 1 bit to a maximum of 11 bits is used.

The result is a representation according to the ISO Intermediate Symbol Sequence Format, which basically alternates the following three pieces of information:

1. the number of subsequent coefficients with the value zero,
2. the number of bits used for the representation of the coefficient that follows, and
3. the value of the coefficient, represented using the specified number of bits.

An advantage of Huffman coding is that it can be used cost-free, since it is not protected by patents. A disadvantage is that the application must provide coding tables, since JPEG does not specify any. In the base mode, two different Huffman tables can be used, one for AC-coefficients and for DC-coefficients.

In sequential coding, the whole image is coded and decoded in a single run. Figure 7-15 shows an example of decoding with immediate presentation on the display. The picture is completed from top to bottom.

Figure 7-15 Example of sequential picture preparation, here using the lossy DCT-based mode.

7.5.3 Expanded Lossy DCT-Based Mode

Image preparation in this mode differs from the previously described mode in terms of the number of bits per pixel. This mode supports 12 bits per sample value in addition to 8 bits. The image processing is DCT-based and is performed analogously to the baseline DCT mode. For the expanded lossy DCT-based mode, JPEG defines progressive coding in addition to sequential coding. In the first decoding run, a very rough, unsharp image appears. This is refined during successive runs. An example of a very unsharp image is shown in Figure 7-16(a). It is substantially sharper in Figure 7-16(b), and in its correct resolution in Figure 7-16(c).

(a) (b) (c)

Figure 7-16 Progressive picture presentation: (a) first phase, very unsharp image; (b) second phase, unsharp image; (c) third phase, sharp image.

Progressive image presentation is achieved by expanding quantization. This is equivalent to layered coding. For this expansion, a buffer is added at the output of the quantizer that temporarily stores all coefficients of the quantized DCT. Progressiveness is achieved in two different ways:

- Using spectral selection, in the first run only the quantized DCT-coefficients of each data unit's low frequencies are passed on to the entropy encoding. Successive runs gradually process the coefficients of higher frequencies.
- In successive approximation, all of the quantized coefficients are transferred in each run, but individual bits are differentiated according to their significance. The most significant bits are encoded before the least significant bits.

Besides Huffman coding, arithmetic entropy coding can be used in the expanded mode. Arithmetic coding automatically adapts itself to the statistical characteristics of an image and thus requires no tables from the application. According to several publications, arithmetic encoding achieves around five to ten percent better compression than Huffman encoding. Other authors presuppose a comparable compression rate. Arithmetic coding is slightly more complex and its protection by patents must be considered (Appendix L in [Org93]).

In the expanded mode, four coding tables are available for the transformation of DC- and AC-coefficients. The simpler mode allows a choice of only two Huffman tables each for the DC- and AC-coefficients of an image. The expanded mode thus offers 12 alternative types of processing, as listed in Table 7-2.

Image Display	Bits per Sample Value	Entropy Coding
Sequential	8	Huffman coding
Sequential	8	Arithmetic coding
Sequential	12	Huffman coding
Sequential	12	Arithmetic coding
Progressive successive	8	Huffman coding
Progressive spectral	8	Huffman coding
Progressive successive	8	Arithmetic coding
Progressive spectral	8	Arithmetic coding
Progressive successive	12	Huffman coding
Progressive spectral	12	Huffman coding
Progressive successive	12	Arithmetic coding
Progressive spectral	12	Arithmetic coding

Table 7-2 Alternative types of processing in expanded lossy DCT-based mode.

7.5.4 Lossless Mode

The lossless mode shown in Figure 7-17 uses single pixels as data units during image preparation. Between 2 and 16 bits can be used per pixel. Although all pixels of an image must use the same precision, one can also conceive of adaptive pixel precision.

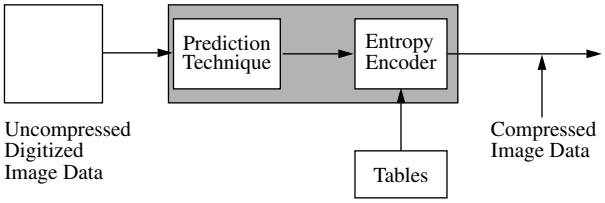


Figure 7-17 Lossless mode, based on prediction.

In this mode, image processing and quantization use a predictive technique instead of transformation coding. For each pixel *X* as shown in Figure 7-18, one of eight possible predictors is selected. The selection criterion is the best possible prediction of the value of *X* from the already known adjacent samples *A, B,* and *C*. Table 7-3 lists the specified predictors.

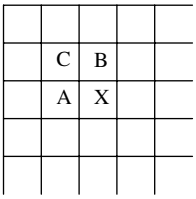


Figure 7-18 Basis of prediction in lossless mode.

The number of the chosen predictor, as well as the difference between the prediction and the actual value, are passed to the subsequent entropy encoding, which can use either Huffman or arithmetic coding.

In summary, this mode supports two types of processing, each with between 2 and 16 bits per pixel. Each variant can use either Huffman coding or arithmetic coding.

Selection Value	Prediction
0	No prediction
1	X=A
2	X=B
3	X=C
4	X=A+B+C
5	X=A+(B-C)/2
6	X=B+(A-C)/2
7	X=(A+B)/2

Table 7-3 Predictors in lossless mode.

7.5.5 Hierarchical Mode

The hierarchical mode can either use one of the lossy DCT-based algorithms described above or alternatively use the lossless compression technique, as the need arises. The main feature of this mode is the encoding of an image at different resolutions, that is, the compressed image contains images at several resolutions. To do this, the prepared digital image is first reduced by a factor of 2^n and compressed. The original image is then reduced by a factor of 2^{n-1} vertically and horizontally. The previously compressed image is subtracted from this, and the result is once again compressed. This process is successively repeated until the full resolution of the image is compressed.

Hierarchical coding is computationally intensive and requires considerable storage space. The advantage is the compressed image is available at different resolutions. Applications working with lower resolutions thus do not need to first decode the whole image and then reduce the resolution. In other words, scaling becomes cheap. According to the authors' practical experience, it is often more efficient to display an image in its full resolution than to first scale it down and display a smaller image. Yet, in the case of images encoded according to the hierarchical JPEG mode, the display of a reduced size picture requires less time to process than one of higher resolution.

7.6 H.261 (*p*×64) and H.263

The driving force behind the H.261 video standard was and is ISDN. One or both B-channels of a narrowband ISDN connection can transfer video data, for example, in addition to speech. This requires that both partners connected via the channel have to use the same coding of video data. In a narrowband ISDN connection, exactly two B-channels and one D-channel are available at the user interface. The European ISDN hierarchy also allows a connection with 30 B-channels, intended for PABX. In the fol-

lowing, when we speak of *the* B-channel, this refers to one or more channels. Early on, the primary applications of ISDN were videophone and videoconferencing systems. These dialogue applications require that coding and decoding be carried out in real time. In 1984, Study Group XV of CCITT established a committee of experts to draw up a standard for the compression of moving pictures [Lio91].

The resulting CCITT Recommendation H.261 *Video CoDec for Audiovisual Services at p×64Kbit/s* [ITUC90] was finalized after five years of work and accepted in December 1990. In this context, codec (Coder/Decoder) refers to encoding and decoding, or to compression and decompression. North America adopted the recommendation with slight modifications. Since data rates of $p \times 64$Kbit/s are considered, the recommendation is also known as $p \times 64$.

The ITU Study Group XV Recommendation H.261 was developed for real-time processing during encoding and decoding. The maximum combined signal delay for compression and decompression must not exceed 150ms. If the end-to-end delay is too great, the subjective interactivity of a dialogue application using this standard suffers considerably.

H.263 is a provisional ITU-T Standard published in 1996 to replace H.261 for many applications. H.263 was designed for low bit rate transmission. Early designs called for data rates under 64Kbit/s, though this was later revised. As part of the ITU-T H.320 family of standards (recommendation for real-time voice, data, and video over V.34 modems using a conventional GSTN telephone network), H.263 is used for a wide range of bit rates (not just low bit rate applications).

With respect to efficiency of video compression, H.263 is one of the best techniques available today. The H.263 coding algorithm is similar to that of H.261, with some improvements and changes to further improve performance and add error correction.

The following are the key differences between the H.261 and H.263 coding algorithms:

- H.263 uses half pixel precision for motion compensation, while H.261 uses full pixel precision with a "loop filter."
- Some parts of the hierarchical structure of the data stream are optional in H.263, so that the codec can be configured for a lower bit rate or better error correction.
- H.263 includes four optional, negotiable parameters to improve performance:
    - the unrestricted motion vector mode,
    - the syntax-based arithmetic coding mode,
    - the advanced prediction mode,
    - and forward and backward frame prediction (similar to the P and B frames in MPEG).

- H.263 can often achieve the same quality as H.261 with less than half as many bits by using the improved negotiable options.
- H.263 supports five resolutions. In addition to QCIF and CIF, which H.261 supports, H.263 also supports SQCIF, 4CIF, and 16CIF. SQCIF has about half the resolution of QCIF. 4CIF and 16CIF correspond to four and 16 times the resolution of CIF, respectively. Support for 4CIF and 16CIF means that the codec can unquestionably vie with other high bit rate coding standards, such as MPEG.

7.6.1 Image Preparation

Unlike JPEG, H.261 defines a very precise image format. The image refresh frequency at the input must be 30000/1001 = 29.97 frames/s. During encoding, it is possible to generate a compressed image sequence with a lower frame rate of, for example, 10 or 15 frames per second. Images cannot be presented at the input to the coder using interlaced scanning. The image is encoded as a luminance signal (Y) and chrominance difference signals C_b, C_r, according to the CCIR 601 subsampling scheme (2:1:1), which was later adopted by MPEG.

Two resolution formats, each with an aspect ratio of 4:3 are defined. The so-called Common Intermediate Format (CIF) defines a luminance component of 352 lines, each with 288 pixels. As per the 2:1:1 requirement, the chrominance components are subsampled with 176 lines, each with 144 pixels.

Quarter CIF (QCIF) has exactly half the resolution in all components (i.e., 176×144 pixels for the luminance and 88×72 pixels for the other components). All H.261 implementations must be able to encode and decode QCIF. CIF is optional.

The following example illustrates the compression ratio necessary to encode even an image with the low resolution of QCIF for the bandwidth of an ISDN B-channel. At 29.97 frames/s, the uncompressed QCIF data stream has a data rate of 9.115 Mbit/s. At the same frame rate, CIF has an uncompressed data rate of 36.45 Mbit/s. The image to be processed should be compressed at a rate of ten frames per second. This leads to a necessary compression ratio for QCIF of about 1:47.5, which can be easily supported by today's technology.

For CIF, a corresponding reduction to about six ISDN B-channels is possible. H.261 divides the Y as well as the C_b and C_r components into blocks of 8×8 pixels. A macro block is the result of combining four blocks of the Y matrix with one block each from the C_b and C_r components. A group of blocks consists of 3×11 macro blocks. A QCIF image thus consists of three groups of blocks and a CIF image consists of twelve groups of blocks.

7.6.2 Coding Algorithms

The H.261 standard uses two different methods of coding: intraframe and interframe. Intraframe coding under H.261 considers only the data from the image being

coded; this corresponds to *intrapicture* coding in MPEG (see Section 7.7.1). Interframe coding in H.261 uses data from other images and corresponds to P-frame coding in MPEG (see Section 7.7.1). The H.261 standard does not prescribe any criteria for using one mode or the other depending on specific parameters. The decision must be made during encoding and thus depends on the specific implementation.

H.263, unlike H.261, recommends four negotiable modes of intraframe coding. These can be used separately or together. An exception is that the advanced prediction mode requires use of the unrestricted motion vector mode.

The new intraframe coding modes added to H.263 are briefly described below:

1. Syntax-based arithmetic coding mode
   defines the use of arithmetic coding instead of variable-length coding. This results in identical image recoverability with better compression efficiency.
2. PB-frame mode
   can increase the frame rate without changing the bit rate by coding two images as one unit. The images must be a predicted, or P frame, and a B frame (as defined by MPEG; see Section 7.7.1), which is predicted bidirectionally from the preceding P frame and the current P frame.
3. Unrestricted motion vector mode
   makes it possible for motion vectors to point outside image boundaries. This is particularly useful for small images with movements in the direction of the edges.
4. Advanced prediction mode
   uses the Overlapped Block Motion Compensation (OBMC) technique for P-frame luminance. The coder can use one 16×16 vector or four 8×8 vectors for each macro block. Using smaller vectors requires more bits but yields better predictions, and in particular, fewer artifacts.

Like JPEG, for intraframe coding, each block of 8×8 pixels is transformed into 64 coefficients using DCT. Here also, DC-coefficients are quantized differently than AC-coefficients. The next step is to perform entropy encoding using variable-length code words.

In interframe coding, a prediction method is used to find the most similar macro block in the preceding image. The motion vector is defined by the relative position of the previous macro block with respect to the current macro block. According to H.261, a coder does not need to be able to determine a motion vector. Thus a simple H.261 implementation can always consider only differences between macro blocks located at the same position of successive images. In this case, the motion vector is always the zero vector. Next, the motion vector and the DPCM-coded macro block are processed. The latter is transformed by the DCT if and only if its value exceeds a certain threshold value. If the difference is less than the threshold, then the macro block is not encoded

any further and only the motion vector is processed. The components of the motion vector are entropy encoded using variable-length coding. This is lossless.

Transformed coefficients are all quantized linearly and entropy encoded using variable-length code words.

Optionally, an optical low pass filter can be inserted between the DCT and the entropy encoding. This filter deletes any remaining high-frequency noise. H.261 implementations need not incorporate this filter.

H.261 uses linear quantization. The step size is adjusted according to the amount of data in a transmission buffer, thereby enforcing a constant data rate at the output of the coder. This feedback also influences the image quality.

7.6.3 Data Stream

According to H.261/H.263, a data stream is divided into several layers, the bottom layer containing the compressed images. Some interesting properties of H.261 and H.263 are mentioned below (for further details, see [ITUC90]):

- The data stream contains information for error correction, although use of external error correction, e.g. H.223, is recommended.
- Each image in H.261 includes a five-bit number that can be used as a temporal reference. H.263 uses eight-bit image numbers.
- During decoding, a command can be sent to the decoder to "freeze" the last video frame displayed as a still frame. This allows the application to stop/freeze and start/play a video scene without any additional effort.
- Using an additional command sent by the coder, it is possible to switch between still images and moving images. Alternatively, instead of using explicit commands, a time out signal can be used.

7.6.4 H.263+ and H.263L

H.263+ is a planned extension of the existing H.263 standard. Improvements will probably be rather small, especially in the area of coding options. Examples of methods that will be incorporated in H.263+ are the 4×4 DCT, improved intra coding, and a deblocking filter in the prediction loop.

H.263L is further improvement on H.263 with a longer time horizon than H.263+. Here greater changes are expected. H.263L could coincide with the development of MPEG-4.

7.7 MPEG

MPEG was developed and defined by ISO/IEC JTC1/SC 29/WG 11 to cover motion video as well as audio coding. In light of the state of the art in CD technology, the goal was a compressed stream data rate of about 1.2Mbit/s. MPEG specifies a

maximum data rate of 1,856,000 bit/s, which should not be exceeded [ISO93b]. The data rate for each audio channel can be chosen between 32 and 448 Kbit/s in increments of 16 Kbit/s. This data rate enables video and audio compression of acceptable quality. Since 1993, MPEG has been an International Standard (IS) [ISO93b]. MPEG explicitly takes into account developments in other standardization activities:

- JPEG: Since a video sequence can be regarded as a sequence of still images and the JPEG standard development was always ahead of the MPEG standardization, the MPEG standard makes use of the results of the JPEG work.
- H.261: Since the H.261 standard already existed during the MPEG work, the MPEG group strived to achieve at least a certain compatibility between the two standards in some areas. This should simplify implementations of MPEG that also support H.261. In any case, technically MPEG is the more advanced technique. Conversely, H.263 borrowed techniques from MPEG.

Although mainly designed for asymmetric compression, a suitable MPEG implementation can also meet symmetric compression requirements. Asymmetric coding requires considerably more effort for encoding than for decoding. Compression is carried out once, whereas decompression is performed many times. A typical application area is retrieval systems. Symmetric compression is characterized by a comparable effort for the compression and decompression processing. This is a requirement for interactive dialogue applications, as is a bounded processing delay for the processing.

Besides the specification of video coding [Le 91, VG91] and audio coding, the MPEG standard provides a system definition, which describes the combination of individual data streams into a common stream.

7.7.1 Video Encoding

In the image preparation phase (according to the reference scheme shown in Figure 7-1), MPEG, unlike JPEG but similar to H.263, defines the format of an image very precisely.

7.7.1.1 Image Preparation

An image must consist of three components. In addition to the luminance Y there are two color difference signals C_r and C_b (similar to the YUV format). The luminance component has twice as many samples in the horizontal and vertical axes as the other components, that is, there is color subsampling. The resolution of the luminance component should not exceed 768×576 pixels. The pixel depth is eight bits in each component.

An MPEG data stream also contains information that is not part of a data stream compressed according to the JPEG standard, for example the pixel aspect ratio. MPEG supports 14 different pixel aspect ratios. The most important are:

- A square pixel (1:1) is suitable for most computer graphics systems.
- For a 625-line image, a ratio of 16:9 is defined (European HDTV).
- For a 525-line image, a ratio of 16:9 is also defined (U.S. HDTV).
- For 702×575 pixel images, an aspect ratio of 4:3 is defined.
- For 711×487 pixel images, an aspect ratio of 4:3 is also defined.

The image refresh frequency is also encoded in the data stream. So far, eight frequencies have been defined (23.976 Hz, 24 Hz, 25 Hz, 29.97 Hz, 30 Hz, 50 Hz, 59.94 Hz, and 60 Hz), so no low image refresh frequencies are permitted.

Temporal prediction of still images usually yields a considerable data reduction. Areas within an image with strong, irregular motions can only be reduced by a ratio similar to that of intraframe coding. The use of temporal predictors requires the storage of a huge amount of previously determined information and image data. The extent to which prediction is employed can be determined by balancing the required storage capacity against the achievable compression rate.

In most cases, predictive coding only makes sense for parts of an image and not for the whole image. The image is thus divided into areas called macro blocks. An MPEG macro block is partitioned into 16×16 pixels for the luminance component and 8×8 pixels for each of the two chrominance components. These sizes are well suited for compression based on motion estimation. This is a compromise between the computational effort required for estimation and the resulting data reduction. A macro block is formed of six blocks of 8×8 pixels, ordered as follows: first four blocks for the luminance component then the two chrominance blocks. There are no user-defined MCUs as in JPEG since, given the defined frame rates, the maximum time to present an image is 41.7 ms. The three components are compressed and decompressed together. From the MPEG user's perspective, there is no fundamental advantage to progressive image display over sequential display.

7.7.1.2 Image Processing

For the image processing stage, MPEG supports four types of image coding. The reason for this is the contradictory demands of efficient coding and random access. In order to achieve a high compression ratio, temporal redundancies of successive images need to be exploited. Fast random access requires that images be coded individually. Hence the following image types are distinguished:

- I frames (intra coded pictures) are coded without using information about other frames (intraframe coding). An I frame is treated as a still image. Here MPEG falls back on the results of JPEG. Unlike JPEG, real-time compression must be possible. The compression rate is thus the lowest within MPEG. I frames form the anchors for random access.
- P frames (predictive coded pictures) require information about previous I and/or P frames for encoding and decoding. Decoding a P frame requires decompression of

the last I frame and any intervening P frames. In return, the compression ratio is considerably higher than for I frames. A P frame allows the following P frame to be accessed if there are no intervening I frames.

- B frames (bidirectionally predictive coded pictures) require information from previous and following I and/or P frames. B frames yield the highest compression ratio attainable in MPEG. A B frame is defined as the difference from a prediction based on a previous and a following I or P frame. It cannot, however, ever serve as a reference for prediction coding of other pictures.
- D frames (DC coded pictures) are intraframe-coded and can be used for efficient fast forward. During the DCT, only the DC-coefficients are coded; the AC-coefficients are ignored.

Figure 7-19 shows a sequence of I, P, and B frames. This example illustrates the prediction for the first P frame and the bidirectional prediction for a B frame. Note that the order in which the images are presented differs from the actual decoding order if B frames are present in an MPEG-coded video stream.

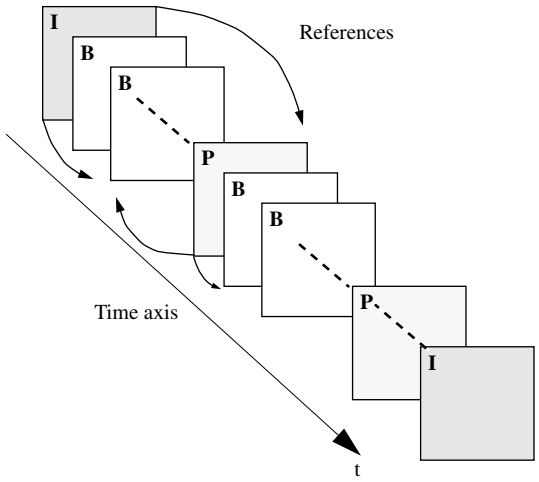


Figure 7-19 Types of individual images in MPEG: I, B, and P frames.

The pattern of I, P, and B frames in a sequence is determined by the MPEG application. For random access, the ultimate resolution would be attained by encoding the entire stream using I frames. The highest compression rate can be achieved by using as many B frames as possible. For practical applications, the sequence `IBBPBBPBB IBBPBBPBB`... has proven to be useful. This permits random access with a resolution of nine still images (i.e., about 330ms) and still provides a very good compression ratio. Every 15 images includes one I frame.

The following detailed description of image processing, quantization and entropy encoding distinguishes the four image types.

7.7.1.3 I Frames

I frames are encoded by performing a DCT on the 8×8 blocks defined within the macro blocks, as in JPEG. The DC-coefficients are then DPCM coded, and differences between consecutive blocks of each component are calculated and transformed into variable-length code words. AC-coefficients are run-length encoded and then transformed into variable-length code words. MPEG distinguishes two types of macro blocks: those that contain only coded data and those that additionally contain a parameter used for scaling the characteristic curve used for subsequent quantization.

7.7.1.4 P Frames

The coding of P frames exploits the fact that in consecutive images, some areas of the image often shift (move), but do not change. To encode a block, the most similar macro block in the preceding image must be determined. This is done by calculating the differences between the absolute values of the luminance component for each macro block in the preceding image. The macro block with the lowest sum of differences is the most similar. MPEG does not specify an algorithm for motion estimation, but rather specifies the coding of the result. Only the motion vector (the spatial difference between the two macro blocks) and the small difference between the macro blocks need to be encoded. The search range, that is, the maximum length of the motion vector, is not defined by the standard. As the search range is increased, the motion estimation becomes better, although the computation becomes slower [ISO93b].

P frames can consist of macro blocks as in I frames, as well as six different predictive macro blocks. The coder essentially decides whether to code a macro block predictively or like an I frame and whether it should be coded with a motion vector. A P frame can thus also include macro blocks that are coded in the same way as I frames.

In coding P-frame-specific macro blocks, differences between macro blocks as well as the motion vector need to be considered. The difference values between all six 8×8 pixel blocks of a macro block being coded and the best matching macro block are transformed using a two-dimensional DCT. Further data reduction is achieved by not further processing blocks where all DCT coefficients are zero. This is coded by inserting a six-bit value into the encoded data stream. Otherwise, the DC- and AC-coefficients are then encoded using the same technique. This differs from JPEG and from the coding of I frame macro blocks. Next, run-length encoding is applied and a variable-length coding is determined according to an algorithm similar to Huffman. Since motion vectors of adjacent macro blocks often differ only slightly, they are DPCM coded. The result is again transformed using a table; a subsequent calculation performs a transformation into variable-length coded words.

7.7.1.5 B Frames

In addition to the previous P or I frame, B-frame prediction takes into account the following P or I frame. The following example illustrates the advantages of bidirectional prediction:

In a video sequence, a ball moves from left to right in front of a static background. In the left area of the scene, parts of the image appear that in previous images were covered by the moving ball. A prediction of these areas would ideally be derived from the following, not from the previous, image. A macro block can be derived from macro blocks of previous and following P and/or I frames. Motion vectors can also point in orthogonal directions (i.e., in x direction and in y direction). Moreover, a prediction can interpolate two similar macro blocks. In this case, two motion vectors are encoded and one difference block is determined between the macro block to be encoded and the interpolated macro block. Subsequent quantization and entropy encoding are performed as for P-frame-specific macro blocks. Since B frames cannot serve as reference frames for subsequent decoding, they need not be stored in the decoder.

7.7.1.6 D Frames

D frames contain only the low-frequency components of an image. A D-frame always consists of one type of macro block and only the DC-coefficients of the DCT are coded. D frames are used for fast-forward display. This could also be realized by a suitable placement of I frames. For fast-forward, I frames must appear periodically in the data stream. In MPEG, slow-rewind playback requires considerable storage. All images in a so-called group of pictures must decoded forwards and stored before they can be displayed in reverse.

7.7.1.7 Quantization

Concerning quantization, it should be noted that AC-coefficients of B and P frames are usually very large values, whereas those of I frames are very small. Thus, MPEG quantization quantization adjusts itself accordingly. If the data rate increases too much, quantization becomes more coarse. If the data rate falls, then quantization is performed with finer granularity.

7.7.2 Audio Coding

MPEG audio coding is compatible with the coding of audio data used for Compact Disc Digital Audio (CD-DA) and Digital Audio Tape (DAT). The most important criterion is the choice of sample rate of 44.1 kHz or 48 kHz (additionally 32 kHz) at 16 bits per sample value. Each audio signal is compressed to either 64, 96, 128, or 192 Kbit/s.

Three quality levels (layers) are defined with different encoding and decoding complexity. An implementation of a higher layer must be able to decode the MPEG audio signals of lower layers [Mus90].

Similar to two-dimensional DCT for video, a transformation into the frequency domain is applied for audio. The Fast Fourier Transform (FFT) is a suitable technique. As shown in Figure 7-20, the relevant portion of the spectrum is divided into 32 non-overlapping subbands. The audio signal is thus split into 32 subbands. Different components of the spectrum can then be quantized differently. In parallel with the actual FFT, the noise level in each subband is determined using a psychoacoustic model. At a higher noise level, a coarser quantization is performed. A lower noise level results in finer quantization. In the first and second layers, the appropriately quantized spectral components are simply PCM-encoded. The third layer additionally performs Huffman coding.

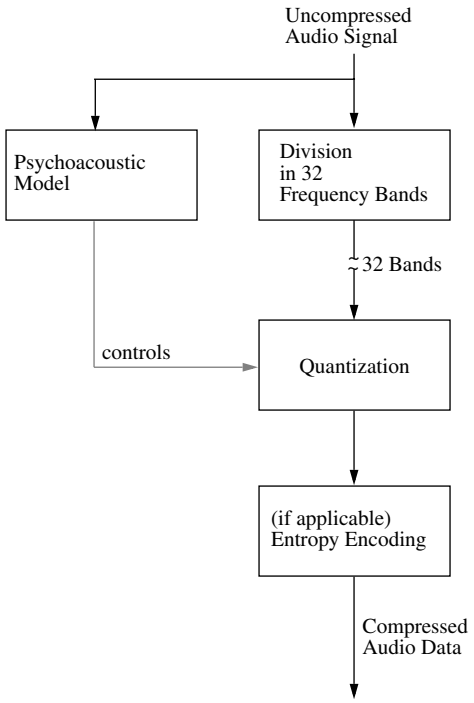


Figure 7-20 MPEG audio encoding.

Audio coding can be performed on stereo sound, a single channel, or two independent channels. MPEG provides for two types of stereo sound. In the first case, two channels are processed completely independently. In the joint stereo mode, MPEG

achieves a higher compression ratio by exploiting redundancies between the two channels.

Each layer defines 14 fixed bit rates for the coded audio data stream, which in MPEG are addressed by a bit rate index. The minimal value is always 32 Kbit/s. The layers support different maximal bit rates: layer 1 allows for a maximum of 448 Kbit/s, layer 2 for 384 Kbit/s, and layer 3 for 320 Kbit/s. For layers 1 and 2, a decoder is not required to support a variable bit rate. In layer 3, a variable bit rate is specified by allowing the bit rate index to be switched. In layer 2, not all combinations of bit rate and mode are allowed:

- 32 Kbit/s, 48 Kbit/s, 56 Kbit/s, and 80 Kbit/s are only allowed for a single channel.
- 64 Kbit/s, 96 Kbit/s, 112 Kbit/s, 128 Kbit/s, 160 Kbit/s, and 192 Kbit/s are allowed for all modes.
- 224 Kbit/s, 256 Kbit/s, 320 Kbit/s, and 384 Kbit/s are allowed for the stereo, joint stereo and dual channel modes.

7.7.3 Data Stream

Like JPEG, MPEG specifies a precise syntax for the compressed audio and video data streams.

7.7.3.1 Audio Stream

An audio stream is comprised of frames, which are made up of audio access units, which in turn are divided into slots. At the lowest coding complexity (layer 1), a slot consists of 4 bytes; otherwise it is one byte. Frames consist of a fixed number of samples. An audio access unit is the smallest compressed audio sequence that can be completely decoded independently of all other data. At 48 kHz, the audio access units contained in a frame have a play time of 8 ms; at 44.1 kHz, the play time is 8.7 ms; and at 32 kHz, the play time is 12 ms. In the case of stereo, data from both channels are included in one frame.

7.7.3.2 Video Stream

A video stream is comprised of 6 layers:

1. At the highest level, the sequence layer, data buffering is handled. For example, it does not make sense to generate a data stream that places excessive demands on storage space in the decoder. Among other things, the beginning of the sequence layer thus includes two entries: the constant bit rate of the sequence and the minimum storage capacity required during decoding.

   A video buffer verifier is added after the quantizer that uses the bit rate resulting from decoding in order to monitor the decoding delay. The video buffer verifier influences the quantizer and forms a type of control loop. Successive sequences

can have varying bit rates. During decoding of several consecutive sequences, often no data are output between the end of one sequence and the beginning of the next sequence because the underlying decoder parameters need to be updated and an initialization performed.

2. The group of pictures layer is the next layer. This layer contains at least an I frame, which must be one of the first images. Random access to this image is always possible. At this layer, it is possible to distinguish between the order of images in the data stream and their display order. The data stream must always begin with an I frame so that the decoder can first decode and store the reference image. In order of presentation, however, B frames can precede the I frame. The following example illustrates the difference between decoding order and display order:

Decoding order													
Type of Frame	I	P	B	B	P	B	B	P	B	B	I	B	B
Frame number	1	4	2	3	7	5	6	10	8	9	13	11	12

Display order													
Type of Frame	I	P	B	B	P	B	B	P	B	B	I	B	B
Frame number	1	2	3	4	5	6	7	8	9	10	11	12	13

3. The picture layer contains a whole still image. The image's temporal reference is defined using an image number. This number is shown below each image in the example above. There are also data fields defined in this layer that are not yet used in MPEG. These fields are reserved for future extensions and may not be used by the decoder.
4. The next layer is the slice layer. Each slice consists of macro blocks, the number of which can vary from image to image. A slice also includes the scaling used for DCT quantization of all its macro blocks.
5. The next layer is the macro block layer. This contains a macro block with the properties described above.
6. The lowest layer is the block layer, also described above.

7.7.3.3 System Definition

The MPEG system definition also specifies the combination of both data streams into a single stream. The most important task of this process is the actual multiplexing. It includes the coordination of input data streams with output data streams, clock

adjustment, and buffer management. Data streams as defined by ISO 11172 are divided into separate packs. The decoder gets the information it needs from the multiplexed data stream. For example, the maximal data rate is included at the beginning of every ISO 11172 data stream in the first pack.

This definition of the data stream makes the implicit assumption that it is possible to read such a header prior to the first (possibly random) data access from a secondary storage medium. In dialogue services over communications networks, such as telephone or videophone, a participant will always get the header information first. On the other hand, this would be inconvenient in a conferencing application where a user can join at any time. The required header information would not then be immediately available, since it transmitted only at the beginning of an ISO 11172 data stream. One could, however, define a protocol to supply the header upon request.

MPEG also supplies time stamps necessary for synchronization in ISO 11172 data streams. These concern the relationship between multiplexed data streams within an ISO 11172 data stream and not other ISO 11172 data streams that may exist.

It should be noted that MPEG does not prescribe compression in real-time. Moreover, MPEG defines the decoding process but not the decoder itself.

7.7.4 MPEG-2

Today one assumes that the quality of a video sequence compressed according to the MPEG standard at the maximum data rate of about 1.5 Mbit/s cannot be substantially improved. Here only the results (the compression ratio and the quality) should count, not the required processing expenditure. Thus a video compression technique is being developed for rates up to 100 Mbit/s. This is known as “MPEG-2” [ISO93a]. The previously established technique is now known as “MPEG-1,” while MPEG-2 aims at a higher image resolution, similar to the CCIR 601 digital video studio standard.

To ensure a harmonized solution that covered a wide range of applications, the ISO/IEC Working Group ISO/IEC JTC1/SC29/WG11 developed MPEG-2 in close cooperation with the ITU-TS Study Group 15 Experts Group for ATM Video Coding. In addition to these two groups, other representatives of the ITU-TS, EBU, ITU-RS, SMPTE, and North American HDTV developers also worked on MPEG-2.

The MPEG group developed the MPEG-2 Video standard, which specifies the coded bit stream for higher quality digital video. As a compatible extension, MPEG-2 video builds on the MPEG-1 standard by supporting interlaced video formats and a number of other advanced features, including support for HDTV.

	Profile name	Simple Profile	Main Profile	SNR Scaling Profile	Spatial Scaling Profile	High Profile
Profiles		no B frames	B frames			
		4:2:0				4:2:0 or 4:2:2
	Characteristics of profile	not scalable		SNR scalable	SNR or spatially scalable	
Levels	**High Level** 1920 pixels/line 1152 lines		≤ 80 Mbit/s			≤ 100 Mbit/s
	High-1440 Level 1440 pixels/line 1152 lines		≤ 60 Mbit/s		≤ 60 Mbit/s	≤ 80 Mbit/s
	Main Level 720 pixels/line 572 lines	≤ 15 Mbit/s	≤ 15 Mbit/s	≤ 15 Mbit/s		≤ 20 Mbit/s
	Low Level 352 pixels/line 288 lines		≤ 15 Mbit/s	≤ 15 Mbit/s		

Table 7-4 MPEG-2 Profiles and Levels. (Empty cells contain undefined values.)

As a generic international standard, MPEG-2 Video was defined in terms of extensible profiles, each of which supports the features needed by a class of applications. The MPEG-2 Main Profile was defined to support digital video transmission in the range of about 2 to 80 Mbit/s using cable, satellite, or other broadcast channels, as well as to support digital storage and other communications applications. The parameters of the Main Profile and the High Profile are suitable for supporting HDTV.

The MPEG experts also extended the features of the Main Profile by defining a hierarchical/scalable profile. This profile aims to support applications such as compatible terrestrial TV, packet-network video systems, backwards compatibility with existing standards (MPEG-1 and H.261), and other applications requiring multi-level coding. For example, such a system could offer a consumer the option of using a small portable receiver to decode standard definition TV from a broadcast signal, or using a larger fixed receiver to decode HDTV from the same signal.

The MPEG-2 profiles arranged in a 5×4 matrix as shown in Table 7-4. The degree of functionality increases along the horizontal axis. The vertical axis indicates levels

with increased parameters, such as smaller and larger frame sizes. For example, the Main Profile in the Low Level specifies 352 pixels/line with 288 lines/frame and 30 frames/s, without any B frames and a data rate not to exceed 4 Mbit/s. The Main Profile in the High Level specifies 1920 pixels/line with 1152 lines/frame and 60 frames/s with a data rate not to exceed 80 Mbit/s.

MPEG-2 incorporates, in a way similar to the hierarchical mode of JPEG, scaling of compressed video [GV92]. In particular, video is compressed at different qualities during encoding, so that different alternatives are available during decompression [Lip91, GV92]. Scaling may act on various parameters:

- Spatial scaling facilitates decompression of image sequences with different horizontal and vertical resolutions. For example, a single data stream could include images with 352×288 pixels (H.261 CIF format), 360×240 pixels, 704×576 pixels (a format according to CCIR 601) and, for example 1,250 lines at an aspect ratio of 16:9. These resolutions refer to the luminance component; the chrominance component is subsampled by a factor of two. This can be implemented using a pyramid for the level of the DCT-coefficients [GV92], that is, 8×8 DCT, 7×7 DCT, 6×6 DCT, and other transformations can be formed. From a technical standpoint, only steps by a factor of two are useful.
- Scaling the data rate allows for playback at different frame rates. In MPEG-1, this functionality is achieved by using D frames. This can be implemented in MPEG-2 by using I frames, given a suitable distribution of I frames within the data stream. This condition must be met not only for a group of pictures, but also for the entire video sequence.
- Amplitude scaling can be interpreted as affecting either the pixel depth or the resolution at which DCT-coefficients are quantized. This then leads to a layered coding and to the possibility of progressive image presentation, which is not relevant for video data. However, this can be of interest if certain images of an image sequence can be extracted from the data stream as still images.

Scaling is one of the most important MPEG-2 extensions to MPEG-1. MPEG-2 also, for example, minimizes the effects of the loss of individual ATM cells in an MPEG-2 coded data stream. Sequences of different frame types (I, P, B) should be defined to minimize end-to-end delay at a given data rate.

The MPEG-2 group developed the MPEG-2 Audio Standard for low bit rate coding of multichannel audio. MPEG-2 supplies up to five full bandwidth channels (left, right, center, and two surround channels), plus an additional low-frequency enhancement channel and/or up to seven commentary/multilingual channels. The MPEG-2 Audio Standard also extends MPEG-1 coding of stereo and mono channels with half sampling rates (16 kHz, 22.05 kHz, and 24 kHz), which significantly improves quality at 64 Kbit/s or less per channel.

The MPEG-2 Audio Multichannel Coding Standard is backwards compatible with the existing MPEG-1 Audio Standard. The MPEG group carried out formal subjective testing of the proposed MPEG-2 Multichannel Audio codecs and up to three nonbackward-compatible codecs working at rates from 256 to 448 Kbit/s.

7.7.4.1 MPEG-2 System

MPEG-2 addresses video together with associated audio. Note that in order to provide a precise description, this section uses terminology from the original MPEG-2 specification. MPEG-2 defines how an MPEG-2 system can combine audio, video, and other data in a single stream, or in multiple streams, suitable for storage and transmission. MPEG-2 imposes syntactic and semantic rules that are necessary and sufficient to synchronize the decoding and presentation of video and audio information, while ensuring that the decoder's coded data buffers do not overflow or run out of data. The streams include time stamps used in the decoding, presentation, and delivery of the data.

In the first step, the basic multiplexing approach adds system-level information to each stream. Each stream is then divided into packets, forming a Packetized Elementary Stream (PES). Next, the PESs are combined into a Program Stream or a Transport Stream. Both streams were developed to support a large number of known or anticipated applications. They thus incorporate a significant degree of flexibility while ensuring interoperability between different device implementations.

- The Program Stream resembles the MPEG-1 stream, but should only be used in a relatively error-free environment. Program stream packets are of variable length. Timing information in this stream can be used to implement a constant end-to-end delay (along the path from the input of the coder to the output of the decoder).
- The Transport Stream bundles the PESs and one or more independent time bases into a single stream. This stream was developed for use with lossy or noisy media. Each packet has a length of 188 bytes, including a four-byte header. The Transport Stream is well-suited for transmission of digital television and video telephony over fiber, satellite, cable, ISDN, ATM, and other networks, as well as for storage on digital video tape and other devices.

A conversion between the Program Stream and the Transport Stream is possible and sensible. Note that the MPEG-2 buffer management specification limits the end-to-end delay for audio and video data to under one second, a value unacceptably high for users of dialogue mode applications.

A typical MPEG-2 video stream has a variable bit rate. By using a video buffer as specified in the standard, a constant bit rate can be enforced, at the cost of varying quality.

MPEG-2 reached the CD (Committee Draft) status in late 1993 and required three additional months to become a DIS (Draft International Standard). After a six-month

ballot period, the DIS became an IS (International Standard). Originally, there were plans to specify an MPEG-3 standard covering HDTV. However, during the development of MPEG-2, it was found that scaling up could easily meet the requirements of HDTV. Consequently, MPEG-3 was dropped.

7.7.5 MPEG-4

Work on another MPEG initiative for very low bit rate coding of audio visual programs started in September 1993 in the ISO/IEC JTC1. Formally designated ISO/IEC 14496, MPEG-4 was published in November 1998 and adopted as an international standard in January 1999.

MPEG-4 incorporates new algorithmic techniques, such as model-based image coding of human interaction with multimedia environments and low bit-rate speech coding for use in environments like the European Mobile Telephony System (GSM). The most important innovation is improved flexibility. Developers can use the compression method in different ways and configure systems to support a multitude of applications. MPEG-4 is thus not a fixed standard suitable only for a few applications. Moreover, MPEG-4 integrates a large number of audiovisual data types, for example natural and synthetic, aimed at a representation in which content-based interactivity is supported for all media types. MPEG-4 thus allows the design of audiovisual systems that are oriented around specific users yet are compatible with other systems.

Most currently available audiovisual services allow only playback functionality. In contrast, MPEG-4 places a strong emphasis on interactivity. Support for random access to audio and video scenes as well as the ability to revise content are thus among the main objectives of the MPEG-4 standardization. MPEG-4 provides for a universal coding of various forms of audiovisual data, called audiovisual objects. In other words, MPEG-4 seeks to represent the real world as a composition of audiovisual objects. A script then describes the spatial and temporal relationships among the objects. Using this form of representation, the user can interact with the various audiovisual objects in a scene in a manner that corresponds to the actions of everyday life.

Although this content-based approach to scene representation may seem obvious to the human user, it actually represents a revolution in video representation since it enables a quantum leap in the functionality that can be offered to the user. If a scene is represented as a collection of a few or many independent audiovisual objects, the user has the opportunity to play with the contents of the scene, for example by changing the properties of some objects (e.g., position, motion, texture, or form), accessing only selected parts of a scene, or even using cut and paste to insert objects from one scene into another scene. Interaction with contents is thus a central concept of MPEG-4.

Another major shortcoming of other audiovisual coding standards is the restricted number of audio and video data types used. MPEG-4 attempts to smoothly integrate natural and synthetic audiovisual objects, for example mono, stereo, and multiple

channel audio. Moreover, MPEG-4 supports either 2-D or 3-D (mono or stereo, respectively) video modes, as well as these from additional camera perspectives (so-called multiview video*).* This integration should be extended with regard to audiovisual interrelationships so that processing can incorporate the mutual influence and dependence between different types of information. Moreover, new and already available analysis and coding tools are being integrated into MPEG-4. In the future, MPEG-4 as a whole will be further developed as new or better tools, data types, or functionality becomes available.

The quick pace of technological progress in recent years has accentuated the inflexibility of standards that do not allow for the continuous development of hardware and methods. Such standards implement only specific solutions that very quickly no longer reflect technological developments. Flexibility and extensibility are thus important objectives of MPEG-4. MPEG-4 provides flexibility and extensibility through the "MPEG-4 Syntactic Description Language (MSDL)." The MSDL approach to extending audiovisual coding standards is revolutionary. Not only can new algorithms be integrated by selecting and integrating predefined tools (level 1), but they can be "learned" in that the coder can download new tools. At the same time, even MSDL is subject to further development since the MPEG-4 standard can always incorporate new tools, algorithms, and concepts in MSDL to support new or improved functionalities.

The areas of telecommunications, computers, and television/film are converging and are in the position of being able to exchange elements that previously were typical for only one of the three areas. This convergence is more likely viewed as evolutionary because it is taking place through gradual transitions. Because qualitatively new multimedia services are emerging from this convergence, a logical consequence is that new requirements will be placed on coding and transmission techniques. With its official focus on the three driving forces of content and interaction*,* integration and flexibility and extensibility*,* MPEG-4 seems to be the appropriate answer to such requirements.

7.7.5.1 MPEG-4 Extensions Compared with MPEG-2

The vision of the MPEG-4 standard can be best explained using the eight new or improved functionalities identified in the MPEG-4 Proposal Package Description. These were the result of a meeting held to determine which features could be of significance in the near future but that were not (or only partially) supported by present-day coding standards.

MPEG-4 must additionally provide some other important so-called standard functions included in previously available standards, for example synchronization of audio and video, low-delay modes, and interoperability between networks. Unlike the new or improved functions, the standard functions can be provided by standards that either exist or are under development.

MPEG-4 makes it possible to scale content, spatial and temporal resolution, quality, and complexity with finer granularity. This content-based scalability implies

the existence of a priority mechanism for a scene's objects. The combination of various scaling methods can lead to interesting scene representations, whereby more important objects are represented using higher spatial-temporal resolution. Content-based scaling is a key part of the MPEG-4 vision, since other features can easily be implemented once a list of a scene's important objects is available. In some cases, these and related features may require analyzing a scene to extract the audiovisual objects, depending on the application and on previous availability of composition information.

MPEG-4 incorporates a syntax and various coding methods to support content-based manipulation and bit stream editing, without requiring transcoding (conversion from one coding system to another). This means that the user should be able to access a specific object within a scene or bit stream, thus making it possible to easily change the object's properties.

MPEG-4 offers efficient, content-based tools to access and organize multimedia data. These features include indexing, addition of hyperlinks, queries, viewing data, uploading or downloading data, and deleting data.

MPEG-4 supports efficient methods of combining synthetic and natural scenes (e.g., text and graphics overlays), the ability to code natural and synthetic audio and video data, as well as methods of mixing synthetic data with conventional video or audio under decoder control. This feature of MPEG-4 enables extensive interaction features. The hybrid coding of natural and synthetic data allows, for the first time, a smooth integration of natural and synthetic audiovisual objects and thus represents a first step towards the complete integration of all sorts of types of audiovisual information.

MPEG-4 supports scenes with different views/sound tracks and can efficiently code them together with sufficient information to synchronize the resulting basic streams. For video applications that use stereoscopic pictures or multiple views, this means that redundancy between multiple views of the same scene can be exploited. Furthermore, this permits solutions that are compatible with normal (mono) video. Coding multiple, simultaneous data streams should provide an efficient representation of natural 3-D objects if a sufficient number of views is available. On the other hand, this can necessitate a complex analysis process. This functionality should especially benefit applications that until now have used almost exclusively synthetic objects, for example in the area of virtual reality applications.

The strong growth of mobile networks in particular has resulted in a need for improved coding efficiency. MPEG-4 is thus needed in order to provide substantially better audiovisual quality than either existing standards or standards under development (e.g. H.263), at comparably low bit rates. It should be noted that the simultaneous support of other functionality does not necessarily further compression efficiency. However, this is not problematic since different coder configurations can be used in different situations. Subjective tests carried out with MPEG-4 in November 1995 showed that, in

terms of coding efficiency, the available coding standards performed very well in comparison to most other proposed techniques.

Universal accessibility implies access to applications over a wide range of networks (both wireless and cable-based) and storage media. MPEG-4 must therefore work robustly in environments susceptible to errors. Especially for low bit rate applications, the system must provide sufficient robustness against serious errors. The approach used is not to replace the error-control techniques that are implemented in the network, but rather to offer elasticity against residual errors. This can be achieved with, for example, selective forward error correction, error containment, or error concealment.

MPEG-4 incorporates efficient methods to improve temporal random access to parts of an audiovisual sequence in a limited time interval with fine resolution. These include conventional techniques for achieving random access at very low bit rates.

7.7.5.2 Audiovisual Objects (AVOs) in MPEG-4

Audiovisual scenes in MPEG-4 are composed of audiovisual objects (AVOs), which are organized in a hierarchical fashion.

Primitive AVOs are found at the lowest level of the hierarchy. Examples are:

- a two-dimensional fixed background,
- the image of a talking person (without background) or
- the speech associated with the person.

MPEG-4 standardizes a number of these primitive AVOs that can be used to represent natural as well as synthetic types of content, which can be either two- or three-dimensional.

MPEG-4 defines the coded representation of such objects, for example:

- text and graphics,
- heads of speaking actors and associated text that the receiver can use to synthesize corresponding speech and animate the head accordingly,
- animated human bodies.

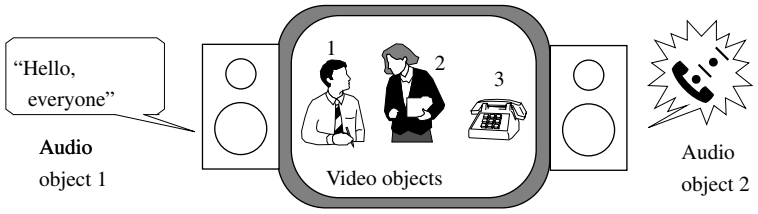


Figure 7-21 Example of an audiovisual scene in MPEG-4.

AVOs are individually coded in order to achieve maximum efficiency. They can be represented independently of other AVOs or of the background information.

MPEG is working with representatives of organizations that manage electronic media rights on defining a syntax for storing information about Intellectual Property Rights (IPR) pertaining to MPEG-4 AVOs and on developing tools to support IPR identification and IPR protection. The MPEG-4 standard thus incorporates support for unambiguous identification using attributes assigned by international naming authorities. These can be used to identify the current holder of the rights to an AVO. Protection of the content will be part of a subsequent version of MPEG-4.

7.7.5.3 Combining AVOs into Scenes

AVOs are combined in a hierarchical fashion to make audiovisual scenes. The result is a dynamic, tree-like structure of AVOs that can be changed through user interaction. Since each AVO possesses spatial and temporal components and can be located in relation to other AVOs, each AVO has coordinates that result from those of its parent AVO in the tree structure for the audiovisual scene.

7.7.5.4 Coding of Visual Objects

MPEG-4 uses a variety of methods to code visual objects. These are defined for natural images and videos as a set of tools and algorithms for the efficient compression of images, videos, textures, and 2-D and 3-D meshes, as well as geometry streams that animate these meshes in a time-varying manner. Tools to randomly access and manipulate all types of visual objects are available. Furthermore, content-based coding as well as content-based spatial, temporal, and qualitative scaling are supported. For natural content, there are mechanisms that provide robustness against errors and elasticity against errors in environments susceptible to errors.

To code synthetic objects, MPEG-4 defines parametric description tools as well as animated streams of human faces and bodies for static and dynamic mesh coding with texture mapping and texture coding. MPEG-4 uses wavelet compression techniques to efficiently code textures. Integration of other standards for synthetic audiovisual content, such as VRML, is planned as a future extension to MPEG-4.

In efficiently coding multimedia content the greatest gains can be gotten from video compression. Video coding is thus a particularly important aspect of the MPEG-4 standard. Three fundamental video coding extensions, which are realized by MPEG-4, are described below in detail:

1. Object-based scene layering and separate coding and decoding of layers

   In order to be able to support content-based functionalities, before processing a video scene, the coder must be able to divide the scene into layers that represent its physical objects. For example, one could divide a video scene into three layers O_1, O_2, and O_3, whereby O_1 represents the background, O_2 represents the person

in the foreground, and O_3 represents the telephone (see Figure 7-22). Each object is then coded separately and transmitted over its respective bit stream layer.

Figure 7-22 Example of division of a video sequence.

The layering approach has the major advantage that each bit stream layer can be independently processed and separately coded and decoded. This achieves efficient coding and permits content-based functionalities.

If transmission takes place in environments susceptible to errors, different layers can be treated differently. For example, better error protection could be used for the foreground layer O_2 than for the two other layers if the recipient is more interested in the foreground object. Then at least the foreground layer can be decoded with sufficient quality if there is significant noise in the transmission channel. In other applications, the user could be interested in showing only the foreground layer when the video is being edited, manipulated, and mixed. The person in the foreground could then be placed in a scene coming from another video with a different background (possibly even a synthetic scene that was created through computer graphics). The bit stream layer O_3 can be accessed directly. This can also be inserted in the bit stream of another video scene without requiring that one of the two scenes be additionally segmented and transcoded.

2. Shape-adaptive DCT coding

After a scene has been successfully segmented, creating the different object layers, each layer is coded separately. This is done using a DCT coding technique that adapts itself to the object's shape. The fundamental structure of this technique can be considered an extension to conventional block-based hybrid DCT algorithms that use motion compensation and that code the different object layers. In contrast to previous standards, images to be coded in each object layer are no longer considered to be rectangular regions. Their shape and position can change between successive frames.

In order to model and optimize the performance of an algorithm for the functional coding of input sequences of arbitrarily shaped images at a bit rate of around 1 Mbit/s, the standard MPEG-1 coding system was extended with a shape coding algorithm and a shape-adaptive DCT technique. Shape-adaptive DCT allows transformation coding of image blocks of any shape.

For each object layer, the object shape information is transmitted first. This is followed by the blocks' motion vectors and the DCT coefficients corresponding to the motion and to the object's texture information as described by luminance and chrominance. In order to permit separate decoding of each object layer, the coding uses no information about shape, motion, or texture outside of the respective layer. As in previous MPEG definitions, input images to be encoded are split into grids of macro blocks and blocks. A block motion vector is transmitted for each macro block. For rectangular blocks, the SA-DCT algorithm reduces to the standard block DCT algorithm.

3. Object-based tool box for motion prediction

   The basic SA-DCT algorithm was developed in order to implement MPEG-4's object-based functionalities. The algorithm reduces temporal redundancy by using block-based motion compensation, comparable to that of the previous MPEG coding algorithm. An important advantage of the SA-DCT object-based approach is the significant increase in compression efficiency achieved by using appropriate tools for motion prediction in each object layer.

   The alternative prediction tools Pred 1–Pred 3, for example, extend the basic SA-DCT motion compensation approach with object-based motion prediction techniques. After the initial scene segmentation and layered description of the video content, each object layer can be classified in view of specific object properties during initial scene analysis (e.g., constant background with or without global camera movement motion, constant foreground objects with global motion or flexible foreground objects with related motion). This classification can be used to apply different tools for temporal prediction in layers with different properties. The set of motion prediction algorithms embedded in the basic SA-DCT coding system is also described as an object-based prediction tool box, in other words as a set of motion prediction tools that have been optimized for the motion statistics of various object layers having different properties in the object model.

   For flexible foreground objects with corresponding motion (e.g., the foreground person O_2) or for constant objects with global motion (e.g., a car) it might be more suitable to code and transmit fixed-motion vector fields or global motion parameters than block vectors. Moreover, a background with global camera motion can be coded very efficiently by estimating and transmitting global camera parameters, such as zoom, rotation, and translation, that can be mapped to a stationary panorama image of a constant background. This implies implementation within the coder and the decoder of a background memory that holds a complete (or as complete as possible) representation of the stationary background panorama. Based on these global parameters and on the panorama image stored in the background memory, a prediction of the background layer image to be coded is generated by the coder and by the decoder using efficient texturing algorithms.

Correction images for the background prediction are generated using the SA-DCT algorithm, similar to the hybrid DPCM/DCT standard coding method.

In the extended SA-DCT approach, conventional block-based motion compensation (Pred 1) can be used for object layers that do not satisfy the special model assumptions of the new prediction tools. In any case, intra-coded frames and regions where the model fails (for each of the prediction tools) are efficiently coded using the SA-DCT algorithm.

The MPEG-4 decoder was specified with different complexity levels in order to support different types of applications. These are: Type 0 (not programmable, with a predetermined set of tools), Type 1 (flexible, with a set of configurable tools) and Type 2 (programmable, with the ability to download new tools from the coder).

7.7.5.5 Streams in the MPEG-4 Standard

Just like MPEG-1 and MPEG-2, MPEG-4 also describes streams. Since MPEG-4 divides content into multiple objects, the stream properties affect multiplexing, demultiplexing, and synchronization of multiple streams.

AVO data is bundled into one or more Elementary Streams. These are characterized by the Quality of Service (QoS) needed for transmission (e.g., maximum bit rate or bit error rate) as well as other parameters, for example, stream type information, which helps determine the resources required in the decoder and the precision of time information in the coder. The manner in which such information about stream characteristics is transported from the source to the sink in a synchronized fashion over networks having different QoS is specified in terms of an Access Unit Layer and a conceptual two-layer multiplexer.

The Access Unit Layer allows Access Units to be identified within Elementary Streams, for example, video or audio frames, and scene description commands. This layer also permits the reestablishment of the time base of an AV-object or of a scene description as well as synchronization between them. The header of an Access Unit can be configured in a variety of ways, permitting a wide range of systems.

The FlexMux Layer (Flexible Multiplex) is fully specified by MPEG-4. It comprises a multiplexing tool that allows Elementary Streams (ESs) to be grouped with minimal multiplexing overhead. For example, Elementary Streams with similar QoS requirements could be grouped.

The TransMux Layer (Transport Multiplexing) models the layer that provides the transport services that match the required QoS. MPEG-4 specifies only the interface to this layer. Thus any suitable transport protocol architecture, such as (RTP)/UDP/IP/(AAL5)/ATM or MPEG-2's Transport Stream over a suitable link layer, may become a specific TransMux instance. The choice is left to the end user/service provider, allowing MPEG-4 to be used in a multitude of operating environments.

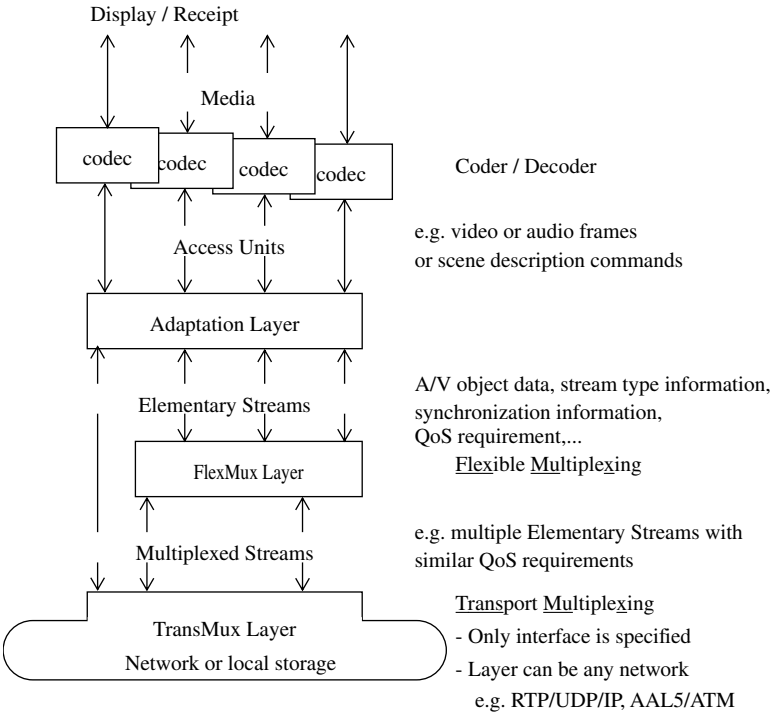


Figure 7-23 MPEG-4 System Layer Model.

Usage of the FlexMux multiplexing tool is optional. As shown in Figure 7-23, this layer can be skipped by the underlying TransMux instance since this provides equivalent functionality. The Access Unit Layer, however, is always present.

With this the following are possible:

1. Identification of Access Units, transport timestamps, clock reference information, and data loss.
2. Optional interleaving of data from different Elementary Streams into FlexMux streams.
3. Control information:

    - to indicate the required QoS for each Elementary Stream and FlexMux stream,
    - to translate such QoS requirements into currently available network resources,
    - to transport the mapping of Elementary Streams associated with AVOs to FlexMux and TransMux channels.

Individual Elementary Streams must be read from incoming data on a network connection or from a storage device. In the MPEG-4 system model, each network connection or file is homogeneously viewed as a TransMux Channel. Multiplexing is carried out partially or completely by layers that are not part of MPEG-4, depending on the application. The Stream Multiplex Interface is used as a reference point for integrating MPEG-4 in system environments. Streams packetized by the Adaptation Layer (AL) are accessed at this interface. The FlexMux layer specifies the optional FlexMux tool. The TransMux interface specifies how either streams packetized by the AL (no FlexMux used) or FlexMux streams are received from the TransMux layer. The TransMux interface thus realizes the passage to transport functionality not defined by MPEG. The interface's data part is considered here, while the control part is dealt with in the context of DMIF.

In the same way that MPEG-1 and MPEG-2 describe the behavior of an ideal operational decoding device together with the syntax and semantics of the bit stream, MPEG-4 defines a System Decoder Model. This allows the precise definition of a terminal's operation without having to make unnecessary assumptions about implementation details. This is essential in order to give developers the freedom to implement MPEG-4 terminals and decoding devices in a multitude of ways. Such devices range from TV receivers, which cannot communicate with the sender, to computers, which are fully capable of exchanging data bidirectionally. Some devices receive MPEG-4 streams over isochronous networks, while others use nonisochronous means (e.g., the Internet) to exchange MPEG-4 information. The System Decoder Model represents a common model that can serve as the basis for all MPEG-4 terminal implementations.

The MPEG-4 demultiplexing step is specified in terms of a conceptual two-layer multiplexer consisting of a TransMux layer and a FlexMux layer as well as the Access Unit Layer, which conveys synchronization information.

The generic term TransMux Layer is used to abstract underlying multiplexing functionality (existing or future) that is suitable for transport of MPEG-4 streams. It should be noted that this layer is not defined in the context of MPEG-4. Examples are the MPEG-2 Transport Stream, H.223, ATM AAL2, and IP/UDP. The TransMux Layer is modelled by means of a protection sublayer and a multiplexing sublayer, which make it clear that this layer is responsible for offering a specific QoS. The functionality of the protection sublayer includes error protection and error detection tools appropriate to the network or storage medium. In some TransMux instances, it may not be possible to separately identify the two sublayers.

Every concrete application scenario uses one or more specific TransMux Instances. Each TransMux Demultiplexer provides access to the TransMux Channels. Requirements for access to the TransMux Channel at the data interface are the same for all TransMux Instances. They include reliable error detection, the delivery or erroneous data with a suitable error indication (if possible), and the division of the payload into

frames. This division consists of either streams that have been packetized by the AL or of FlexMux streams. The requirements are summarized in an informal fashion in the TransMux interface, in the system part of the MPEG-4 standard.

On the other hand, MPEG fully describes the FlexMux Layer. It provides a flexible tool for optional interleaving of data with minimal overhead and low delay and is particularly useful if the underlying TransMux Instance has a large packet size or high overhead. The FlexMux itself is not robust against errors. It can either be used on top of TransMux Channels with high QoS or to bundle Elementary Streams, both of which are equivalent in terms of error tolerance. The FlexMux requires reliable error detection and sufficient division of the FlexMux packets into frames (for random access and error correction), which must be provided by the underlying layer. These requirements are summarized in the Stream Multiplex Interface, which defines data access to individual transport channels. The FlexMux demultiplexer receives AL-packetized streams from FlexMux Streams.

The Access Unit Layer has a minimal set of tools for checking consistency, padding headers, conveying time base information, and transporting time-stamped access units of an Elementary Stream. Each packet consists of an access unit or fragment thereof. These time-stamped units represent the only semantic structure of Elementary Streams that is visible in this layer. The AU Layer requires reliable error detection and framing of individual packets of the underlying layer, which can for example be performed by the FlexMux. The manner in which the compression layer accesses data is summarized in the informal Elementary Stream interface, also in the system part of the MPEG-4 standard. The AU layer retrieves Elementary Streams from the streams packetized by the AL.

Depending on the degree of freedom allowed by the author of a scene, the user can interact with the scene's contents. The user could, for example, navigate through the scene, move objects to different positions, or trigger a sequence of events by clicking on a specific object with the mouse (for example, starting or stopping a video stream or choosing the desired language if multiple language channels are available). It would also incorporate more complex behaviors. For example, a virtual telephone rings and the user answers it, establishing a communications link.

Streams that come from the network (or from a storage device) as TransMux Streams are demultiplexed into FlexMux Streams and passed on to appropriate FlexMux demultiplexers, which receive Elementary Streams (see Figure 7-24).

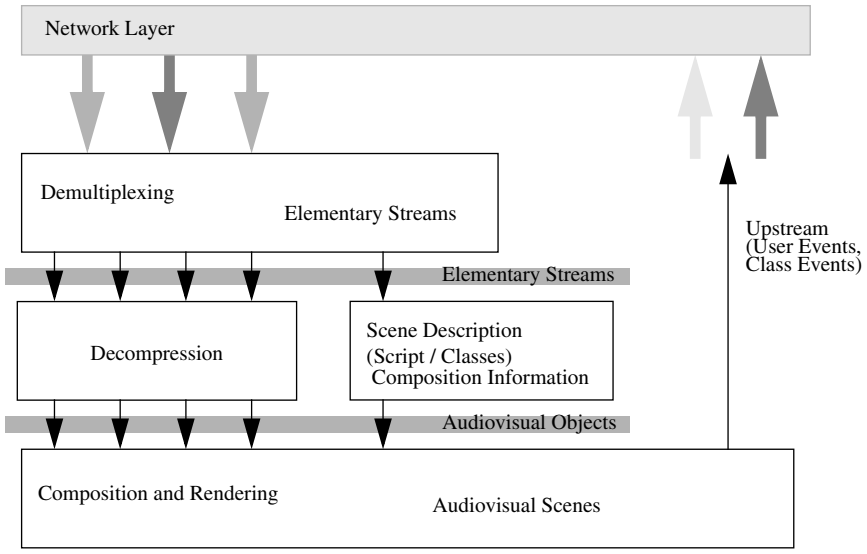


Figure 7-24 Important components of an MPEG-4 terminal.

Parts of the control functionality are only available in conjunction with a transport control unit, such as the DMIF environment, which MPEG-4 defines for precisely this purpose. The Delivery Multimedia Integration Framework (DMIF) covers operations of multimedia applications over interactive networks, in broadcast environments or from hard disks. The DMIF architecture is structured such that applications that use DMIF for communication do not need to have any knowledge of the underlying communications methods. The DMIF implementation takes care of the network details, presenting the application with a simple interface. DMIF can be placed between an MPEG-4 application and the transport network (see Figure 7-25).

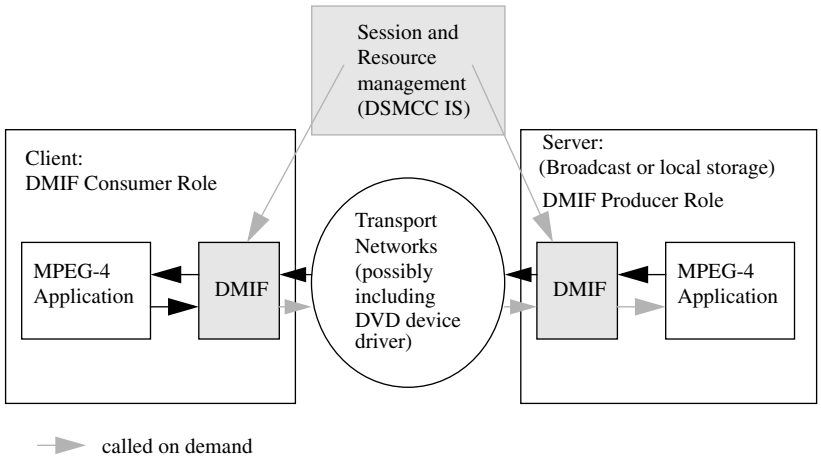


Figure 7-25 DMIF architecture.

To predict the behavior of a decoder during decompression of the various elementary data streams that make up an MPEG-4 session, the System Decoder Model makes it possible for the coder to specify and monitor the minimum buffer resources needed to decode the session. The required resources are conveyed to the decoder with object descriptors during establishment of the MPEG-4 session so that the decoder can decide if it is capable of proceeding with the session.

By managing the finite amount of buffer space, for example, the model allows the sender to transmit nonreal-time data too early, if sufficient space is available to store it at the receiver. The prestored data can be accessed at the appropriate moment, at which time real-time information can use a greater portion of the channel capacity, if required.

Real-time operation assumes a timing model where the end-to-end delay from the signal output of the coder to the signal input of the decoder is constant. Furthermore, the transmitted data streams must carry either implicit or explicit time information. There are types of such information. The first is used to convey the clock speed, or the encoder time base, to the decoder. The second, which consists of time stamps attached to parts of the coded AV data, contains desired decoding times for access units or the composition and expiration time for composition units. This information is conveyed in the AL-PDU headers generated by the Access Unit Layer. This timing information can be used to adapt the interval between pictures and the audio sampling rate at the decoder to match the encoder's values in order to achieve synchronized operation.

Different AV objects can be encoded by encoders with different time bases together with the slightly different speeds that occur. At any time, it is possible to map these time bases to the receiving terminal's time base. However, in this case no real

implementation of a receiving terminal can avoid the occasional repetition or loss of AV data caused by temporal aliasing (relative reduction or extension of the time scale).

Although system operation without any timing information is allowed, it is not possible to define a buffer model in this case.

7.7.6 MPEG-7

After MPEG-3 did not materialize, it was decided not to allow MPEG-5 or MPEG-6 either, in order to preserve the logic of the sequence MPEG-1 (+1) to MPEG-2 (+2) to MPEG-4 (+3) to MPEG-7 (+4) ...

MPEG-7 is not meant as another compression format, rather the intention of the MPEG-7 ISO Working Group was to establish a metadata standard to supplement content coded in other formats (such as MPEG-4). The use of metadata aims at a new type of extension to the coding of multimedia data: mainly improved searching techniques and display strategies, but also consistency checking and, for example, scaling that incorporates consistency and priority. MPEG-7 calls for the integration and extension of other media content formats (especially MPEG-4) in this way.

In addition to the coding of metadata, MPEG-7 will define interfaces (and only the interfaces) for working in tandem with tools for automatic content analysis and search engines, but not these services themselves.

7.8 Fractal Compression

Fractal image and video compression represents an entirely different coding method. In this technique, no actual pixel information is transmitted, rather only a transformation function that contains an image similar to the target image as a fixed point. The decoding process then consists of an iterative application of this function to any beginning image.

Various specific attributes of this coding technique follow from this. First of all, the decoding process is progressive and the decoding efficiency is scalable, since the quality of the decoded image increases with each iteration step. Additionally, the process is independent of the resolution. The mapping function can be applied repeatedly in order to get more detail. The third attribute is an asymmetry between the encoding and decoding processes.

This raises the question of how an image can be coded as a transformation function. The algorithm makes use of a fractal attribute of images: their self-similarity. Images consist of regions that are similar to one another. A transformation function then consists of a mapping of each region of an image to the most similar part of the image. The mapping consists of contracting, stretching, rotating, and skewing the form of the image regions and adjusting their contrast and brightness. This is a type of vector quantization that does not use a fixed set of quantization vectors. The mapping will result in an image if the existence of a fixed point can be guaranteed. The only

condition required by the theory for this is that of contractivity. The absolute value of the contraction factor must lie in the interval $[0, 1)$.

The mapping is achieved in current implementations [BH93] by dividing the original image into blocks (of 8×8 pixels) and, for each block, finding the most similar block of image regions (i.e., overlapping 16×16 pixel blocks). The set of blocks compared with each (8×8) original block is increased by the possible geometric transformations. The most similar block can be found by minimizing a distance measure. Usually the sum of the quadratic pixel differences is minimized.

For natural images, fractal compression can achieve high compression ratios (up to 1000:1) with very good image quality. The biggest disadvantage of this coding method is the complexity of calculation and its low efficacy when applied to graphics images. In order to keep the complexity within practicable limits, only a subset of all transformations are considered, for example only rotation angles of 0^o, 90^o, 180^o, and 270^o. Nevertheless each original block must be compared with a very large number of blocks in order to find the mapping to the most similar block. In addition to the complexity of calculation, this coding technique is lossy since it uses only the similarity of blocks, not their identity.

7.9 Conclusions

The important compression techniques used in multimedia systems all represent a combination of many well known algorithms:

7.9.0.1 JPEG

JPEG is the standard for still image coding that will have the most significance in the future. Its far-reaching definition allows a large number of degrees of freedom. For example, an image can have up to 255 components, that is, levels. An image can consist of up to 65,535 lines, each of which can contain up to 65,535 pixels. Compression performance is measured in bits per pixel. This is an average value calculated as the quotient of the total number of bits contained in the coded picture and the number of pixels contained in the picture. That said, the following statements can be made for DCT-coded still images [Wal91]:

- 0.25 to 0.50 bit/pixel: moderate to good quality; sufficient for some applications.
- 0.50 to 0.75 bit/pixel: good to very good quality; sufficient for many applications.
- 0.75 to 1.50 bit/pixel: excellent quality; suitable for most applications.
- 1.50 to 2.00 bit/pixel: often barely distinguishable from the original; sufficient for almost all applications, even those with the highest quality requirements.

In lossless mode, a compression ratio of 2:1 is achieved despite the remarkable simplicity of the technique. Today JPEG is commercially available in software as well as in hardware and is often used in multimedia applications that require high quality.

The primary goal of JPEG is compression of still images. However, in the form of Motion JPEG, JPEG can also be used for video compression in applications such as medical imaging.

7.9.0.2 H.261 and H.263

H.261 and H.263 are already established standards. These were mainly supported by associations of telephone and wide area network operators. Due to the very restricted resolution of the QCIF format and reduced frame rates, implementing H.261 and H.263 encoders and decoders does not cause any significant technical problems today. This is especially true if motion compensation and the optical low-pass filter are not components of the implementation, although the quality is not always satisfactory in this case. If the image is encoded in CIF format at 25 frames/s using motion compensation, the quality is acceptable. H.263 is mostly used for dialogue mode applications in network environments, for example video telephony and conferencing. The resulting continuous bit rate is eminently suitable for today's wide area networks operating with ISDN, leased lines, or even GSM connections.

7.9.0.3 MPEG

MPEG is the most promising standard for future audio and video compression use. Although the JPEG group has a system that can also be used for video, it is overly focussed on the animation of still images, instead of using the properties of motion pictures. The quality of MPEG video (without sound) at about 1.2 Mbit/s, appropriate for CD-ROM drives, is comparable to that of VHS recordings [Le 91]. The compression algorithm works very well at a resolution of about 360×240 pixels. Obviously, higher resolutions can also be decoded. However, at a resolution of, for example 625 lines, quality is sacrificed. The future of MPEG points towards MPEG-2, which defines a data stream compatible with MPEG-1, but provides data rates up to 100 Mbit/s. This significantly improves the currently available quality of MPEG-coded data.

MPEG also defines an audio stream with various sampling rates, ranging up to DAT quality, at 16 bit/sample. Another important part of the MPEG group's work is the definition of a data stream syntax.

Further, MPEG was optimized by making use of the retrieval model for application areas such as tutoring systems based on CD-ROMs and interactive TV. Embedding this optimization in MPEG-2 will allow TV and HDTV quality at the expense of a higher data rate. MPEG-4 will provide high compression ratios for video and associated audio and is furthermore an appropriate tool for creating whole classes of new multimedia applications. However, currently the complex and still very new MPEG-4 standard is little used in widespread commercial applications.

JPEG, H.263, MPEG, and other techniques should not be viewed as competing alternatives for data compression. Their goals are different and partly complementary. Most of the algorithms are very similar, but not identical. Technical quality, as well as

market availability, will determine which of these techniques will be used in future multimedia systems. This will lead to cooperation and convergence of the techniques. For example, a future multimedia computer could generate still images using JPEG, use H.263 or MPEG-4 for video conferencing, and need MPEG-2 to retrieve stored multimedia information. This is, however, a purely hypothetical conception and is not in any way meant to prejudge future development or strategies for these systems.