Chapter 5

Video and Animation

Motion video and computer-based animation have become basic media for multimedia systems. In this chapter we present concepts and developments in these areas to develop understanding of the *motion video* and *animation* media. We describe current (analog) and upcoming (digital) video, and graphics animation with respect to the properties of human perception.

5.1 Basic Concepts

The human eye views pictures and motion pictures. The immanent properties of the eye determine, in connection with neuronal processing, some essential conditions related to video systems.

5.1.1 Video Signal Representation

In conventional black-and-white TV sets, the video signal is displayed using a CRT (Cathode Ray Tube). An electron beam carries corresponding pattern information, such as intensity in a viewed scene.

To understand later reasoning behind data rates of motion video and computer-based animation, we focus on the description of their respective signals rather than specific camera or monitor technologies. We analyze the video signal coming from a camera and the resulting pictures (using USA standards) [BF91].

Video signal representation includes three aspects: the *visual representation*, *transmission* and *digitalization*.

Visual Representation

A central objective is to offer the viewer a sense of presence in the scene and of participation in the events portrayed. To meet this objective, the televised image should convey spatial and temporal content of the scene. Important measures are:

1. *Vertical Detail and Viewing Distance*

   The geometry of the field occupied by the television image is based on the ratio of the picture width *W* to height *H*. It is called *aspect ratio*. The conventional aspect ratio is *4/3=1.33*. Figure 5.1 shows an example of aspect ratio.

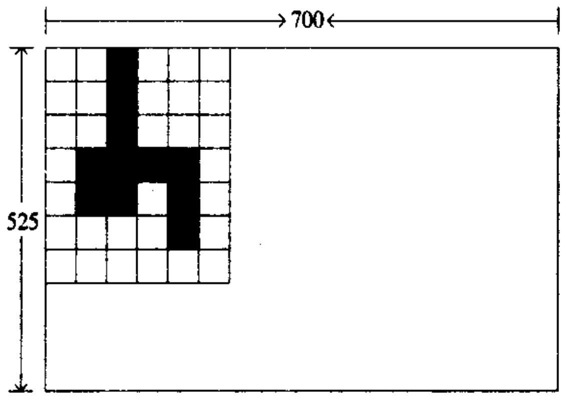

Figure 5.1: *Decomposition of a picture using an aspect ratio of 4:3 (NTSC standard).*

   The *viewing distance* *D* determines the angle *h* subtended by the picture height. This angle is usually measured by the ratio of the viewing distance to the picture height (*D/H*).

The smallest detail that can be reproduced in the image is a *pixel*. Ideally, each detail of the scene would be reproduced by one pixel. Practically, however, some of the details in the scene inevitably fall between scanning lines, so that two lines are required for such picture elements. Thus, some vertical resolution is lost. Measurements of this effect show that only about 70% of the vertical detail is presented by the scanning lines. The ratio is known as the Kell factor; it applies irrespective of the manner of scanning, whether the lines follow each other sequentially (a progressive scan) or alternately (an interlaced scan).

2. *Horizontal Detail and Picture Width*

   The picture width chosen for conventional television service is $4/3 \times$*picture height*. Using the aspect ratio, we can determine the horizontal field of view from the horizontal angle. Table 5.1 lists the resolutions and fields of view for different systems.

3. *Total Detail Content of the Image*

   The vertical resolution is equal to the number of picture elements separately presented in the picture height, while the number of elements in the picture width is equal to the horizontal resolution times the aspect ratio. The product of the number of elements vertically and horizontally equals the total number of picture elements in the image.

4. *Perception of Depth*

   In natural vision, perception of the third spatial dimension, depth, depends primarily on the angular separation of the images received by the two eyes of the viewer. In the flat image of television, a considerable degree of depth perception is inferred from the perspective appearance of the subject matter. Further, the choice of the focal length of lenses and changes in depth of focus in a camera influence the depth perception.

5. *Luminance and Chrominance*

   Color vision is achieved through three signals, proportional to the relative intensities of Red, Green and Blue light (*RGB*) in each portion of the scene. The three signals are conveyed separately to the input terminals of the picture tube, so that the tube reproduces at each point the relative intensities of red, green and blue discerned by the camera. During the transmission of the signals from the camera to the receiver (display), a different division of signals in comparison to the RGB division is often used. The color encoding during transmission uses *luminance* and two *chrominance* signals. We will detail these signals later in this section.

6. *Temporal Aspects of Illumination*

   Another property of human vision is the boundary of *motion resolution*. In contrast to continuous pressure waves of an acoustic signal, a discrete sequence of individual pictures can be perceived as a continuous sequence. This property is used in television and motion pictures, i.e., motion is the presentation of a rapid succession of slightly different still pictures (frames). Between frames, the light is cut off briefly. To represent visual reality, two conditions must be met. First, the rate of repetition of the images must be high enough to guarantee smooth motion from frame to frame. Second, the rate must be high enough so that the persistence of vision extends over the interval between flashes.

7. *Continuity of Motion*

   It is known that we perceive a continuous motion to happen at any frame rate faster than 15 frames per second. Video motion seems smooth and is achieved at only 30 frames per second, when filmed by a camera and not synthetically generated. Movies, however, at 24 frames/s, often have a jerkiness about them, especially when large objects are moving fast and close to the viewer, as sometimes happens in a panning scene. The new *Showscan* technology [Dep89] involves making and showing movies at 60 frames per second and on 70-millimeter films. This scheme produces a bigger picture, which therefore occupies a larger portion of the visual field, and produces much smoother motion.

   There are several standards for motion video signals which determine the frame rate to achieve proper continuity of motion. The USA standard for motion video signals, NTSC (National Television Systems Committee) standard, specified the frame rate initially to 30 frames/s, but later changed it to 29.97 Hz to maintain the visual-aural carrier separation at precisely 4.5 MHz. NTSC scanning equipment presents images at the 24 Hz standard, but transposes them to the 29.97 Hz scanning rate. The European standard for motion video, PAL (Phase Alternating Line), adopted the repetition rate of 25 Hz, and the frame rate therefore is 25 frames/s.

8. *Flicker*

   Through a slow motion, a periodic fluctuation of brightness perception, a *flicker effect*, arises. The marginal value to avoid flicker is at least 50 refresh cycles/s. To achieve continuous flicker-free motion, we need a relatively high refresh frequency. Movies, as well as television, apply some technical measures to work with lower motion frequencies.

   For example, to run a movie with 16 pictures per second without any technical measures taken would be very disturbing. To reduce the flicker effect, the light wave is interrupted additionally two times during the picture projection, so that additionally to the original picture projection, the picture can be redrawn twice during the interruptions; thereby, a picture refresh rate of $3 \times 16\ Hz = 48\ Hz$ is achieved.

   In the case of television, flicker effect can be alleviated through a ***display refresh buffer***. The data are written in the refresh buffer at a higher frequency than the motion resolution requires (e.g., 25 Hz). The picture is displayed at a frequency so that the flicker effect is removed (e.g., 70 Hz). For example, the 70-Hz-motion frequency corresponds to the motion frequency of a good computer display. A full TV picture is divided into two half-pictures which consist of interleaved scanning lines. Each half-picture after another is transmitted, using the line-interleaving method. In the case of a full TV picture, where the transmission occurs at 30 Hz (actually 29.97 Hz), or 25 Hz in Europe, the half-pictures must be scanned at higher frequency of $2 \times 30\ Hz = 60\ Hz$, or $2 \times 25\ Hz = 50\ Hz$, to achieve the scanning rate of 30 Hz, respectively 25 Hz, for the full pictures. Figure 5.2 shows the situation described above.

9. *Temporal Aspect of Video Bandwidth*

   An important factor to determine which video bandwidth to use to transmit motion video is its temporal specification. Temporal specification depends on the rate of the visual system to scan pixels, as well as on the human eye's scanning capabilities. For example, in a regular TV device, the time consumed in scanning lines and frames is measured in microseconds. In an HDTV (High Definition TV) device, however, a pixel can be scanned in less than a tenth of a millionth of a second. From the human visual perspective, the eye requires that a video frame be scanned every 1/25 second. This time is equivalent to the time during which a human eye does not see the flicker effect.

The rates of scanning pixels are stated in video frequencies, but there is no one-on-one mapping. At best, during one cycle of video frequency, two horizontally adjacent pixels can be represented, so the scanning rate of, for example, 22.5 million pixels per second (HDTV scanning rate) requires video frequency up to 11.25 MHz.

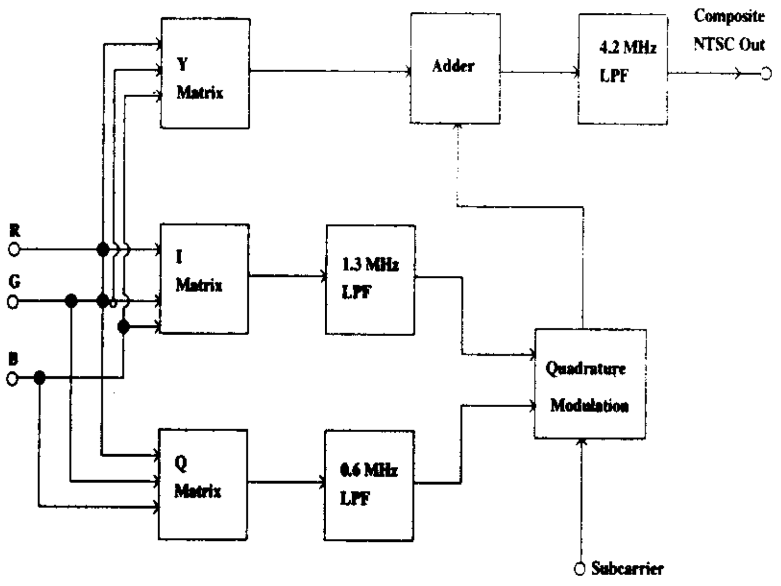

Figure 5.2: *Flicker-effect: Eye irritation by the motion frequency of 30 Hz and 60 Hz.*

Transmission

Video signals are transmitted to receivers through a single television channel. The NTSC channel is shown in Figure 5.3.

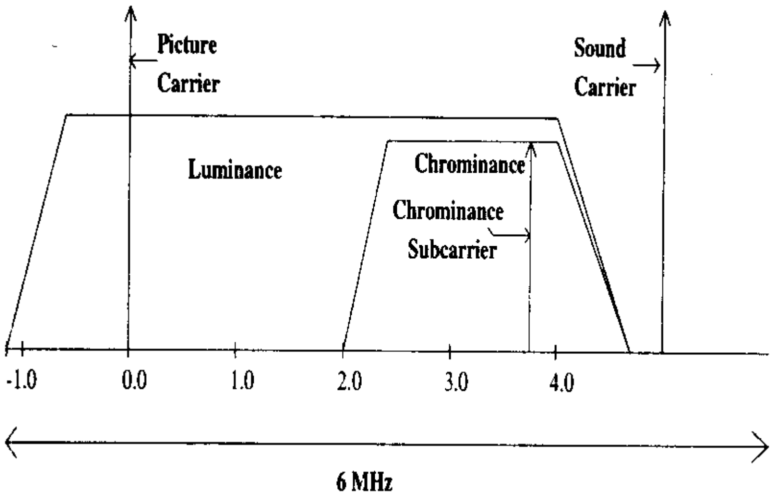

Figure 5.3: *Bandwidth of the NTSC system.*

To encode color, a video signal is a composite of three signals. For transmission purposes, a video signal consists of one luminance and two chrominance signals. In NTSC and PAL systems, the composite transmission of luminance and chrominance signals in a single channel is achieved by specifying the chrominance subcarrier to be an odd multiple of one-half of the line-scanning frequency. This causes the component frequencies of chrominance to be interleaved with those of luminance. The goal is to separate the two sets of components in the receiver and avoid interference between them prior to the recovery of the primary color signals for display. In practice, degradation in the image, known as *cross-color* and *cross-luminance*, occurs. These effects have pushed manufacturers of NTSC receivers to limit the luminance bandwidth to less than 3 MHz below the 3.58 MHz subcarrier frequency and far short of the 4.2 MHz maximum of the broadcast signal. This causes the horizontal resolution in such receivers to be confined to about 25 lines. The filtering employed to remove chrominance from luminance is a simple *notch filter* tuned to the subcarrier's frequency; currently it is the *comb filter*. The transmitter also uses the comb filter during the luminance-chrominance encoding process.

Several approaches of color encoding are summarized below:

- *RGB signal*

In the case of separate signal coding the color can be encoded in the *RGB signal*, which consists of separate signals for red, green, and blue colors. Other colors can be coded as a combination of these primary colors. For example, if the values R (red), G (green) and B (blue) are normalized with $R+G+B=1$, we get the neutral white color.

- *YUV signal*

  As human perception is more sensitive to brightness than any chrominance information, a more suitable coding distinguishes between luminance and chrominance. This means instead of separating colors, one can separate the brightness information (luminance Y) from the color information (two chrominance channels U and V). The luminance component must always be transmitted because of compatibility reasons. For black-and-white reception, the utilization of chrominance components depends on the color capability of the TV device.

  The component division for *YUV* signal is :

$$Y = 0.30\ R + 0.59\ G + 0.11\ B$$
$$U = (B\text{-}Y) \times 0.493$$
$$V = (R\text{-}Y) \times 0.877$$

  Any error in the resolution of the luminance (Y) is more important than in the chrominance (U, V) values. Therefore, the luminance values can be coded using higher bandwidth than the chrominance values.

  Because of these different component bandwidths, the coding is often characterized by a ratio between the luminance component and the two chrominance components. For example, the YUV encoding can be specified as (4:2:2) signal. Further, the YUV encoding is sometimes specified as the Y, B-Y, R-Y signal because of the dependencies between U and B-Y and V and R-Y in the above equations.

  The CD-I (Compact Disc-Interactive) and DVI video on demand CD developments adopted the *YUV* signal decomposition.

- *YIQ signal*

  A coding similar to the YUV signal described above is the *YIQ* signal, which builds the basis for the NTSC format.

$$Y = 0.30R + 0.59G + 0.11B$$
$$I = 0.60R - 0.28G - 0.32B$$
$$Q = 0.21R - 0.52G + 0.31B$$

  A typical NTSC encoder is shown in Figure 5.4. It produces the *I* and *Q* signals, limits their passbands, uses them to modulate the subcarrier in a quadrature and adds the moduled subcarrier to the luminance *Y*, blanking and synchronizing signal waveform.

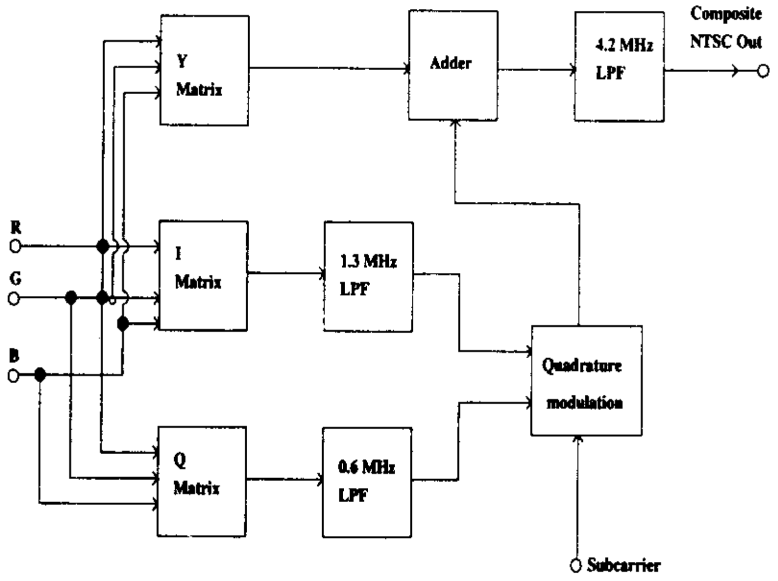

Figure 5.4: *YIQ encoding operations of an NTSC system.*

- *Composite signal*

  The alternative to component coding composes all information into one signal; consequently, the individual components (*RGB, YUV* or *YIQ*) must be combined into one signal. The basic information consists of luminance information and chrominance difference signals. During the composition into one signal, the chrominance signals can interfere with the luminance. Therefore, the television technique has to adopt appropriate modulation methods to eliminate the interference between luminance and chrominance signals.

The basic video bandwidth required to transmit luminance and chrominance signals is 4.2 MHz for the NTSC standard. In HDTV, the basic video bandwidth is at least twice of the conventional standard (e.g., NTSC). Moreover, in the separate component method of transmission, the bandwidths occupied by the three signals (*RGB* or *YUV*) are additive, so the total bandwidth prior to signal processing is of the order of 20-30 MHz.

Digitalization

Before a picture or motion video can be processed by a computer or transmitted over a computer network, it needs to be converted from analog to digital representation. In an ordinary sense, digitalization consists of *sampling the gray (color) level* in the picture at $M \times N$ array of points. Since the gray level at these points may take any value in a continuous range, for digital processing, the gray level must be *quantized.* By this we mean that we divide the range of gray levels into K intervals, and require the gray level at any point to take on only one of these values. For a picture reconstructed from quantized samples to be acceptable, it may be necessary to use 100 or more quantizing levels. When samples are obtained by using an array of points or finite strings, a fine degree of quantization is very important for samples taken in regions of a picture where the gray (color) levels change slowly [RK82].

The result of sampling and quantizing is a digital image (picture), at which point we have obtained a rectangular array of integer values representing pixels. Digital images were described in more detail in Section 4.1.

The next step in the creation of digital motion video is to digitize pictures in time and get a sequence of digital images per second that approximates analog motion video.

5.1.2 Computer Video Format

The computer video format depends on the input and output devices for the motion video medium.

Current video digitizers differ in digital image (*frame*) resolution, quantization and frame rate (frames/s). For example, as described in Section 4.1.2., IRIS's video board *VINO* takes NTSC video signal and after digitalization can achieve spatial resolution of 640 x 480 pixels, quantization of 8 bits/pixel (256 shades of gray) and a frame rate of 4 frames/second. The *SunVideo* digitizer from Sun Microsystems, on the other hand, captures NTSC video signal in the form of the RGB signal with frame resolution of 320 x 240 pixels, quantization of 8 bits/pixel, and a frame rate of 30 frames/second.

The output of the digitalized motion video depends on the display device. The most often used displays are raster displays, described in the previous chapter. A common raster display system architecture is shown in Figure 5.5.

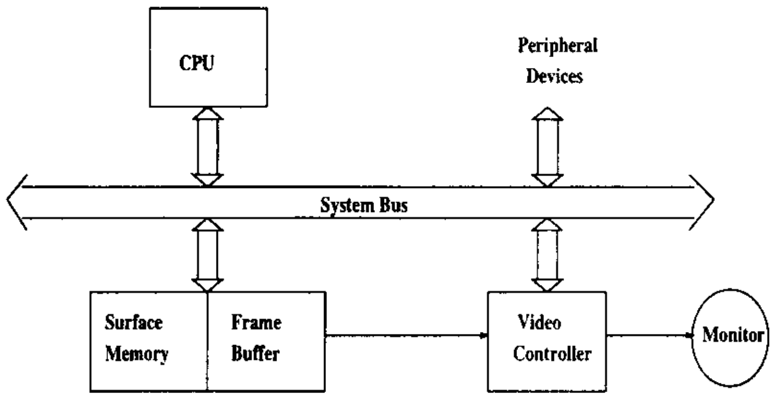

Figure 5.5: *A common raster display system architecture.*

The *video controller* displays the image stored in the frame buffer, accessing the memory through a separate access port as often as the raster scan rate dictates. The constant refresh of the display is its most important task. Because of the disturbing flicker effect, the video controller cycles through the frame buffer, one scan line at a time, typically 60 times/second. For presentation of different colors on the screen, the system works with a *Color Look Up Table (CLUT or lut)*. At a certain time, a limited number of colors (n) is prepared for the whole picture. The set of n colors, used mostly, is chosen from a color space, consisting of m colors, where generally $n << m$.

Some computer video controller standards are given here as examples. Each of these systems supports different resolution and color presentation.

- The *Color Graphics Adapter (CGA)* has a resolution of 320 x 200 pixels with simultaneous presentation of four colors. Therefore, the storage capacity per image is:

$$320 \times 200 \text{ pixels} \times \frac{2bits/pixel}{8bit/byte} = 16{,}000 \text{ bytes}$$

- The *Enhanced Graphics Adapter (EGA)* supports a resolution of 640 x 350 pixels with 16-color presentation. Therefore, the storage capacity per image is:

$$640 \times 350 \text{ pixels} \times \frac{4bits/pixel}{8bits/byte} = 112{,}000 \text{ bytes}$$

- The *Video Graphics Array (VGA)* works mostly with a resolution of 640 x 480 pixels. In this case, 256 colors can be displayed simultaneously. The monitor is controlled through an RGB output. The storage capacity per image is:

$$640 \times 480 \text{ pixels} \times \frac{8bits/pixel}{8bits/byte} = 307{,}200 \text{ bytes}$$

- The *8514/A Display Adapter Mode* can present 256 colors with a resolution of 1024 x 768 pixels. The storage capacity per image is:

$$1024 \times 768 \text{ pixels} \times \frac{8bits/pixel}{8bits/byte} = 786{,}432 \text{ bytes}$$

- The *Extended Graphics Array (XGA)* supports a resolution of 640 x 480 pixels and 65,000 different colors. With the resolution of 1024 x 768 pixels, 256 colors can be presented. In this case, we have the same storage capacity per image as the 8514/A adapter.

- The *Super VGA (SVGA)* offers resolutions up to 1024 × 768 pixels and color formats up to 24 bits per pixel. The storage capacity per image is:

  $$1024 \times 768 \text{ pixels} \times \frac{24bits/pixel}{8bits/byte} = 2{,}359{,}296$$

  Low-cost SVGA video adaptors are available with *video accelerator chips* that pretty much overcome the speed penalty in using a higher resolution and/or a greater number of colors [Lut94]. The role of video accelerator boards is to play back video that would originally appear in a 160 × 120 window at full screen. Hence, video accelerators improve the playback speed and quality of captured digital video sequences [Ann94c].

5.2 Television

Television is the most important application that has driven the development of motion video. Since 1953, television has gone through many changes. In this section we present an overview of the many changes that have occurred, from conventional systems used in black-and-white and color television, to enhanced television systems, which are an intermediate solution, to digital interactive television systems and their coming standards. A summary of the most important characteristics of all television systems is presented in Tables 5.1 and 5.2.

5.2.1 Conventional Systems

Black-and-white television and current color television is based on the representation range described in Section 5.1.1. Early on, different video format standards were established in different parts of the world. Conventional television systems employ the following standards:

- *NTSC*

  NTSC *National Television Systems Committee*, developed in the U.S., is the oldest and most widely used television standard. The color carrier is used with approximately 4.429 MHz or with approximately 3.57 MHz. NTSC uses

System	Total Lines	Active Lines	Vertical Resolution	Optimal Viewing Distance (m)	Aspect Ratio	Horizontal Resolution	Total Picture Elements
HDTV-p USA	1050	960	675	2.5	16/9	600	720,000
HDTV-p Europe	1250	1000	700	2.4	16/9	700	870,000
HDTV-p NHK	1125	1080	540	3.3	16/9	600	575,000
NTSC-i	525	484	242	7.0	4/3	330	106,000
NTSC-p	625	484	340	5.0	4/3	330	149,000
PAL-i	625	575	290	6.0	4/3	425	165,000
PAL-p	625	575	400	4.3	4/3	425	233,000
SECAM-i	625	575	290	6.0	4/3	465	180,000
SECAM-p	625	575	400	4.3	4/3	465	248,000

Table 5.1: *Spatial characteristics of television systems [BF91].*

a quadrature amplitude modulation with a suppressed color carrier and works with a motion frequency of approximately 30 Hz. A picture consists of 525 lines.

For NTSC television, 4.2 MHz can be used for luminance and 1.5 MHz for each of the two chrominance channels. Home television and VCRs employ only 0.5 MHz for chrominance channels.

- *SECAM*

  SECAM (*SEquential Couleur Avec Memoire*) is a standard used in France and Eastern Europe. In contrast to NTSC and PAL, it is based on frequency modulation. It uses a motion frequency of 25 Hz, and each picture has 625 lines.

- *PAL*

  PAL(*Phase Alternating Line*) was invented by W. Bruch (Telefunken) in 1963. It is used in parts of Western Europe. The basic principle of PAL is a quadrature amplitude modulation similar to NTSC, but the color carrier is not sup-

System	Total Channel Width (MHz)	Video Basebands (MHz)			Scanning Rates (Hz)		
		Y	R-Y	B-Y	Camera	HDTV Display	Conventional Display
HDTV USA	9.0	10.0	5.0	5.0	59.94-p	59.94-p	59.94-i
HDTV Europe	12.0	14.0	7.0	7.0	50-p	100-p	50-i
HDTV Japan	30.0	20.0	7.0	3.0	60-i	60-i	NA
NTSC	6.0	4.2	1.0	0.6	59.94-i	NA	59.94-i
PAL	8.0	5.5	1.8	1.8	50-i	NA	50-i
SECAM	8.0	6.0	2.0	2.0	50-i	NA	50-i

Table 5.2: *Temporal characteristics of television systems [BF91] (NA - Not Available, i-interlaced, p-progressive [non-interlaced]).*

pressed. The color carrier is computed as follows: first, the color carrier is multiplied directly by the color difference of the signal U; the color carrier is shifted at 90 degrees, and then multiplied by the color difference of the signal V; both results are added together (these three steps represent the regular quadrature amplitude modulation); and one phase of the modulated V signal is added to each second line for the purpose of phase errors reduction.

5.2.2 Enhanced Definition Systems

Enhanced Definition Television Systems (EDTV) are conventional systems modified to offer improved vertical and/or horizontal resolution.

Comb filters are one means for improving horizontal resolution. Used in NTSC broadcasts, they essentially make full use of the 4.2 MHz bandwidth of luminance, producing better than 30% improvement in horizontal resolution. Another improvement in horizontal resolution has been offered in VCRs [BF91].

Vertical resolution improvements were achieved by introducing progressive scanning into receivers designed for NTSC, PAL and SECAM services. For example, in the progressively scanned mode intended for the NTSC service, there are 525 lines in each field versus the 265.5 lines per field as broadcast. An extra "blank" line is presented in the display between each pair of "active" lines. The blank line may be filled in by interpolation of the video information from the active lines above and below it, plus that from the corresponding line of the previous field.

We briefly describe intermediate television systems emerging in the USA and Europe, which offer improvements on their conventional system components (NTSC and PAL).

- *IDTV*

  In the U.S., the intermediate step between NTSC television and HDTV (see Section 5.2.3) is known as IDTV (*Improved-Definition Television*). IDTV is not a new television standard but an attempt to improve NTSC image by using digital memory to double the scanning lines from 525 to 1,050 (the same number as two proposed digital HDTV formats). The pictures are only slightly more detailed than NTSC images because the signal does not contain any new information. Vertical resolution is enhanced, because 1,050-line IDTV images are displayed at once, in 1/60 of a second. Normally, 525-line TV pictures are presented in two interlaced half pictures, each 1/60 second apart. The double lines on IDTV sets, however, are small enough to disappear, at least to the naked eye. In addition, line-doubling helps to eliminate the interline flicker caused by standard interlaced scanning. By separating the chrominance and luminance parts of the video signal, and thus preventing the former from leaking into the latter, IDTV zaps the crawling dots. The improved digital separation of signals also helps cross-color interference.

  An example of improved and advanced television systems is the ACTV-I (Advanced Compatible Television, First System). The channel occupancy of the ACTV-I system is shown in Figure 5.6.

- *D2-MAC*

  D2-MAC (*Duobinary Multiplexed Analogue Components*) is envisioned as the intermediate level between current television and European HDTV. This intermediate solution is already introduced, for example, in Germany as a successor of the PAL standard. D2-MAC uses a time-multiplexing mechanism for component transmission. Figure 5.7 shows the time split for one line of a motion

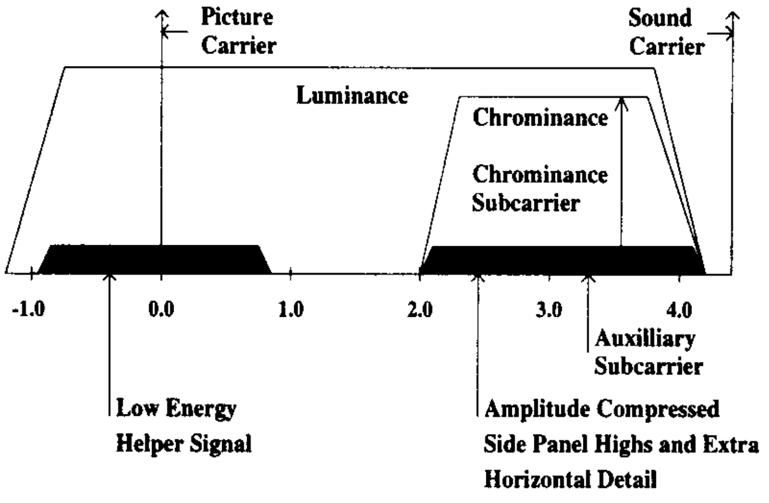

Figure 5.6: *Bandwidth of ACTV-I systems.*

picture. It means that the 64 μs time interval is divided, so that 34.4 μs are

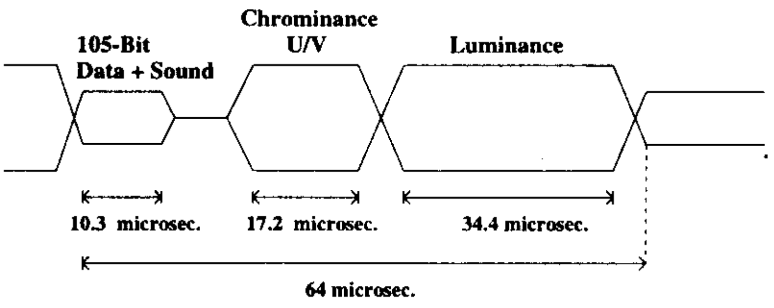

Figure 5.7: *D2-MAC – time multiplexing.*

used for the luminance signal, 17.2 μs for chrominance signals and 10.3 μs for voice and data.

Further characteristics of the D2-MAC system are: 625 lines are transmitted, but only 574 lines are visible. The width to height ratios of 4:3 and 16:9 are supported. The audio and further data are transmitted in a duobinary coding (D2 using together approximately 105 bits/64 μs = 1.64 Mbits/s). Audio channels can be coded as two high-quality stereo signal channels, or up to eight channels of lower audio quality.

5.2.3 High-Definition Systems

The next generation of TV is known as HDTV (*High-Definition Television*). HDTV is, in principle, defined by the image it presents to the viewer:

- *Resolution*

  The HDTV image has approximately twice as many horizontal and vertical pixels as conventional systems. The increased vertical definition is achieved by employing more than 1000 lines in the scanning patterns. The increased luminance detail in the image is achieved by employing a video bandwidth approximately five times that used in conventional systems. Additional bandwidth is used to transmit the color values separately, so that the total bandwidth is from five to eight times that used in existing color television services.

- *Aspect Ratio*

  The aspect ratio of the proposed HDTV images is *16/9=1.777.*

- *Viewing Distance*

  Since the eye's ability to distinguish details is limited, the more detailed HDTV image should be viewed closer than is customary with conventional systems.

Digital codings are essential in the design and implementation of HDTV. Digital television is still existent in research labs as prototypes and sophisticated professional studios, although an international digital television standard was already specified by the CCIR (Consultative Committee International Radio) in 1982. This standard describes specific resolution sampling and coding. There are two kinds of possible digital codings: *composite coding* and *component coding.*

Composite Coding

The simplest possibility for digitizing video signal is to sample the composite analog video signal. Here, all signal components are converted together into a digital representation. Composite coding of the whole video signal is in principle easier than a digitalization of separate signal components (luminance and two chrominance signals), but there are also severe problems with this approach:

- There is often disturbing cross-talk between the luminance and chrominance information.
- The composite coding of the color television signal depends on the television standard. Therefore, there would exist, besides the different number of lines and motion frequencies, another difference among the various standards. Even using multiplex techniques for further signal transmission, the standard difference would be disturbing because we would then have to apply different transmission techniques to digital television systems of different standards.
- Since luminance information is more important than chrominance, this information should be allocated more bandwidth. The sampling frequency in composite coding cannot be adapted to the bandwidth requirements of individual components; hence, data reduction cannot be adapted to the properties of the individual components.
- Using component coding, the sampling frequency is not coupled with the color carrier frequency.

All the above described disadvantages of composite coding require closer consideration of component coding.

Component Coding

The principle of component coding consists of separate digitization of various image components or planes; for example, coding of luminance and color difference (chrominance) signals. These digital signals can be transmitted together using multiplexing. The luminance signal is sampled with 13.5 MHz as it is more important than the chrominance signal. These chrominance signals (R-Y, B-Y) are sampled with 6.75 MHz. The digitized luminance and chrominance signals are then quantized uniformly with eight bits. Because of the component bandwidth ratio (4:2:2), we get 864 sample values per line for the luminance (720 samples are visible) and 432 for each chrominance component (360 samples are visible).

In PAL, for example, a picture consists of 575 lines with 25 frames per second. The high data rate and the fact that these signals do not fit into the European PCM-hierarchy (139.264 MBits/s, 34.368 MBits/s, etc.) causes problems. Therefore, different *substandards* were introduced and lower data rates defined. These substandards were derived from the bandwidth of the components. We describe some of these substandards in Section 5.2.4.

HDTV Systems

Worldwide, three different HDTV systems have and are being developed:

- *United States*

  HDTV will be the follower of IDTV. The effort goes toward compatibility with the NTSC format, but several changes will be visible as the width-to-height ratio of the HDTV image will be 16 to 9 with 1,025 lines and a 59.94 motion frequency. The HDTV format will be a full-digital solution. The compatibility with NTSC will be achieved through using IDTV. Parameter characteristics of the U.S. HDTV service are shown in Tables 5.1 and 5.2.

- *Europe*

  The transmission method of the European HDTV system is known as HD-MAC (*High Definition Multiplexed Analog Components*) [VHC89]. The parameters of this HDTV service are shown in Figure 5.6.

  For compatibility with PAL and D2-MAC, the HD-MAC will use a two-level sampling scheme. The particular high-resolution HDTV motion pictures will be transmitted to each of the 625 lines with reduced full-picture motion. Full resolution and full-picture motion can be shown in the HD-MAC receiver using digital image storage. This reduction of the number of lines and halving fo the full-picture motion allows a simpler conversion of this signal into current PAL or D2-MAC signals. The HD-MAC method was specified in Eureka Project EU 95. This project began in 1986 and includes 35 European producers.

television studios and research institutes. However, since 1993, Europe tends to follow the North American approach leading to a fully digital system.

- *Japan*

  The first HDTV service available to the public was the NHK (Japan Broadcasting Company) system, known as MUSE, which began operation in late 1989. MUSE is a Direct-Broadcast-from-Satellite (DBS) system that employs a 1125-line image at its input. It converts these lines to the narrow channel required by the satellite transponders. This conversion retains the full detail of the 1125 line image, but only when the scene is stationary. When motion occurs, the definition is reduced by approximately 50%. The spatial and temporal characteristics of HDTV are shown in Tables 5.1 and 5.2.

5.2.4 Transmission

For multimedia systems, the *data rates* created by motion video are important. We discuss the U.S. HDTV format first and then the European HDTV format, followed by data rates of the substandards for digital television.

For the U.S. HDTV image we assume 720,000 total pixels per frame (Table 5.1). To compute the data rate for HDTV motion video, we need further parameters such as the number of bits/pixel and number of frames/second. If the quantization is 24 bits/pixel, and the frame rate is approximately 60 frames/second, then the data rate for HDTV will be 1.296×10^8 bytes/second or 1.0368×10^9 bits/second (1036.8 Mbits/second). Using a compression method, data rate reduction to 34 Mbits/second is possible without noticeable quality loss.

In the case of European HDTV, the data rate is 1.152×10^9 bits/second, in general.

To *digital television*, different *substandards* and data reductions can be applied, and hence the data rate can be calculated as follows:

- *Substandard 1*

  Substandard 1 works with a luminance sampling frequency of 11.25 MHz (5/6 of the standard) and with a chrominance sampling frequency of 5.625 MHz

(5/6 of the standard). The data rate is 180 x 10^6 bits/second or 22.5 x 10^6 bytes/second.

- *Substandard 2*

  Substandard 2 specifies a luminance sampling frequency of 10.125 MHz (3/4 of the standard) and a chrominance sampling frequency of 3.375 MHz (1/2 of the standard). This rate fits into the 140 Mbits/second transmission channels and it is a good compromise. The data rate is 125 x 10^6bits/second or 16.875 x 10^6 bytes/second.

- *Substandard 3*

  Substandard 3 specifies a luminance sampling frequency of 9.0 MHz (2/3 of the standard) and a chrominance sampling frequency of 2.25 MHz (1/3 of the standard). The data rate is 108 x 10^6 bits/second = 13.5 x 10^6 bytes/second.

Further reduction of the data rates is possible: first, the *sampling gaps* can be left out, which means that only the visible areas are coded. For example, the luminance consists of 648 sample values per line, but from this number only 540 are visible. The chrominance is digitized with 216 sample values per line, but only 180 pixels are visible. Hence, if we assume that the picture consists of 575 visible lines, then this coding leads to a data rate of:

- (540 + 180 + 180) sample values/line x 575 visible lines/picture = 517,500 sample values/picture
- 517,500 sample values/picture x 8 bits/sample value x 25 pictures/second = 103.5 x 10^6bits/second

Second, *a reduction of vertical chrominance resolution* can be performed. In this case, only the chrominance difference signals of each second line are transmitted. Alternating lines of both chrominance components are digitized (*R-Y, B-Y, R-Y, etc.*). Therefore, using Substandard 2, the following data rate can be computed:

- (540 + 90 + 90) sample values/line x 575 visible lines/picture = 414,000 sample values/picture

- 414,000 sample values/picture $\times$ 8 bits/sample value $\times$ 25 pictures/s = 82.8 $\times$ 10^6 bits/second

Third, different kinds of *source coding* can be applied to the components. For further information, see Chapter 6. Here we present only one result: with an intra-frame working ADPCM, a data reduction of 3 bits/sample (instead of 8 bits/sample) for luminance and chrominance can be achieved. Using Substandard 2, the following data rate can be computed:

- (540 + 90 + 90) sample values/line $\times$ 575 visible lines/picture = 414,000 sample values/picture

- 414,000 sample values/picture $\times$ 3 bits/sample value $\times$ 25 pictures/s = 31.050 $\times$ 10^6 bits/s

5.3 Computer-based Animation

To *animate* something is, literally, to bring it to life. An animation covers all changes that have a visual effect. Visual effects can be of different nature. They might include time-varying positions (*motion dynamics*), shape, color, transparency, structure and texture of an object (*update dynamics*), and changes in lighting, camera position, orientation and focus.

A computer-based animation is an animation performed by a computer using graphical tools to provide visual effects. We concentrate in this section on computer-based animation because this kind of animation will become part of multimedia systems, although traditional (non-computer) animation is a discipline itself and exerts considerable influence over computer-based animation. Conversely, many stages of conventional animation seem ideally suited to computer assistance.

5.3.1 Basic Concepts

Input Process

Before the computer can be used, drawings must be digitized because *key frames*, meaning frames in which the entities being animated are at extreme or characteristic positions, must be drawn. This can be done through optical scanning, tracing the drawings with a data tablet or producing the original drawings with a drawing program in the first place. The drawings may need to be post-processed (e.g., filtered) to clean up any glitches arising from the input process.

Composition Stage

The *composition stage*, in which foreground and background figures are combined to generate the individual frames for the final animation, can be performed with *image-composition techniques* [FDFH92]. By placing several low-resolution frames of an animation in a rectangular array, a trail film (*pencil test*) can be generated using the pan-zoom feature available in some frame buffers. The frame buffer can take a particular portion of such an image (*pan*) and then enlarge it to fill the entire screen (*zoom*). This process can be repeated on several frames of the animation stored in the single image. If it is done fast enough, it gives the effect of continuity. Since each frame of the animation is reduced to a very small part of the total image (1/25 or 1/36), and then expanded to fill the screen, the display device's resolution is effectively lowered.

Inbetween Process

The animation of movement from one position to another needs a composition of frames with intermediate positions (*intermediate frames*) inbetween the key frames. This is called the *inbetween process*. The process of *inbetweening* is performed in computer-based animation through *interpolation*. The system gets only the starting and ending positions. The easiest interpolation in such a situation is linear interpolation (sometimes called *lerping* – Linear intERPolation), but it has many limitations. For instance, if *lerping* is used to compute intermediate positions of a ball that is thrown in the air using the sequence of three key frames shown in Figure 5.8 (a), the resulting track of the ball shown in Figure 5.8 (b) is entirely unrealistic. Because of the drawbacks of lerping, *splines* are often used instead to

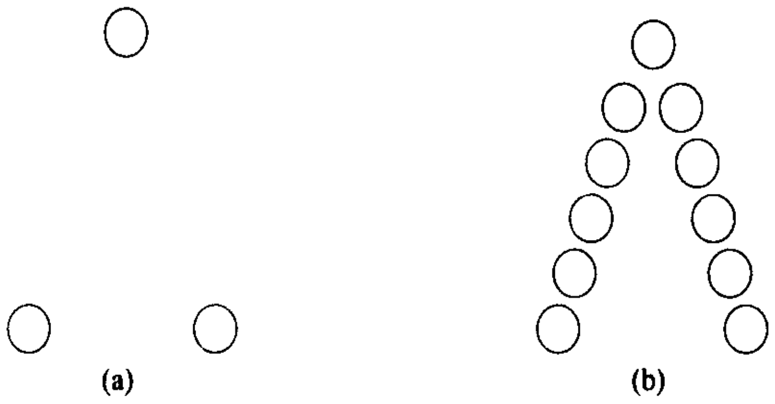

Figure 5.8: *Linear interpolation of the motion of a ball.*

smooth out the interpolation between key frames. Splines can be used to vary any parameter smoothly as a function of time. Splines can make an individual point (or individual objects) move smoothly in space and time, but do not entirely solve the inbetweening problem.

Inbetweening also involves interpolating the shapes of objects in intermediate frames. Several approaches to this have been developed, including one by Burtnyk and Wein [BW76]. They made a *skeleton* for a motion by choosing a polygonal arc describing the basic shape of a 2D figure (or portion of a figure) and a neighborhood of this arc. The figure is represented in a coordinate system based on this skeleton. Inbetweening is performed by interpolating the characteristics of the skeleton between the key frames. A similar technique can be developed for 3D, but generally interpolation between key frames is a difficult problem.

Changing Colors

For changing colors, computer-based animation uses CLUT (lut) in a frame buffer and the process of double buffering. The lut animation is generated by manipulating the lut. The simplest method is to cycle the colors in the lut, thus changing the colors of the various pieces of the image. Using lut animation is faster than sending an entire new pixmap to the frame buffer for each frame. Assuming 8 color bits per pixel in a 640×512 frame buffer, a single image contains 320 Kbytes of information. Transferring a new image to the frame buffer every 1/30 of a second requires a bandwidth of over 9 Mbytes per second. On the other hand, new values for the lut can be sent very rapidly, since luts are typically on the order of a few hundred to a few thousand bytes.

5.3.2 Animation Languages

There are many different languages for describing animation, and new ones are constantly being developed. They fall into three categories:

- *Linear-list Notations*

  In linear-list notations for animation each event in the animation is described by a starting and ending frame number and an action that is to take place (*event*). The actions typically take parameters, so a statement such as

  ```
  42, 53, B, ROTATE ''PALM'',1,30
  ```

  means "between frames 42 and 53, rotate the object called PALM about axis 1 by 30 degrees, determining the amount of rotation at each frame from table B" [FDFH92]. Many other linear-list notations have been developed, and many are supersets of the basic linear-list idea. An example is *Scefo (SCEne FOrmat)* [Str88], which also includes the notion of groups and object hierarchy and supports abstractions of changes (called *actions*) using higher-level programming language constructs.

- *General-purpose Languages*

Another way to describe animation is to embed an animation capability within a general-purpose programming language. The values of variables in the language can be used as parameters to the routines, which perform the animation.

ASAS is an example of such a language [Rei82]. It is built on top of LISP, and its primitive entities include vectors, colors, polygons, solids, groups, points of view, subworlds and lights. *ASAS* also includes a wide range of geometric transformations that operate on objects. The *ASAS* program fragment below describes an animated sequence in which an object called my-cube is spun while the camera pans. This fragment is evaluated at each frame to generate the entire sequence.

```
(grasp my-cube) ; The cube becomes the current object

(cw 0.05) ; Spin it clockwise by a small amount

(grasp camera) ; Make the camera the current object

(right panning-speed) ; Move it to the right
```

- *Graphical Languages*

One problem with textual languages is inability to visualize the action by looking at the script. If a real-time previewer for textual animation languages were available, this would not be a problem; unfortunately the production of real-time animation is still beyond the power of most computer hardware.

Graphical animation languages describe animation in a more visual way. These languages are used for expressing, editing and comprehending the simultaneous changes taking place in an animation. The principal notion in such languages is substitution of a visual paradigm for a textual one. Rather than explicitly writing out descriptions of actions, the animator provides a picture of the action.

Examples of such systems and languages are *GENESYS*^{TM}[Bae69], *DIAL* [FSB82] and *S-Dynamics System* [Inc85].

5.3.3 Methods of Controlling Animation

Controlling animation is independent of the language used for describing it. Animation control mechanisms can employ different techniques.

Full Explicit Control

Explicit control is the simplest type of animation control. Here, the animator provides a description of everything that occurs in the animation, either by specifying simple changes, such as scaling, translation, and rotation, or by providing key frame information and interpolation methods to use between key frames. This interpolation may be given explicitly or (in an interactive system) by direct manipulation with a mouse, joystick, data glove or other input device. An example of this type of control is the *BBOP* system [Ste83].

Procedural Control

Procedural control is based on communication between various objects to determine their properties. Procedural control is a significant part of several other control mechanisms. In particular, in ***physically-based systems,*** the position of one object may influence the motion of another (e.g., balls cannot pass through walls); in ***actor-based systems,*** the individual actors may pass their positions to other actors to affect the other actors' behaviors.

Constraint-based Systems

Some objects in the physical world move in straight lines, but many objects move in a manner determined by other objects with which they are in contact; and this compound motion may not be linear at all. Such motion can be modeled by constraints. Specifying an animated sequence using constraints is often much easier to do than using explicit control. Systems using this type of control are Sutherland's *Sketchpad* [Sut63] or Borning's *ThingLab* [Bor79].

The extension of constraint-based animation systems to support a hierarchy of constraints and to provide motion where constraints are specified by the dynamics of physical bodies and structural characteristics of materials is a subject of active research.

Tracking Live Action

Trajectories of objects in the course of an animation can also be generated by tracking live action. Traditional animation uses *rotoscoping.* A film is made in which people/animals act out the parts of the characters in the animation, then animators draw over the film, enhancing the background and replacing the human actors with their animated equivalents.

Another live-action technique is to attach some sort of indicator to key points on a person's body. By tracking the positions of the indicators, one can get locations for corresponding key points in an animated model. An example of this sort of interaction mechanism is the data glove, which measures the position and orientation of the wearer's hand, as well as the flexion and hyperextension of each finger point.

Kinematics and Dynamics

Kinematics refers to the position and velocity of points. A kinematic description of a scene, for example, might say, "The cube is at the origin at time $t = 0$. It moves with a constant acceleration in the direction (1,1,5) thereafter."

By contrast, *dynamics* takes into account the physical laws that govern kinematics (e.g., Newton's laws of motion for large bodies, the Euler-Lagrange equations for fluids, etc.). A particle moves with an acceleration proportional to the forces acting on it, and the proportionality constant is the mass of the particle. Thus, a dynamic description of a scene might be, "At time $t = 0$ seconds, the cube is at position (0 meters, 100 meters, 0 meters). The cube has a mass of 100 grams. The force of gravity acts on the cube." Naturally, the result of a dynamic simulation of such a model is that the cube falls.

5.3.4 Display of Animation

To display animations with raster systems, animated objects (which may consist of graphical primitives such as lines, polygons, and so on) must be *scan-converted* into their pixmap in the frame buffer. To show a rotating object, we can scan-convert into the pixmap successive views from slightly different locations, one after another. This scan-conversion must be done at least 10 (preferably 15 to 20) times per second to give a reasonably smooth effect· hence a new image must be created in no more than 100 milliseconds. From these 100 milliseconds, scan-converting should take only a small portion of time. For example, if scan-converting of an object takes 75 milliseconds, only 25 milliseconds remain to erase and redraw the complete object on the display, which is not enough, and a distracting effect occurs. Double-buffering is used to avoid this problem. The frame buffer is divided into two images, each with half of the bits per pixel of the overall frame buffer. As an example, we describe the display of the rotation animation [FDFH92]. Let us assume that the two halves of the pixmap are $image_0$ and $image_1$.

```
Load look-up table to display values as background color
Scan-convert object into $image_0$
Load look-up table to display only $image_0$
**Repeat**
    Scan-convert object into $image_1$
    Load look-up table to display only $image_1$
    Rotate object data structure description
    Scan-convert object into $image_0$
    Load look-up table to display only $image_0$
    Rotate object data structure description
**Until** (termination condition).
```

If rotating and scan-converting the object takes longer than 100 milliseconds, the animation is quite slow, but the transition from one image to the next appears to be instantaneous. Loading the look-up table typically takes less than one millisecond [FDFH92].

5.3.5 Transmission of Animation

As described above, animated objects may be represented symbolically using graphical objects or scan-converted pixmap images. Hence, the transmission of animation over computer networks may be performed using one of two approaches:

- The symbolic representation (e.g., circle) of animation objects (e.g., ball) is transmitted together with the operation commands (e.g., roll the ball) performed on the object, and at the receiver side the animation is displayed as described in Section 5.3.4. In this case, the transmission time is short because the symbolic representation of an animated object is smaller in byte size than its pixmap representation, but the display time at the receiver takes longer because the scan-converting operation has to be performed at the receiver side.

  In this approach, the transmission rate (bits/second or bytes/second) of animated objects depends (1) on the size of the symbolic representation structure, where the animated object is encoded, and (2) on the size of the structure, where the operation command is encoded, and (3) on the number of animated objects and operation commands sent per second.

- The pixmap representation of the animated objects is transmitted and displayed on the receiver side. In this case, the transmission time is longer in comparison to the previous approach because of the size of the pixmap representation, but the display time is shorter because the scan-conversion of the animated objects is avoided at the receiver side. It is performed at the sender side where animation objects and operation commands are generated.

  In this approach, the transmission rate of the animation is equal to the size of the pixmap representation of an animated object (graphical image) multiplied by the number of graphical images per second.

5.3.6 Comments

The current development of workstations toward support of motion video and animation is progressing very quickly. Silicon Graphics^{TM}workstations (e.g., Indigo^{TM}XS24A, Indigo Elan, and others) provide high-quality color graphics displays, as well as video boards for capturing motion video. The major emphasis in hardware (e.g., cameras, video boards, workstations) is on achieving real-time motion video and computer-based animation. This allows researchers to achieve better results in areas such as "human facial animation based on speech intonation, emotion, and dialogue models" (a research project in the Center for Human Modeling and Simulation at the University of Pennsylvania), "3-dimensional tracking, focus ranging, and precision measurements of objects from a 2-axis camera" (a research project in the GRASP Laboratory at the University of Pennsylvania) and others.

Chapter 6

Data Compression

6.1 Storage Space

Uncompressed graphics, audio and video data require considerable storage capacity which in the case of uncompressed video is often not even feasible given today's CD technology. The same is true for multimedia communications. Data transfer of uncompressed video data over digital networks requires very high bandwidth to be provided for a single point-to-point communication. To provide feasible and cost-effective solutions, most multimedia systems handle compressed digital video and audio data streams. Note that in this chapter the terms *coding* and *compression* are treated as synonyms.

As shown in [ACM89, ACM91, JSA92a, JSA92b, Lu93, ACD+93] there already exist many compression techniques that are in part competitive and in part complementary. Most of them are already used in today's products, while other methods are still undergoing development or are only partly realized (see also [SPI94]). Whereas *fractal image compression* [BH93] might someday be relevant, today and in the near future, the most important compression techniques are JPEG (for single pictures [Org93, PM93, Wal91]), H.261 (px64, for video [Le 91, ISO93a]), MPEG (for video and audio [Lio91, ITUC90]) and proprietary developments including HQ Learn's DVI (for still images, audio and video [HKL+91]), Intel's *Indeo*, Microsoft's *Video for Windows*, IBM's *Ultimotion Matinee*, Apple's *QuickTime* or *DigiCipher II* developed by General Instruments Corp. and AT&T.

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 superficial or dedicated to 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 relevant techniques (JPEG, H.261, MPEG, DVI) to provide understanding of their advantages and disadvantages, their common abilities and differences and their suitability for today's multimedia systems. First, a short motivation for the need of compression is given. Subsequently, some requirements for these techniques are derived. In Section 6.4, source, entropy and hybrid coding techniques are explained in detail, while Sections 6.5 through 6.8 provide details on JPEG, H.261, MPEG and DVI, respectively.

6.2 Coding Requirements

Images have considerably higher storage requirements than text; audio and video have even more demanding properties for data storage. Not only is a huge amount of storage required, but the data rates for the communication of continuous media are also significant. With the below specified numbers we want to clarify the qualitative transition from simple text to full-motion video data and derive the need for compression. To compare data storage and bandwidth requirements of different visual media (text, graphics, image), the following specifications are based on a typical window size of 640 × 480 pixels on a screen:

- For the representation of text, two bytes are used for each character, allowing for some Asian language variants. Each character is displayed using 8 × 8 pixels, which is sufficient for the display of ASCII characters.

- For the presentation of vector-graphics, a typical still image is composed of 500 lines [BHS91]. Each line is defined by its horizontal position, vertical position and an 8-bit attribute field. The horizontal axis is represented using 10 bits (log2(640)), and the vertical axis is coded using 9 bits (log2(480)).

- In very simple color display modes, a single pixel of a bitmap can be represented by 256 different colors; therefore, one byte per pixel is needed.

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

- An uncompressed audio signal of telephone quality is sampled at 8 kHz and quantized with 8 bits per sample. This leads to a bandwidth requirement of 64 kbits/second and storage requirement of 64 kbits to store one second of playback.

- An uncompressed stereo audio signal of CD quality is sampled at a rate of 44.1 kHz and is quantized with 16 bits per sample; hence, the storage requirement is (44.1 kHz x 16 bits) = 705.6 x 10^3 bits to store one second of playback and the bandwidth (throughput) requirement is 705.5 x 10^3 bits/second.

- According to the European PAL standard, video is defined by 625 lines and 25 frames per second. The luminance and color difference signals are encoded separately. The resulting digital data streams are transformed using a multiplexing technique (4:2:2). Corresponding to CCIR 601 (a studio standard for digital video), a sampling rate of 13.5 MHz is used for luminance Y. The sampling rate for chrominance (R-Y and B-Y) is 6.75 MHz. If the result is a uniform 8-bit coding of each sample, the bandwidth requirement is (13.5 MHz + 6.75 MHz + 6.75 MHz) x 8 bits = 216 x 10^6 bits/second. HDTV doubles the number of lines and uses an aspect ratio of 16/9. This leads to a data rate which increases by a factor of 5.33 compared to today's TV rate.

  To determine storage requirements for the PAL standard, we assume the same image resolution as used before (640 x 480 pixels) and 3 bytes/pixel to encode the luminance and chrominance components. Hence, the storage requirement for one image (frame) is (640 pixels x 480 pixels X 3 bytes) = 921,600 bytes or 7,372,800 bits. To store 25 frames/second, the storage requirement is 230.4 x 10^5 bytes or 184.32 x 10^6 bits.

The storage and throughput requirements of a computer system that processes still images and in particular continuous media are illustrated by these few examples.

In the case of video, processing uncompressed data streams in an integrated multimedia system leads to secondary storage requirements in the range of at least giga-bytes, and in the range of mega-bytes for buffer storage. The throughput in a multimedia system can be as high as 140 Mbits/second, which must be transferred between different systems. This kind of data transfer rate is not realizable with today's technology, or in the near future with reasonably priced hardware. However, the use of appropriate compression techniques considerably reduces the data transfer rates [NH88, RJ91], and fortunately research, development and standardization have rapidly progressed in this area during the last few years [ACM91]. Some compression techniques for different media are often mentioned in the literature and product descriptions: JPEG for still image compression and H.261 (px64) for video conferencing. Additionally, some audio coding techniques, developed for ISDN and mobile communications, could also be used for compression of speech and music. MPEG is used for video and audio compression, while DVI can be used for still images, as well as for continuous media. DVI consists of two different modes for video coding: Presentation-Level Video (PLV) and Real-Time Video (RTV).

Compression in multimedia systems is subject to certain constraints. The quality of the coded, and later on, decoded 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 of the algorithm must not exceed certain time spans.

For each compression technique, there are requirements that differ from those of other techniques (see, for example, the requirements of [ISO93a]). One can distinguish between requirements of an application running in a "dialogue" mode and in a "retrieval" mode, where a dialogue mode means an interaction among human users via multimedia information and a retrieval mode means a retrieval of multimedia information by a human user from a multimedia database. Some compression techniques like px64 are more suitable for dialogue mode applications. Other techniques, like the DVI PLV mode, are optimized for use in retrieval mode applications.

In a *dialogue mode application*, the following requirement, based on human perception characteristics, must be considered: the *end-to-end delay* should not exceed 150 milliseconds (for compression and decompression). A delay in the range of 50 milliseconds should be achieved to support "face-to-face" dialogue applications. The number 50 milliseconds relates to the delay introduced by compression and decompression only. The overall end-to-end delay additionally comprises any delay in the network, in the involved communication protocol processing at the end system and in the data transfer from and to the respective input and output devices.

In a *retrieval mode application*, the following demands arise:

- *Fast forward and backward data retrieval* with simultaneous display should be possible. This implies a fast search for information in multimedia databases.
- *Random access* to single images and audio frames of a data stream should be possible, making the access time less than 0.5 second. This access should be faster than a conventional CD audio system to maintain the interactive character of the application.
- Decompression of images, video or audio should be possible *without a link to other data units*. This allows random access and editing.

For both dialogue and retrieval mode, the following requirements apply:

- To support scalable video in different systems, it is necessary to define a format independent of frame size and video frame rate.
- Various audio and video data rates should be supported; usually this leads to different qualities. Thus, depending on specific system conditions, the data rates can be adjusted.
- It must be possible to synchronize audio with video data, as well as with other media.
- To make an economical solution possible, coding should be realized using software (for a cheap and low-quality solution) or VLSI chips (for a high-quality solution).
- It should be possible to generate data on one multimedia system and reproduce these data on another system. The compression technique should be compatible. This compatibility is relevant in the case of tutoring programs available on a CD for example; it allows different users to read the data on different

systems, thus being independent of the manufacturers. As many applications exchange multimedia data using communication networks, the compatibility of compression techniques is required. Standards like CCITT, ISO and ECMA and/or defacto standards are used to achieve this compatibility.

This set of requirements is taken into account to a varying extent by the various compression schemes.

6.3 Source, Entropy and Hybrid Coding

Compression techniques fit into different categories, as shown in Table 6.1. For their use in multimedia systems, we can distinguish among *entropy*, *source* and *hybrid encoding*. Entropy encoding is a *lossless* process, while source encoding is a *lossy* process. Most multimedia systems use hybrid techniques, which are a combination of the two.

Entropy coding is used regardless of the media's specific characteristics. The data stream to be compressed is considered to be a simple digital sequence and the semantics of the data is ignored. Entropy encoding is an example of lossless encoding as the decompression process regenerates the data completely. Run-length coding is an example of entropy encoding that is used for data compression in file systems.

Source coding takes into account the semantics of the data. The degree of compression that can be reached by source encoding depends on the data contents. In the case of lossy compression techniques, a one-way relation between the original data stream and the encoded data stream exists; the data streams are similar but not identical. Different source encoding techniques make extensive use of the characteristics of the specific medium. An example is the sound source coding, where sound is transformed from time-dependent to frequency-dependent sound concatenations, followed by the encoding of the formants (see Chapter 3 - Speech Generation). This transformation substantially reduces the amount of data. *Formants* are defined as being the maxima of the voice spectrum. In most cases, three to five formants are sufficient to reconstruct the original signal in the time domain. The major problem is the correct reproduction of the transitions between individual voice units in the

Entropy Encoding	Run-length Coding	
	Huffman Coding	
	Arithmetic Coding	
Source Coding	Prediction	DPCM
		DM
	Transformation	FFT
		DCT
	Layered Coding	Bit Position
		Subsampling
		Sub-band Coding
	Vector Quantization	
Hybrid Coding	JPEG	
	MPEG	
	H.261	
	DVI RTV, DVI PLV	

Table 6.1: *A rough classification of coding/compression techniques in multimedia systems.*

time domain.

A content prediction technique can make use, for example, of spatial redundancies within still images. Other techniques perform a transformation of the spatial domain into the two-dimensional frequency domain by using the *Discrete Cosine Transformation* (DCT). Low frequencies define the average color, and the information of high frequencies contains the sharp edges. Hence, low frequencies are much more important than the higher ones, which is a key feature used in DCT-based compression.

Table 6.1 shows examples of coding and compression techniques that are applicable to multimedia applications in relation to the entropy, source and hybrid coding classification. For a better and clearer understanding of *hybrid schemes* we will identify in all schemes (entropy, source and hybrid) a set of typical processing steps.

Figure 6.1 shows this typical sequence of operations performed in the compression of still images, video and audio data streams. The following four steps describe one

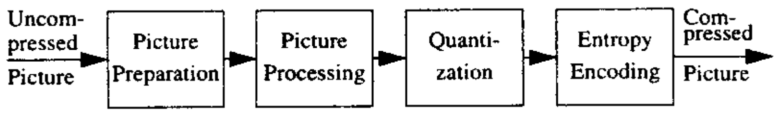

Figure 6.1: *Major steps of data compression.*

image compression.

1. *Preparation* includes analog-to-digital conversion and generating an appropriate digital representation of the information. An image is divided into blocks of 8 × 8 pixels, and represented by a fixed number of bits per pixel.

2. *Processing* is actually the first step of the compression process which makes use of sophisticated algorithms. A transformation from the time to the frequency domain can be performed using DCT. In the case of motion video compression, interframe coding uses a motion vector for each 8 × 8 block.

3. *Quantization* processes the results of the previous step. It specifies the granularity of the mapping of real numbers into integers. This process results in a reduction of precision. This can also be considered as the equivalence of the μ-law and A-law, which apply to audio data [JN84]. In the transformed domain, the coefficients are distinguished according to their significance. For example, they could be quantized using a different number of bits per coefficient.

4. *Entropy encoding* is usually the last step. It compresses a sequential digital data stream without loss. For example, a sequence of zeroes in a data stream can be compressed by specifying the number of occurrences followed by the zero itself.

Processing and quantization can be repeated iteratively several times in feedback loops, such as in the case of *Adaptive Differential Pulse Code Modulation* (ADPCM). After compression, the compressed video builds a data stream, where a specification of the image starting point and an identification of the compression technique may be part of this data stream; the error correction code may also be added to the stream. Figure 6.1 shows the compression process applied to an image; the same principle can be applied to video and audio data.

Decompression is the inverse process of *compression*. The specific encoders and decoders can function in various ways. Symmetric applications, e.g., dialogue applications, should be characterized by more or less the same costs for encoding and decoding. In the case of asymmetric techniques, the decoding process is less costly than the encoding process. This is used for applications in which: (1) the compression process is performed only once and sample time is available, and (2) the decompression is performed frequently and needs to be done quickly. For example, an ***audio-visual tutoring program*** will be produced once but it will be used by many students; therefore, it will be decoded many times. In this case, real-time decoding is a fundamental requirement, whereas encoding need not be performed in real-time. This asymmetric processing can be exploited to increase the quality of the images.

6.4 Some Basic Compression Techniques

Hybrid compression techniques are a combination of well-known algorithms and transformation techniques that can be applied to multimedia systems. For example, all hybrid techniques shown in Table 6.1 use entropy encoding (in the form of run-length encoding and/or a statistical compression).

The simplest compression techniques are based on *interpolation* and *subsampling*. Here, it is possible to make use of the specific physiological characteristics of the human eye or ear. The human eye is more sensitive to changes in brightness than to color changes. Therefore, it is reasonable to divide the image into YUV components instead of RGB components (see Chapter 5 on YUV and RGB encodings).

Sampled images, audio and video data streams often contain sequences of the same bytes. By replacing these repeated byte sequences with the number of occurrences, a substantial reduction of data can be achieved. This is called *run-length coding*, which is indicated by a special flag that does not occur as a part of the data stream itself. This flag byte can also be realized by using any other of the 255 different bytes in the compressed data stream. To illustrate such a *byte-stuffing*, we define the exclamation mark "!" as a special flag. A single occurrence of this exclamation flag is interpreted as a special flag during decompression. Two consecutive exclamation flags are interpreted as an exclamation mark occurring within the data. The overall run-length coding procedure can be described as follows: if a byte occurs at least four consecutive times, the number of occurrences is counted. The compressed data contains this byte followed by the special flag and the number of its occurrences. This allows the compression of between 4 and 259 bytes into three bytes only. Remembering that we are compressing at least 4 consecutive bytes, and the number of occurrences can start with an offset of -4. Depending on the algorithm, one or more bytes can be used to indicate the length. In the following example, the character "C" occurs 8 consecutive times and is "compressed" to 3 characters "C!8":

Uncompressed data: ABCCCCCCCCDEFGGG

Run-length coded: ABC!8DEFGGG

Run-length encoding is a generalization of *zero suppression,* which assumes that just one symbol appears particularly often in sequences. The blank (null character – space) in text is such a symbol; single blanks or pairs of blanks are ignored. Starting with a sequence of three blanks, they are replaced by an M-byte (M-byte has the same function as the exclamation mark before) and a byte that specifies the number of blanks of this sequence. Sequences of three, to a maximum of 258 bytes, can be reduced to two bytes. The number of occurrences can be indicated with an offset of -3 (because three blanks are being suppressed). Further variations are *tabulators* used to substitute a specific number of zeros (or blanks). The substitution depends on the relative position within a line and the definition of different M-bytes to specify a different number of zero bytes (or blanks). The flag M4-byte could replace 8 zero bytes, and another M5-byte could substitute a sequence of 16 zero bytes. An M5-byte followed by an M4-byte would represent 24 zero bytes.

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, a table entry with the most similar pattern is identified. Each pattern in the table is associated with an index. Such a table can be multi-dimensional; in this case, the index will be a vector. A decoder uses the same table to generate an approximation of the original data stream. For further details and refinements see, e.g., [Gra84].

A technique that can be used for text compression substitutes single bytes for patterns that occur frequently. This pattern substitution replaces, for instance, the terminal symbols of high-level languages ("Begin," "End," "If"). Using an escape-byte, a larger number of patterns can be considered; this escape-byte indicates that an encoded pattern will follow. The next byte is an index used as a reference to 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 often better to perform an approximation that looks for the most similar pattern instead of searching for the same pattern, in either case leading to the above described vector quantization.

Diatomic encoding is a variation of run-length encoding based on a combination of two data bytes. This technique determines the most frequently occurring pairs of bytes. According to an analysis of the English language, the most frequently occurring pairs are the following (note, there are blanks included in the pairs "e " "t ", " a" and "s "): "E ", "T ", "TH", " A", "S ", "RE", "IN" and "HE". Replacement of these pairs by special single bytes that do not occur anywhere else in the text leads to a data reduction of more than 10%.

Different characters do not have to be coded with a fixed number of bits. The Morse alphabet is based on this idea. Frequently-occurring characters are coded with shorter strings and seldom-occurring characters are coded with longer strings. Such statistical encoding depends on the occurrence frequency of single characters or sequences of data bytes. There are different techniques that are based on these statistical methods, the most prominent of which are *Huffman* and *Arithmetic encoding.*

Given the characters that must be encoded, together with the probability of their occurrences, the Huffman coding algorithm determines the optimal code using the minimum number of bits [Huf52]. Hence, the length (number of bits) of the coded characters will differ. In text, the shortest code is assigned to those characters that occur most frequently. To determine a Huffman code, it is useful to construct a binary tree. The leaves (node) of this tree represent the characters that are to be encoded. Every node contains the *occurrence probability* of one of the characters belonging to this subtree. 0 and 1 are assigned to the branches (edges) of the tree.

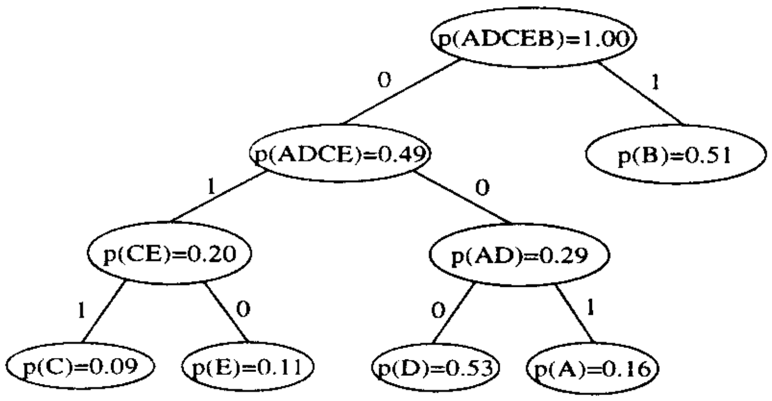

Figure 6.2: *Huffman encoding – an example.*

The following example illustrates this process:

- In Figure 6.2, characters A, B, C, D and E have the following probability of occurrence:

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

- Characters with the lowest probabilities are combined in the first binary tree, thus C and E are the leaves. The combined probability of their root node CE is 0.20. The edge from node CE to node C is assigned a 1 and the edge from CE to E becomes a 0. This is an arbitrary assignment; therefore, with the same data one can get different Huffman codes.

- The following nodes remain:

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

  Again, the two nodes with the lowest probabilities are combined into a binary subtree: the nodes A and D are such leaves and the combined probability of their root AD is 0.29. The edge from AD to A is assigned a 1 and the edge from AD to D a 0. Note, if there are root nodes of different subtrees with the same probabilities, the trees with the shortest maximal path between their roots and their nodes should be combined. This keeps the code length more constant.

- The following nodes remain:

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

  The nodes with the smallest probabilities are *AD* and *CE*. They 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* a 1.

- Two nodes remain:

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

  They are combined to a binary tree with the root *ADCEB*. The edge from *ADCEB* to *B* is assigned a 1, and the edge from *ADCEB* to *CEAD* is assigned a 0.

- Figure 6.2 shows the resulting Huffman code in the form of a binary tree. The result is the following code that is stored in a table:

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

If the information of an image can be transformed into a bit stream, such a table can be used to compress the data without any loss. The most simple way to generate 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. Such a stream can be determined for each image or for a set of images. For videos, a Huffman table can be used for a single sequence of images, for a set of scenes or even for an entire film clip. The same Huffman table must be available for both encoding and decoding.

If we consider run-length coding and all the other methods described so far, which produced the same consecutive symbols (bytes) quite often, it is certainly a major objective to transform images and videos into a bit stream.

From the information theory point of view, arithmetic encoding [Lan84, PMJA88], like Huffman encoding, is optimal. Therefore, the length of the encoded data stream is also minimal. Unlike Huffman coding, arithmetic coding does not encode each symbol separately; each symbol is instead coded by considering the prior data. Therefore, an encoded data stream 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].

Transformation encoding pursues a different approach. Data is transformed into another mathematical domain which is more suitable for compression. The inverse transformation must exist and is known to the encoding process. The most widely known example is the Fourier transformation, which transforms data from the time into the frequency domain. The most effective transformations for image compression are the *Discrete Cosine Transformation* (DCT) (see also Section 6.5.2) and to some extent, the *Fast-Fourier-Transformation* (FFT).

Unlike transformation encoding that transforms all data into another domain, selective frequency transformation sub-band coding considers a spectral selection of the signal in predefined frequency bands. The number of bands is an important criterion for quality. This technique is well-suited for the compression of speech and often makes use of FFT for spectral filtering.

Instead of compressing single bytes or sequences of bytes, *differential encoding* can be used. This is also known as *prediction* or *relative encoding.* Let us consider an example of a sequence of characters whose values are clearly different from zero, but which do not differ much. In this case, the calculation of the difference from previous values could be profitable for compression. The following examples explain this technique for different media:

- For still images, during the calculation of differences between nearby pixels or pixel groups, the edges are represented by large values, whereas areas with similar luminance and chrominance are characterized by small values. A homogeneous area is characterized by a large number of zeros which could be further compressed using run-length encoding.

- For video, the use of relative coding in the time domain can lead to encoding of the differences from the previous image only. For newscast and video telephone applications, the background often remains the same for a long time; therefore, the *difference* between subsequent images is very small, leading to a large set of zeros. Another very popular technique is motion compensation (see, for example, [PA91]). Blocks of 8×8 or 16×16 pixels in subsequent pictures are compared. For example, let us consider a car moving from left to right. An area further left in a previous picture would be very similar to the same area of the current picture, here this 'motion' can be identified by a motion vector.

- Audio techniques often apply *Differential Pulse Code Modulation* (DPCM) to a sequence of *Pulse Code Modulation* (PCM)-coded samples (see e.g., [JN84]). It is not necessary to store the whole number of bits of each sample. It is sufficient to represent only the first PCM-coded sample as a whole and all following samples as the difference from the previous one.

- *Delta Modulation (DM)* is a modification of DPCM (see [JN84]). When coding the differences, it uses 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, a smaller number of bits is sufficient.

Difference encoding is an important feature of techniques used in multimedia systems. Further "delta" methods applied to images are described in Section 6.5.2.

Most of the compression techniques described so far are based on already known characteristics of the data, e.g., sequences of bytes occurring frequently or the probability of the occurrence of certain bytes. A non-typical sequence of characters will not be compressed using these methods.

Some compression techniques adapt a particular compression technique to the particular data to be compressed on the fly. This adaptation can be implemented in different ways:

- For each symbol to be encoded, there is a predefined coding table, e.g., as invented by Huffman, containing the corresponding code and a counter in an additional column. This counter is reset to zero at the beginning; for the first symbol to be encoded, the coder determines the code according to the table. Additionally, the counter of the corresponding table entry is increased by one. To reduce access to individual entries, table entries are sorted according to the values of the counters. The most frequently-occurring characters are at the beginning of the table with the higher values of the counters. This procedure leads to encoding symbols with the highest values by using the shortest codes.

- A prominent adaptive compression technique is *Adaptive DPCM* (ADPCM). It is a successive development of DPCM. Here, differences are encoded using a small number of bits only (e.g., 4 bits). Therefore, either rough transitions are coded correctly (these bits represent bits with a higher significance) or small changes are coded exactly (DPCM-encoded values are the less-significant bits). 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 this "significance" for a particular data stream as follows: the coder divides the value of DPCM samples by a suitable coefficient and the decoder multiplies the compressed data by the same coefficient, i.e., the step size of the signal changes.

  The value of the coefficient is adapted to the DPCM-encoded signal by the coder. In the case of a high-frequency signal, very high DPCM coefficient values occur. The coder determines 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. For a signal with permanently relatively low DPCM values, i.e., with few portions of high frequencies, the coder will determine a small coefficient. Thereby, a good resolution of the dominant low frequency signal portions is guaranteed. If high-frequency portions of the signal suddenly occur in such a passage, a signal distortion in the form of a slope-overload arises. Considering the actually defined step size, the greatest possible change using the existing number of bits will not be large enough to represent the DPCM value with an ADPCM value. The transition of the PCM signal will be faded.

It is possible to explicitly change the coefficient that is adaptively adjusted to the data during compression. Alternatively, the decoder is able to calculate the coefficients itself from an ADPCM-encoded data stream. An audio signal with frequently changing frequency portions of extreme high or low frequencies turns out not to be very suitable for such an ADPCM encoding. In the G.700 series of standards, CCITT has standardized a version of the ADPCM technique using 32 kbits/s for telephone applications that is based on 4 bits per sample and an 8-kHz sampling rate.

Apart from the whole set of basic compression techniques described in this section, some additional well-known techniques are being used today, namely:

- Video compression techniques often use Color Look-Up Tables (CLUT) to achieve data reduction. For instance, this technique is used in distributed multimedia systems [LE91, LEM92].

- A simple technique for audio is silence suppression; in this case, data are only encoded if the volume level exceeds a certain threshold. This can be seen as a special case of run-length encoding.

CCITT incorporates some of the basic audio coding schemes into the G.700 series of standards: G.721 defines PCM coding for 3.4 kHz quality over 64 kbits/second channels; G.728 defines 3.4 kHz quality over 16 kbits/second channels. [ACG93] provides a more detailed description of various audio coding techniques.

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

6.5 JPEG

Since June 1982, WG8 (Working Group 8) of ISO (International Standard Organization) has been working on the standardization of compression and decompression for still images [HYS88]. In June 1987, ten different techniques for still color and gray-scaled images were presented. These techniques were compared, and three of them were analyzed further. An adaptive transformation coding technique based on the DCT achieved the best (subjective) results and, therefore, was adopted for JPEG [LMY88, WVP88]. JPEG is a joint project of ISO/IECJTC1/SC2/WG10 and the commission Q.16 of CCITT SGVIII. In 1992, JPEG became an ISO International Standard (IS) [Org93].

JPEG applies to color and gray-scaled still images [LOW91, MP91, Wal91]. A fast coding and decoding of still images is also used for video sequences known as *Motion JPEG*. Today, parts of JPEG are already available as software-only packages or together with specific hardware support. It should be taken into consideration that in most cases, only the very basic JPEG algorithms, with limited spatial resolution, are supported by these products.

In addition to the requirements described in Section 6.2, JPEG fulfills the following requirements to guarantee further distribution and application [Wal91]:

- The JPEG implementation should be independent of image size.
- The JPEG implementation should be applicable to any image and pixel aspect ratio.
- Color representation itself should be independent of the special implementation.
- Image content may be of any complexity, with any statistical characteristics.
- The JPEG standard specification should be state-of-the-art (or near) regarding the compression factor and achieved image quality.
- Processing complexity must permit a software solution to run on as many available standard processors as possible. Additionally, the use of specialized hardware should substantially enhance image quality.

- Sequential decoding (line-by-line) and progressive decoding (refinement of the whole image) should be possible. A lossless, hierarchical coding of the same image with different resolutions similar to Photo-CD images should be supported.

The user can select the quality of the reproduced image, the compression processing time and the size of the compressed image by choosing appropriate individual parameters.

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. If the compression and decompression process agree on a common set of coding tables to be used, for example, their data of the respective table need not be included in the data stream. There exists a common context between coding and decoding. The interchange format can have an abbreviated format which does not guarantee inclusion of the necessary tables; however, some may be provided (see Appendix B of [Org93]). The interchange format in regular mode (i.e., the non-abbreviated format) includes all of the information necessary for decoding without any previous knowledge of the coding process.

Figure 6.3 outlines the steps of JPEG compression in accordance with the overall scheme shown in Figure 6.1. Four different variants of image compression can be

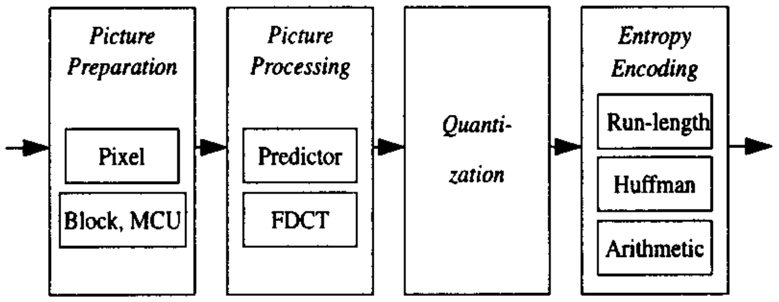

Figure 6.3: *Steps of the JPEG compression process.*

determined that lead to four modes. Each mode itself includes further combinations:

- The lossy sequential DCT-based mode (baseline process) must be supported by every JPEG implementation.
- The expanded lossy DCT-based mode provides a set of further enhancements to the baseline process.
- The lossless mode has a low compression ratio that allows perfect reconstruction of the original image.
- The hierarchical mode accommodates images of different resolutions and selects its algorithms from the three modes defined above.

The ***baseline process*** takes the following techniques: ***Block***, *MCU*, *FDCT*, ***Run-length*** and *Huffman*, which are explained with the other modes in more detail in this section. In the next section, image preparation for all modes is presented; the remaining steps of image processing, quantization and entropy encoding are described.

6.5.1 Image Preparation

For the first step of image preparation, JPEG specifies a very general image model. With this model it is possible to describe most of the well-known two-dimensional image representations. For instance, the model is not based on three image components with 9-bit *YUV* coding and a fixed number of lines and columns. Mapping of encoded chrominance values is not coded either. This fulfills the demand of image parameter independence, like image size, image and pixel aspect ratio.

A source image consists of at least one and at most 255 components or planes, as shown on the left side of Figure 6.4. Each component C_i may have a different number of pixels in the horizontal (X_i) and vertical (Y_i) axes. Note, the index denotes the number of the component or plane. These components may be assigned to the three colors *RGB*, *YIQ* or *YUV* signals, for example.

Figure 6.5 shows three components of an image, each with the same resolution, and each having a rectangular array C_i of $X_i \times Y_i$ pixels. The three X_i values and three Y_i values are the same.

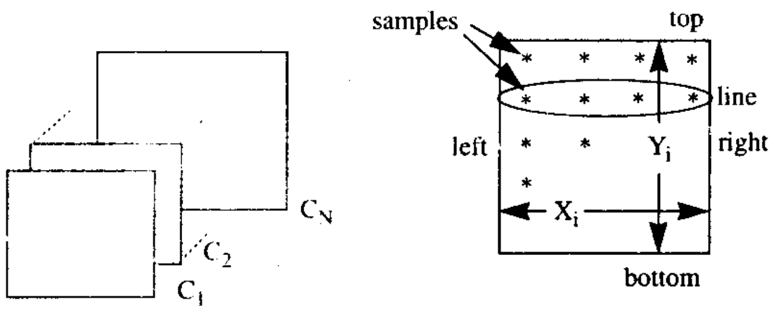

Figure 6.4: *Digital uncompressed still image with the definition of the respective image components according to the JPEG standard.*

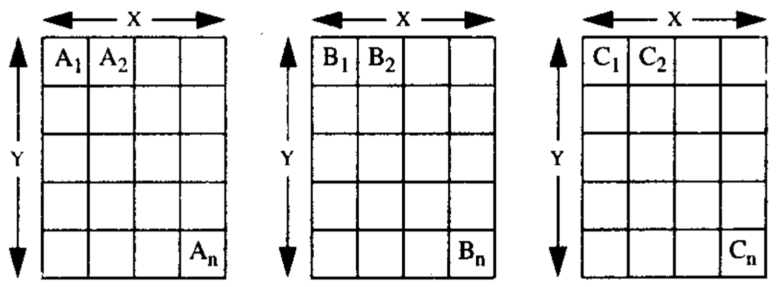

Figure 6.5: *Example of JPEG image preparation with three components having the same resolution.*

The resolution of the individual components may be different. Figure 6.6 shows an image with half the number of columns (i.e., half the number of horizontal samples) in the second and third planes as compared to the first plane: $Y_1 = Y_2 = Y_3$, and $X_1 = 2X_2 = 2X_3$.

A gray-scale image will, in most cases, consist of a single component. An *RGB* color representation has three components with equal resolution (i.e., the same number of lines $Y_1 = Y_2 = Y_3$, and same number of columns $X_1 = X_2 = X_3$). For JPEG, YUV color image processing uses $Y_1 = 4Y_2 = 4Y_3$ and $X_1{=}4X_2{=}4X_3$.

Each pixel is represented by p bits with values in the range of *0* to $2^p - 1$. All pixels of all components within the same image are coded with the same number of bits. The lossy modes of JPEG use a precision of either 8 or 12 bits per pixel. Lossless modes use a precision of 2 up to 12 bits per pixel. If a JPEG application makes use

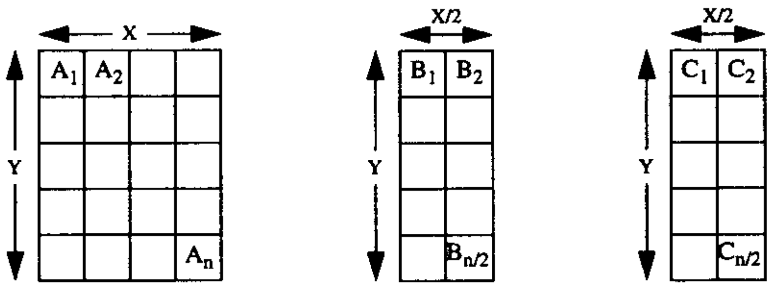

Figure 6.6: *Example of JPEG image preparation with three components having different resolution.*

of any other number of bits, the application itself must perform a suitable image transformation to the well-defined number of bits in the JPEG standard.

The dimensions of a compressed image are defined by new values X (the maximum of all X_i), Y (the maximum of all Y_i), H_i and V_i. H_i and V_i are the relative horizontal and vertical sampling ratios specified for each component i. H_i and V_i must be integer values in the range between 1 and 4. This awkward-looking definition is needed for the interleaving of components described later.

Let us consider the following example, also shown in [Org93]. A picture is given with the maximum horizontal and vertical resolution of 512 pixels and the following sampling factors:

Level 0: $H_0 = 4, V_0 = 1$

Level 1: $H_1 = 2, V_1 = 2$

Level 2: $H_2 = 1, V_2 = 1$

Assuming X = 512, Y = 512, $H_{max} = 4$ and $V_{max} = 2$, this leads to

Level 0: $X_0 = 512, Y_0 = 256$

Level 1: $X_1 = 256, Y_1 = 512$

Level 2: $X_2 = 128, Y_2 = 256$

With the given ceiling functions, X_i and Y_i are calculated as follows: for the use of compression, the image is divided in data units. The lossless mode uses one pixel as one data unit. The lossy mode uses blocks of 8 x 8 pixels. This definition of data units is a result of DCT, which always transforms connected blocks. In most cases, the data units are processed component by component, and passed on, as shown in Figure 6.3, for the image processing step. As shown in Figure 6.7, for one component, the processing order of data units is left-to-right and top-to-bottom, one component after the other. This is known as *non-interleaved data ordering.* Using

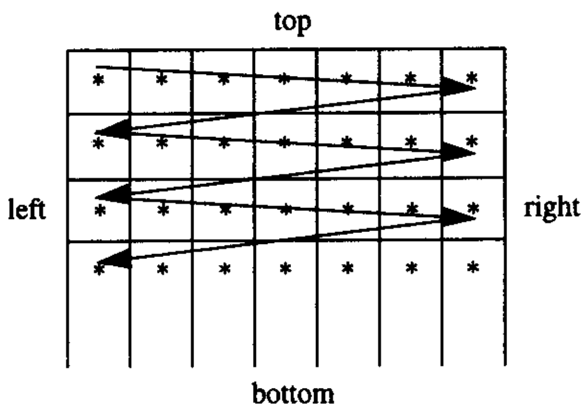

Figure 6.7: *Non-interleaved order of data units, orthe processing of one component according to the JPEG standard.*

this non-interleaved mode for an *RGB*-encoded image with very high resolution, the display would initially present only the red component, then, in turn, the blue and green would be drawn resulting in the original image colors being reconstructed. Due to the finite processing speed of the JPEG decoder, it is therefore often more suitable to interleave the data units as shown in Figure 6.8.

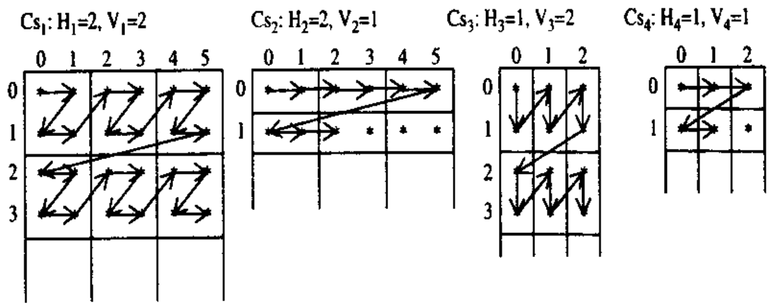

Figure 6.8: *Interleaved data units: an example with four components as derived from the JPEG standard.*

Interleaved data units of different components are combined into *Minimum Coded Units* (MCUs). If all components have the same resolution ($X_i \times Y_i$), an MCU consists of exactly one data unit for 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 resolutions for single components, the construction of MCUs becomes more complex (see Figure 6.8). For each component, the *regions* of the data units (if necessary, with different numbers of data units) are determined. Each component consists of the same number of regions. For example, Figure 6.8 shows six regions for each component. An MCU consists of exactly one region in each component. Again, the data units within one region are ordered left-to-right and top-to-bottom.

Figure 6.8 shows an example with four components. The values of H_i and V_i for each component are provided in the figure. The first component has the highest resolution in both dimensions and the fourth component has the lowest resolution. The arrows indicate the sampling direction of the data units for each component. The 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 $C_{s1} : d^1_{00}...d^1_{31}$

The data units of the second component are $C_{s2} : d^2_{00}...d^2_{11}$

The data units of the third component are $C_{s3} : d^3_{00}...d^3_{30}$

The data units of the fourth component are $C_{s4} : d^4_{00}...d^4_{10}$

According to the JPEG standard, up to four components can be encoded using the interleaved mode. Each MCU consists of at most ten data units. Within an image, some components can be encoded in the interleaved mode and others in the non-interleaved mode.

6.5.2 Lossy Sequential DCT-based Mode

Image Processing

After image preparation, the uncompressed image samples are grouped into data units of 8 × 8 pixels and passed to the encoder; the order of these data units is defined by the MCUs. In this baseline mode, single samples are encoded using p = 8 bits. Each pixel is an integer in the range of 0 to 255.

The first step of image processing in the baseline mode (baseline process), as shown in Figure 6.9, is a transformation performed by DCT [ANR74, NP78]. The pixel values are shifted into the range [-128,127], with zero as the center. These 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. Each of these values is then transformed using *Forward DCT* (FDCT):

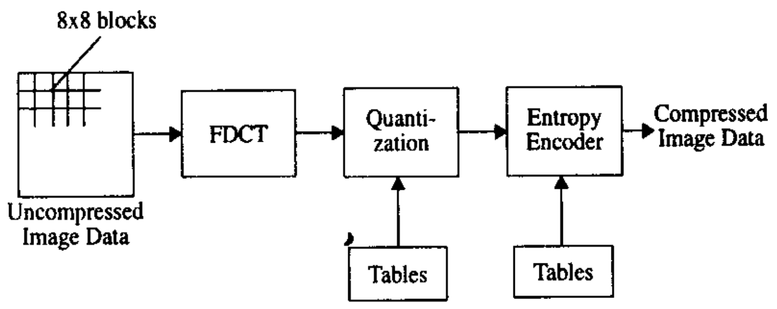

Figure 6.9: *Steps of the lossy sequential DCT-based coding mode.*

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

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 of S_{vu}.

DCT is similar to *Discrete Fourier Transformation* (DFT); it maps the values from the time to the frequency domain. Therefore, each coefficient can be regarded as a two-dimensional frequency.

The coefficient S_{00} corresponds to the lowest frequency in both dimensions. It is known as the DC-coefficient, which determines the fundamental color of the data unit of 64 pixels. The DC-coefficient is the DCT-coefficient for which the frequency is zero in both dimensions. The other coefficients are called AC-coefficients. AC-coefficients are DCT-coefficients for which the frequency in one or both dimensions is non-zero. For instance, S_{70} represents the highest frequency that occurs in the horizontal direction, which is the closest possible separation of vertical lines in the 8× 8 data unit. S_{07} represents the highest frequency in the vertical dimension, i.e., the closest separation of horizontal lines. S_{77} indicates the highest frequency appearing equally in both dimensions. The absolute value of S_{77} is greatest if the source 8 × 8 data unit consists of a full matrix, i.e., with as many 1 × 1 components as possible. One or both dimensions are non-zero. Accordingly, for example, S_{44} will be greatest if the block consists of 16 squares of 4 × 4 pixels. Taking a closer look at the above FDCT formula, we recognize that the cosine expressions only depend upon x and u,

y and v respectively, but do not depend on S_{yx}. Therefore, these cosine expressions represent constants that do not have to be calculated over and over again. There are many effective techniques and implementations of DCT. Important contributions can be found in [DG90, Fei90, Hou88, Lee84, LF91, SH86, VN84, Vet85].

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

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

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

If the FDCT, as well as the IDCT, could be calculated with full precision, it would be possible to reproduce the 64 source pixels exactly. From a theoretical point of view, DCT would be lossless in this case. In practice, precision is restricted and DCT is lossy; however, the JPEG standard does not define any precision. For this reason, two different implementations of a JPEG decoder could generate different images as output of the same compressed image. JPEG merely defines the maximum tolerance.

Most of the areas of a typical image consist of large regions of a single color which, after applying DCT, are represented by many coefficients with very low values. The edges, however, are transformed into coefficients which represent high frequencies. Images of average complexity consist of many AC-coefficients with a value of almost zero. Therefore, entropy encoding is used to achieve considerable data reduction.

Quantization

Following the steps in Figure 6.3, the quantization of all DCT-coefficients is performed. This is a lossy transformation. For this step, the JPEG application provides a table with 64 entries. Each entry will be used for the quantization of one of the 64 DCT-coefficients. Thereby, each of the 64 coefficients can be adjusted separately. The application has the possibility to affect the relative significance of the different coefficients and specific frequencies can be given more importance than others. These coefficients should be determined according to the characteristics of the source image. The possible compression is influenced at the expense of the achievable image quality.

Each table entry is an 8-bit integer value called Q_{vu}. The quantization process becomes less accurate as the size of the table entries increases. Quantization and

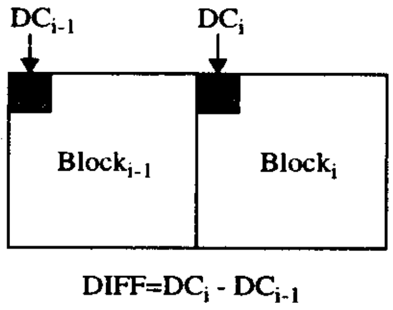

Figure 6.10: ***Preparation of DCT DC-coefficients for entropy encoding, including the calculation of the difference between neighboring values.***

de-quantization must use the same tables. No default values for quantization tables are specified in JPEG; applications may specify values which customize the desired picture quality according to the particular image characteristics.

Entropy Encoding

During the initial step of entropy encoding, the quantized DC-coefficients are treated separately from the quantized AC-coefficients. The processing order of the whole set of coefficients is specified by the *zig-zag sequence* as shown in Figure 6.11.

- The DC-coefficients determine the basic color of the data units. Between adjacent data units the variation of color is fairly small. Therefore, a DC-coefficient is encoded as the difference between the current DC-coefficient and the previous one. Only the differences are subsequently processed (see Figure 6.10).
- The DCT processing order of the AC-coefficients using the zig-zag sequence illustrates that coefficients with lower frequencies (typically with higher values)

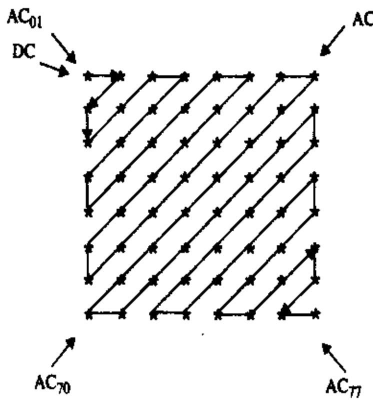

Figure 6.11: *Preparation of DCT AC-coefficient for entropy encoding, in order by increasing frequency.*

are encoded first, followed by the higher frequencies (with typically small, almost zero values). The result is an extended sequence of similar data bytes, permitting very efficient entropy encoding. Note that the arrow between the DC-coefficient and the first AC-coefficient denotes that this DC value has the lowest frequency.

JPEG specifies Huffman and arithmetic encoding as entropy encoding methods. For the lossy sequential DCT-based mode, discussed in this section, only Huffman encoding is allowed. In both methods, a run-length encoding of zero values of the quantized AC-coefficients is applied first. Additionally, non-zero AC-coefficients, as well as the DC-coefficients, are transformed into a spectral representation to compress the data even more. The number of required bits depends on the coefficient's value. A non-zero AC-coefficient will be 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 specifies the following information:

- The number of subsequent coefficients with the value zero.
- The number of bits used for the representation of the coefficient that follows.
- The value of the coefficient represented using the specified number of bits.

The major advantage of Huffman encoding over arithmetic encoding is the free implementation, as the former is not protected by a patent.

Disadvantageous is the fact that the application must provide encoding tables since JPEG does not predefine any of them. This baseline mode allows the use of different Huffman tables for AC- and DC-coefficients.

In the case of *sequential encoding*, the whole image is coded and decoded in a single run. Figure 6.12 shows an example of decoding with immediate presentation; the picture is presented from top-to-bottom.

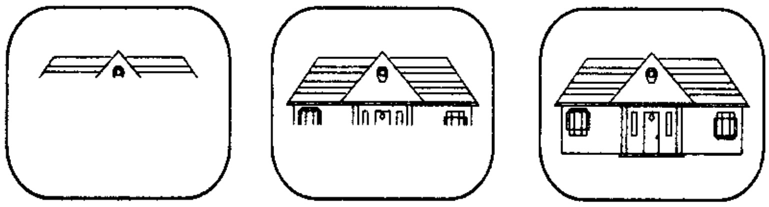

Figure 6.12: *Sequential picture presentation used in the lossy DCT-based mode.*

6.5.3 Expanded Lossy DCT-based Mode

Image preprocessing in this mode differs from the previously described mode in terms of the number of bits per sample. Specifically, a sample precision of 12 bits per sample, in addition to 8 bits per sample, can be used. The image processing is DCT-based and follows rules analogous to the baseline DCT mode.

For the expanded lossy DCT-based mode, JPEG specifies *progressive encoding* in addition to sequential encoding. In the first run, a very rough representation of the image appears which looks out of focus and is refined during successive steps. A schematic example is shown in Figure 6.13.

Progressive image representation is achieved by an expansion of quantization. This is also known as *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:

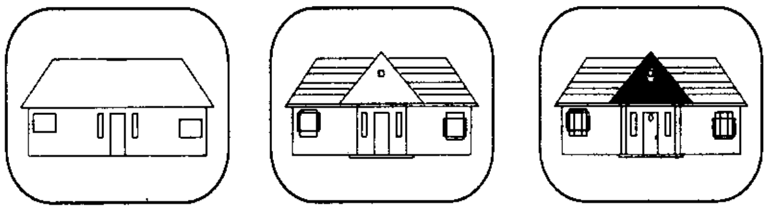

Figure 6.13: *Progressive picture presentation used in the expanded lossy DCT-based mode.*

- By using a *spectral selection* in the first run only, the quantized DCT-coefficients of low frequencies of each data unit are passed to the entropy encoding. In successive runs, the coefficients of higher frequencies are processed.

- *Successive approximation* transfers all of the quantized coefficients in each run, but single bits are differentiated according to their significance. The most-significant bits are encoded first, then the less-significant bits.

Besides Huffman encoding, arithmetic entropy encoding can be used in this mode. The arithmetic encoding requires no tables for the application as it is automatically adapted to the statistical characteristics of an image. According to several publications, the compression achieved by arithmetic coding is sometimes between 5% and 10% better than that achieved by Huffman encoding. Other authors assume a similar compression rate. Arithmetic coding is slightly more complex and its protection by patents must be considered (see [Org93], Appendix L).

Four coding tables for the transformation of DC- and AC-coefficients can be defined by the JPEG application. In a simpler mode, a choice of only two Huffman tables each for the DC- and AC-coefficients of one image is allowed. For this reason, twelve alternative types of processing can be used in this mode (see Table 6.2). The most widely-used display mode is the sequential display mode with 8 bits per sample and Huffman encoding.

Image Display	Bits per Sample	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 6.2: *Types of image processing in the extended lossy DCT-based mode.*

6.5.4 Lossless Mode

The lossless mode shown in Figure 6.14 uses data units of single pixels for image preparation. Any precision between 2 and 16 bits per pixel can be used.

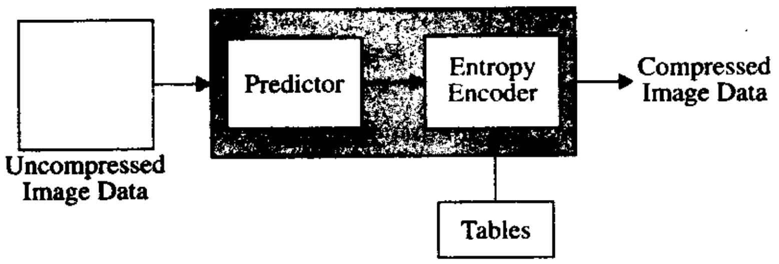

Figure 6.14: *Lossless mode based on a prediction.*

In this mode, image processing and quantization use a predictive technique instead of a transformation encoding technique. As shown in Figure 6.15, for each pixel X, one of eight possible predictors is selected. The selection criterion is a prediction

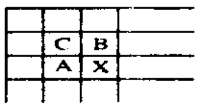

Figure 6.15: *Principle of the prediction in the lossless mode.*

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 6.3: *Predictors for lossless coding.*

that is as good as possible of the value of X from the already known adjacent samples A, B and C. The specified predictors are listed in Table 6.3.

The number of the chosen predictor, as well as the difference of the prediction to the actual value, is passed to the subsequent entropy encoding. Entropy encoding can use either Huffman or arithmetic encoding techniques.

6.5.5 Hierarchical Mode

The hierarchical mode uses either the lossy DCT-based algorithms described above or alternatively the lossless compression technique. The main feature of this mode is the encoding of an image at different resolutions, i.e., the encoded data contains images at several resolutions. The prepared image is initially sampled at a lower resolution (reduced by the factor 2^n). Subsequently, the resolution is reduced by a factor 2^{n-1} vertically and horizontally. This compressed image is then subtracted from the previous result. The process is repeated until the full resolution of the image is compressed.

Hierarchical encoding requires considerably more storage capacity, but the compressed image is immediately available at different resolutions. Therefore, applications working with lower resolutions do not have to decode the whole image and subsequently apply image processing algorithms to reduce the resolution – in other words, scaling becomes cheap. According to the authors' experiences with scaled images in the context of DVI, any scaling performed by the application consumes considerable time. It takes less CPU processing time to display an image with full resolution than to process a scaled-down image and display it with a reduced number of pixels. Yet, in the case of images coded according to the JPEG hierarchical mode, the display of a reduced size picture consumes less processing power than any higher resolution.

6.6 H.261 (px64)

The driving force behind the H.261 (px64) video coding standard is ISDN. The two B-channels of an ISDN connection (or part of them) can be used to transfer video in addition to audio data. This implies that both users connected via the B-channel have to use the same codec for video signals. Note that *codec* means encoder and decoder, i.e., encoding and decoding, compression and decompression. In the case of an ISDN connection, exactly two B-channels and one D-channel are available at the user interface. The European ISDN hierarchy allows a connection with 30 B-channels, which were originally intended for PABX. Here, we use *B-channels* to specify one or more ISDN channels. The prime considered ISDN applications were videophone and video conferencing systems. For these *dialogue applications*, coding and decoding must be carried out in real-time. In 1984, study group XV of CCITT established a committee that worked on this standard for the compression of moving pictures [Lio91].

First, a compressed data stream with a data rate of $m \times 384$ kbits/second (m = 1, 2, ..., 5) was foreseen. Later, a demand for standardization with $n \times 64$ kbits/second (n = 1, 2, ..., 5) arose. Due to advances in video-coding technology and the necessary support of narrowband ISDN, a decision was made in favor of video compression with a data rate of $p \times 64$ Kbits/second (p = 1, 2,..., 30). After five years, CCITT Recommendation H.261 *Video Codec* for Audiovisual Services at $p \times 64$ kbits/second was finalized in December 1990 [ITUC90]. This recommendation is also known as *px64*, because of the compressed data rate of $p \times 64$ kbits/second. North America adopted the recommendation with slight modifications.

The CCITT recommendation H.261 was developed for real-time processing of encoding and decoding. The maximum signal delay of both compression and decompression must not exceed 150 milliseconds. If the end-to-end delay is too long, an application using this technology will be affected considerably.

6.6.1 Image Preparation

Unlike JPEG, H.261 defines a very precise image format. The image refresh frequency at the input must be 29.97 frames per second. During encoding, it is possible to generate a compressed image sequence with a lower frame rate of, for example, 10 or 15 still images per second. Only non-interleaved images are allowed at the input of the coder. The image is encoded as luminance signal (Y) and chrominance difference signals C_b, C_r, according to the CCIR 601 subsampling scheme (2:1:1). Later this was also adopted by MPEG.

Two resolution formats each with an aspect ratio of 4:3 are specified. The so-called *Common Intermediate Format* (CIF) defines a luminance component of 288 lines, each with 352 pixels. The chrominance components have a resolution with a rate of 144 lines and 176 pixels per line to fulfill the 2:1:1 requirement. *Quarter-CIF* (QCIF) has exactly half of the CIF resolution, 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 necessary compression ratio for images with the low QCIF resolution (determined by the bandwidth of an ISDN B-channel) is illustrated by means of the following example. The uncompressed QCIF is composed of a data stream with 29.97 frames per second and a data rate of about 9.115 Mbits/second; for CIF (with the same number of images per second), an uncompressed data rate of about 36.45 Mbits/second is produced. The image should be reduced to a frame rate of 10 pictures per second. This leads to a compression ratio of about 1:47.5, which can be supported with today's technology. Using a CIF format with the same compression ratio a reduction to about the bandwidth of six ISDN B-channels is possible.

In H.261, data units of the size 8 × 8 pixels are used for the representation of the Y, as well as the C_b and C_r components. A macro block is the result of combining four blocks of the Y-matrix each with one block of the C_b and C_r components. A ***group of blocks*** is defined to consist of 33 macro blocks. Therefore, a QCIF-image consists of three groups of blocks, and a CIF-image comprises twelve groups of blocks.

6.6.2 Coding Algorithms

The H.261 standard uses two different methods of coding: *intraframe* and *interframe*. In the case of intraframe coding, no advantage is taken from the redundancy between frames. This coding technique corresponds to the I-frame coding of MPEG (see Section 6.7.1). For interframe coding, information from previous or subsequent frames is used; this corresponds to the P-frame encoding of MPEG (see Section 6.7.1). The H.261 standard does not provide any criteria for mode choice. The decision must be made during the coding process, depending on the specific implementation.

Similar to JPEG, for intraframe encoding, each block of 8 × 8 pixels is transformed into 64 coefficients using DCT. The quantization of DC-coefficients differs from the quantization of AC-coefficients. The next step is to apply entropy encoding to the AC- and DC-parameters, resulting in a variable-length encoded word. ***Interframe coding*** is based on a prediction for each macro block of an image. This is determined by a comparison of macro blocks from previous images and the current image. The motion vector is defined by the relative position of the previous macro block with respect to the current macro block. Note that according to H.261, the coder need not be able to determine a motion vector. Therefore, a simple H.261 implementation considers only the differences between macro blocks located at the same position of subsequent images. In such cases, the motion vector is always a zero vector.

Subsequently, the motion vector and DPCM-coded macro block are processed. The DPCM-coded macro block is transformed by DCT if and only if its value exceeds a certain threshold. If the difference is less than this threshold, the corresponding macro block is not encoded any further – only the respective motion vector is processed. The components of the motion vector are entropy encoded using a lossless variable-length coding system. All of the transformed coefficients are quantized linearly and variable-length encoded.

Additionally, an optical low pass filter can operate between the DCT and entropy encoding process. This filter deletes any remaining high-frequency noise. Note, such a filter is optional and few implementations actually incorporate it.

For H.261, quantization is a linear function and the step size is adjusted according to the amount of data in the transformation buffer. This mechanism enforces a constant data rate at the output of the coder. Therefore, the quality of the encoded video data depends on the contents of individual images, as well as on the motion within the respective video scene.

6.6.3 Data Stream

According to H.261, a data stream has a hierarchical structure composed of several layers, the bottom layer containing the compressed picture. H.261 has the following characteristics (for further details, see [ITUC90]):

- The data stream of an image includes information for error correction.
- For each image, a 5-bit image number is used as a temporal reference.
- If a certain command is passed from the application to the decoder, the image displayed last is *frozen* as a still image. This allows the application at the decoding station to stop/freeze and start/play a video scene without any additional effort.
- Using further commands sent by the encoder (and not by the application), it is also possible to switch between still images and moving images. Alternatively, a time-out signal can also be used instead of this explicit command.

H.261 was designed for conferencing systems and video telephony.

6.7 MPEG

The MPEG standard was developed by ISO/IEC JTC1/SC 29/WG11 to cover motion video as well as audio coding according to the ISO/IEC standardization process. Considering the state of the art in CD-technology digital mass storage, MPEG strives for a data stream compression rate of about 1.2 Mbits/second, which is today's typical CD-ROM data transfer rate. MPEG can deliver a data rate of at most 1856000 bits/second, which should not be exceeded [ISO93a]. Data rates for audio are between 32 and 448 Kbits/second; this data rate enables video and audio compression of acceptable quality. In 1993, MPEG was accepted as the International Standard (IS) [ISO93a] and the first commercially available MPEG products entered the market.

The MPEG standard explicitly considers functionalities of other standards::

- *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 standard, the MPEG standard makes use of JPEG.

- *H.261.* Since the H.261 standard was already available during the work on the MPEG standard, the working group strived for compatibility (at least in some areas) with this standard. Implementations that are capable of H.261, as well as of MPEG, may arise, however, MPEG is the more advanced technique.

MPEG is suitable for symmetric as well as asymmetric compression. Asymmetric compression requires more effort for coding than for decoding. Compression is carried out once, whereas decompression is performed many times. A typical application area is retrieval systems. Symmetric compression is known to expect equal effort for the compression and decompression processes. Interactive dialogue applications make use of this encoding technique, where a restricted end-to-end delay is required.

Besides the specification of video [Le 91, VG91] and audio coding, the MPEG standard provides a *system definition*, which specifies the combination of several individual data streams.

6.7.1 Video Encoding

In contrast to JPEG, but similar to H.261, the image preparation phase of MPEG, according to our reference scheme shown in Figure 6.1, exactly defines the format of an image. Each image consists of three components (similar to the *YUV* format); the luminance component has twice as many samples in the horizontal and vertical axes as the other two components – this is known as *color-subsampling.* The resolution of the luminance component should not exceed 768 x 576 pixels; for each component, a pixel is coded with eight bits.

The MPEG data stream includes more information than a data stream compressed according to the JPEG standard. For example, the aspect ratio of a pixel is included. MPEG provides 14 different image aspect ratios per pixel. The most important are:

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

The image refresh frequency is also encoded in the data stream. Eight frequencies are defined: 23.976 Hz, 24 Hz, 25 Hz, 29.97 Hz, 30 Hz, 50 Hz, 59.94 Hz and 60 Hz.

A temporal prediction of still images leads to a considerable compression ratio. Moving images often contain non-translational moving patterns such as rotations or waves at the seaside. Areas in an image with these irregular patterns of strong motion can only be reduced by a ratio similar to that of intraframe encoding. The use of temporal predictors requires the storage of a great amount of information and image data. There is a need to balance this required storage capacity and the achievable compression rate. In most cases, predictive encoding only makes sense for parts of images and not for the whole image. Therefore, each image is divided into areas called *macro blocks*. Each macro block is partitioned into 16×16 pixels for the luminance component and 8×8 pixels for each of the two chrominance components. These macro blocks turn out to be quite suitable for compression based on motion estimation. This is a compromise of costs for prediction and the resulting data reduction.

Due to the required frame rate, each image must be built up within a maximum of 41.7 milliseconds. From the user's perspective there are no advantages to progressive image display over sequential display. The user has neither the need nor possibility to define the MCUs (Minimum Coded Units) in MPEG (in contrast to JPEG).

MPEG distinguishes four types of image coding for processing, as shown Figure 6.16. The reasons behind this are the contradictory demands for an efficient coding scheme and fast random access. To achieve a high compression ratio, temporal redundancies of subsequent pictures must be exploited (interframe), whereas the demand for fast random access requires intraframe coding. The following types of images are distinguished (*image* is used as a synonym for *still image* or *frame*):

- *I-frames* (*Intra-coded images*) are self contained, i.e., coded without any reference to other images. An I-frame is treated as a still image. MPEG makes use of JPEG for I-frames. However, contrary to JPEG, compression must often be executed in real-time. The compression rate of I-frames is the lowest within MPEG. I-frames are points for random access in MPEG streams.

  I-frames use 8×8 blocks defined within a macro block, on which a DCT is performed. The DC-coefficients are then DPCM coded; differences of successive blocks of one component are computed and transformed using variable-length coding. MPEG distinguishes two types of macro blocks – the first type includes only the encoded data and the second covers a parameter used for scaling by adjustment of the quantization characteristics.

- *P-frames* (*Predictive-coded frames*) require information of the previous I-frame

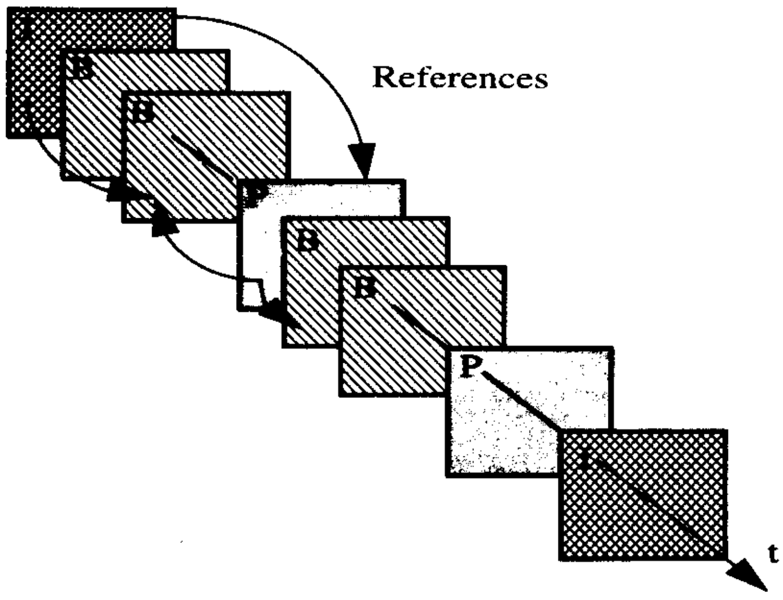

Figure 6.16: *Types of images in MPEG.*

and/or all previous P-frames for encoding and decoding.

The coding of P-frames is based on the fact that, by successive images, their areas often do not change at all but instead, the whole area is shifted. In this case of ***temporal redundancy***, the block of the last P- or I-frame that is most similar to the block under consideration is determined. Several methods for motion estimation are available to the encoder. The most processing-intensive methods tend to give better results, so the following trade-offs must be made in the encoder: computational power, and hence cost, versus video quality [ISO93a]. Several matching criteria are available, e.g., the differences of all absolute values of the luminance component are computed. The minimal number of the sum of all differences indicates the best matching macro block. Thereby, MPEG does not provide a certain algorithm for motion estimation, but instead specifies the coding of the result. Only the motion vector (the difference between the spatial location of the macro blocks) and the small difference in content of these macro blocks are left to be encoded. The search range, i.e., the maximum size of the motion vector, is not defined in the standard, but it is constrained by the definable motion vector range. The larger the search range the better the motion estimation, although the computation is slower.

Like I-frames, P-frames consist of I-frame maçro blocks and six predictive macro blocks. The coder must determine if a macro block should be coded predictively or as a macro block of an I-frame, and furthermore, if there is a motion vector that must be encoded. A P-frame can contain macro blocks that are encoded using the same technique as I-frames. The coder for specific macro blocks of P-frames must consider the differences of macro blocks, as well as the motion vector. The difference of all six 8×8 pixel blocks of the best matching macro block and the macro block to be coded are transformed using a two-dimensional DCT. For further data rate reduction, blocks that only have DCT-coefficients with all values of zero are not processed further. These are stored using 6-bit values, which are added to the encoded data stream. Subsequently, the DC- and the AC-coefficients are encoded using the same technique. Note that this differs from JPEG and from the coding of macro blocks of I-frames. In the next step, a run-length encoding and the determination of a variable-length code (not according to Huffman, but similar) is applied. Since the motion vectors of adjacent macro blocks often differ only slightly, DPCM encoding is used. The result is again transformed using a table leading to a variable-length encoded word.

- *B-frames* (*Bi-directionally predictive-coded frames*) require information of the previous and following I- and/or P-frame for encoding and decoding. The highest compression ratio is attainable by using these frames. A B-frame is defined as the difference of a prediction of the past image and the following P- or I-frame. B-frames can never be directly accessed in a random fashion.

  For the prediction of B-frames, the previous as well as the following P- or I-frames are taken into account. The following example illustrates the advantages of a *bi-directional prediction.* In a video scene, 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 the former image were covered by the ball. A prediction of these areas can be derived from the following but not from the previous image. A macro block may be derived from the previous or the next macro block of P- or I-frames. Apart from a motion vector from the previous to the next image, a motion vector in the other direction can also be used. *Interpolative* motion compensation that uses both matching macro blocks is allowed. In this case, two motion vectors are encoded. The difference of the macro block to be encoded and the interpolated macro block is determined. Further quantization and entropy encoding are performed like P-frame specific macro blocks. B-frames must not be stored in the decoder as a reference for subsequent decoding of images.

- *D-frames* (*DC-coded frames*) are intraframe-encoded. They can be used for fast forward or fast rewind modes. The DC-parameters are DCT-coded; the AC-coefficients are neglected.

  D-frames consist only of the lowest frequencies of an image. They only use one type of macro block and only the DC-coefficients are encoded. D-frames are used for display in fast-forward or fast-rewind modes. This could also be realized by a suitable order of I-frames. For this purpose, I-frames must occur periodically in data stream. Slow-rewind playback requires huge storage capacity. Therefore, all images that were combined in a group must be decoded in the forward mode and stored, after which a rewind playback is possible. This is known as the *group of pictures* in MPEG.

Figure 6.16 shows a sequence of I-, P- and B-frames. For example, the prediction for the first P-frames and a bi-directional prediction for a B-frame is shown. Note that by using B-frames the order of the images in a MPEG-coded data stream often differs from the actual decoding order. A P-frame to be displayed after the related B-frame must be decoded before the B-frame because its data is required for the decompression of the B-frame. This fact introduces an additional end-to-end delay.

The regularity of a sequence of I-, P- and B-frames is determined by the MPEG application. For fast random access, the best resolution would be achieved by coding the whole data stream as I-frames. On the other hand, the highest degree of compression is attained by using as many B-frames as possible. For practical applications, the following sequence has proved to be useful, "IBBPBBPBB IBBPBBPBB ..." In this case, random access would have a resolution of nine still images (i.e., about 330 milliseconds), and it still provides a very good compression ratio.

Concerning quantization, it should be mentioned that AC-coefficients of B- and P-frames are usually large values, whereas those of I-frames are smaller values. Thus, the MPEG quantization is adjusted respectively. If the data rate increases over a certain threshold, the quantization enlarges the step size. In the opposite case, the step size is reduced and the quantization is performed with finer granularity.

6.7.2 Audio Encoding

MPEG audio coding uses the same sampling frequencies as Compact Disc Digital Audio (CD-DA) and Digital Audio Tape (DAT), i.e., 44.1 kHz and 48 kHz, and additionally, 32 kHz is available, all at 16 bits.

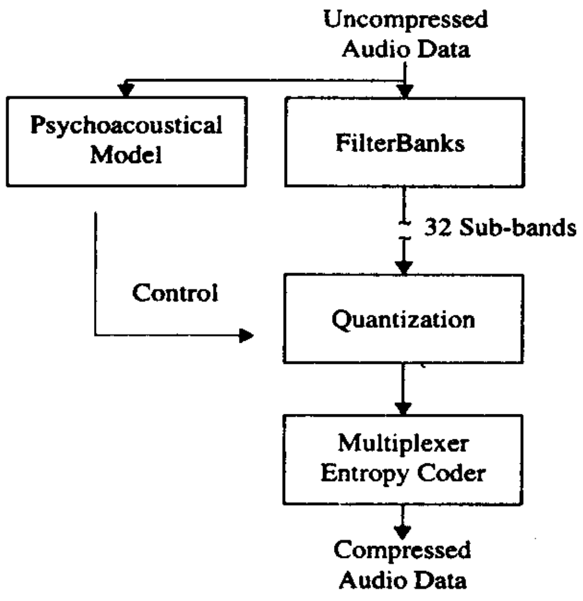

Figure 6.17: *MPEG basic steps of audio encoding.*

Three different layers (Figure 6.17) of encoder and decoder complexity and performance are defined. 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. *Fast Fourier Transformation* (FFT) is suitable for this coding, and the spectrum is split into 32 non-interleaved subbands. For each subband, the amplitude of the audio signal is calculated. Also for each subband, the noise level is determined simultaneously to the actual FFT by using a *psychoacoustic model*. At a higher noise level, a rough quantization is performed, and at a lower noise level, a finer quantization is applied. The quantized spectral portions of layers one and two are PCM-encoded and those of layer three are Huffman-encoded. The audio coding can be performed with a single channel, two independent channels or one stereo signal. In the definition of MPEG, there are two different stereo modes: two channels that are processed either independently or as *joint stereo*. In the case of joint stereo, MPEG exploits redundancy of both channels and achieves a higher compression ratio.

Each layer defines 14 fixed bit rates for the encoded audio data stream, which in MPEG are addressed by a bit rate index. The minimal value is always 32 Kbits/second. These layers support different maximal bit rates: layer 1 allows for a maximal bit rate of 448 Kbits/second, layer 2 for 384 Kbits/second and layer 3 for 320 Kbits/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 switching the bit rate index. For layer 2, not all combinations of bit rate and mode are allowed:

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

6.7.3 Data Stream

Audio Stream

MPEG specifies a syntax for the interleaved audio and video data streams. An audio data stream consists of frames, which are divided into audio access units. Each audio access unit is composed of slots. At the lowest complexity (layer 1), a slot consists of four bytes. In any other layer, it consists of one byte. A frame always consists of a fixed number of samples. Most important is the *audio access unit*, which is the smallest possible audio sequence of compressed data that can be completely decoded independent of all other data. The audio access units of one frame lead to a playing time of 8 milliseconds at 48 kHz, of 8.7 milliseconds at 44.1 kHz, and 12 milliseconds at 32 kHz. In the case of stereo signals, data from both channels are merged into one frame.

Video Stream

A video data stream is comprised of six layers:

1. At the highest level, the *sequence layer*, data buffering is handled. A data stream should have low requirements in terms of storage capacity. For this reason, at the beginning of the *sequence layer* there are the following two entries: the constant bit rate of the sequence and the storage capacity that is needed for decoding. In the processing scheme, a *video-buffer-verifier* is inserted after the quantizer. The resulting data rate is used to verify the delay caused by decoding. The *video-buffer-verifier* influences the quantizer and forms a kind of control loop. Several successive sequences could have a varying data rate. During decoding of several immediately following sequences there is no direct relationship between the end of one sequence and the beginning of the next one. The basic parameters of the decoder are set again and an initialization is executed at this time.

2. The *group of pictures layer* is the next layer. This layer consists of a minimum of one I-frame, which is the first frame. Random access to this image is always possible. At this layer, it is possible to distinguish the order of images in a data stream and during display. The first image of a data stream always has to be an I-frame. Therefore, the decoder decodes and stores the reference frame first. In the order of display, a B-frame can occur before an I-frame.

   Display Order:

Type of Frame	B	B	I	B	B	P	B	B	P	B	B	P
Number of Frame	0	1	2	3	4	5	6	7	8	9	10	11

Decoding order:

Type of Frame	I	B	B	P	B	B	P	B	B	P	B	B
Number of Frame	2	0	1	5	3	4	8	6	7	11	9	10

3. The *picture layer* contains a whole picture. The temporal reference is defined by an image number. Note that there are data fields defined in this layer which are not yet used in MPEG. The decoder is not allowed to use these data fields, as they are designated for future extensions.

4. The next layer is the *slice layer*. Each slice consists of a number of macro blocks that may vary from one image to the next. Additionally, the DCT quantization of each macro block of a slice is specified.

5. The fifth layer is the *macro block layer*. It contains the sum of the features of each macro block as described above.

6. The lowest layer is the *block layer* (described above).

The MPEG standard also specifies the combination of data streams into a single data stream in the system definition. The same idea was pursued in DVI to define the AVSS (Audio/Video Support System) data format. The most important task of this process is the actual multiplexing. It includes the coordination of input data streams and output data streams, the adjustment of clocks and buffer management. Therefore, the data stream defined by ISO 11172 is divided into single *packs*. The decoder gets the information necessary for its resource reservation from this multiplexed data stream. The maximal data rate is included in the first pack at the beginning of each ISO 11172 data stream. The definition of this data stream makes the following implicit assumption: for data stored on a secondary storage medium, it is possible to read such a header first (if necessary, by random access). In dialogue services like telephone or videophone applications using communication networks, the user will always get the header information first. In a conferencing application, using an MPEG stream might be inconvenient because a new user might like to join an existing conference after the data streams have already been setup. Therefore, the necessary header information would not be directly available to her/him.

For a data stream generated according to ISO 11172, MPEG provides time stamps that are necessary for synchronization. They refer to the relationship between multiplexed data streams, but not between other existing ISO 11172 data streams.

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

6.7.4 MPEG-2

The quality of a video sequence compressed according to the MPEG standard is near the target maximum data rate of about 1.5 Mbits/second. This optimum is in quality and not in performance. Further developments in the area of video coding techniques are based on a target rate of up to 40 Mbits/second; this is known as *MPEG-2* [ISO93b]. MPEG-2 strives for a higher resolution, similar to the digital video studio standard CCIR 601 and leading towards the video quality needed in HDTV. Note that most of the following information on MPEG-2 and MPEG-4 was gleaned by the authors from press releases and many personal communications with members of the MPEG expert group [Liu93].

To ensure that a harmonized solution to the widest range of applications is achieved, the ISO/IEC working group designated ISO/IEC JTC1/SC29/WG11, has been working jointly with ITU-TS Study Group 15 *Experts Group for ATM Video Coding*. MPEG-2 also collaborates with representatives from other parts of ITU-TS, EBU, ITU-RS, SMPTE and the North American HDTV community.

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

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

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 576 lines	≤ 15 Mbit/s	≤ 15 Mbit/s	≤ 15 Mbit/s		≤ 20 Mbit/s
Low Level 352 pixels/line 288 lines		≤ 4 Mbit/s	≤ 4 Mbit/s		
LAYERS and PROFILES	**Simple Profile** No B-frames 4:2:0 Not Scalable	**Main Profile** B-frames 4:2:0 Not Scalable	**SNR Scalable Profile** B-frames 4:2:0 SNR Scala-ble	**Spatially Scalable Profile** B-frames 4:2:0 SNR Scalable or Spatial Scalable	**High Profile** B-frames 4:2:0 or 4:2:2 SNR Scalable or Spatial Scalable

Table 6.4: *MPEG-2 profiles and levels with their most important characteristics. Note that cells in the table without entries are not defined as compliance points (adapted from [Scha93]).*

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/HDTV, packet-network video systems, backward-compatibility with existing standards (MPEG-1 and H.261) and other applications for which multilevel coding is required. For example, such a system could give the consumer the option of using either a small portable receiver to decode standard definition TV, or a larger fixed receiver to decode HDTV from the same broadcast signal.

All profiles are arranged in a 5 × 4 matrix as shown in Table 6.4. The horizontal axis denotes profiles with an increasing number of functionalities to be supported. 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, 288 lines/frame and 30 frames/second, in which B-frames are allowed to occur and a data rate not to exceed 4 Mbits/second; the *Main Profile* in the *High Level* specifies 1920 pixels/line, 1152 lines/frame and 60 frames/second with a data rate not to exceed 80 Mbits/s.

MPEG-2 considers a structure similar to that of the hierarchical mode of JPEG. The hierarchy consists of the compressed motion images scaling, i.e., video is encoded at different *qualities* [Lip91, GV92]. The scaling may act on the following different parameters:

- Spatial scaling facilitates decompression of image sequences with dissimilar horizontal and vertical resolutions. A single data stream could include, for example, images with 352×288 pixels (H.261 CIF format), 360×240 pixels, 704×576 pixels (a format according to CCIR 601) and, for example, 1250 lines at an aspect ratio of 16:9 (European HDTV). These resolutions refer to the luminance component; the chrominance components are subsampled with ratio 1:2. This can be implemented using a pyramid for the level of the DCT-coefficients [GV92]. Thereby, an 8×8 DCT, 7×7 DCT, 6×6 DCT and other transformations can be performed. From the technical point of view, only steps with a factor of two are useful.
- Scaling the data rate allows for playback at a lower frame rate or for fast-forward at a constant frame rate. In MPEG-1 this is defined by using D-frames. D-frames are not allowed in MPEG-2. Hence, fast-forward can be realized using I-frames if there is a suitable distribution of them within the data stream of the entire video clip and not only a group of pictures.
- Scaling in amplitude can be interpreted as a different resolution of different pixels or a different quantization of the DCT-coefficients. This leads to layered coding and to the possibility of progressive image presentation. Progressive coding is not at all important for the presentation of video data. However, it should be possible to extract certain images out of a sequence as a still image from the data stream, in which case progressive coding may be of interest. Layered coding can also be used for data partitioning to transmit more important data with better error correction than less important data.

Scaling is an essential extension from MPEG-1 to MPEG-2. Additionally, MPEG-2 considers current developments in the broadband ISDN world. *Asynchronous Transfer Mode* (ATM) is the realization of B-ISDN, based on the transfer of small packets known as cells. The potential loss of single ATM cells, containing MPEG-2 encoded data, is taken into account in MPEG-2 development. In this case, effects on other images and parts of the video data stream must be minimized. It should also be possible to define sequences of different types of images (I-,P-,B-frames) that minimize the end-to-end delay for a given target data rate.

MPEG group developed the *MPEG-2 Audio Standard* for low bit rate coding of multichannel audio. MPEG-2 audio coding 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 *stereo and mono coding* of the MPEG-1 Audio Standard to half sampling-rates (16 kHz, 22.05 kHz and 24 kHz) to improve quality for bit rates at or below 64 kbits/second per channel.

The *MPEG-2 Audio Multichannel Coding Standard* provides backward-compatibility with the existing MPEG-1 Audio Standard. MPEG organized formal subjective testing of the proposed MPEG-2 multichannel audio codecs and up to three non-backward-compatible codecs. These codecs work with rates ranging from 256 to 448 Kbits/second.

Note that to provide a very accurate description, in following text, the notation and terminology according to the original MPEG-2 specification is used. MPEG-2 addresses video as well as associated audio; it provides the MPEG-2 system with a definition of how audio, video and other data are combined into single or multiple streams which are suitable for storage and transmission. Therefore, it imposes syntactical and semantical rules which are necessary and sufficient to synchronize the decoding and presentation of the video and audio information, while ensuring that coded data buffers in the decoder do not overflow or underflow. The streams include time stamps concerning the decoding, presentation and delivery of these data.

In the first step, the basic multiplexing approach adds to each individual stream system-level information and each individual stream is packetized to produce the *Packetized Elementary Stream* (PES). In the subsequent step, the PESs are combined to form a Program or Transport Stream. Both streams are designed to support a large number of known and anticipated applications, and they retain a significant amount of flexibility as may be required, while providing interoperability between different device implementations.

- The *Program Stream* is similar to the MPEG-1 stream. It is aimed at a relatively error-free environment. The Program Stream's packets may be of variable length. The timing information in this stream can be used to implement a constant end-to-end delay (covering the path from the input of the encoder to the output of the decoder).
- The *Transport Stream* combines the PESs and one or several independent time bases into a single stream. The Transport Stream is designed for use in lossy or noisy media. The respective packets are 188 bytes long, including the 4-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, and also for storage on digital video tape and other devices.

A conversion between the Program and Transport Stream is possible and reasonable. Note that the MPEG-2 specification of its buffer management constrains the end-to-end delay to below one second for audio and video data, a value which is too high (i.e., not humanly acceptable) for applications in the dialogue mode.

A typical MPEG-2 video stream has a variable bit rate. With the use of a video buffer as specified in this standard, it is possible to also enforce a constant bit rate leading to varying quality.

MPEG-2 standard, at the CD (Committee Draft) status in late 1993, required 3 months to become a DIS (Draft International Standard) and then required a six-month ballot period before becoming an IS (International Standard). Originally, there were plans to specify a MPEG-3 standard approaching HDTV. However, during the development of the MPEG-2 standard, it was found scaling up could adequately meet HDTV requirements. Subsequently, MPEG-3 was dropped.

6.7.5 MPEG-4

Work on another MPEG initiative for very low bit rate coding of audio-visual programs started in September 1993 at ISO/IEC JTC1. It is scheduled to achieve CD status in 1995 or 1996.

This work will require the devélopment of fundamentally new algorithmic techniques, including 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).

6.8 DVI

Digital Video Interactive (DVI) is a technology that includes coding algorithms. The fundamental components are a VLSI chip set for the video subsystem, a well-specified data format for audio and video files, an application user interface to the audio-visual kernel (AVK, the kernel software interface to the DVI hardware) and compression, as well as decompression, algorithms [HKL+91, Lut91, Rip89]. In this section, we will concentrate mainly on compression and decompression. DVI can process ***data, text, graphics, still images, video*** and ***audio.*** The original essential characteristic was the asymmetric technique of video compression and decompression known as Presentation-Level Video (PLV).

DVI has a very interesting history, an understanding of which helps to explain some of its current features. DVI stated as a project at the David Sarnoff Research Center of the RCA company in Princeton in 1984. At that time, the major goals – to compress video and audio at the data rate appropriate for a CD, and to decompress it in real-time – were defined. In 1986, the first draft of a DVI-specific chip using a ***silicon compiler*** was developed. Also in 1986, General Electrics (GE) took over this technology. The DVI development team became employees of GE and the project continued. The first public presentation took place at the second *Microsoft CD-ROM Conference* in Seattle in March 1987. For the first time, the real-time play-back of video stored on a CD-ROM was demonstrated. In 1989 at the fourth *Microsoft CD-ROM Conference*, IBM and Intel announced their cooperation concerning DVI. The DVI team was later taken over by Intel. The first generation of PS/2 boards were introduced as *ActionMedia 750.* In April 1992, the second generation of these boards for Microchannel and ISA bus machines (*ActionMedia II*) became available. In 1993, the software-only decoder became available as the product Indeo.

Concerning audio, the demand for a hardware solution that can be implemented at a reasonable price is met by using a standard signal processor. Processing of still images and video is performed by a video processor. The video hardware of a DVI board is shown in Figure 6.18. It consists of two VLSI chips containing more than

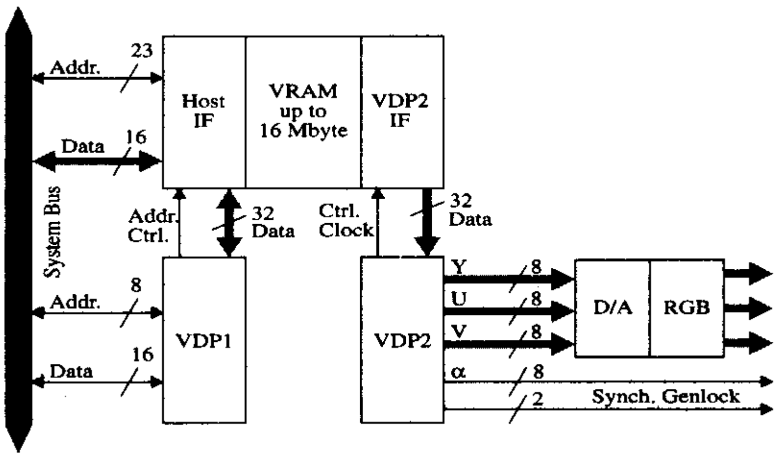

Figure 6.18: *DVI video processing according to [Rip89].*

265,000 transistors each. This Video Display Processor (VDP) consists of the pixel processor VDP1 and the display processor VDP2. VDP1 processes bitmaps and is programmed in microcode; VDP2 generates analog *RGB* signals out of the different bitmap formats and its configuration is also programmable. The coupling of the processors is carried out by the Video-RAM (VRAM). An important characteristic is the capability for microprogramming. It allows one to change and adapt the compression and decompression algorithms to new developments without investing in new hardware.

6.8.1 Audio and Still Image Encoding

Audio signals are digitized using 16 bits per sample and are either PCM-encoded or compressed using the Adaptive Differential Pulse Code Modulation (ADPCM) technique. Thereby, a reduction to about four bits per sample is achieved at a quality corresponding to stereo broadcasting. Different sampling frequencies are supported (11025 Hz, 22050 Hz and 44100 Hz for one or two PCM-coded channels; 8268 Hz, 31129 Hz and 33075 Hz for ADPCM for one channel).

When encoding still images, different video input formats can be used. These can be composite, as well as component video signals like *RGB*. In the case of an *RGB* signal, the color of each pixel is split into portions of the three colors of the spectrum – red, green and blue – and each color is processed separately.

For image preparation, DVI assumes an internal digital *YUV* format, i.e., any video input signal must first be transformed into this format. Note, by DVI we refer to the *Action Media II* version. The color of each pixel is split into a luminance component *Y* and the two chrominance components *U* and *V*. The luminance represents a gray scale image. White is not a basic color but a mixture of colors. In the case of an *RGB* signal, a pure white pixel consists of about 30% red, 59% green and 11% blue.

Starting with an *RGB* signal, DVI computes the *YUV* signal using the following relationship:

$$Y = 0.30\ R + 0.59\ G + 0.11\ B,\ U = B - Y,\ V = R - Y$$

leading to:

$U = -0.30\ R - 0.59\ G + 0.89\ B$

$V = 0.70\ R - 0.59\ G - 0.11\ B$

Therefore, DVI determines the components *YUV* according to the following relation:

$Y = 0.299\ R + 0.587\ G + 0.144\ B + 16$

$U = 0.577\ B - 0.577\ Y + 137.23$

$V = 0.730\ R - 0.730\ Y + 139.67$

This is realized in software using fixed-point arithmetic based on the following:

$109\ Y = 32\ R + 64\ G + 16\ B + 1744$ *with* $0\ R,G,B\ 255$

$111\ U = 64\ B - 64\ Y + 15216$ *with* $16\ Y\ 235$

88 V = 64 R - 64 Y + 12253 with 16 U, V 240

DVI always combines all chrominance components of 4×4 pixel blocks into a single value. The chrominance component of the top left pixel of such a block is used as the reference value for the 16 pixels. Therefore, each pixel has an 8-bit value for the luminance Y, and for 16 related pixels, a single 8-bit chrominance value U, and another 8 bits defining the value for V. The result is a 9-bit YUV format.

To increase image quality during presentation, an interpolation technique is applied to the chrominance values of adjacent blocks. Note that this is the reason for the recognizable color distortion at the right and at the bottom edges of the images. Additionally, DVI is able to process images in the 16-bit YUV format and the 24-bit YUV format. The 24-bit format uses 8 bits for each component. The 16-bit YUV format codes the Y component of each pixel with 6 bits and the color difference components with 5 bits each. This is the reason for two different bitmap formats, *planar* and *packed*. For the planar format, all data of the Y component are stored first, followed by all U component values and then all V values (9-bit YUV format or 24-bit YUV format). For the packed bitmap format, the Y, U and V information of each pixel is stored together followed by the data of the next pixel (16-bit YUV format).

Single images in the 24-bit format can be stored immediately or transmitted. Images in the 16-bit format can be compressed using a lossless algorithm which is known as *PIC 1.0*. In 9-bit format, there is a choice between a lossy algorithm and JPEG baseline mode. For backward compatibility to previous DVI algorithms, two additional compression schemes can be applied.

6.8.2 Video Encoding

For motion video encoding DVI distinguishes two techniques with different resolutions and dissimilar goals:

- *Presentation-Level Video (PLV)* is characterized by its better quality. This is achieved at the expense of a very time-consuming asymmetric compression performed by specialized compression facilities. In the early stages of DVI technology, PLV compression required, for example, a *Meico Transputer System* with more than 60 transputers to compress one image in three seconds, which corresponds to a 90-fold increase in the time needed for such an operation compared to real-time constraints. PLV is suitable for applications distributed on CD-ROMs. The development of such DVI applications using PLV mode follows the process shown in Figure 6.19.

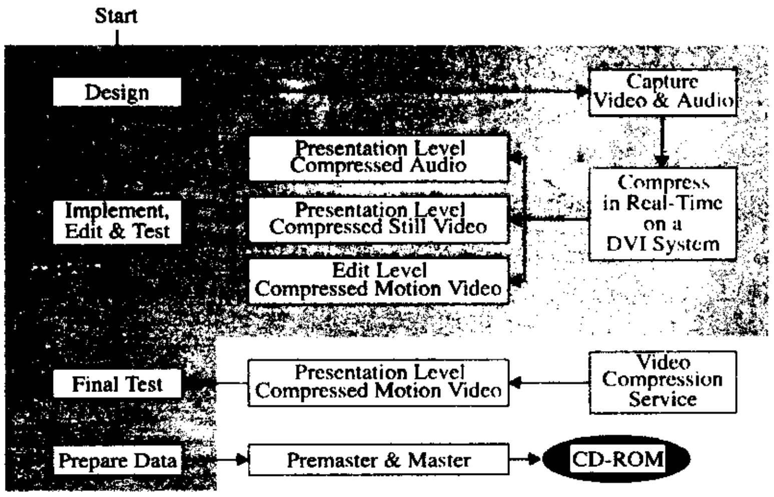
Start
Design
Implement, Edit & Test
Final Test
Prepare Data
Presentation Level Compressed Audio
Presentation Level Compressed Still Video
Edit Level Compressed Motion Video
Presentation Level Compressed Motion Video
Premaster & Master
Capture Video & Audio
Compress in Real-Time on a DVI System
Video Compression Service
CD-ROM

Figure 6.19: *DVI generation process of a PLV-coded video sequence as shown in [Rip89].*

- *Real-Time Video (RTV)* is a symmetric compression technique that works with hardware and software and can be performed in real-time (RTV version 2.0 uses the i750PB chip). Known as *Indeo,* it can also run in real-time on processors such as the Intel 386/486, with certain limitations such as reduced image quality. In previous versions, RTV was known as *Edit-Level Video (ELV).* ELV was conceived to enable the developers of DVI applications to see their video sequences with reduced quality during the construction phase. Afterwards, they would send the videotapes to a DVI compression facility to be compressed in the PLV mode, and would get in return compressed video sequences of higher quality than RTV. Today, RTV is most often used for interactive communication in the same manner as px64.

With respect to the compression steps shown in Figure 6.1, we can again distinguish the various steps:

The image preparation phase of RTV distinguishes three components of an image, in which all pixels are coded with eight bits each. As subsampling is used, the luminance has a higher resolution than the chrominance components. For each 16 pixels of luminance, there is one pixel in each of the chrominance components U and V. Consequently, the luminance component consists of four times the number of lines and columns as the other two components. In a block of 16 Y pixels, one U pixel and one V pixel are encoded together. The result is the 9-bit YUV format of RTV (for each 8 Y bits, there is one U bit or one V bit).

It should be mentioned that the RTV algorithms described below may also make use of other image preparation schemes which would have to be supported by the *AVK (Audio Video Kernel)*. The following processing in the RTV algorithm treats all three components under the same scheme.

RTV image processing distinguishes between interframe coding and intraframe coding as follows:

- *Intraframe coding* is based on individual images. The difference between the value of each pixel and the adjacent pixel above is calculated. For the first line, a fictitious line above is used, which has a constant value. This calculation is performed for all components, and it results in many zero values, which is excellent for the consecutive entropy coding steps.

- *Interframe coding* determines the difference between the value of a pixel in the current image and the value of the pixel located at the same place in the preceding image. This is performed for all three components – normally the differences also consist of many zero values.

A quantization is not necessary because of the simple subtraction operations mentioned above. The entropy encoding is based on a linear data stream. It immediately follows the calculation of differences, and is used for both interframe and intraframe coding. A distinction between run-length encoded data bytes (zero bytes) and the remaining vector is made:

- Sequences of existing zero bytes are compressed using run-length coding.
- All other bytes are compressed using a two-dimensional vector-encoding technique. An index into one of the eight available tables is determined that corresponds to two adjacent pixels to be compressed. The different components of an image are typically encoded using different tables.

In a last step, the run-length encoded values and the indices previously determined are transformed according to another table and subsequently Huffman encoded. It is possible to select a new Huffman table that is adapted to the specific content of each image.

PLV is an asymmetric compression technique that is proprietary to Intel and not published in detail [HKL+91]; however, its fundamental principles are well-known. Each picture is divided into rectangular blocks and processed using motion compensation. For each block, a prediction in the form of a block of the previous image is determined. If its position has changed, its motion is recorded in a motion vector. An exceptional feature should be mentioned: the motion vector is measured in terms of pixels, but its values can be real numbers. The interpolation between the values of pixels can result in a motion vector with values of fractions of pixel widths and heights, leading to better resolution. However, the disadvantage is the penalty (computational cost) caused by processing real numbers instead of integers.

The coding of the difference of the predicted block and the actual block of the previous picture is performed as previously described (i.e., two-dimensional vector encoding and Huffman encoding).

6.8.3 Data Stream

Besides the actual compression technique, DVI defines a data stream. For example, when a data stream including audio and PLV-encoded video is used, a subdivision into single images is the first step. In addition to the actual image data, the following information is included:

- Version label.

- Information on the choice of interframe or intraframe encoding.
- Height and width of the image (number of pixels).
- Information on which of the eight tables must be used for the two-dimensional vector encoding of this image.
- Huffman tables.
- Information if half resolution in vertical and/or horizontal dimension is used.

Additionally, PCM- or ADPCM-encoded audio data are included in the stream.

6.9 Comments

All of the important compression techniques used in multimedia systems turn out to be combinations of different known algorithms.

JPEG must be considered the future standard for the coding of still images. It incorporates a remarkable variety of alternative modes with a high degree of freedom. For example, there could be up to 255 components or planes, an image may comprise up to 65535 lines and each line can have up to 65525 pixels. As a measure of efficiency, the required bits per pixel can be composed, i.e., the average value determined by the ratio of the number of encoded bits and the number of pixels of the image. The following statements apply to DCT-encoded still images [Wal91]:

- 0.25 to 0.50 bits/pixel – moderate to good quality; sufficient for some applications.
- 0.50 to 0.75 bits/pixel – good to very good quality; sufficient for many applications.
- 0.75 to 1.50 bits/pixel – excellent quality suitable for most applications.
- 1,50 to 2.00 bits/pixel – in most cases, not distinguishable from the original; sufficient for almost all applications, even for highest quality requirements.

In lossless mode, a compression ratio of 2:1 is achieved despite the remarkable simplicity of the technique. Today, JPEG is available as a software, as well as hardware solution. However, in most cases, only the baseline mode and predefined image formats are supported. JPEG is often used in multimedia applications which require high quality. The primary goal is to compress an image. However, it is also used as *Motion JPEG* for video compression in applications such as medical imaging.

H.261 is an established standard strongly supported by telecommunication operators. Due to the very restricted resolution in the QCIF format and reduced frame rates, the implementation of H.261 coders and decoders is possible with few, if any technical problems. This is certainly true if motion compensation and the optical low-pass filter are not components of the implementation, though the quality is not always satisfactory in this case. If the image is encoded in CIF format at 25 or 30 images per second using motion compensation, the quality is acceptable. H.261 is mostly used in applications with the dialog mode in a networked environment: video telephony and conferencing. The resulting continuous bit rate is eminently suitable for today's wide area networks operating with ISDN and leased lines.

MPEG is the most promising standard for future compressed digital video and audio. While the JPEG group has a system that can also be used for video, it is too oriented towards *animating* stills rather than using the properties of motion pictures. Currently, the quality of MPEG video can be compared to VHS video recordings, i.e., a data rate of 1.2 Mbits/second, appropriate for CD-ROM drives. The compression algorithm is very good for a resolution of 360 × 240 pixels. Obviously, higher resolutions can also be decoded (e.g., a resolution of 625 lines), but the quality is affected. The future of MPEG points towards MPEG-2, which defines a data stream compatible with MPEG-1, but provides data rates up to 40 Mbits/second. This substantially improves the currently available quality of MPEG-coded data. MPEG also defines an audio stream, with various sampling rates, up to DAT quality of 16 bits/sample. One other important part of the MPEG group's work is the definition of a data stream syntax, which has proved to be relatively successful for DVI technology. Further, MPEG was optimized by making use of the retrieval model in application areas such as tutoring systems based on CD-ROMs and interactive TV. This optimization embedded in MPEG-2 will allow TV and HDTV quality at the expense of a higher data rate. MPEG-4 will provide a very high compression ratio coding of video and associated audio.

DVI, as opposed to the other systems, is a proprietary invention now under development by Intel. It defines two quality encoding variants: one for real-time compression given the appropriate hardware, and another that must be compressed off-line, but allows for decompression with the same hardware. The resolutions are 512×480 (interpolated from 256×240) and 256×240 respectively. As mentioned above, DVI also features a file syntax known as AVSS. Due to the standardization of the other formats mentioned, as well as the demand for interchangeable formats, it can be expected that RTV, PLV or both will incorporate more features from these standards to provide compatibility. DVI also specifies high-quality audio and still image compression. For still images, a certain configuration of the JPEG format is supported. The video quality in PLV mode is very good and allows for use in retrieval mode applications similar to MPEG. The RTV mode is good and quite convincing for many applications. As described in this chapter, RTV allows dialogue mode applications. However, many available implementations suffer from considerable compression/decompression delay above 150 milliseconds.

JPEG, H.261, MPEG and DVI are not alternative techniques for data compression. Their goals are different and partly complementary. Most of the algorithms used are very similar but not the same. Technical quality, as well as market availability, determine which techniques will be used in future multimedia systems. This will lead to a *cooperation* and *convergence* of techniques. For example, a future multimedia computer could generate still images in JPEG, use H.261 for a video conference and MPEG-2, as well as DVI PLV, for retrieval of stored multimedia information.