

## Chapter 2

# Understanding the Application: An Overview of the H.264 Standard

**Abstract** Before any attempt to parallelize an application can be made, it is necessary to understand the application. Therefore, in this chapter we present a brief overview of the state-of-the-art H.264/AVC video coding standard. The H.264/AVC standard is based on the same hybrid structure as previous standards, but contains several new coding tools that increase the coding efficiency and quality. These new features increase the computational complexity of video encoding as well as decoding, however. Therefore, parallelism is a solution to obtain the performance required for real-time processing. The goal of this chapter is not to provide a detailed overview of H.264/AVC, but to provide sufficient background to be able to understand the remaining chapters.

## 2.1 Introduction

A video consists of a sequence of pictures or *frames*. When these frames are displayed rapidly in succession and provided the *frame rate* is high enough (typically 20-25 frames per second), viewers have the illusion that motion is occurring. To be able to store and transmit large video sequences, they need to be compressed or *encoded*. For example, to store a 2-hour uncompressed video of Full High Definition (HD) resolution frames (the resolution refers to the frame size and FHD corresponds to a frame size of  $1920 \times 1080$  picture elements or *pixels*), a capacity of  $2 \text{ (hours)} \times 60 \text{ (minutes per hour)} \times 60 \text{ (seconds per minute)} \times 25 \text{ (frame rate, frames per second)} \times 1920 \times 1080 \text{ (frame size in pixels)} \times 3/2 \text{ (number of bytes per pixel)} = 559,9 \text{ GB}$  is required, which far exceeds the capacity of current optical disc storage (50 GB for dual layer Blu-ray Disc). Therefore, videos have to be encoded before they can be stored or transmitted and decoded before they can be displayed.

To encode a video sequence, *spatial* and *temporal redundancy* can be exploited. Spatial redundancy means that pixels that are spatially close to each other typically have similar values. For example, if one pixel is “green”, then its neighboring pixels are often also green or slightly lighter or darker green. This redundancy can be

exploited to compress the pixel values in fewer bits. Another form of redundancy is temporal redundancy. Temporal redundancy refers to the fact that consecutive frames are typically very similar. Very often, consecutive frames are almost identical except that they have shifted a little due to motion. This redundancy can be exploited by storing the difference or *error* between a frame and a previous or next frame corrected by the amount of shift. Since the differences are usually small, they can be encoded in fewer bits than the original pixel values. This process is known as *motion compensation* and the amount of shift is given by the *motion vector*.

Currently, the best video coding standard, in terms of compression rate and quality, is H.264/AVC (Advanced Video Coding) [1], also known as MPEG-4 part 10<sup>1</sup>. It is used in Blu-ray Disc and many countries are using or will use it for terrestrial television broadcast, satellite broadcast, and mobile television services. It improves the compression ratio of previous standards such as H.262/MPEG-2 and MPEG-4 Advanced Simple Profile (ASP) by a factor of 2 or more. The H.264 standard [13] was designed to serve a broad range of application domains ranging from low to high bitrates, from low to high resolutions, and a variety of networks and systems, e.g., Internet streams, mobile streams, disc storage, and broadcast. It was jointly developed by ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG).

H.264 is based on the same block-based motion compensation and transform-based coding framework as prior MPEG and ITU-T video coding standards [10]. It provides higher coding efficiency by adding features and functionality that, in turn, increase the computational complexity. H.264 includes many new coding tools for different application scenarios but in this chapter only a summary of these tools can be presented, focusing on the tools which effect the performance of the decoder, which is the focus of this book, the most.

## 2.2 High-level Overview

The input of the video decoder is a compressed video, also called a *bitstream*. After a series of steps, referred as video decoding *kernels*, the decoder produces a reconstructed video than can be displayed on a screen. Real-time is achieved when each frame is processed within a limited amount of time, and the required frame rate is commonly expressed in terms of frames per second (fps).

A high-level view of the H.264 decoder is presented in Figure 2.1. The decoder is composed of four main stages: entropy decoding, inverse quantization and inverse discrete cosine transform (ICDT), prediction, and the deblocking filter.

First, the compressed bitstream has to be *entropy decoded*. Entropy coding is a lossless data compression scheme that replaces each fixed-length input symbol by a variable-length code word. Since shorter code words are used for more frequent symbols, this compresses the input data. Entropy decoding decompresses the bit-

---

<sup>1</sup> For brevity, in the remainder of this book, H.264/AVC will be referred to as H.264

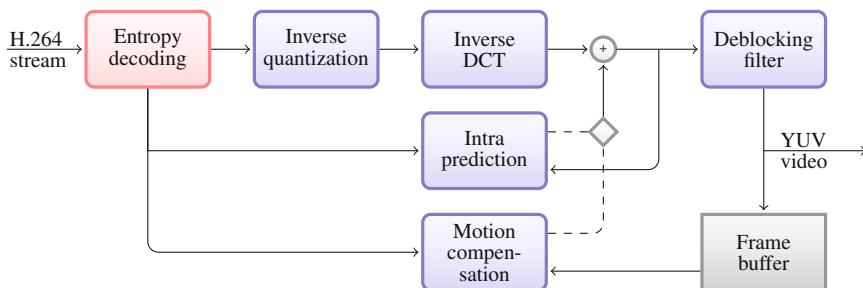


Fig. 2.1: High-level block diagram of the H.264 decoder.

stream and produces a series of syntax elements that specify, e.g., the coding modes, the quantized coefficients, motion vectors, etc. These elements are used by the other stages to reconstruct the final output video.

After entropy decoding, an inverse quantization step is applied to the quantized coefficients. At the encoder side, the coefficients are quantized by dividing them by a value greater than one. Again, the goal is to make the values smaller so that they can be encoded in fewer bits. Quantization is the main reason why video coding is *lossy*, i.e., some information is lost. Usually, however, the error is too small to be noticeable for the human eye. Then, an inverse DCT (IDCT) is applied to the reconstructed coefficients to obtain the residual data. The residual data represent the difference between the original signal and the predicted signal. The DCT, which is performed by the decoder, concentrates the signal information in a few components, which again compresses the information, and the decoder has to perform the inverse transform.

The predicted signal is created either with *intra-* or *inter-prediction*. In intra-prediction the predicted signal is formed using only samples from the same picture. Typically, the first frame from a scene is encoded using intra-prediction, since there is no previous frame very similar to this one. With inter-prediction, the signal is predicted from data of other frames, called *reference frames*, using *motion compensation*. Basically, the encoder partitions each frame into square blocks called *macroblocks* (MBs). For each MB it searches for similar blocks (not necessarily MBs) in the reference frames and it encodes the difference between the MB and the blocks that are similar (we do not have to repeat that the goal is to produce smaller values that can be encoded in fewer bits). The thus obtained motion vectors are included in the bitstream and the decoder uses this information to predict the signal. Thereafter, the decoder adds the predicted signal to the residual data to obtain the reconstructed signal.

Partitioning each frame into MBs, however, can lead to visible quality losses known as *blocking artifacts*. To improve the visual quality of the final output signal, a *deblocking filter* is finally applied to the reconstructed signal.

If you do not fully understand this high-level description of H.264, please do not despair. In Section 2.5 we describe each phase, or *coding tool*, in somewhat more detail.

## 2.3 Elements of a Video Sequence

H.264 is a *block-based* coder/decoder (codec), meaning that each frame is divided into small square blocks called macroblocks (MBs). The coding tools / kernels are applied to MBs rather than to whole frames, thereby reducing the computational complexity and improving the accuracy of motion prediction. Figure 2.2 depicts a generic view of the data elements in a video sequence. It starts with the sequence of frames that comprise the whole video. Several frames can form a Group of Pictures (GOP), which is an independent set of frames. Each frame can be composed of independent sections called *slices*, and slices ones, in turn, consist of MBs. Each MB can be further divided into sub-blocks, which in turn, consist of pixels.

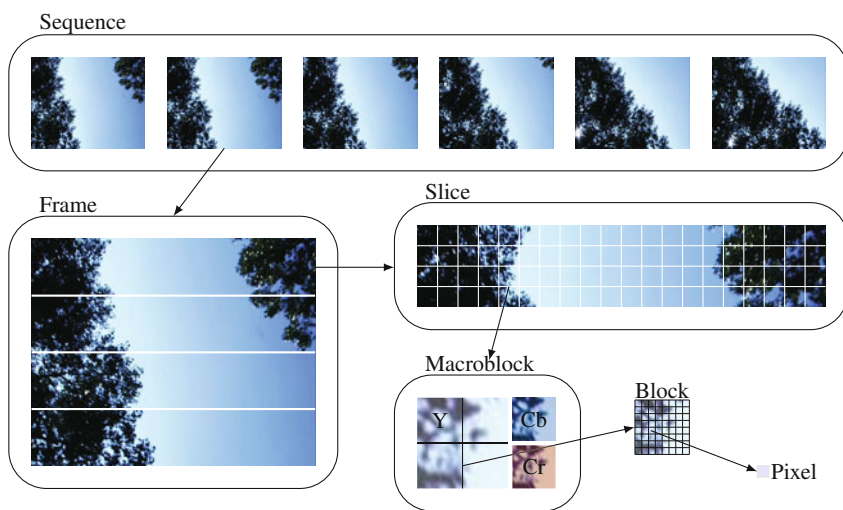


Fig. 2.2: Elements of a video sequence

A MB<sup>2</sup> consists of separated blocks for luma (denoted by Y) and chroma signals (denoted by Cb and Cr). A pre-processing step has to be applied to convert video from a different color component format (such as red-green-blue, RGB) to the YCbCr color model. Chroma sub-sampling is applied to reduce the amount of

<sup>2</sup> The word macroblock is used so often in this book that we introduce an acronym for it: MB. When reading the text, however, we read it as macroblock. Therefore, we write “a MB”, while it should be written as “an MB” when the acronym is pronounced as “em-bee”.

color information, since the human eye is more sensitive to brightness (Y) than to color (Cb and Cr) [8]. The most common color structure is denoted by 4:2:0 in which the chroma signals (Cb and Cr) are sub-sampled by 2 in both dimensions. As a result, in H.264, as in most MPEG and ITU-T video codecs, each MB typically consists of one  $16 \times 16$  luma block and two  $8 \times 8$  chroma blocks.

## 2.4 Frame Types

H.264 defines three main types of frames: *I*-, *P*-, and *B*-frames. An I-frame uses intra-prediction and is independent of other frames. In intra-prediction, each MB is predicted based on adjacent blocks from the same frame. A P-frame (Predicted frame) uses motion estimation as well as intra-prediction and depends on one or more previous frames, which can be either I-, P- or B-frames. Motion estimation is used to exploit temporal correlation between frames. Finally, B-frames (Bidirectionally predicted frames) use bidirectional motion estimation and can depend on previous frames as well as future frames [3].

Figure 2.3 illustrates a typical I-P-B-B (first an I-frame, then two B-frames between P-frames) sequence. The arrows indicate the dependencies between frames caused by motion estimation. In order to ensure that a reference frame is decoded before the frames that depend on it, and because B-frames can depend on future frames, the decoding order (the order in which frames are stored in the bitstream) differs from the display order. Thus a reordering step is necessary before the frames can be displayed, adding to the complexity of H.264 decoding.

## 2.5 H.264 Coding Tools

The H.264 standard has many coding tools each one with several options. Here we can only briefly mention the key features and compare them to previous standards.

### 2.5.1 Entropy Coding

H.264 includes two different entropy coding techniques. The first one is *Context Adaptive Variable Length Coding* (CAVLC), which is an adaptive variant of Huffman coding and targeted at applications that require a slightly simpler entropy decoder. The second one is *Context Adaptive Binary Arithmetic Coding* (CABAC), which is based on arithmetic coding techniques. Arithmetic coding differs from Huffman coding in that it does not encode each symbol separately, but encodes an entire message in a single number between 0.0 and 1.0. This results in compression ratios that are 10% to 14% higher than CAVLC [6]. In this work we focus on H.264

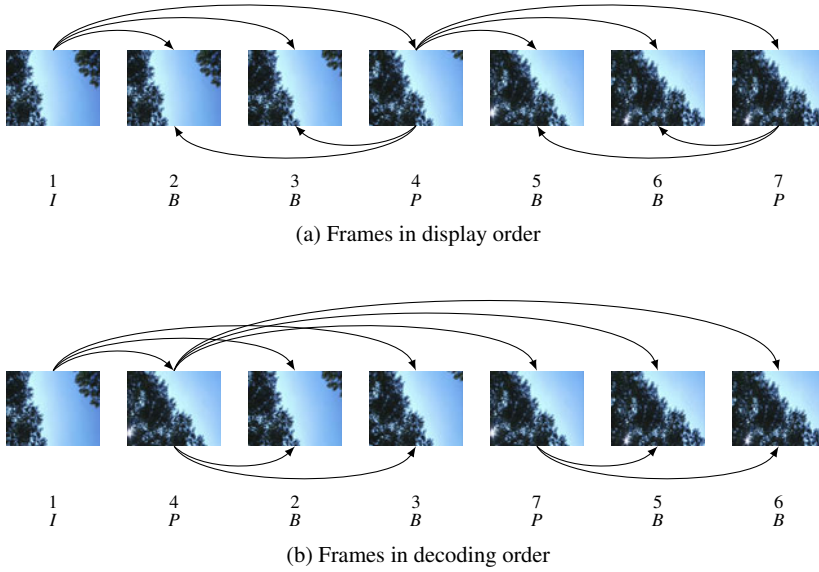


Fig. 2.3: Types of frames and display order versus decoding order.

decoders that employ CABAC, because it achieves higher compression ratios than CAVLC, and because it is widely used in HD applications.

### 2.5.2 Integer Transform

The Discrete Cosine Transform (DCT) is a mathematical transform that converts input data (e.g. residual data) into a different domain (called the frequency domain). The main purpose of this conversion is to separate the input data into independent components and concentrate most of the information on a few coefficients [9]. The 2-dimensional Discrete Cosine Transform (2D-DCT) is the most widely used transform for block-based video codecs because its mathematical properties allow efficient implementations [2].

H.264 transform is an integer approximation of the DCT allowing implementations with integer arithmetic for decoders and encoders. The main transform is applied to  $4 \times 4$  blocks and is useful for reducing ringing artifacts (negative coding effects that appear as bands near the edges of objects in an image). The H.264 High Profile (described later) allows another transform for  $8 \times 8$  blocks which is useful in HD video for the preservation of fine details and textures [11]. The encoder is allowed to select between the  $4 \times 4$  and  $8 \times 8$  transforms on a macroblock by macroblock basis. The transforms employ only integer arithmetic without multiplications, with coefficients and scaling factors that allow for 16-bit arithmetic

computation [5]. In the decoder, an Inverse Discrete Cosine Transform (IDCT) is applied to the transformed coefficients in order to reconstruct the original residual data.

Matrices 2.1 and 2.2 illustrate a  $8 \times 8$  IDCT using data from a real H.264 video.  $X$  represents a block with input transformed coefficients, most of them with zero value:

$$\mathbf{X} = \begin{bmatrix} -936 & -1020 & -460 & 0 & 0 & 68 & 0 & 0 \\ 0 & 128 & 344 & 128 & 0 & 0 & 0 & 0 \\ 276 & 86 & 232 & 0 & 0 & 0 & 0 & 0 \\ 136 & 64 & 0 & 0 & 0 & 0 & 0 & 0 \\ 216 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.1)$$

After applying the IDCT the reconstructed version of the coefficients  $Y$  is obtained:

$$\mathbf{Y} = \begin{bmatrix} -11 & -18 & -18 & -14 & -4 & 5 & 14 & 11 \\ -31 & -35 & -31 & -23 & -12 & -4 & 3 & 0 \\ -51 & -49 & -35 & -24 & -12 & -7 & -3 & -6 \\ -50 & -45 & -26 & -14 & -3 & 1 & 2 & -2 \\ -53 & -43 & -19 & -6 & 3 & 4 & 3 & -1 \\ -56 & -45 & -20 & -8 & 0 & -1 & -2 & -4 \\ -56 & -43 & -18 & -6 & 0 & 0 & 1 & 1 \\ -52 & -39 & -13 & -1 & 6 & 5 & 9 & 10 \end{bmatrix} \quad (2.2)$$

### 2.5.3 Quantization

The encoder quantizes the transformed coefficients by dividing them by values greater than 1, so the quantized coefficients can be coded using fewer bits. The decoder has to perform the inverse operation, i.e., multiply the quantized coefficients with the same values. Quantization is the main reason why video coding is *lossy*, meaning that some information is lost due to rounding errors. The loss of information is, however, usually too small for humans to notice.

### 2.5.4 Inter-Prediction

Advances in inter-prediction (predicting a MB from one or several MBs in other frame(s)) is one of the main contributors to the compression improvement of H.264. The standard allows variable block sizes ranging from  $16 \times 16$  down to  $4 \times 4$ , and each block has its own motion vector(s). The motion vector is quarter sample ac-

curate, meaning that motion vectors can also point to positions between pixels and the best matching block is obtained using interpolation. Multiple reference frames can be used in a weighted fashion [3]. Using multiple reference frames significantly improves coding *occlusion areas*, where an accurate prediction can only be made from a frame further in the past. Figure 2.4 illustrates motion estimation.

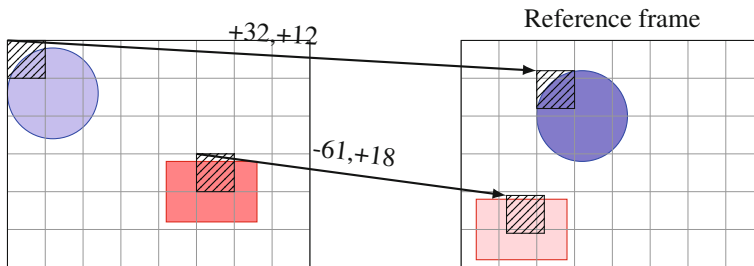


Fig. 2.4: Motion estimation. For each MB the best matching block in the reference frame is found. The encoder codes the differences (errors) between the MBs and their best matching blocks. Arrows indicate motion vectors and are labeled by the vector coordinates. In this example the shapes are identical but their colors are slightly larger/darker.

### 2.5.5 Intra-Prediction

The first frame in a new scene is typically not similar to a previous frame. Therefore, these frames are often I-frames which are not predicted from previous or next frames but using intra-prediction, meaning from itself. P-frames employ inter- as well as intra-prediction. H.264 supports three types of intra coding, denoted by Intra\_4x4, Intra\_8x8, and Intra\_16x16.

Intra\_4x4 is well suited for coding parts of a picture with significant spatial detail. The  $16 \times 16$ -pixel MB is divided into sixteen  $4 \times 4$  sub-blocks and intra-prediction is applied to each sub-block. The standard defines nine prediction modes that the encoder can choose independently for each sub-block. Samples are predicted using previously decoded samples from the blocks to the north, north-east, and west.

The High Profile defines intra-prediction for  $8 \times 8$  blocks. The prediction modes are basically the same as in  $4 \times 4$  intra-prediction with the addition of low-pass filtering to improve prediction performance [11].

Intra\_16x16 predicts a whole MB and is well suited for coding frame areas with smooth variations of tone and color. Four different prediction modes are available for this type of prediction: vertical, horizontal, DC-, and plane-prediction. The first three ones are very similar to the modes available for  $4 \times 4$  blocks. Plane-prediction employs a linear function between neighboring samples [13, 7].



### 2.5.6 Deblocking filter

Processing a frame in MBs can produce artifacts at the block edges, generally considered the most visible artifact in standards prior to H.264. These *blocking artifacts* can be reduced so that they are no longer visible by applying a deblocking filter to the pixels around the block edges, which basically smooths the block transitions. The filter strength is adaptable through several syntax elements [4]. In H.263+ this feature was optional, but in H.264 it is standard, significantly improving the quality of the produced pictures. Furthermore, the deblocking filter is applied before the frame is used as a reference frame, which improves the efficacy of motion compensation.

### 2.5.7 Comparison With Previous Standards

In Table 2.1 the main features of H.264 are summarized and compared to the MPEG-2 and MPEG-4 standards [13, 12]. One of the main differences between H.264 and the other video codecs is that H.264 allows a variable block size for motion compensation, while MPEG-2 only supports  $16 \times 16$  pixel blocks and MPEG-4  $16 \times 16$  down to  $8 \times 8$  pixel blocks. Additionally, H.264 employs quarter sample resolution for motion estimation, a feature that is optional in MPEG-4 and not available in MPEG-2. Another important difference is that H.264 supports multiple reference frames for motion compensation, while the other two standards support only one reference frame. As explained before, by doing so H.264 improves the coding of occlusion areas.

Furthermore, H.264 features more than one intra-prediction mode, which results in higher intra-compression than the single DC-prediction of MPEG-2 and the prediction of transformed coefficients of MPEG-4. H.264 also includes a mandatory in-loop deblocking filter that is not available in MPEG-2 and MPEG-4, and is optional in H.263. In addition, H.264 includes a binary arithmetic coder (CABAC) which achieves higher compression than the conventional entropy coders based on Variable Length Coding (VLC) employed in previous standards. Finally, H.264 employs an adaptive transform size ( $4 \times 4$  and  $8 \times 8$ ), and an integer transform that is faster than the fractional transforms of previous standards.

## 2.6 Profiles and Levels

The H.264 standard was designed to suite the requirements of a broad range of video application domains such as video conferencing, mobile video, as well as consumer and high definition broadcast. Each domain, however, is expected to use only a subset of all available options. For this reason *profiles* and *levels* were specified to mark conformance points. Encoders and decoders that conform to the same profile are

Codec	MPEG-2	MPEG-4 Part II ASP H.264 High	
Macroblock size	$16 \times 16$	$16 \times 16$	$16 \times 16$
Block size	$8 \times 8$	$16 \times 16, 16 \times 8, 8 \times 8$	$16 \times 16, 16 \times 8, 8 \times 16, 8 \times 8, 4 \times 8, 8 \times 4, 4 \times 4$
Transform	$8 \times 8$ DCT	$8 \times 8$ DCT	$8 \times 8, 4 \times 4$ integer transform
Pel-Accuracy	1, 1/2 pel	1, 1/2, 1/4 pel	1, 1/2, 1/4 pel
Reference frames	One frame	One frame	Multiple frames (up to 16 frames)
Bidirectional prediction	forward/backward	forward/backward	forward/backward forward/forward backward/backward
Intra-prediction	DC-pred.	coeff. pred.	$4 \times 4, 8 \times 8, 16 \times 16$ spatial
Deblocking filter	No	No	Yes
Weighted prediction	No	No	Yes
Entropy Coding	VLC	VLC	CAVLC, CABAC

Table 2.1: Comparison of video coding standards

guaranteed to interoperate correctly. Profiles define sets of coding tools and algorithms that can be used, while levels place constraints on the bitstream parameters.

The standard initially defined two profiles, but has since then been extended several times with additional profiles for new applications, such as intra-only profiles (for video editing), scalable profiles for Scalable Video Coding (SVC), and multi-view profiles for Multiview Video Coding (MVC). Currently, the main profiles are:

- **Baseline Profile (BP):** the simplest profile mainly used for video conferencing and mobile video.
- **Main Profile (MP):** this profile was intended to be used for consumer broadcast and storage applications, but has been overtaken by the high profile.
- **Extended Profile (XP):** this profile is intended for streaming video, and includes special capabilities to improve robustness.
- **High Profile (HiP):** this profile is intended for high definition broadcast and disc storage, and is used in Blu-ray.

Besides HiP there are three other high profiles that support up to 14 bits per sample (normally a sample is 8-bit), 4:2:2 and 4:4:4 sampling, as well as other features [11].

Besides profiles, 16 levels are currently defined which are used for all profiles. A level specifies, for example, the upper limit for the picture size, the decoder processing rate, the size of the multi-picture buffers, and the video bitrate. Levels have profile independent as well as profile specific parameters.

## 2.7 Conclusions

In this chapter we have presented a, necessarily brief, overview of the H.264/AVC standard. Hopefully we have convinced you that it is a complex application with many options, compression algorithms, and dependencies between the different parts. Therefore, discovering the parallelism in it is less than obvious. This is the step that will be undertaken in the next chapter.

## References

1. Advanced Video Coding for Generic Audiovisual Services, Recommendation H.264 and ISO/IEC 14 496-10 (MPEG-4) AVC, International Telecommunication Union-Telecommun. (ITU-T) and International Standards Organization/International Electrotechnical Commission (ISO/IEC) JTC 1 (2003)
2. Blinn, J.: What's that deal with the DCT? *IEEE Computer Graphics and Applications* **13**(4), 78–83 (1993)
3. Flierl, M., Girod, B.: Generalized B Pictures and the Draft H. 264/AVC Video-Compression Standard. *IEEE Transactions on Circuits and Systems for Video Technology* **13**(7), 587–597 (2003)
4. List, P., Joch, A., Lainema, J., Bjøntegaard, G., Karczewicz, M.: Adaptive Deblocking Filter. *IEEE Transactions on Circuits and Systems for Video Technology* **13**(7), 614–619 (2003)
5. Malvar, H., Hallapuro, A., Karczewicz, M., Kerofsky, L.: Low-Complexity Transform and Quantization in H. 264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology* **13**(7), 598–603 (2003)
6. Marpe, D., Schwarz, H., Wiegand, T.: Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. *IEEE Transactions on Circuits and Systems for Video Technology* **13**(7), 620–636 (2003)
7. Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., Wedi, T.: Video Coding with H.264/AVC: Tools, Performance, and Complexity. *IEEE Circuits and Systems Magazine* **4**(1), 7–28 (2004)
8. Richardson, I.E.G.: *Video Codec Design: Developing Image and Video Compression Systems*. John Wiley and Sons (2002)
9. Richardson, I.E.G.: *H.264 and MPEG-4. Video Compression for Next-generation Multimedia*. Wiley, Chichester, England (2004)
10. Sikora, T.: Trends and Perspectives in Image and Video Coding. *Proceedings of the IEEE* **93**(1), 6–17 (2005)
11. Sullivan, G., Topiwala, P., Luthra, A.: The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. In: *Proceedings SPIE Conference on Applications of Digital Image Processing XXVII*, pp. 454–474 (2004)
12. Tamhankar, A., Rao, K.: An Overview of H. 264/MPEG-4 Part 10. In: *Proceedings 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, p. 1 (2003)
13. Wiegand, T., Sullivan, G.J., Bjøntegaard, G., Luthra, A.: Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology* **13**(7), 560–576 (2003)

Scalable Parallel Programming Applied to H.264/AVC  
Decoding

Juurlink, B.; Alvarez-Mesa, M.; Chi, C.C.; Azevedo, A.;

Meenderinck, C.; Ramirez, A.

2012, XIII, 101 p. 35 illus., Softcover

ISBN: 978-1-4614-2229-7