Chapter 2
Video Coding Fundamentals

This chapter provides an overview of the fundamentals of the video coding, including the tool chain, from acquisition of the video sequence via coding and transmission to display. The premier focus is on the aspects that concern the encoding and decoding process. These include the representation format of video sequences including the representation of color. The fundamental concept and the main building blocks of hybrid video coding are presented. The aim of the presentation is to provide a conceptual overview of the components and how they interact. In the following chapters, the realization of the building blocks of this scheme in HEVC will be presented and analyzed in greater detail.

2.1 Video Coding Systems

The setup of a complete video coding system can be organized according to the block diagram in Fig. 2.1. The building blocks of the video coding system are summarized below:

- Video Acquisition—Source of the video sequence which is output in a digital form. The acquisition process may be temporally and locally decoupled from the following processing steps.
- Pre-Processing—Operations on the raw uncompressed video source material, such as trimming, color format conversion, color correction, or de-noising.
- Encoding—Transformation of the input video sequence into a coded bitstream. The aim of encoding is to generate a compact representation of the input video sequence which is suitable for the transmission method in the given application scenario.
- Transmission—Packaging the bitstream into an appropriate format and transmission over the channel. Transmission includes sending and delivery of the video data to the receiver side, as well as potential methods for loss protection and loss recovery. In some scenarios, like in conversational applications with real-time

M. Wien, *High Efficiency Video Coding*, Signals and Communication Technology,
DOI 10.1007/978-3-662-44276-0_2

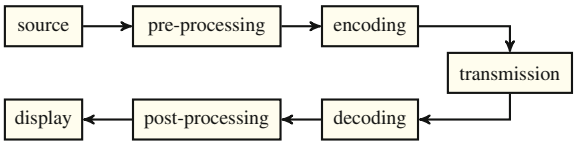

Fig. 2.1 General block diagram of a video coding system

requirements, feedback from the receiver side to the sender side is available and can be used to control the encoder configuration. Such feedback can further be used e.g. to request re-transmission in case of lost information.

- Decoding—Transformation of the received bitstream into a reconstructed video sequence. Since video encoding usually requires lossy compression to achieve the target transmission bitrate constraints, the decoded video constitutes an approximation of the original source video. If unrecoverable transmission losses have occurred, the decoder applies concealment strategies to recover the corrupted video sequence as much as possible.
- Post-Processing—Operations on the reconstructed video sequence for enhancement or for adaptation of the sequence for display. These operations can e.g. include color correction, trimming, or re-sampling. Also special effects may be applied as determined by the application.
- Display—Presentation of the video sequence for viewing. The video sequence needs to be transferred into the appropriate color format for display. The output timing of the pictures of the video sequence is an important aspect to achieve the intended visual impression.

The processing steps described above represent a simplified view on the tool chain that is used. In real applications, e.g. multiple pre-/post-processing steps in conjunctions with iterative de- and re-encoding and transmission may be used. In many applications, transcoding techniques are applied, where the incoming bitstream is transformed into a bitstream with different properties (e.g. lower bitrate), or into a bitstream following a different video coding specification.

2.1.1 Video Acquisition

Video sequence can originate from a variety of sources. These include e.g. capture of natural scenes, scanning of photographed film material, analogue recording of video, or synthetically generated video material like computer generated animation. Furthermore, mixtures and combinations of the above mentioned sources may be possible. Depending on the origin of the video material, the signal reveals different characteristic properties concerning the represented scene content as well as the presence of other signal components which interfere with the content, such as camera noise, film grain, motion blur, or artifacts from digitized analogue or physical source

material. In a simple signal model, an acquired video sequence can be described as an undistorted video signal which is additively distorted by a noise component. Further, non-additive artifacts in the video sequence may for example relate to geometrical distortions or color alignment issue due to camera lense properties. When natural scenes are captured, camera properties like the construction and the resolution of the sensor chip or the shutter speed play an important role for the resulting video sequence properties and the picture quality.

2.1.2 Pre-processing

As indicated in the previous section, the incoming video source material may have various degradations and show undesirable artifacts, which should be removed. Further pre-processing operations include trimming, geometrical format conversion, and color format conversion. Furthermore, the picture rate at which the video sequence is captured might be subject to modifications. Since different distribution methods operate with different picture rates, picture rate conversion is an important task.

In the pre-processing stage, the incoming raw video material may therefore undergo multiple signal processing and signal enhancement operations in preparation for the next steps in the video tool chain. For video coding, specific pre-processing may be applied in order to adapt the signal to the requirements of the target application. This includes e.g. the picture rate, the picture resolution, and the color format. Further dedicated pre-processing may be applied to the source material to support the encoder for an improved compression efficiency or adjusted visual impression. Applicable operations include e.g. local lowpass filtering or denoising filtering.

2.1.3 Encoding

The encoder stage transforms the input video sequence into the coded representation of a bitstream. The bitstream in contemporary video coding standards has a packet oriented structure where the encoded information is hierarchically organized according to the employed specification.

The encoder must generate a bitstream to the application requirements. These include e.g. the selection of tools applied for encoding (which correspond to a certain profile in a standard video coding scheme) or constraints on decoder-side buffer requirements and the resulting bitrate for transmission of the coded video.

The optimization of the encoder control decisions is a crucial aspect of encoder design. On the picture level, the selection of coding tools as well as the bit allocation over the regions of the picture has to be considered. Two independent encoder implementations which are conforming to the same video coding specification may produce very different reconstructed video quality.

Depending on the time that is available for encoding the video, the amount and effort spent for optimization varies. In real-time application, the encoder can only

use past information of the video it is encoding to adapt to the content. Furthermore, transmission requirements and other unpredictable events influence the encoding process.

In production systems that prepare the video for storage and media distribution, the video is processed for large scale distribution to a potentially very high number of customers. Here, off-line processing of the video can be applied and serious effort for encoder optimization and parameter tuning may be invested.

In preparation for transmission to the decoder side, the encoded bits are encapsulated into a transport format according to the application requirements.

2.1.4 Transmission

The transmission building block in the diagram of Fig. 2.1 represents the delivery of the encoded video sequence to the receiving side. This transmission may be in real-time and even bi-directional, e.g. in conversational applications. Here, the end-to-end delay between the sender and the receiver side is crucial. In live broadcast applications (like sports events) a certain delay may be tolerable as usually no direct feedback between receiver and sender is required. In storage and video distribution applications, the transmission may be temporally completely decoupled from the production process. Such applications include video-on-demand systems or storage on optical discs like DVD or Blu-ray.

Design aspects of the transmission system include the accessibility of the desired signal. This is e.g. important in multi-channel applications like television where the ability of entering to or switching between different channels is an important feature. In conversational applications where more than two participants are connected in a joint video call, the organization and redirection of the video signals to the correct receiver sides is required. In streaming and storage applications, features like fast-forward or backward search operations are desirable.

A further important aspect is the robustness of the transmission signal against disruptions, distortions, and losses. In packet-oriented transmission scenarios, an error-free transmission of the packets is often required. Distortions in the transmission signal may induce the loss of the affected packets in such systems. Therefore, mechanisms for protection against losses, and potentially the recovery of lost packets are desirable.

2.1.5 Decoding

The decoder stores the received bitstream in a buffer and reconstructs the encoded data into a video sequence in the format indicated by the encoder. In a standard video coding system, the decoding process follows the normative procedures as defined in the underlying video coding specification. If no losses occur, the reconstructed

video at the output of the decoder exactly matches the video as held available for reference at the encoder side. The bitstream may include information that only parts of the decoded video are displayed. It may even be indicated that complete decoded pictures are suspended from display (while potentially being used for prediction).

An important aspect of decoder control is the buffer management for the decoded pictures as well as for the incoming bitstream. Decoded pictures in the decoded picture buffer may be further used for reference in the decoding process. Further, in systems that operate at a defined picture rate for presentation, it must be asserted that the decoding process makes the decoded pictures available for display according to the timing constraints imposed by the employed format.

In case of transmission losses, the decoder must be able to act on missing data and potentially unforeseen conditions. In such cases, error resilience is required and the decoder must be able to apply error concealment strategies.

2.1.6 Post-processing

The pictures that are output by the decoder may be fed into a post-processing stage where operations for image enhancement and for display adaptation may be performed. Also post-decoder error concealment methods may be applied.

If the format of the decoded video material does not correspond to the format required for display, application of conversion operations similar to those listed for pre-processing is indicated. These include e.g. the color format, the picture resolution or the picture rate.

2.1.7 Display

The display device is the interface to the human visual system (HVS) as the final sink for the video signal.

As indicated before, not all samples of the coded video sequence are necessarily used or even available for display. For example, parts of the left or right boundary of the picture (or both) may be excluded from the area for display. If the remaining part of the picture is to be displayed on the full screen, this mode of operation is called *overscan*. If number of coded samples is smaller than the number of samples available on the display, the video may either be 'stretched' to match the display resolution, or the video is displayed without modification and the display area is not completely covered by the video. This mode of operation is called *underscan*.

The video format, the color space of the source material, and other helpful information that may be utile for adequate preparation of the video for display can be signaled with the video data. Such information is called *Video Usability Information*and is further described in Sect. 5.7. While it does not necessarily have an impact on the normative decoding process, keeping the information associated with the video

itself ensures that the intended parameters for display are conveyed to the decoding side. In some application scenarios, the video usability information may be fixed and therefore be hard-coded for the system.

2.2 Structure of a Video Sequence

A *video sequence* consists of a series of pictures, which are presented with constant or variable time intervals between successive pictures. In order to achieve the impression of motion, a picture rate of at least about 24 pictures per second is required. The minimum picture rate depends on the lighting conditions and the content to be displayed [1]. The unit of the picture rate is Hertz (Hz), with $1\,\mathrm{Hz} = 1\,\mathrm{s}^{-1}$. The picture rate is also referred to as the frame rate using the acronym 'fps' (frames per second). High definition (HD) video today applies picture rates of 50–60 Hz. For Ultra HD formats, picture rates of up to 120 Hz are specified [2]. Even higher picture rates are discussed, see e.g. [3].

In the context of video coding, an input video sequence is encoded into a bitstream which contains basically all information and data that is necessary to decode the bitstream and reconstruct the output video sequence for display at the receiving side. In the vast majority of video coding applications, encoding the video sequence into a bitstream corresponds to a lossy compression, i.e. the reconstructed video sequence at the decoder only approximates the original input video sequences. The transmission between the sender, where the original sequence is encoded, and the receiver, which includes the decoder, may be instantaneous or close to real-time, e.g. in video communication or live broadcast scenarios. It may also be completely decoupled, e.g. if the data is stored on a Blu-ray disc, DVD, or in streaming applications.

2.2.1 Pictures, Frames, and Fields

A *picture* is an array or a set of arrays of *samples* with intensity values. A sample in the array is also denoted as *pixel* (derived from *pic*ture *el*ement, or a *pel*). If the picture is monochrome, i.e. it has one single color component, the picture consists of a single sample array. For representation of color in video or pictures, usually three color components are employed as detailed below. Correspondingly, a color picture consists of three intensity arrays, with one array for each component. In some applications a fourth array is included which represents the 'opaqueness' of the pixels. This array is often called the α component. Such information can be used when mixing multiple videos into one scene, e.g. to add a foreground speaker to a background video.

Historically, the video format for capture, distribution, and display of television programs was interlaced. In interlaced video, the even and odd sample lines are collected in the *top* and *bottom field* pictures, respectively. The two fields are alternately

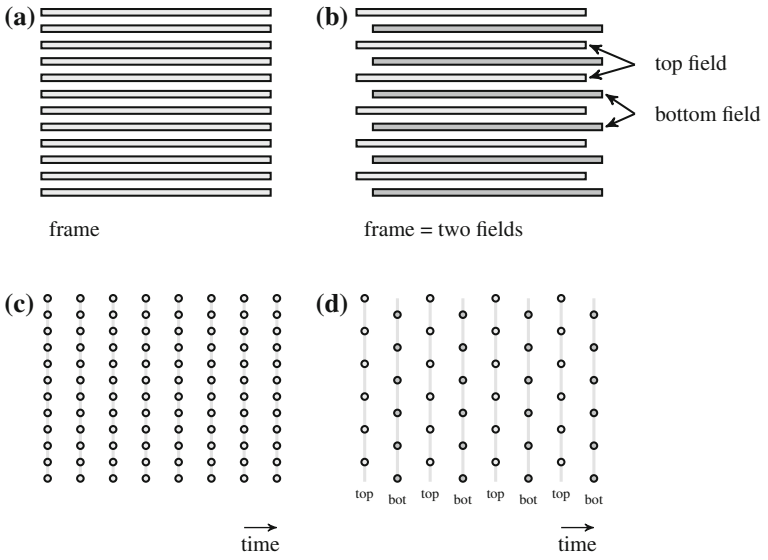

Fig. 2.2 Frames and fields in progressive and interlaced video. **a** Lines of a progressive frame. **b** *Lines* of the *top* and *bottom* fields of an interlaced frame. **c** One-dimensional representation of the scanning of progressive frames over time. **d** One-dimensional representation of the scanning of interlaced frames over time

displayed. The displays were built with CRT screens[1] with relatively slow fading characteristics. The alternating display of the even and odd lines on these screens was the state of the art in display technology until the end of the 20th century.

In interlaced video, a pair of a top and a bottom field constitutes a *frame*. For non-interlaced material, a frame holds a single picture which has been acquired in *progressive scan*. The progressive and interlaced scans are visualized in Fig. 2.2, including an illustration of the relation of the fields over time. For details on the interlaced and progressive video formats, the reader is referred to [1].

2.2.2 Sample Shape

When capturing video, each picture is represented by a set of lines and a set of samples (i.e. samples) per line. The shape of the samples in a picture is not necessarily square. There exists a variety of other aspect ratios of width and height of the samples that are commonly used in video coding systems. Different sample aspect ratios can be used to adapt the display size of the video to the aspect ratio of the screen in dependency of the number of lines and the number of samples per line in a picture.

[1] CRT = Cathode Ray Tube

Table 2.1 Common sample aspect ratios

Index[a]	Sample aspect ratio	Pixel shape	Usage examples	
1	1:1		7680 × 4320 frame	No horizontal overscan
			3840 × 2160 frame	No horizontal overscan
			1280 × 720 frame	No horizontal overscan
			1920 × 1080 frame	No horizontal overscan
			640 × 480 frame	No horizontal overscan
2	12:11		720 × 576 4:3 frame	With horizontal overscan
			352 × 288 4:3 frame	No horizontal overscan
3	10:11		720 × 480 4:3 frame	With horizontal overscan
			352 × 240 4:3 frame	No horizontal overscan
4	16:11		720 × 576 16:9 frame	With horizontal overscan
			528 × 576 4:3 frame	No horizontal overscan
5	40:33		720 × 480 16:9 frame	With horizontal overscan
			528 × 480 4:3 frame	No horizontal overscan
6	24:11		352 × 576 4:3 frame	No horizontal overscan
			480 × 576 16:9 frame	With horizontal overscan
7	20:11		352 × 480 4:3 frame	No horizontal overscan
			480 × 480 16:9 frame	With horizontal overscan
8	32:11		352 × 576 16:9 frame	No horizontal overscan
9	80:33		352 × 480 16:9 frame	No horizontal overscan
10	18:11		480 × 576 4:3 frame	With horizontal overscan
11	15:11		480 × 480 4:3 frame	With horizontal overscan
12	64:33		528 × 576 16:9 frame	No horizontal overscan
13	160:99		528 × 480 16:9 frame	No horizontal overscan
14	4:3		1440 × 1080 16:9 frame	No horizontal overscan
15	3:2		1280 × 1080 16:9 frame	No horizontal overscan
16	2:1		960 × 1080 16:9 frame	No horizontal overscan

[a] Index indication according to the HEVC Video Usability Information

A set of commonly used sample aspect ratios is presented in Table 2.1. The table provides the aspect ratio and visualizes the shape of the samples under the assumption of identical height. For each aspect ratio, typical application formats and the corresponding application of horizontal overscan are indicated [4].

2.3 Representation of Color

The perception of color in the eye is stimulated by electromagnetic waves. The visible range of the electromagnetic waves includes the approximate range from 380 to 780 nm. The impression of color corresponds to the perception of an intensity

density distribution over the visible spectral range. Colors corresponding to a single wavelength are called spectral colors or *primary colors*. These cover a range from violet over blue, green, yellow, to red. A large set of observed color impressions are created by the combination of multiple spectral colors (e.g. magenta or brown).

The human visual system has three color receptors (cone cells) with maximum sensitivity in the wavelength areas of red, green, and blue. It further comprises 'gray-scale' receptors (rod cells) which are especially responsive in low lighting conditions, e.g. at night. The impression of different colors is generated by the combined stimulus of the three color receptors. The human visual system has the feature that the same color impression can be achieved by multiple different combinations of spectral colors. Stimuli leading to the same color impressions are characterized to be metameric. This metamerism is utilized for presentation of color on displays, where usually a mixture of three light sources (red, green, and blue) is used to generate the impression of color.

2.3.1 The CIE Standard Observer

Visual perception can be split into perception of brightness (light and dark) and chromaticity (the color impression). While the brightness is driven by the summarized intensity of the observed spectrum, the color impression is driven by the shape of the intensity distribution. A functional expression to represent perceived color by a mathematical description was first established and standardized in the CIE 1931 Standard Observer.[2] The model assigns each color a point in a three-dimensional XYZ space. The X, Y, Z values are derived from the observed spectrum using three color matching functions $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$ which simulate the sensitivity of the human color receptors, as shown in Fig. 2.3. As can be seen from the figure, $\bar{x}(\lambda)$ has a large peak in the range of the color red and a small peak in the range of blue, $\bar{y}(\lambda)$ has a peak in the range of green and $\bar{z}(\lambda)$ has its peak in the range of blue.

The color impression translates into the contribution of each color matching function. Let $I(\lambda)$ be the spectral intensity at wavelength λ. The X, Y, Z values are derived as

$$X = \int_{380\,\mathrm{nm}}^{780\,\mathrm{nm}} I(\lambda)\bar{x}(\lambda)\,\mathrm{d}\lambda, \quad Y = \int_{380\,\mathrm{nm}}^{780\,\mathrm{nm}} I(\lambda)\bar{y}(\lambda)\,\mathrm{d}\lambda, \quad Z = \int_{380\,\mathrm{nm}}^{780\,\mathrm{nm}} I(\lambda)\bar{z}(\lambda)\,\mathrm{d}\lambda. \tag{2.1}$$

[2] Defined in 1931 by the Commission internationale de l'éclairage (CIE), specified in ISO 11664-1 [5].

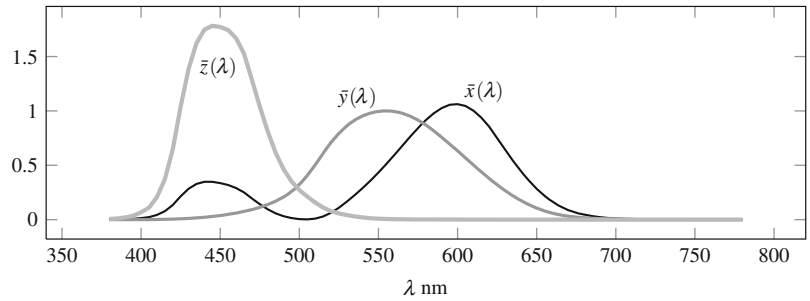

Fig. 2.3 CIE1931 standard observer color matching functions

In order to get an expression of the chromaticity, which is independent of the observed brightness, the X, Y, Z values are normalized as

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}. \tag{2.2}$$

Since $x + y + z = 1$, the chromaticity can be specified by the (x, y)-pair. These (x, y) values are used to specify reference colors in various standards which deal with coding and display of images and video, see e.g. Table E-3 in HEVC Annex E.

This color representation further allows for the definition of a 'white point', i.e. the combination of x, y, z values which corresponds to a defined color impression of white. Several definitions of white points exist, including 'white C' specified by the CIE with the standard observer in 1931 and now deprecated, and 'white D65' which was specified by the CIE later on. See the references for further reading [1, 6].

2.3.2 Color Primaries

As stated above, display systems usually employ a mixture of red, green, and blue light sources to generate the impression of color on the screen. The mixture of red, green, and blue spans the RGB color space. As these colors are spectral or primary colors, they are called the *color primaries*.

In order to specify the perceived color impression generated from an input signal, color primaries are specified in all relevant video transmission and video display specifications. A set of commonly used color primaries is provided in Table 2.2.

2.3.3 Display Transfer Characteristics

The relation between the optical intensity and the corresponding signal representation is not necessarily linear. In order to deal with this effect, opto-electronic transfer

Table 2.2 Example color primaries used in ITU-R recommendations

Primary	Red		Green		Blue	
	x	y	x	y	x	y
ITU-T BT.601 625	0.640	0.330	0.290	0.600	0.150	0.060
ITU-T BT.601 525	0.630	0.340	0.310	0.595	0.155	0.070
ITU-T BT.709	0.640	0.330	0.300	0.600	0.150	0.060
ITU-T BT.2020	0.708	0.292	0.170	0.797	0.131	0.046

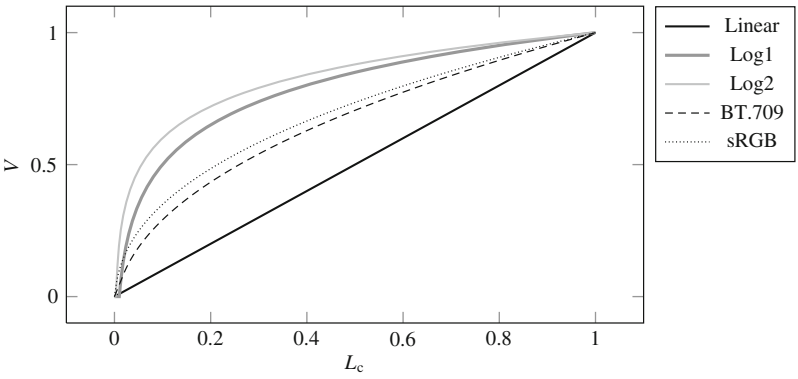

Fig. 2.4 Visualization of transfer characteristics between linear intensity input L_c and the value V of the signal component

characteristics are used, which describe the relation between a linear optical intensity input L_c and the nominal value V.[3]

Figure 2.4 shows some representative transfer characteristics according to Table E-4 in HEVC Annex E. A linear transfer characteristic is compared to logarithmic and other characteristics. Other characteristics very similar to the ITU-T BT.709 curve are specified in SMPTE-240M or ITU-T BT.2020. The curves for IEC 61966-2-4 and the extended gamut system in ITU-T BT.1361 are not shown. These specifications represent a much wider value range and include negative values for L_c.

2.3.4 Color Conversion

While capture and display usually are operating in the RGB color space, the usage of a *luminance* and two *chrominance* signals is established for coding and

[3] The reference electro-optical transfer function (EOTF) for flat panel displays used in HDTV production is specified in ITU-T BT.1886 [9].

transmission.[4] The luminance component represents the gray level intensity while the two chrominance components are required to specify the chromaticity as described in Sect. 2.3.1. Together with the color primaries and the transfer characteristics, the conversion parameters between the two color spaces define the complete settings of the color processing chain. If information on the parameters used in the generation process of the coded video sequence is available, the receiver is enabled to reproduce (or at least approximate) the intended color impression for display.

Let the linear intensity of the RGB components based on the color primaries as e.g. listed in Table 2.2 be denoted by E_R, E_G, and E_B. The direct conversion of these values would turn these intensities into a luminance and two chrominance signals. Since the system transfer characteristics for the color impression are not linear in general, these have to be taken into account by applying a transfer characteristics function as shown in Fig. 2.3. The resulting modified values are denoted by E'_R, E'_G, and E'_B. The conversion result from these modified intensity values is denoted by *luma* and *chroma* to indicate the different conversion source. The luma component is indicated by Y or L while the two chroma components are indicated by Cb and Cr. The derivation of the luma and chroma signals from the modified intensities is performed as follows.

Let the components E'_R, E'_G, E'_B be real valued and have a value range of [0, 1] each, where $E'_R = E'_G = E'_B = 0$ corresponds to nominal black and $E'_R = E'_G = E'_B = 1$ corresponds to nominal white. Let $k_R, k_G = 1-k_R-k_B$, and k_B be weighting factors for the three components E'_R, E'_G, and E'_B, respectively. The real valued luma and chroma signals are derived as

$$E'_Y = k_R \cdot E'_R + (1 - k_R - k_B) \cdot E'_G + k_B \cdot E'_B, \tag{2.3}$$

$$E'_{PB} = \frac{1}{2} \cdot \frac{1}{1 - k_B} \cdot (E'_B - E'_Y), \tag{2.4}$$

$$E'_{PR} = \frac{1}{2} \cdot \frac{1}{1 - k_R} \cdot (E'_R - E'_Y). \tag{2.5}$$

The luma signal E'_Y has a range from nominal black ($E'_Y = 0$) to nominal white ($E'_Y = 1$). The two chrominance components have nominal range from -0.5 to 0.5. Example values for k_R and k_B are listed in Table 2.3 [2, 7, 8].

Table 2.3 Example color conversion coefficients used in ITU-R recommendations

	k_R	k_B
ITU-T BT.601	0.299	0.114
ITU-T BT.709	0.2126	0.0722
ITU-T BT.2020	0.2627	0.0593

[4] Analog television started off with presentation of luminance only (black and white). By additional transmission of two chrominance signals, a backward compatible transmission of color and monochrome television signals was enabled [1].

For digital representation, the signals need to be quantized to a predefined precision which corresponds to a specified bit depth. In many applications, the bit depth of the component signals is set to 8 bit. In HD applications, 10 bit or even higher bit depths like 12 bit, 14 bit, or even 16 bit may be employed.

Since signal processing like filtering and prediction may induce some signal overshoot or undershoot, the quantization of the real-valued E'_{Y}, E'_{PB}, E'_{PR} signals to integer values of the specified bit depth B_{d} is performed using a slightly reduced value range, see HEVC, Annex E [4][5]:

$$Y = \text{round}\left\{219 \cdot E'_{\mathrm{Y}} + 16\right\} \tag{2.6}$$

$$C_{\mathrm{b}} = \text{round}\left\{224 \cdot E'_{\mathrm{PB}} + 128\right\} \tag{2.7}$$

$$C_{\mathrm{r}} = \text{round}\left\{224 \cdot E'_{\mathrm{PR}} + 128\right\} \tag{2.8}$$

The quantization process is denoted for 8 bit signals here. For higher bit depth B_{d}, the (2.6)–(2.8) are scaled by $2^{B_{\mathrm{d}}-8}$. The resulting integer Y, C_{b}, and C_{r} components are denoted as the luma component and the two chroma components, respectively. The color space defined by these components is denoted as the YCbCr color space.

2.3.5 Chroma Sub-sampling

The human visual system is less sensitive to color than it is to structure and texture information. Therefore, in many application scenarios, it is more important to provide a high resolution luma component than to provide such detail for the chroma components. In high-definition high-quality applications like production, full resolution components at a high bit depth may be advisable. In consumer applications, sub-sampling of the chroma components is commonly applied.

Various types of chroma sub-sampling exist. Here, two chroma sub-sampling formats explicitly considered in the HEVC specification are briefly summarized. A comprehensive treatment on the subject is provided e.g. in [1]. The chosen color sub-sampling is commonly expressed in the relation between the number of luma samples compared to the number of chroma samples. The common notation is

$$\text{YCbCr}\ \ Y{:}X_1{:}X_2.$$

The first value Y denotes the number of luma samples, the value X_1 and X_2 describe the sub-sampling format of the chroma components relative to the luma value. In the most common formats, $Y = 4$ is used. The X_1 value specifies the horizontal sub-sampling. The value $X_2 = 0$ indicates that the same X_1 sub-sampling factor is applied for the vertical direction. $X_2 = X_1$ indicates that no vertical sub-sampling

[5] Using $\text{round}\{v\} = \text{sgn}(v)\left\lfloor |v| + \frac{1}{2} \right\rfloor$.

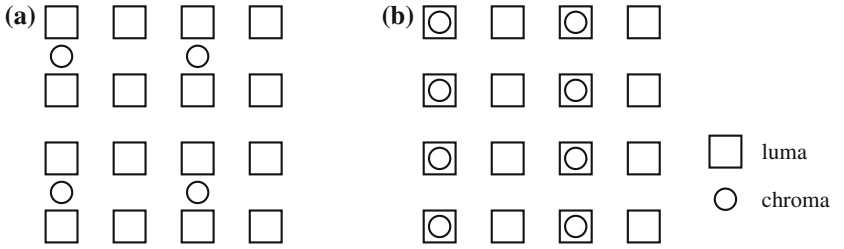

Fig. 2.5 Chroma locations in typical video formats as used in the specification of the HEVC decoding process. **a** YCbCr 4:2:0 video. **b** YCbCr 4:2:2 video

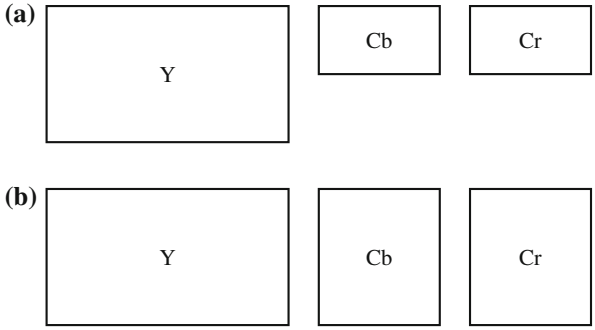

Fig. 2.6 Illustration of the YCbCr sample arrays for pictures using the chroma sub-sampling shown in Fig. 2.5 for video sequences with 16:9 picture size. **a** YCbCr 4:2:0. **b** YCbCr 4:2:2

is performed and both chroma components apply the same horizontal sub-sampling factor. A visualization for the two most common formats YCbCr 4:2:0 and YCbCr 4:2:2 is shown in Fig. 2.5. The corresponding sample arrays as used in the decoding process are shown in Fig. 2.6.

The locations of the chroma samples shown in Fig. 2.5 corresponds to the locations that are used in the specification of the HEVC decoding process (as it was done for H.264 | AVC). While these chroma locations are used for coding, the consideration of the correct chroma locations as present in the original source material is required for display. Otherwise, display of the video may include an observable shift between the luma component and the chroma component.

A variety of chroma locations may be considered for this purpose. For a coded video sequence, the applicable locations can be indicated in the Video Usability Information, see Sect. 5.7. The possible locations are summarized in Fig. 2.7. The numbering used in the Figure corresponds to the numbering used for signaling of the chroma locations in HEVC Annex E. In the specification of the decoding process for YCbCr 4:2:0 video, chroma location "0" is employed. The presentation includes the

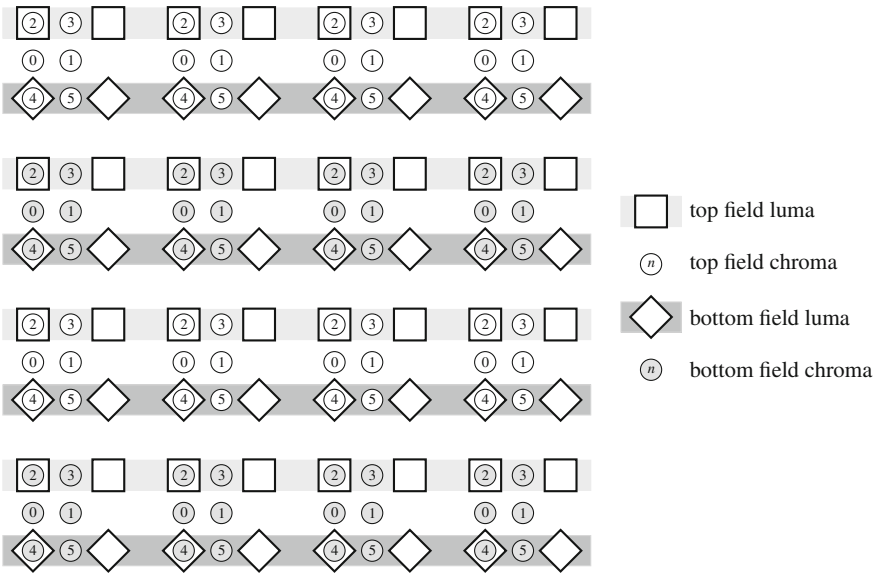

Fig. 2.7 Possible locations for chroma samples relative to the luma samples in video source material

distinction between top field and bottom field locations for both luma and chroma. In case of interlaced video material, the top field chroma samples are associated with the top field luma samples in a top field picture, while the bottom field chroma samples are associated with the bottom field luma samples in a bottom field picture. The chroma locations shown in Fig. 2.5 correspond to chroma location "0" for YCbCr 4:2:0 video and "2" and "4" for YCbCr 4:2:2 video. For each of these locations, a half way shift of the chroma location between two neighboring luma locations may have been used (e.g. in H.261, or JPEG). Such shifted chroma locations can be indicated by a "+1" operation on the indices mentioned previously.

2.4 The Hybrid Video Coding Scheme

Since H.261, the hybrid video coding scheme has been the basic structure for all video coding standards and recommendations of ITU-T and ISO/IEC MPEG. While the structure has not been changed, the algorithms represented by the building blocks have been refined and the applicable configuration for the algorithms has become more and more flexible over the last 25 years. The coding scheme is called *hybrid* as it combines temporal prediction between pictures of the video sequence with transform coding techniques for the prediction error [10].

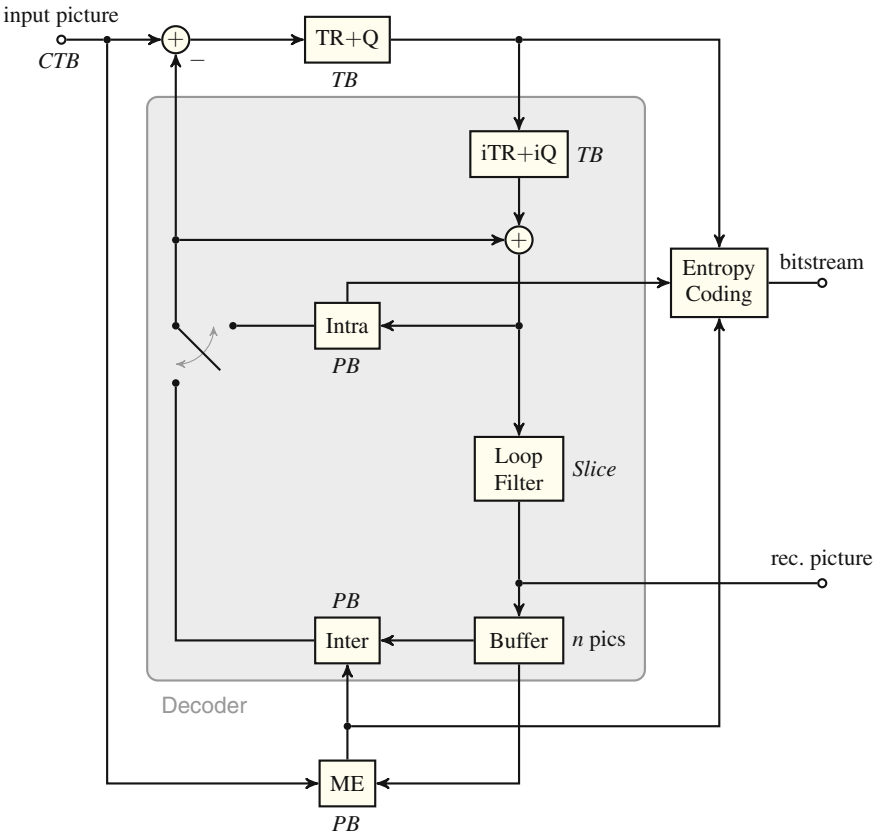

Fig. 2.8 Encoder block diagram for the hybrid video coding scheme (*CTB* coding tree block; *ME* motion estimation; *PB* prediction block; *Q* quantization; *TB* transform block; *TR* transform)

The hybrid video coding scheme has proved to be a well-suited tool to efficiently compress a video signal into a bitstream of smallest possible size. By prediction and transformation of the prediction error signal, it eliminates *redundant* information from the data. With the application of quantization after the transform step, *irrelevant* parts of the information are removed.[6]

The basic structure of the hybrid video coding scheme is shown in Fig. 2.8. The figure is simplified to a great extent to make the structure and the interdependencies visible. It corresponds to a classical DPCM loop[7]: The pictures of the input video sequence are fed to the encoder. A prediction signal generated from information available both, encoder and decoder is subtracted from the input signal. The residual, rep-

[6] If too strong quantization is applied, also relevant parts of the video signal content may be affected.

[7] DPCM: Differential Pulse Code Modulation.

resenting the resulting prediction error, is transformed, quantized, and encoded into the bitstream. The prediction parameters needed to reproduce the prediction signal at the decoder side are encoded into the bitstream as well. Under the assumption of error-free transmission, the encoder and decoder sides are synchronized since the encoder includes the complete prediction structure of the decoder. In Fig. 2.8, the building blocks which are included in both encoder and decoder are marked by a gray box.

Previously decoded pictures can be used for *inter* prediction in the current picture, which is to be encoded. The encoder can also choose to employ already coded neighboring samples within the current picture for *intra* prediction. The first coded picture of a video sequence can only apply intra prediction as no previous pictures are available. For the following pictures, the encoder decides between the two prediction options based on a decision criterion (e.g. rate-distortion optimization).

The process operates in a block-based fashion where the complete input picture is segmented into non-overlapping blocks, which are processed one after the other. For construction of the prediction signal, the transformed and quantized prediction error is first reconstructed and added with the available prediction. The signal is then processed by the loop filter (e.g. a deblocking filter). If the full picture has been processed, the reconstructed picture is available, as it also is at the decoder side. The reconstructed picture is stored in the decoded picture buffer to make it available for prediction.

For inter prediction, the motion estimation stage (ME) searches for the best prediction available for the current picture block in the decoded picture buffer. For intra prediction, sample values from already reconstructed neighboring blocks of the current picture can be used for prediction. Depending on the encoder decision which prediction mode has been selected, either the intra or the inter prediction signal is used for prediction of the current block.

The block diagram shown in Fig. 2.8 does not include the building blocks required for encoder control. An encoder implementation requires a control engine which decides on the applicable prediction modes, prediction and filtering parameters, as well as the applicable quantization parameters. Control information to inform the decoder on the selected prediction tools and configurations is included in the bitstream as well.

In the following, the building blocks of the hybrid video coding scheme are briefly described. The acronyms used in Fig. 2.8 are explained. For simplification purpose, the description focuses on the luma component. Prediction and transform coding are performed similarly for the chroma components.

2.4.1 Picture Partitioning

Each picture is partitioned into a complete set of non-overlapping blocks. An illustration of the basic partitioning of a picture is shown in Fig. 2.9. In previous video coding specifications, the basic block structure was the so-called *macroblock* of 16×16 luma samples and the corresponding chroma samples. As a generalization

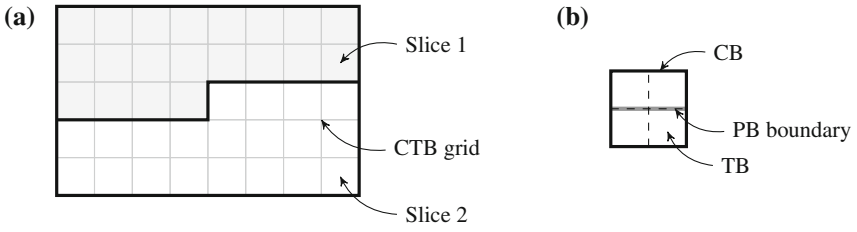

Fig. 2.9 Partitioning. **a** Picture into slices and coding tree blocks (*CTBs*). **b** Partitioning of a coding block into prediction blocks (*PBs*) and transform blocks (*TBs*)

of the macroblock concept, the coding tree block (CTB) is introduced in HEVC. It can be split into multiple coding blocks (CBs). For prediction, each coding block is partitioned into a set of one or more prediction blocks (PB). In parallel, the coding block can be partitioned into transform blocks (TB).

The picture can further be partitioned into one or more slices, which consist of an integer number of CTBs and are independently decodable.

2.4.2 Intra Prediction

Intra prediction is used to remove correlation within local regions of a picture. The basic assumption for intra prediction is that texture of a picture region is similar to the texture in the local neighborhood and can thus be predicted from there.

Intra prediction is performed on a prediction block basis. In the block diagram of Fig. 2.8, it is denoted by the building block "intra" For intra prediction, samples from the reconstructed neighborhood of the prediction block under consideration are employed to form the prediction signal. Intra prediction is applied when no pictures for inter prediction are available, or if inter prediction would be less efficient or more expensive (in the sense of an applicable cost criterion) than an applicable inter prediction.

The direct neighbor samples are commonly employed for prediction, i.e. samples from the sample line above the current block and samples from the last column of the reconstructed blocks to the left of the current block. The values of the available neighboring samples are combined to form a e.g. directional or planar prediction signal. An illustration of directional intra prediction is shown in Fig. 2.10.

The quality and the frequency of usage of intra prediction depends on various aspects. Criteria are the available intra prediction directions, the method of potential pre-filtering of the prediction samples before application, but also the availability of other prediction methods that are not directional (like DC or planar prediction). As for inter prediction, an efficient signaling of the applicable intra prediction mode is essential. The more variants are available for use, the more efficient signaling is needed in order not to trade the prediction benefit for the signaling cost.

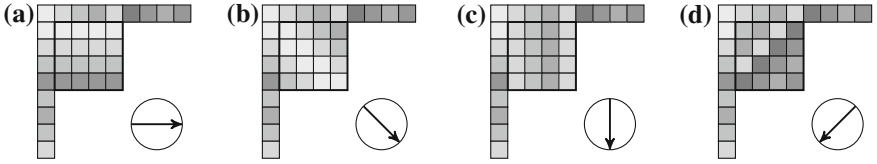

Fig. 2.10 Examples for directional intra prediction of the current block from neighboring samples. **a** Horizontal. **b** Diag. down right. **c** Vertical. **d** Diag. down left

2.4.3 Inter Prediction

A basic assumption in the context of inter-picture prediction is that a significant part of the content of the pictures in a video sequence consists of objects which move in the scene. From picture to picture, only small differences between the scene content are observed and these differences are mostly due to motion. If this motion is used for prediction, the scene can be efficiently represented by motion vectors and a prediction error signal. The motion vectors indicate how picture regions should be moved from the reference picture to the current picture to form the prediction. The prediction error signal contains the part of the scene content that could not be described by the applied motion model.

Inter prediction is performed on a prediction block basis. In the block diagram of Fig. 2.8, it is denoted by the building block "inter". Usually, inter prediction is performed by selecting an available reference picture from the decoded picture buffer and indicate a displacement relative to the location of the current prediction block in the selected picture.

This displacement is motivated by the assumption of planar motion of rigid objects in the observed scene, see Fig. 2.11a. More complex motion, like movement in direction of the camera or rotation, or motion of non-rigid objects, cannot be represented by this very simple model. Examples for pure rotational motion of the rigid object

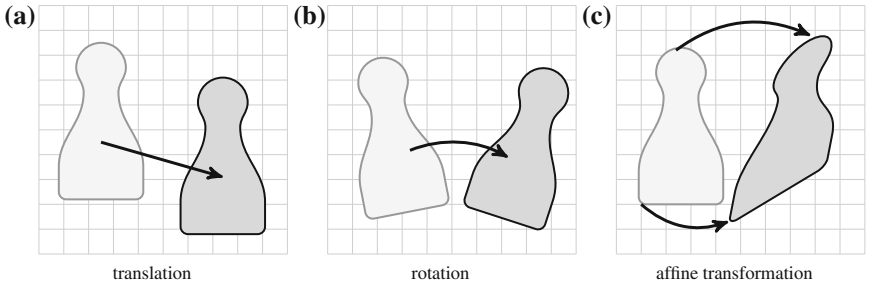

Fig. 2.11 Illustration of motion models for objects between successive pictures. **a**, **b** rigid transformation. **c** non-rigid transformation

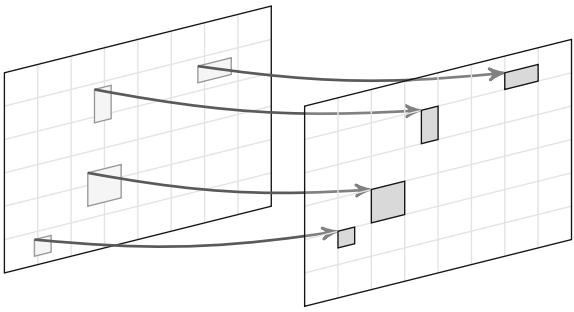

Fig. 2.12 Block displacement using a fixed grid for inter prediction from the reference picture

and affine transformation where the shape of the object is affected are shown in Fig. 2.11b, c. More complex objects, e.g. like waves or smoke, but also effects like shadow, reflections, or transparency reduce the possibility to predict from picture to picture by simple displacement even more. However, by adapting the applicable prediction block size, the prediction quality using these displacements has been considered to be sufficient and suitable for video coding applications.

In terms of prediction precision, motion parameters which e.g. describe the rotational or affine motion might be the an appropriate solution for some of the example scenarios. However, the signaling cost for the required parameters, the estimation effort needed to find the right model parameters, and not least the complexity impact when implementing higher order motion models in the decoder have so far prevented the specification of higher order motion models in video coding specifications. Since only the translational motion model, using displacement vectors, is available, more complex motion has to be (somewhat) approximated by adapting the applicable size and shape of the prediction blocks.

For inter prediction, the current picture is partitioned into non-overlapping rectangular blocks. For each of the blocks, a displacement vector is applied to get the prediction for the block from the displaced area of the reference picture, see the illustration in Fig. 2.12. For each prediction block in a fixed block raster in the current picture, a related area in the reference picture is determined. While the blocks in the current picture follow the given block grid, the reference blocks in the reference picture are located arbitrarily subject to the optimization criterion for motion estimation.

In terms of motion vector precision, the most simple realization would allow for integer displacements on the sample grid. Since motion in the captured scene will not necessarily correspond to full sample deltas between successive pictures in the sequence, sub-sample motion vector precision is required for high compression performance. The prediction signal at sub-sample locations is interpolated using interpolation filters. In H.264 | AVC and HEVC, quarter-sample precision is applied for motion vectors in the luma component. The usage of higher precision luma motion vectors, as well as the usage of adaptive interpolation filters, has been investigated during the development of HEVC but has not been adopted into a video coding specification due to the observed trade-off between the compression gain improvement and the implementation complexity cost.

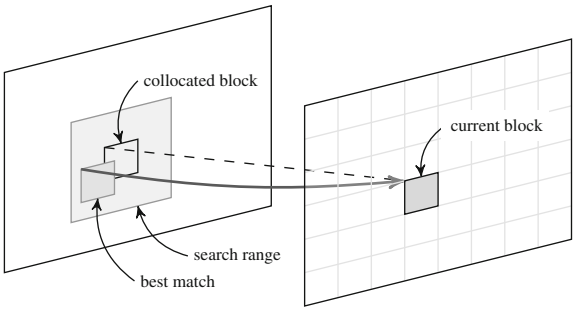

Fig. 2.13 Motion estimation in a search range around the collocated block in the reference picture

2.4.4 Motion Estimation

The motion estimation stage operates on a prediction block level and is only part of the encoder.[8] The estimator takes the current prediction block to be used and tries to find the best matching area in an available reference picture. The determination of what the best match would be is subject to the employed cost criterion.

A traditional search method is to shift the current prediction block over a search area around the collocated block position in the reference picture, and to determine the cost criterion for each position, see the illustration in Fig. 2.13. If all available positions are tested, the method is called a full search. Fast search methods may only test a subset of the available positions, or e.g. use criteria from the neighborhood to generate a candidate set of motion vectors to be considered during the search operation. Since motion estimation is one of the computationally most demanding tasks in the encoder, the implementation of an efficient search algorithm is crucial for a successful and fast encoder.

2.4.5 Residual Coding

Intra or inter prediction remove correlation within pictures and between pictures of a video sequence. The subtraction of the prediction signal from the current block generates the residual signal, containing the part of the original signal which could not be predicted by the selected prediction method. While prediction already reduces the correlation of the residual signal, it still contains information which can be further compressed. This decorrelation is performed by the application of a transformation, which is applied to represent the correlated parts of the residual signal in the residual block by a potentially small number of transform coefficients. These coefficients are then quantized and coded into the bitstream.

[8] Motion estimation at the decoder side to circumvent the transmission of motion vector information has been evaluated [11], but so far has not become part of a video coding specification.

The transform is applied on a transform block basis. In the block diagram of Fig. 2.8, forward transform and quantization are denoted by the building block "TR+Q". The inverse quantization and inverse transform block is denoted by "iTR+iQ". Usually the transform block size is smaller or equal to the prediction block size. Thereby, transformation across artificial boundaries is prevented. Such boundaries may occur if the prediction sources for two neighboring prediction blocks diverge (e.g. when predicting from different reference pictures).

2.4.5.1 Transform Type

By concept, the Karhunen-Loàve Transform (KLT) provides optimum decorrelation and energy compaction of the residual signal [12]. As it would not be practical to compute and signal the applicable KLT for a given residual characteristics, the application of transforms approximating the features of the KLT is advisable. For this purpose, parametric models of the prediction error signal have been used. Jain [13] has shown that for an important parametric model which includes the first order Markov process, a family of sinusoidal unitary transforms constitutes a complete orthonormal set of bases (eigenvectors). This family comprises the type I–VIII Discrete Cosine Transform (DCT) and the type I–VIII Discrete Sine Transform (DST). The different types of the DCT and DST vary in the way the transform matrices are constructed from cosine or sine functions. Details can be found e.g. in [14]. Here, only the transforms which are used in the context of video compression are detailed, see below. For prediction error signals whose characteristics can be approximated by a process following this parametric model, a corresponding sinusoidal transform can hence be considered as a suitable choice for decorrelation.

In all standardized video coding specifications which are based on the hybrid video coding scheme, the DCT-II/DCT-III transform pair is used for transformation of the prediction error, where the DCT-II is the forward and the DCT-III the corresponding inverse transform. Since this is the most commonly used DCT pair, it is often referred to as *the* DCT. In the context of HEVC, also the DST-VI / DST-VII pair is applied as further detailed in Chap. 8. The suitability of this transform pair for application with intra prediction error signals has been presented in [15]. The DST-VI will be referred to as *the* DST in this book.

2.4.5.2 The DCT Matrix

The DCT is a block transform and can be represented in matrix notation. A DCT matrix of size $N \times N$ is denoted by $\mathbf{T}_{\mathrm{DCT},N}$ with base vectors $\mathbf{t}_n$ of length N,

$$\mathbf{T}_{\mathrm{DCT},N} = \begin{bmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_{N-1} \end{bmatrix}. \tag{2.9}$$

The row vector $\mathbf{t}_n$ contains the n-th base function or base vector of the transform. The elements $t_n(m)$ of this vector constitute the coefficients of the $N \times N$ DCT matrix $\mathbf{T}_{\mathrm{DCT},N}$. For the DCT-II, these are defined by

$$t_n^{\mathrm{II}}(m) = \sqrt{\frac{2}{N}} \cdot a(n) \cdot \cos\left(\frac{\pi(2m+1)n}{2N}\right), \quad m, n = 0, ..., N-1, \tag{2.10}$$

with

$$a(n) = \begin{cases} 1/\sqrt{2} & : \quad n = 0, \\ 1 & : \quad n = 1, ..., N-1. \end{cases} \tag{2.11}$$

This is the orthonormal definition of the DCT-II, i.e. $\mathbf{T}_{\mathrm{DCT},N} \cdot \mathbf{T}_{\mathrm{DCT},N}^{-1} = \mathbf{I}_N$, where $\mathbf{I}_N$ is the $N \times N$ identity matrix. Other not normalized definitions of the DCT-II matrix are sometimes used as well [16]. As can be seen from (2.10), the base vectors are sorted with respect to increasing frequency of the cos() function in $\mathbf{T}_{\mathrm{DCT},N}$.

The DCT-II has symmetric base functions on the even positions and anti-symmetric base functions on the odd positions, i.e.

$$t_n^{\mathrm{II}}(N-m-1) = \begin{cases} t_n^{\mathrm{II}}(m) & : \quad m = 0, \ldots, N/2-1, \ n \text{ even}, \\ -t_n^{\mathrm{II}}(m) & : \quad m = 0, \ldots, N/2-1, \ n \text{ odd}. \end{cases} \tag{2.12}$$

Hence, each DCT base function can only have $N/2$ different coefficient values at maximum.

Furthermore, the even base functions of a $2N \times 2N$ DCT can be generated from the base functions of the $N \times N$ DCT by scaling and mirroring,

$$\mathbf{t}_{2n,2N}^{\mathrm{II}} = \frac{1}{\sqrt{2}} \cdot \left[\mathbf{t}_{n,N}, \quad \mathbf{J}_N \cdot \mathbf{t}_{n,N}^{\mathrm{II}}\right], \tag{2.13}$$

where $\mathbf{J}_N$ is the opposite identity or reversal matrix of size $N \times N$. From these symmetry properties it can be seen that overall, a $N \times N$ DCT-II matrix contains $N-1$ different coefficient values.

The elements of the DCT-III transform matrix are defined as

$$t_n^{\mathrm{III}}(m) = \sqrt{\frac{2}{N}} \cdot a(m) \cdot \cos\left(\frac{\pi m(2n+1)}{2N}\right), \quad m, n = 0, ..., N-1, \tag{2.14}$$

with $a(m)$ given in (2.11). As stated above, the DCT-III is the inverse of the DCT-II. Furthermore, the matrices are unitary, i.e.

$$\mathbf{T}_{\mathrm{DCT-II},N}^{-1} = \mathbf{T}_{\mathrm{DCT-II},N}^{\mathsf{T}} \left(= \mathbf{T}_{\mathrm{DCT-III},N}\right). \tag{2.15}$$

The contribution of each transform coefficient to the reconstruction of the corresponding signal block can be visualized by plotting the set of DCT base pictures

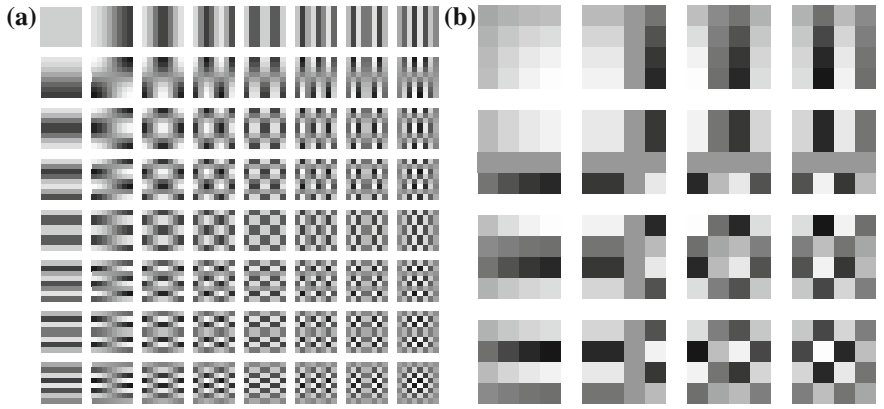

Fig. 2.14 Transform base pictures. *Black* indicates large negative values, *white* indicates large postive values. **a** 8×8 DCT. **b** 4×4 DST

for each transform coefficient. In Fig. 2.14a, the set of DCT base pictures is shown for the DCT of block size $N = 8$. These are generated by reconstructing a $N \times N$ block of transform coefficients with a single non-zero coefficient each. Thereby, the contribution of each coefficient in the block can be visualized. In the figure, the DC base image is located in the top-left corner. The horizontal and vertical frequencies increase to the right and to the bottom, respectively.

2.4.5.3 The DST Matrix

Similar to the DCT-II/DCT-III transform matrices, the matrices for the DST-VI/DST-VII pair can be defined. The DST matrix is denoted as

$$\mathbf{T}_{\mathrm{DST},N} = \begin{bmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_{N-1} \end{bmatrix}. \tag{2.16}$$

The elements of the $N \times N$ DST-VI matrix are defined as

$$t_{n-1}^{\mathrm{VI}}(m-1) = \frac{2}{\sqrt{2N+1}} \cdot \sin\left(\frac{\pi(2m-1)n}{2N+1}\right), \quad m, n = 1, \ldots, N. \tag{2.17}$$

The elements of the $M \times M$ DST-VII matrix are defined as

$$t_{n-1}^{\mathrm{VII}}(m-1) = \frac{2}{\sqrt{2N+1}} \cdot \sin\left(\frac{\pi m(2n-1)}{2N+1}\right), \quad m,n = 1,\ldots,N. \tag{2.18}$$

The DST matrices do not show the same symmetry properties as the DCT. However, some factorization is possible. Approaches for factorization and fast computation have been proposed [17].

Similar to the DCT matrices, the DST-VII is the inverse of the DCT-VI. The matrices are unitary as well, i.e.

$$\mathbf{T}_{\mathrm{DST-VI},N}^{-1} = \mathbf{T}_{\mathrm{DST-VI},N}^{\mathsf{T}} \left(= \mathbf{T}_{\mathrm{DST-VII},N}\right). \tag{2.19}$$

The contribution of the transform coefficients to the reconstructed block can be visualized by base pictures. In Fig. 2.14b, the set of DST base pictures is shown for the DST of block size $N = 4$. The lowest frequency base image is located in the top-left corner. The horizontal and vertical frequencies increase to the right and to the bottom, respectively.

2.4.5.4 The Hadamard Transform Matrix

Due to its low complex implementation (basically only additions) the Hadamard transform is often applied in video encoders for calculation of the sum of absolute transformed difference (SATD), see Sect. 2.5.1.3.

The Hadamard transform matrix is orthogonal with identical elements that only differ in their sign. Omitting normalization, the $N \times N$ matrix with $N = 2^n$ can be recursively derived as[9]

$$\mathbf{T}_{\mathrm{H}1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{T}_{\mathrm{H}n} = \begin{bmatrix} \mathbf{T}_{\mathrm{H}n-1} & \mathbf{T}_{\mathrm{H}n-1} \\ \mathbf{T}_{\mathrm{H}n-1} & -\mathbf{T}_{\mathrm{H}n-1} \end{bmatrix}. \tag{2.20}$$

The resulting matrix has only entries with value ± 1. For normalization, the $\mathbf{T}_{\mathrm{H}n}$ must be multiplied with $\sqrt{2}^{-n}$.

2.4.5.5 Transform and Quantization Scheme

Let $\mathbf{T}$ and $\mathbf{B}$ denote the transform matrix and a residual block, both of size $N \times N$. The transform is unitary, i.e.

$$\mathbf{T}^{\mathsf{T}} \cdot \mathbf{T} = \mathbf{I}, \tag{2.21}$$

[9] The Hadamard transform shares the base vectors with the Walsh transform. In the Walsh matrix, the base vectors are sorted according to increasing 'frequency', i.e. increasing number of sign changes within one base vector, which is comparable to the organization of the DCT base vectors. Omitting normalization, the Walsh transform matrix could also be derived as $\mathbf{T}_{\mathrm{W}} = \mathrm{sgn}\{\mathbf{T}_{\mathrm{DCT}}\}$.

with **I** denoting the identity matrix. The block **B** is transformed into the transform coefficients **C** by horizontal and vertical application of the transform,

$$\mathbf{C} = \mathbf{T} \cdot \mathbf{B} \cdot \mathbf{T}^{\mathsf{T}}. \tag{2.22}$$

With (2.21), the picture block can be perfectly reconstructed applying the inverse transformation procedure,

$$\begin{aligned} \mathbf{B} &= \mathbf{T}^{-1} \cdot \mathbf{C} \cdot \left(\mathbf{T}^{-1}\right)^{\mathsf{T}} \\ &= \mathbf{T}^{\mathsf{T}} \cdot \left(\mathbf{T} \cdot \mathbf{B} \cdot \mathbf{T}^{\mathsf{T}}\right) \cdot \mathbf{T} = \mathbf{I} \cdot \mathbf{B} \cdot \mathbf{I}. \end{aligned} \tag{2.23}$$

In the compression application, the transform coefficients are quantized and encoded for transmission in the bitstream. Since the transform is orthonormal, the Parsival theorem can be applied [10]. Thereby, the quantization error energy introduced by quantization of the transform coefficients is identical to the reconstruction error energy after inverse transform. The quantization aims at the removal of the most irrelevant information for a given bitrate budget. At the decoder side, the quantized transform coefficients $\mathbf{C}_{\mathrm{q}}$ are transformed back to form the reconstructed residual signal $\tilde{\mathbf{B}}$,

$$\tilde{\mathbf{B}} = \mathbf{T}^{\mathsf{T}} \cdot \mathbf{C}_{\mathrm{q}} \cdot \mathbf{T}.$$

The inverse transformation distributes the quantization error induced by a single transform coefficient over the whole reconstructed block. This effect helps to conceal the introduced quantizer distortion for the observer.

2.4.5.6 Quantization

The quantization process maps signal amplitudes to a predefined set of representative values. If individual values are quantized, the process is called scalar quantization. For multidimensional values vector quantization can be applied [10]. In HEVC, as in previous video coding specifications, scalar quantization is used.

Quantization is an inherently non-linear lossy operation, which cannot be inverted. The quantization process is the part of the hybrid video coding scheme which inserts signal degradation and distortion by removing signal information from the coded representation. By careful control of the quantizer settings within the coded picture and over the coded video sequence, the amount of removed irrelevant information (i.e. information which can be removed without noticeable degradation) compared to the amount of removed relevant information, which leads to visible distortions, can be optimized.

The design of the quantizer is driven by the probability distribution of the observed signal amplitudes, balancing the rate needed to encode the quantized values and the distortion introduced by mapping amplitude intervals to a defined reconstruction

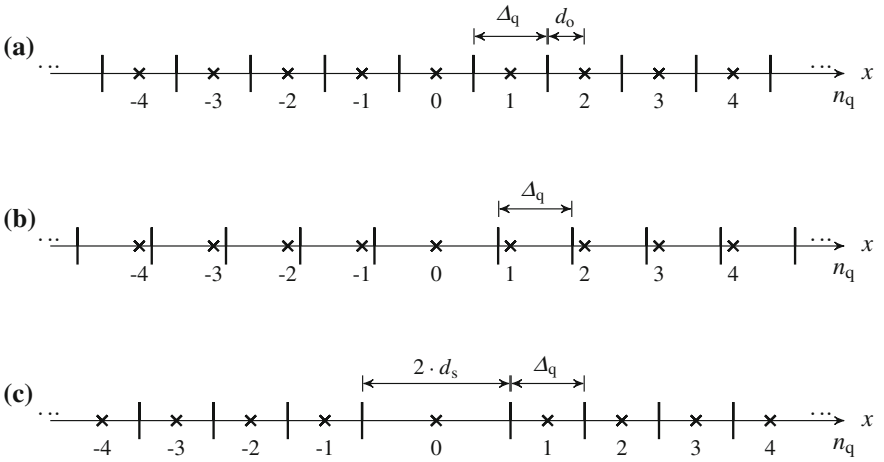

Fig. 2.15 Example scalar quantizer types. An input value x is mapped to the reconstruction value of the n_qth interval. **a** Uniform quantizer (uniform distribution). **b** Uniform quantizer with shifted interval boundaries (Laplacian distribution). **c** Dead-zone quantizer with extended interval around 0

value. Solutions for this optimization problem are described in literature [18]. For a set of known amplitude probability distributions like the uniform or Laplacian distributions, optimum quantizers can be derived and described in a closed form [19]. DCT transform coefficients can be modelled to follow a Laplacian distribution, which lead to the present quantizer design used in DCT based hybrid video coding schemes [20, 21].

Three classical scalar quantizer schemes are shown in Fig. 2.15. For quantization, an input value x is mapped to the n_qth reconstruction value representing the interval which the input value x falls into. n_q is called the quantizer *level*. The intervals are scaled to achieve finer or coarser quantization according to the application needs. The quantizer scaling is expressed by the quantizer step size Δ_q.

In a uniform quantizer, the level n_q for a quantized value x is calculated by

$$n_q = \mathrm{sgn}(x) \cdot \left\lfloor \frac{|x|}{\Delta_q} + d_o \right\rfloor, \tag{2.24}$$

where d_o is an offset which determines the position of the interval boundaries. For example in Fig. 2.15a, b, $d_o = \frac{1}{2}$ and $d_o = \frac{1}{6}$, respectively. These correspond to optimized quantizers for the uniform distribution and the Laplacian distribution, respectively. Reconstruction of the quantized value is trivially achieved by scaling with the quantizer step size,

$$x_q = \Delta_q \cdot n_q. \tag{2.25}$$

The corresponding equations for a dead-zone quantizer as shown in Fig. 2.15c can be given as follows

$$n_{\mathrm{q}} = \begin{cases} 0, & |x| < d_{\mathrm{s}} \\ \mathrm{sgn}(x) \cdot \left\lfloor \frac{|x| - d_{\mathrm{s}}}{\Delta_{\mathrm{q}}} + 1 \right\rfloor, & \text{otherwise} \end{cases} \tag{2.26}$$

and

$$x_{\mathrm{q}} = \begin{cases} 0, & n_{\mathrm{q}} = 0 \\ \mathrm{sgn}(n_{\mathrm{q}}) \big[d_{\mathrm{s}} + \Delta_{\mathrm{q}} \cdot (|n_{\mathrm{q}}| - 1) + d_{\mathrm{o}} \big], & \text{otherwise.} \end{cases} \tag{2.27}$$

In the example of Fig. 2.15c, the parameters were set to $d_{\mathrm{s}} = \Delta_{\mathrm{q}}$ and $d_{\mathrm{o}} = \frac{\Delta_{\mathrm{q}}}{2}$.

The number and values of the applicable quantizer step sizes in a video coding system is limited. The applicable quantizer step size is usually indicated by the *quantization parameter* (QP), which serves as an index to a predefined set of applicable quantization step sizes. Commonly, low quantization parameters correspond to fine quantization and high quantization parameters correspond to coarse quantizer step sizes. A high granularity of the quantizer step sizes is beneficial to allow for precise rate control in the encoded bitstream. On the other hand, the signaling for a high number of available quantizer step sizes induces additional coding cost, which needs to be considered. The specification has to balance granularity and coding cost.

For the orthonormal transform, application of the same quantizer step size to the transform coefficients regardless of their frequency index induces the minimum mean squared error for the reconstructed block [12]. However, this approach does not consider the subjective impact of quantization on the human visual system if applied to coefficients which correspond to different spectral frequencies in the block. In order to enable adaptation of the quantization scheme in this regard, a quantization weighting matrix $\mathbf{W}_{\mathrm{q}}$ of transform block size $N \times N$ can be applied. This matrix defines an additional scaling of the quantizer step size in dependence of the transform coefficient which is quantized,

$$\mathbf{W}_{\mathrm{q}} = \begin{bmatrix} w_{\mathrm{q}0,0} & w_{\mathrm{q}0,1} & \cdots & w_{\mathrm{q}0,N-1} \\ w_{\mathrm{q}1,0} & w_{\mathrm{q}1,1} & \cdots & w_{\mathrm{q}1,N-1} \\ \vdots & & \ddots & \vdots \\ w_{\mathrm{q}N-1,0} & w_{\mathrm{q}N-1,0} & \cdots & w_{\mathrm{q}N-1,N-1} \end{bmatrix}. \tag{2.28}$$

For the transform coefficient $c_{i,j}$, the applicable quantizer step size is scaled to

$$\Delta'_{\mathrm{q}}(QP, i, j) = w_{\mathrm{q}}(i, j) \cdot \Delta_{\mathrm{q}}(QP) \tag{2.29}$$

for both, quantization and inverse quantization. Quantization weighting matrices can e.g. be designed to provide the finest quantization for the low frequencies while increasing the quantizer step size with increasing frequency index.

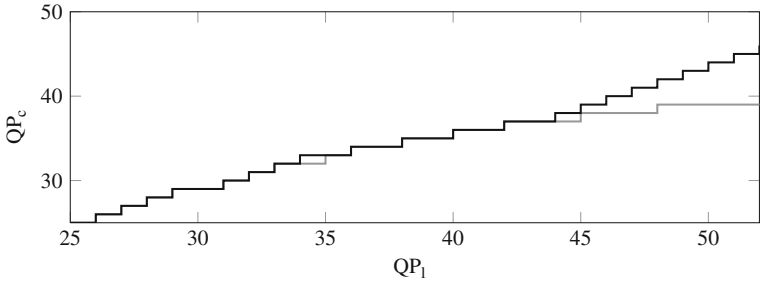

Fig. 2.16 Schematic presentation of the mapping of the luma quantization parameter QP_l to the chroma quantization parameter QP_c for HEVC (*black*) and H.264 | AVC (*gray*)

For YCbCr 4:2:0 video, the signal characteristics of luma and the two chroma components are quite different. Specifically, chroma often exhibits a strong lowpass character. If strong quantization is applied, the chroma information may be completely quantized to zero, which would lead to the complete loss of color. Accordingly, in order to reduce this the quantizer step size for chroma is adapted by reducing the chroma quantizer step size for high QP values.

As an example, the correspondence between the luma quantization parameter QP_l and the chroma quantization parameter QP_c is shown for the standards H.264 | AVC and HEVC in Fig. 2.16. For QP values below $QP_l = 30$ in the QP range $0 \leq QP_l \leq 51$, the chroma quantization parameter is aligned to luma. At high QP values, the chroma is kept at a finer quantization than the luma component.

2.4.5.7 Integer Transforms

Before H.264 | AVC, the transformation in video coding standards was specified in floating point operations. Since deviations in different floating point implementations of encoders and decoders may easily lead to differences in the inverse transform result, a significant effort was spent to define conformance conditions such that it could be ensured that two independent DCT implementations would induce the same (or almost the same) reconstructed block. The normative requirements for this task were specified by the IEEE in [22]. After the withdrawal of this standard in 2003, MPEG approved the replacement specification ISO/IEC 23002-1 in 2006 [23].

In order to dispose this transform conformance issue, the specification of integer transforms has been introduced. In H.264 | AVC and HEVC, the inverse transforms are specified in integer arithmetic. In fact, all mathematical operations specified in the decoding process are in integer arithmetic in these standards. As a recommended solution for the previously existing standards, a fixed-point 8×8 IDCT and DCT have been specified in ISO/ICE 23002-2 [24]. This transform pair fulfills the requirements of ISO/IEC 23002-1 and is recommended for implementation as it enables bit-exact reconstruction if used for both, encoder and decoder implementations.

Integer DCT Matrices

The integer approximations of the DCT are called integer DCTs in the following. Several design aspects rule into the development of these transforms. These include the required dynamic range in the transformation process, the approximation precision, and the implementation friendliness in terms of required multiplications, additions, and bit-shifts.

Three example 8×8 integer DCT matrices which follow different design criteria are given below:

$$\mathbf{T}_{8,\mathrm{a}} = \begin{bmatrix} 17 & 17 & 17 & 17 & 17 & 17 & 17 & 17 \\ 24 & 20 & 12 & 6 & -6 & -12 & -20 & -24 \\ 23 & 7 & -7 & -23 & -23 & -7 & 7 & 23 \\ 20 & -6 & -24 & -12 & 12 & 24 & 6 & -20 \\ 17 & -17 & -17 & 17 & 17 & -17 & -17 & 17 \\ 12 & -24 & 6 & 20 & -20 & -6 & 24 & -12 \\ 7 & -23 & 23 & -7 & -7 & 23 & -23 & 7 \\ 6 & -12 & 20 & -24 & 24 & -20 & 12 & -6 \end{bmatrix} \tag{2.30}$$

$$\mathbf{T}_{8,\mathrm{b}} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \tag{2.31}$$

$$\mathbf{T}_{8,\mathrm{c}} = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix} \tag{2.32}$$

Matrix $\mathbf{T}_{8,\mathrm{a}}$ in (2.30) is an orthogonal, single-norm integer approximation of the 8×8 DCT matrix [25]. The matrix $\mathbf{T}_{8,\mathrm{b}}$ in (2.31) corresponds to the inverse 8×8 transform in the High profiles of H.264 | AVC [26, 27]. The coefficient values of this matrix allow for an implementation of the inverse transform process with 16-bit dynamic range. The base vectors of this matrix are orthogonal but have diverging norms. These are dealt with in the design of the corresponding integer quantization process. The last matrix $\mathbf{T}_{8,\mathrm{c}}$ in (2.32) corresponds to the inverse 8×8 transform in

HEVC [4]. The base vectors of this transform have approximately the same norm but are not exactly orthogonal, i.e. the product $\mathbf{T}_{8,c} \cdot \mathbf{T}_{8,c}^{\mathsf{T}}$ does not result in a diagonal matrix. The deviation from orthogonality however is so small that it is leveled out by normalization during the quantization and reconstruction processes. The HEVC transform and quantization design is further detailed in Chap. 8.

Normalization and Quantization

By definition, integer DCT matrices can not be orthonormal. Therefore, normalization must be implemented into the coding scheme in this case. For the discussion, a uniform quantizer and a $N \times N$ orthogonal integer DCT matrix are assumed, i.e.

$$\mathbf{T} \cdot \mathbf{T}^{\mathsf{T}} = \|\mathbf{T}\|^2 \cdot \mathbf{I}. \tag{2.33}$$

The transform and quantization operation to generate the quantizer levels $\mathbf{n}_{\mathrm{q}}$ for a residual block $\mathbf{X}$ at the encoder side can be written as

$$\mathbf{n}_{\mathrm{q}} = \mathrm{round}\left\{ \frac{1}{\|\mathbf{T}\|^2 \cdot \Delta_{\mathrm{q}}} \cdot \mathbf{T} \cdot \mathbf{X} \cdot \mathbf{T}^{\mathsf{T}} \right\}, \tag{2.34}$$

where Δ_{q} is the quantizer step size. Accordingly, the normalization at the decoder side has to be included in the reconstruction $\mathbf{X}_{\mathrm{r}}$ of the residual block:

$$\mathbf{X}_{\mathrm{r}} = \mathrm{round}\left\{ \frac{\Delta_{\mathrm{q}}}{\|\mathbf{T}\|^2} \cdot \mathbf{T}^{\mathsf{T}} \cdot \mathbf{n}_{\mathrm{q}} \cdot \mathbf{T} \right\}. \tag{2.35}$$

In order to enable the implementation of the divisions in (2.33) and (2.35) as bit-shift operations, the quotient factors can be designed as

$$\frac{f_{\mathrm{q}}}{2^{N_{\mathrm{e}}}} \approx \frac{1}{\left\|\mathbf{T}^2\right\| \cdot \Delta_{\mathrm{q}}} \quad \text{and} \quad \frac{g_{\mathrm{q}}}{2^{N_{\mathrm{d}}}} \approx \frac{\Delta_{\mathrm{q}}}{\|\mathbf{T}\|^2}, \tag{2.36}$$

with appropriate bit-shift values N_{e} and N_{d} at the encoder and decoder side, respectively. For integer DCTs with base vectors of different norm (e.g. in H.264 | AVC), separate f_{q} and g_{q} values have to be specified according to the applicable normalization of the respective transform coefficients.

2.4.6 In-loop Filtering

Loop filtering is a means to improve the reconstruction quality of the picture for display. Since the filter is located within the loop, the enhancement not only affects the quality of the output pictures but also the reference pictures, which are available

for prediction when coding succeeding pictures. Thereby, loop filters have a strong impact on the overall performance of the video coding scheme. Optimization criteria can be in terms of visual quality but also in terms of objective fidelity measures such as the minimization of the mean squared error between the original picture and the reconstructed picture.

Conceptually, two classes of loop filter operations can be distinguished: The first class comprises linear filters which are applied to an area of a picture or even a complete picture. For the selected area, the filtering operation can be performed as a convolution with the filter impulse response in the spatial domain or as a multiplication with the filter transfer function in the frequency domain. The filter configuration can be flexible or static over the video sequence and the applicable filter parameters may be determined at the encoder side according to a selected optimization criterion, e.g. using a Wiener filter approach [10, 12].[10]

The filters of the second class of loop filters operate on a local spatial domain of the picture. Based on local features, the applicable filtering operation for a small set of samples is determined. The H.264 | AVC and HEVC deblocking filters are representatives of this class. A deblocking filter deals with block boundary structures which are induced by block-wise motion compensated prediction and block-wise processing of the residual signal. The edges of these blocks become visible in the reconstructed pictures due to quantization of the residual signal such that perfect reconstruction of the original picture is not achieved. In order to mitigate this effect, the deblocking filter operates in a local area across prediction and transform block boundaries. The strength of the deblocking filter is locally controlled by the detected amount of 'blockiness' at the processed boundary. The realization of the HEVC deblocking filter and the corresponding deblocking filter control are discussed in Sect. 9.1.

Another representative of local loop filters is sample-adaptive offset (SAO) filtering, which is newly specified in HEVC and has not been included in a ITU-T or ISO/IEC video coding standard before. SAO operates on a sample basis, independent from the block structures of the reconstructed picture. Two types of SAO filtering are defined in HEVC: Derivation of a sample level correction based on the local neighborhood, or derivation of a sample level correction based on the intensity level of the sample itself. Conceptually, sample-adaptive offset filtering can be considered as a picture restoration operation. While such filters may commonly be applied to the reconstructed video in a post-processing stage, it is integrated into the coding loop in this case. SAO can be used e.g. to effectively reduce ringing artifacts or to correct level values which have been shifted due to quantization. The operation of the HEVC sample-adaptive offset filter is further detailed in Sect. 9.2.

The applicable loop filters are jointly denoted by "Loop Filter" in the block diagram of Fig. 2.8. The filters operate on a slice basis, as the filtering is applied across

[10] During the development of HEVC, the specification of an adaptive loop filter was evaluated. This filter partitioned the picture into filtering blocks on a quadtree basis and applied adaptive filters to the partitions. In the final design, the overall trade-off between compression improvement and implementation cost was considered a too high burden and this loop filter type was not included in the HEVC specification [28].

block boundaries within slices. In H.264 | AVC and HEVC, the deblocking filter can also be configured to be applicable across slice boundaries.

A filter process within the coding loop has already been present in the H.261 specification [29]. There, a separable three-tap averaging filter can be applied on an 8×8 block basis. The filtering is activated on a macroblock basis. In contrast to H.264 | AVC and HEVC, which apply the loop filter after reconstruction as shown in Fig. 2.8, the H.261 loop filter is located after the motion compensation stage. Thereby, the inter prediction signal before addition of the reconstructed residual is filtered instead of the reconstructed picture. Since H.261 only specifies full-sample motion compensated prediction, this application of the loop filter can be interpreted as a replacement for missing sub-sample motion vector precision. If a block does not fit the full-sample motion compensation, the prediction is improved by lowpass filtering the motion compensated block.

2.4.7 The Decoded Picture Buffer

The decoded picture buffer holds the decoded reconstructed pictures until they are scheduled for display. This is specifically necessary if the decoding order of the pictures diverges from the output order of the pictures. In addition to the storage for display, pictures can be stored for use as reference pictures in inter prediction. In the block diagram of Fig. 2.8, the decoded picture buffer is denoted by "Buffer".

The size of the buffer defines limits for the applicable temporal coding structures applied to a video sequence. Conceptually, two types of reference pictures are differentiated in the buffer: short-term and long-term reference pictures. Short-term reference pictures denote reference pictures from the temporal proximity of the current picture to be predicted. Long-term reference pictures are explicitly marked as such. These pictures can be used to store a picture with certain scene content for reference usage on a larger temporal scale, e.g. if some scene content appears repeatedly after interruptions by other content. The handling of reference pictures and the decoded picture buffer in HEVC are detailed in Sects. 4.3 and 5.6.2.

2.4.8 Entropy Coding

The syntax elements which represent the quantized transform coefficients, as well as the applicable prediction modes, motion vectors, intra prediction directions, etc. need to be coded into the bitstream. The values of the decoded syntax elements are used in the decoding process to derive the intended reconstructed video sequence. The operation of mapping the syntax elements into the bitstream is done by the building block "Entropy Coding" in the block diagram of Fig. 2.8. The entropy coding stage maps the incoming syntax elements, such as flags, indices, vectors, modes, or coefficients, to binary code words and bit-string representations. By the design and

layout of the assigned coding methods, the entropy coding stage has significant impact on the overall compression efficiency of a video coding scheme.

Depending on the type of coded information, the design criteria for the entropy coding stage differ. Syntax elements indicating high-level properties of the bitstream are usually coded using code word representations which are easy to access not only by the decoder but also by other applications which may want to acquire information regarding general features of the bitstream. Here, *fixed length codes* are often employed. Such information is often placed at prominent positions in the bitstream for easy access. Byte aligned positions relative to defined starting points alleviate access to relevant syntax elements in the bitstream. Since such information usually comprises only a very small portion of the bitstream, fixed-length code words can be used without much penalty in the coding efficiency. Syntax elements which convey information at the picture or slice level are coded using *variable length codes* (VLCs) as appropriate. For these code words, a trade-off regarding decoding complexity and compactness of the representation has to be found during the design of the coding scheme. The highest effort in terms of an efficient entropy coding representation is spent on the slice and block level. This part of the coded information typically covers the vast majority of the bitstream. State-of-the-art video coding schemes apply context-dependent adaptive coding at this level.[11]

In H.264 | AVC, a context adaptive VLC based scheme (CAVLC) and a context adaptive binary arithmetic coding scheme (CABAC) were specified and employed on the block level, depending on the selected standard profile. In HEVC, only context-based adaptive binary arithmetic coding has been selected after thorough investigation on the performance, computational complexity, and description complexity impact [31].

In the following, the basic principles of coding with variable length codes and arithmetic coding are summarized. For further study, the reader is referred to literature, e.g. starting from [10, 32, 33]. VLCs and CABAC as used in HEVC are detailed in Chap. 10.

2.4.8.1 Entropy and Information Content

Let $S = \{s_0, s_1, \ldots, s_{N-1}\}$ be an alphabet of N symbols s_n. Let the symbols be sent by a source, let the symbols be independent and identically distributed, and let each symbol s_n occur with a probability $p_S(n)$. When a source sends symbols of this alphabet, the information content of each symbol can be computed as

$$I(s_n) = \log_2\left(\frac{1}{p_S(n)}\right) = -\log_2\left(p_S(n)\right). \tag{2.37}$$

[11] For the HEVC Random Access configuration according to the JCT-VC common testing conditions [30], the portion of the bitstream which is not encoded with CABAC is in the range of 0.1–1.0 %.

The logarithm to the base 2 applies for a coded representation with binary values. For such a binary code, the information content can be interpreted as the minimum required code word length to represent the given symbol s_n. The entropy of the source is defined as the average information content of the source,

$$H = \sum_{n=0}^{N-1} p_S(n) \cdot I(s_n) = -\sum_{n=0}^{N-1} p_S(n) \cdot \log_2 \big(p_S(n)\big). \tag{2.38}$$

According to the foundations of information theory, the entropy corresponds to the least number of bits per symbol that need to be spent for encoding, if the symbols shall be uniquely decodable.

When designing a coding scheme for a given source, the entropy of the source thereby provides the lower bound on the achievable compression. The assumption of independent or uncorrelated symbols, which was used in the definition above, does not generally hold. Instead, determining and utilizing the correlation between successive symbols as well as the impact on the coding context on the symbol probability distribution is crucial for the derivation of a (close-to) optimum coding scheme. However, the achievable gain in terms of representation compactness always has to be balanced with the accompanying algorithmic effort that has to be spent.

2.4.8.2 Fixed and Variable Length Codes

As stated above, fixed length codes are used for coding high-level syntax in a video coding scheme. Trivially, a flag which has two states can be coded with 1 bit. For syntax elements with more values, the applicable number of bits for the fixed-length code must be specified. Typically, the value of the syntax element is directly coded as its binary representation. Depending on the probability of the available syntax element values and observing (2.38), this representation can be more or less efficient.

When designing VLCs, various strategies can be applied. Under the assumption that the probability distribution of the symbols in the alphabet is known (i.e. here the probability for the values of the syntax element under consideration), the VLC design method proposed by Huffman can be applied [34]. The method guarantees a prefix-free set of resulting code words, i.e. a bitstream consisting of these code words can be unambiguously decoded. The length of the resulting code words increases with decreasing probability of the corresponding symbol. It can be shown that by design, the average code length of a Huffman code is bound by $H + 1$ [33].

While Huffman codes provide an efficient coded representation for a given source alphabet, the resulting code words are not systematic. Therefore, code word tables have to be stored at both encoder and decoder side in the coding application. An alternative method is the usage of *systematic codes*, which do not require the storage of code word tables at the decoder side but are rather generated on the fly by a specified construction rule, see e.g. [35].

The construction of the applicable VLCs can further be made adaptive to better fit variable source statistics. Strategies for the design of context adaptive VLCs include

Table 2.4 Unary code

v	$C_{\mathrm{u}}(v)$
0	1
1	0 1
2	0 0 1
3	0 0 0 1
⋮	⋮

context-based selection of VLC tables or adaptive assignment of code words to syntax element values, see e.g. [36].

In the following, some systematic VLCs are presented which play a role in the context of HEVC. The presentation focuses on unsigned integer values. A codeword that is assigned to a value $v \geq 0$ is denoted by $c = C(v)$. The value is decoded from the codeword by $v = C^{\mathrm{i}}(c)$.

2.4.8.3 Unary Code

A *unary* code $C_{\mathrm{u}}(v)$ is a systematic code that represents an non-negative integer value v by v bits of one parity and a stop-bit of the other parity. A schematic presentation of the code is given in Table 2.4. Given a value v, the code word consists of v 0-bits and a terminating 1-bit. The length of the code word is

$$n_{\mathrm{c}} = v + 1. \tag{2.39}$$

Accordingly, the value is decoded from the code word by

$$\begin{aligned} v &= C_{\mathrm{u}}^{\mathrm{i}}(c) \\ &= n_{\mathrm{c}} - 1. \end{aligned} \tag{2.40}$$

2.4.8.4 Golomb Codes

Golomb codes are a family of systematic codes that can be adapted to the source statistics and are thereby well suited for coding applications [35, 37]. Golomb codes are generally constructed by a prefix and a suffix part.

Golomb-Rice Codes

A Golomb-Rice code $C_{\mathrm{gr}\,k}(v)$ of grade k is constructed by a unary coded prefix and k suffix bits [38]. An example is given in Table 2.5 for $k = 4$. In the table and in the following $x_0, x_1, \ldots, x_n$ denote bits of the code word with $x_i \in \{0, 1\}$.

Table 2.5 Golomb-Rice code of order $k = 4$

v	$C_{\mathrm{gr}4}(v)$
$0, \ldots, 15$	$1\ x_3\ x_2\ x_1\ x_0$
$16, \ldots, 31$	$0\ 1\ x_3\ x_2\ x_1\ x_0$
$32, \ldots, 47$	$0\ 0\ 1\ x_3\ x_2\ x_1\ x_0$
$\vdots$	$\vdots$

Let the code be used for unsigned integer values and the suffix be the k-bit binary representation of an integer $0 \leq i < 2^k$. The number of prefix bits is denoted by n_{p}, the number of suffix bits is denoted by n_{s}. For the Golomb-Rice code, the number of suffix bits is $n_{\mathrm{s}} = k$. When encoding a value v, the number of prefix bits is determined by

$$n_{\mathrm{p}} = 1 + \left\lfloor \frac{v}{2^k} \right\rfloor. \tag{2.41}$$

The suffix then is the n_{s}-bit binary representation of

$$v_{\mathrm{s}} = v - 2^k (n_{\mathrm{p}} - 1). \tag{2.42}$$

Accordingly, the value v can be reconstructed from the code word c by

$$\begin{aligned} v &= C^{\mathrm{i}}_{\mathrm{gr}\,k}(c) \\ &= 2^k (n_{\mathrm{p}} - 1) + \sum_{i=0}^{k-1} x_i \cdot 2^i. \end{aligned} \tag{2.43}$$

Exp-Golomb Codes

While the Golomb-Rice codes use a suffix of fixed length, it is also possible to determine the length of the suffix by the length of the prefix. Exponential Golomb codes (Exp-Golomb) follow this approach [39].

A kth-order Exp-Golomb code $C_{\mathrm{eg}\,k}(v)$ is constructed by a unary prefix code and a suffix of configurable length. The number of bits in the suffix n_{s} is determined by the value n_{p} of the prefix code, where

$$n_{\mathrm{s}} = k + n_{\mathrm{p}} - 1. \tag{2.44}$$

The number of prefix bits n_{p} of $C_{\mathrm{eg}\,k}(v)$ is determined from the value v by

$$2^k \left(2^{n_{\mathrm{p}}-1} - 1\right) \leq v < 2^k \left(2^{n_{\mathrm{p}}} - 1\right). \tag{2.45}$$

Table 2.6 Exp-Golomb codes of order $k = 0$ and $k = 1$

v	$C_{\mathrm{eg}\,0}(v)$	v	$C_{\mathrm{eg}\,1}(v)$
0	1	0,1	$1\ x_0$
1,2	$0\ 1\ x_0$	$2, \ldots, 5$	$0\ 1\ x_1\ x_0$
$3, \ldots, 6$	$0\ 0\ 1\ x_1\ x_0$	$6, \ldots, 13$	$0\ 0\ 1\ x_2\ x_1\ x_0$
$7, \ldots, 14$	$0\ 0\ 0\ 1\ x_2\ x_1\ x_0$	$14, \ldots, 29$	$0\ 0\ 0\ 1\ x_3\ x_2\ x_1\ x_0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

The suffix is then the n_s-bit binary representation of $v_\mathrm{s} = v - 2^k \left(2^{n_\mathrm{p}-1} - 1\right)$.

A value v can be reconstructed from the code word $c = C_{\mathrm{eg}\,k}(v)$ code as

$$\begin{aligned} v &= C^{-1}_{\mathrm{eg}\,k}(c) \\ &= 2^k \left(2^{n_\mathrm{p}-1} - 1\right) + \sum_{i=0}^{n_\mathrm{s}-1} x_i \cdot 2^i . \end{aligned} \tag{2.46}$$

Examples for Exp-Golomb codes with $k = 0$ and $k = 1$ are given in Table 2.6. The 0th-order Exp-Golomb code $C_{\mathrm{eg}\,0}(v)$ is used for coding of signed and unsigned values in HEVC, see Sect. 10.1.

2.4.8.5 Arithmetic Coding

With *arithmetic coding*, the direct connection between the symbol and the bitstream representation is abolished. In contrast to variable length codes, symbols are represented by coded number intervals instead of explicit code words. A sequences of symbols can be represented by a single value from a defined interval in a given number range. Thereby, arithmetic coding conceptionally allows for encoding symbols at fractional numbers of bits. This is particularly useful for symbols which occur with a very high probability. According to (2.37), such symbols have an information content of less than 1 bit, which would lead to an inefficient representation by a VLC code word (which by nature must have a length of at least 1 bit). Arithmetic coding is used in H.264 | AVC and HEVC at the slice level for highest compression efficiency. It has been specified and used in image and video compression standards such as H.263 Annex E or JPEG2000 before [35, 40]. The basic concept of arithmetic coding is illustrated here. The arithmetic coding scheme using integer arithmetic as specified in HEVC is detailed in Chap. 10.

Let the available number range be [0, 1], and let the symbols of an alphabet {A, B} occur with probability $p_\mathrm{A} = 0.7$ and $p_\mathrm{B} = 1 - p_\mathrm{A} = 0.3$, respectively. Since the probability for symbol A is higher than for symbol B, symbol A is called the most probable symbol (MPS). Symbol B is denoted the least probable symbol (LPS). For encoding a sequence of these symbols, a representative interval is derived and the

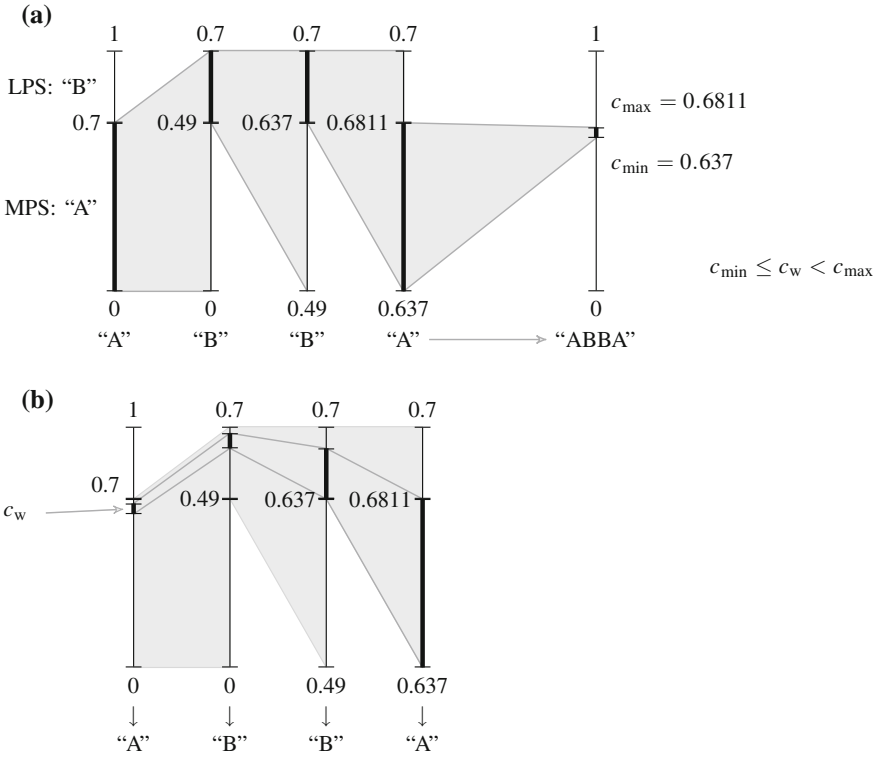

Fig. 2.17 Example for arithmetic coding of a sequence of binary symbols "A, B, B, A" with probabilities $p_A = 0.7$ and $p_B = 0.3$. **a** Arithmetic encoding corresponds to an interval subdivision. The sequence can be represented by a code word c_w from the last interval. **b** Arithmetic decoding compares the code word to the interval boundaries to successively reconstruct the sequence

sequence can be identified by any number from that interval. The process of interval determination is illustrated in Fig. 2.17a. The symbols are assigned to intervals of the available range according to their probability. In the coding step for each symbol, the corresponding interval is selected and taken as the available range for the next symbol. With each step, the corresponding interval is refined. If a pre-defined minimum interval size is reached, a rescaling operation is applied and the encoding process is continued. This is also the point in time where the bits for the previous interval indicators are written into the bitstream. For decoding, a corresponding set of steps has to be performed as illustrated in Fig. 2.17b. The decoded interval indicator is successively compared to the interval boundaries and the symbols corresponding to the intervals are thereby successively decoded. If the successively reduced interval size indicates the need for rescaling, further bits are read from the bitstream and the decoding process continues. The binary scheme as shown here can be similarly extended to alphabets with more than two symbols, e.g. as used in H.263.

A substantial advantage of arithmetic coding is the possibility to update the relation of the interval boundaries for each symbol after each coding step. Thereby, the size of the interval can be adapted to the estimated statistics of the encoded symbols. Context-based adaptive binary arithmetic coding as used in H.264 | AVC and HEVC utilizes this possibility for highest compression efficiency. Conceptually, similar results could be achieved with non-binary arithmetic coding. In terms of implementation complexity but also in terms of symbol representation (binarization), binary arithmetic coding has been found to be the better choice [41].

2.5 Encoder Control

The block diagram in Fig. 2.8 does not include the notion of encoder control to simplify the diagram. In fact, virtually all building blocks in the block diagram are affected by it. The encoder control takes all decisions related to coding of the pictures of the video sequence into the bitstream according to the application requirements.

As a fundamental property, the encoder control must ensure to generate a bitstream conforming to the requirements of the given video coding specification.The most important task in most applications is, within these constraints, to optimize the bitrate (or the bitstream size) and the reconstructed video quality. This could e.g. mean to minimize the bitrate given that the reconstruction quality does not fall below a predefined level, or to maximize the reconstructed video quality subject to given bitrate constraints. Another very important aspect of encoder control is to ensure that the amount of bits sent to the decoder side is regulated such that the decoder buffer for the incoming bitstream does neither overfill nor run empty. This task is e.g. achieved by application of a hypothetical reference decoder model as further discussed in Sect. 5.6. In low-delay and live applications, a fast encoder operation must be achieved which satisfies real-time constraints. In such application scenarios, the video is usually directly processed upon availability to the encoder. If the video material is processed offline for later usage, the encoding speed is of less importance. In such application scenarios, elaborate encoder optimization strategies can be applied and multi-pass encoding may be used. Multi-pass encoding allows the encoder control to make use of 'look-ahead' knowledge when encoding the picture of the video sequence. Thereby, bitrate can be efficiently invested for better prediction in preparation of forthcoming content.

The decisions the encoder control needs to take include the applicable partitioning of the picture as well as the partitioning of the coding blocks. Furthermore, the applicable inter and intra prediction modes, motion vectors, loop filtering modes, the applicable transforms, and the reference picture management are concerned. The selection of the applicable motion vectors and the corresponding reference pictures are a major task. The motion vector determination by the motion estimation stage usually covers a significant portion of the encoder processing time. A variety of fast search strategies have been developed to reduce the

computational complexity burden of this task while keeping the prediction quality as high as possible.

The fundamental decision criterion at the encoder side is to balance the bitrate to be spent against the corresponding reconstruction quality that is achieved. The criterion is used on the block level for decision on prediction modes as well as on a larger level, e.g. when deciding on the applicable picture structure. The bitrate cost can be directly measured by counting the bits that would be spent to encode a block in a given mode. The more complete this analysis is, the more precise and reliable the result will be (e.g. including the encoding cost of the quantized transform coefficients). In a VLC-based solution this can be achieved by corresponding table look-ups. With arithmetic coding, determining the precise number of bits to be spent requires more effort as trial encoding and appropriate coder state management needs to be implemented for a precise measurement.

For determination of the reconstruction quality, the objective distortion of the reconstructed picture compared to the original picture is often measured. This measurement is commonly taken on a sample by sample basis. The corresponding distortion measures as used in the HEVC reference software are summarized in the following subsection. Conceptually, also more elaborate measures that e.g. take into account the modelled subjective impression by an observer can be applied. Such measures might be able to better describe the visual impact of a encoder decision. For effective application in an encoder implementation such measures must be operable on a block basis as well as a picture basis. Due to their effectiveness, the simple difference measures as presented below are widely used.

2.5.1 Distortion Measures

In the following, the most commonly used sample-based distortion measures are summarized. Let the current block be denoted by $\mathbf{B}_c$ with $N \times M$ samples $b(n, m)$, and let the prediction block be $\mathbf{P}$ with $N \times M$ prediction samples $p(n, m)$.

2.5.1.1 SSD—Sum of Squared Difference

The sum of squared differences (SSD) or sum of squared error (SSE) is an evaluation measure commonly used in signal processing. The sum of the squared differences between $\mathbf{B}_c$ and $\mathbf{P}$ corresponds to the squared Euclidian norm or the squared L_2-norm in a vector space. It is defined as

$$D_{\mathrm{SSE}} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left| b(n, m) - p(n, m) \right|^2. \tag{2.47}$$

The measure corresponds to the energy of the error between $\mathbf{P}$ and $\mathbf{B}_c$. Considering $\mathbf{P}$ as an approximation of $\mathbf{B}_c$, the difference signal can be interpreted as noise that has been added to the original signal block.

2.5.1.2 SAD—Sum of Absolute Differences

The sum of absolute differences is defined similarly to the SSE but omits squaring the summation elements. The SAD corresponds to the L_1 norm in a vector space and is defined as

$$D_{\mathrm{SAD}} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \big| b(n,m) - p(n,m) \big|. \tag{2.48}$$

The SAD can be seen as the simplest similarity metric. It is used for motion estimation and also in image analysis applications.

2.5.1.3 SATD—Sum of Absolute Transformed Differences

The sum of absolute transformed differences applies a Hadamard transform before calculating the sum of absolute differences. While the SAD itself serves as a measure for the difference between the samples in the blocks, the preceding transform in the SATD provides an additional estimate for the effort needed for transform coding the difference signal, the residual. Thereby, the SATD serves as a combined metric, for the block difference as well as for the coding cost of the residual.

For calculation of the SATD, the prediction residual is transformed by the Hadamard transform as

$$\mathbf{R}_{\mathrm{H}} = \mathbf{T}_{\mathrm{H}} \cdot (\mathbf{B} - \mathbf{P}) \cdot \mathbf{T}_{\mathrm{H}}^{\mathsf{T}}. \tag{2.49}$$

The SATD is then calculated as

$$D_{\mathrm{SATD}} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \big| r_{\mathrm{H}}(n,m) \big|. \tag{2.50}$$

Though having an increased complexity compared to the simple SAD, the features described above make the SATD a very suitable metric for distortion measurement in the context of motion estimation.

2.5.1.4 PSNR—Peak Signal-to-Noise Ratio

Since the conduction of formal subjective tests is an expensive and tedious process, *objective* testing methods are employed, which are considered to sufficiently

approximate results of corresponding subjective testing campaigns. Objective testing methods have the advantage of being exactly reproducible by independent evaluators for a given set of test material.

The most commonly used objective quality measure in video coding is the peak signal-to-noise ratio, PSNR. The PSNR measured the relation of the mean squared error between the original signal (i.e. picture) p_{orig} and the reconstructed signal p_{rec} to the available maximum amplitude of the original signal,

$$\begin{aligned}\mathrm{PSNR} &= 10 \cdot \log_{10}\left(\frac{A_{\max}^2}{\frac{1}{N_{\mathrm{P}} N_{\mathrm{L}}} \sum_{n=0}^{N_{\mathrm{P}}-1} \sum_{m=0}^{N_{\mathrm{L}}-1} \left|p_{\mathrm{orig}}(n,m) - p_{\mathrm{rec}}(n,m)\right|^2}\right) \\ &= 10 \cdot \log_{10}\left(N_{\mathrm{P}} N_{\mathrm{L}} \cdot \frac{A_{\max}^2}{D_{\mathrm{SSE}}}\right) \mathrm{dB}, \end{aligned} \tag{2.51}$$

with $A_{\max} = 255 = 2^8 - 1$ for 8-bit video.

Commonly, the PSNR is measured on a picture basis and the average is taken as the PSNR value for a full sequence. This measurement method is valid if quality fluctuations between successive pictures of the decoded sequence do not specifically need to be taken into account. Otherwise, the D_{SSE} for the full sequence and the corresponding PSNR can be evaluated.

For evaluation, rate-distortion (RD) plots are used, showing the sequence PSNR over the bitrate of the corresponding bitstream. The bitrate is usually calculated from the file size of the coded bitstream divided by the duration of the decoded sequence in seconds and is measured in kbit/s. Depending on the measurement task, the bitrate can also be calculated only from parts of the bitstream, e.g. if specific side information like supplemental enhancement information shall not be taken into account.

2.5.2 Rate-Distortion Optimization

Let each coding tree block (or each macroblock) be coded using a specific coding mode. The coding mode comprises the applicable prediction mode, the corresponding prediction parameters, and the applicable block partitioning for transform coding. With all available coding modes at hand, the encoder has to decide which coding mode with what prediction parameters to apply for each block (e.g. each coding tree block or each macroblock) in each picture of the video sequence, in order to optimize the output in terms of the used bitrate R and observed distortion D induced by the selected coding modes. This problem can be solved by using classical rate-distortion optimization, applied on a picture basis [42].

Let the complete picture be partitioned into a set of N coding tree blocks, $P = \{\mathrm{CTB}_n\}, n = 0, \ldots, N-1$, and let the coding mode for coding tree block CTB_n be denoted by a coding mode index M_n. The set of applied coding modes for the picture

is denoted by $M = \{M_n\}$, $n = 0, \ldots, N-1$. The optimization task for the encoder is to find the best coding mode set M_{opt} for the current picture subject to a given rate constraint R_c, such that the distortion in the picture is minimized,

$$\min_M \big(D(P, M)\big), \text{ with } R(P, M) < R_c. \tag{2.52}$$

Using the Lagrangian weighting factor λ, this constrained problem is transformed into an unconstrained problem,

$$M_{\text{opt}} = \arg\min_M \big(J(P, M|\lambda)\big), \tag{2.53}$$

with the joint cost criterion

$$J(P, M|\lambda) = D(P, M) + \lambda \cdot R(P, M). \tag{2.54}$$

When encoding a picture, it is assumed that the cost for the coding tree blocks is additive. Under this assumption, the total minimum joint cost for coding the picture can be determined by selecting the minimum joint cost for each coding tree block,

$$\min_M \sum_{n=0}^{N-1} J(\text{CTB}_n, M|\lambda) = \sum_{n=0}^{N-1} \min_{M_n} \big(J(\text{CTB}_n, M_n|\lambda)\big). \tag{2.55}$$

This assumption does not take into account potential dependencies of the coding decision for coding tree blocks on future blocks. Also, potentially existing inter-dependencies between pictures are neglected. At an increased computational effort, such information could be included in the coding decision e.g. if look-ahead coding strategies are applied.

The applicable weighting factor λ depends on the employed distortion measure and also on the coding configuration for the given video sequence. It has to be determined by the application.

2.6 Compression Artifacts

Prediction, decorrelation, and entropy coding of a signal can be used to generate a compact coded representation which allows for perfect reconstruction of the signal. The quantization operation removes information from the signal which cannot be recovered. This results in typical compression artifacts which are briefly reviewed in this section.

In terms of artifacts potentially occurring within a single picture, three types are present which occur in the context of predictive coding. These are *ringing*, *blurring*, and *blocking*. These artifacts can be considered to be 'local' to a picture as they are

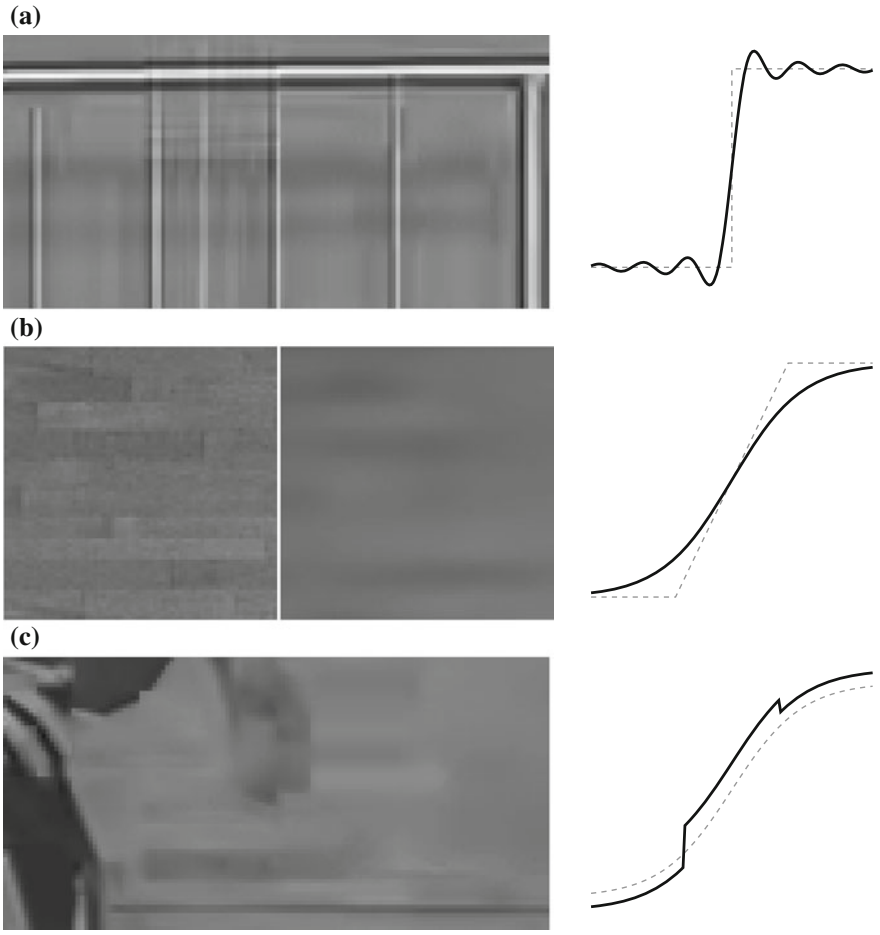

Fig. 2.18 Examples and illustrations for compression artifacts in video coding. **a** Ringing. **b** Blurring. **c** Blocking

induced by prediction and reconstruction of the quantized prediction error of the picture under consideration and do appear in still-image coding schemes as well as in video coding schemes. For video sequences, additional artifacts may be introduced which can severely harm the visual impression of the reconstructed video. *Motion jerkiness* can be introduced when the coded motion vectors do not match the motion observed in the scene.[12] *Pumping* of the reconstruction quality over time

[12] This effect may e.g. be observed with rate-distortion optimized H.264 | AVC encoding. Here, the rate-distortion optimization favours the skip mode, which is very cheap in terms of coding cost while it omits an update of the motion information according to the scene motion. Thereby, skip-coded

may occur, e.g. if the quantizer control periodically switches between fine and coarse quantization.

Cause and effect for each of ringing, blurring, and blocking are briefly described here. An example with a snapshot from a reconstructed video and a one-dimensional illustration for each of the artifacts is shown in Fig. 2.18.

Ringing occurs in the context of transformation and quantization. If a steep edge in the signal is transformed and the coefficients are quantized, some of the transform coefficients—specifically at high frequencies—are quantized to zero. As a consequence, the missing frequency components induce artifacts in the reconstructed signal, which are characterized as overshoots and undershoots. This effect occurs also when filtering a step function with a lowpass filter where the effect is referred to as the Gibbs phenomenon [43]. Fig. 2.18a demonstrates the ringing artifact for a sample area of a reconstructed video sequence and shows the Gibbs phenomenon for a lowpass filtered one-dimensional step response.

Blurring is observed if strong lowpass filtering is applied to a signal. The lowpass filtering removes the detail and smoothes out the contour of the signal. This artifact is induced by strong quantization of the signal. It can be even enhanced by the operation of the deblocking filter, which adds a smoothing operation at otherwise visible block boundaries. A combination of small transform block sizes and strong deblocking filtering abets a blurred impression of the reconstructed video as few detail information is preserved by both operations. This effect can be observed e.g. for video coded at low bitrates with H.264 | AVC Baseline profile. An example can be found in Fig. 2.18b, providing a blurred sample area of a reconstructed picture and an illustration for a one-dimensional signal.

Blocking is an artifact which is induced by two different sources. These are blockwise prediction and block transform coding. The blocks of a picture from a video sequence are either intra predicted from a spatial neighborhood or are inter predicted using motion compensation of a reference picture. Both types of prediction operate on a block basis where the sample values at the boundaries of the neighboring blocks do not necessarily relate to each other. As a consequence, the block structure of the prediction operation becomes visible. The second source of blocking artifacts is the block transform nature of the transform coding stage in the video coding scheme. With an increasing amount of quantization, the block boundaries between neighboring transform blocks become more and more visible as the reconstructions of the neighboring blocks are independent. If the residual signal is even quantized to zero, the blocking artifact of the prediction operation comes to full effect. These blocking effects can be mitigated by deblocking filtering. Other approaches such as overlapped block motion compensation [44] and overlapped transforms [45] alleviate the effect as well but induce an increased computational complexity and integration issue in the overall design. Fig. 2.18(c) shows an example for blocking artifacts in a reconstructed picture. The one-dimensional illustration exemplifies the transition discontinuities if shifted parts of the signal are attached for approximation.

---

regions in a scene may appear to 'jump' back and forth in successive pictures, depending on how coarse the motion approximation by the skip modes has been.

References

1. Poynton, C.: Digital Video and HD: Algorithms and Interfaces. Morgan Kaufman Publishers, Waltham (2012)
2. Parameter values for ultra-high definition television systems for production and international programme exchange. ITU-R Rec, BT.2020-0. http://www.itu.int/rec/R-REC-BT.2020/en (2012). Accessed 14 Apr 2014
3. Salmon, R., et al.: Higher Frame Rates for more Immersive Video and Television. British Broadcasting Corporation. http://www.bbc.co.uk/rd/publications/whitepaper209 (2011). Accessed 14 Apr 2014
4. High efficiency video coding. ITU-T Rec. H.265 (HEVC). http://www.itu.int/rec/T-REC-H.265/en (2013). Accessed 14 Apr 2014
5. Colorimetry—Part 1: CIE standard colorimetric observers. ISO 116641:2007. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=52495 (2007). Accessed 14 Apr 2014
6. Hunt, R.G.: The Reproduction of Colour, 6th edn. Whiley-VCH, Chichester (2004)
7. Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. ITU-R Rec, BT.601-7. http://www.itu.int/rec/R-REC-BT.601/en (2011). Accessed 14 Apr 2014
8. Parameter values for the HDTV standards for production and international programme exchange. ITU-R Rec, BT.709-5. http://www.itu.int/rec/R-REC-BT.709/en (2002). Accessed 14 Apr 2014
9. Reference electro-optical transfer function for flat panel displays used in HDTV studio production. ITU-R Rec, BT.1886-0. http://www.itu.int/rec/R-REC-BT.1886/en (2011). Accessed 14 Apr 2014
10. Ohm, J.-R.: Multimedia Communication Technology. Springer, Berlin, Heidelberg (2004)
11. Kamp, S., Wien, M.: Decoder-side motion vector derivation for block-based video coding. IEEE Trans. Circ. Syst. Video Technol. **22**(12), 1732–1745 (2012). doi:10.1109/TCSVT.2012.2221528
12. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice-Hall, Englewood Cliffs (1989)
13. Jain, A.K.: A sinusoidal family of unitary transforms. IEEE Trans. Pattern Anal. Mach. Intell. **1**(4), 356–365 (1979). doi:10.1109/TPAMI.1979.4766944
14. Britanak, V., et al.: Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations. Academic Press, New York (2006)
15. Han, J., et al.: Towards jointly optimal spatial prediction and adaptive transform in video/image coding. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '10), pp. 726–729 (2010). doi:10.1109/ICASSP.2010.5495043
16. Rao, K.R., Yip, P.: Discrete Cosine Transform. Academic Press, San Diego, CA (1990)
17. Chivukula, R.K., Reznik, Y.A.: Fast computing of discrete cosine and sine transforms of types VI and VII. In: Tescher, A.G. (ed) Applications of Digital Image Processing XXXIV, vol. 8135 (2011). SPIE, San Diego, CA. doi:10.1117/12.903685
18. Gray, R.M., Neuhoff, D.L.: Quantization. IEEE Trans. Inf. Theory **44**(6), 2325–2383 (1998). doi:10.1109/18.720541
19. Sullivan, G.J.: Efficient scalar quantization of exponential and Laplacian random variables. IEEE Trans. Inf. Theory **42**(5), 1365–1374 (1996). doi:10.1109/18.532878
20. Reininger, R.C., Gibson, J.D.: Distributions of the two-dimensional DCT coefficients for images. IEEE Trans. Commun. **31**(6), 835–839 (1983). doi:10.1109/TCOM.1983.1095893
21. Lam, E.Y., Goodman, J.W.: A mathematical analysis of the DCT coefficient distributions for images. IEEE Trans. Image Process. **9**(10), 1661–1666 (2000). doi:10.1109/83.869177
22. IEEE Standard Specifications for the Implementations of 8×8 Inverse Discrete Cosine Transform. IEEE 1180 (1991). doi:10.1109/IEEESTD.1991.101047
23. Information technology—MPEG video technologies—Part 1: Accuracy requirements for implementation of integer-output 8×8 inverse discrete cosine transform. ISO/IEC

23002-1:2006 (MPEG-C). http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=42030 (2006). Accessed 14 Apr 2014
24. Information technology—MPEG video technologies—Part 2: Fixed-point 8 × 8 inverse discrete cosine transform and discrete cosine transform. ISO/IEC 23002-2:2008 (MPEG-C). http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=45433 (2008). Accessed 14 Apr 2014
25. Wien, M., et al.: Integer transforms for H.26L using adaptive block transforms. Doc. 11th meeting: ITU-T SG16/Q15 VCEG, Q15-K-24, Portland (2000)
26. Advanced video coding for generic audiovisual services. ITU-T Rec. H.264 (AVC). http://www.itu.int/rec/T-REC-H.264/en (2014). Accessed 14 Apr 2014
27. Information technology—Coding of audio-visual objects—Part 10: Advanced video coding. ISO/IEC 14496–10:2012 (AVC). http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=61490 (2012). Accessed 14 Apr 2014
28. Sullivan, G.J, Ohm, J.-R.: Meeting report of the tenth meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Stockholm, SE. Doc. JCTVC-J1000. 10th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Stockholm, SE (2012)
29. Video codec for audiovisual services at p × 64 kbit/s. ITU-T Rec. H.261. http://www.itu.int/rec/T-REC-H.261/en (1993). Accessed 14 Apr 2014
30. Bossen, F.: Common test conditions and software reference configurations. Doc. JCTVC-K1100. 11th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Shanghai, CN (2012)
31. Sullivan, G.J., Ohm, J.-R.: Meeting report of the seventh meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Geneva, CH. Doc. JCTVC-G1100. 7th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Geneva, CH (2011)
32. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, New York (1991)
33. Sayood, K.: Introduction to Data Compression, 3rd edn. Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann Publishers Inc., San Francisco (2005). ISBN: 012620862X
34. Gallager, R.G.: Variations on a theme by Huffman. IEEE Trans. Inf. Theory **24**, 668–674 (1978). doi:10.1109/TIT.1978.1055959
35. Taubman, D.S., Marcellin, M.W.: JPEG2000: Image Compression Fundamentals, Standards and Practice. Kluwer, Boston (2002)
36. Ugur, K., et al.: Description of video coding technology proposal by Tandberg, Nokia, Ericsson. Doc. JCTVC-A119. 1st Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Dresden, Germany (2010)
37. Golomb, S.W.: Run-length encodings. IEEE Trans. Inf. Theory **12**(3), 399–401 (1996). doi:10.1109/TIT.1966.1053907
38. Weinberger, M.J., et al.: The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. IEEE Trans. Image Process. **9**(8), 1309–1324 (2000). doi:10.1109/83.855427
39. Teuhola, J.: A compression method for clustered bit-vectors. Inform. Process. Lett. **7**(6), 308–311 (1978). doi:10.1016/0020-0190(78)90024-8
40. Video coding for low bit rate communication. ITU-T Rec. H.263. http://www.itu.int/rec/T-REC-H.263/en (2005). Accessed 14 Apr 2014
41. Marpe, D., et al.: Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 620–637 (2003). doi:10.1109/TCSVT.2003.815173
42. Sullivan, G.J., Wiegand, T.: Rate-distortion optimization for video compression. IEEE Signal Process. Mag. **15**(6), 74–90 (1998). doi:10.1109/79.733497
43. Oppenheim, A.V., et al.: Signals and Systems, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ (1997)

44. Orchard, M.T., Sullivan, G.J.: Overlapped block motion compensation: an estimation-theoretic approach. IEEE Trans. Image Process. **3**(5), 693–699 (1994). doi:10.1109/83.334974
45. Malvar, H.S.: Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts. IEEE Trans. Signal Process. **46**(4), 1043–1053 (1998). doi:10.1109/78.668555

http://www.springer.com/978-3-662-44275-3

High Efficiency Video Coding
Coding Tools and Specification
Wien, M.
2015, XXIV, 314 p. 127 illus., 22 illus. in color.,
Hardcover
ISBN: 978-3-662-44275-3