Poznań University of Technology
Faculty of Electronics and Telecommunications
Chair of Multimedia Telecommunication and Microelectronics

Doctoral Dissertation

Advanced Adaptation Algorithms of Arithmetic Coding in Hybrid Video Compression

Damian Karwowski

Supervisor: Prof. dr hab. inż. Marek Domański

Poznań, 2008

Table of contents

List of Tables

List of Figures

List of symbols and abbreviations

a_s – Number of zeros in node *s* in the context tree

ASIC – Application-specific integrated circuit

AVC – Advanced Video Coding

AVS – Audio and Video Coding Standard of China

b_s – Number of ones in node *s* in the context tree

CABAC – Context-based Adaptive Binary Arithmetic Coding

CAVLC – Context-Adaptive Variable Length Coding

Cb, Cr – Chroma components

CTW – Context-Tree Weighting

CTW+PPMA – Joint application of the CTW and the PPMA technique

D – Depth of the context tree

DAG – Directed Acyclic Graph

DCT – Discrete Cosine Transform

DWT – Discrete Wavelet Transform

EG*k* – *k*-th order Exp-Golomb coding

GOP – Group of Pictures

GRASP – Growing, Reordering and Selection by Pruning

HDL – Hardware description language

HDTV – High-Definition Television

HVS – Human Visual System

$H(S)$ – Entropy of source *S*

H263AE – Core of arithmetic encoder from H.263 video coding standard

H263AD – Core of arithmetic decoder from H.263 video coding standard

FPGA – Field-Programmable Gate Array

FSM – Finite-State Machine

IPTV – Internet Protocol Television

$I(x_k)$ – Self-information of symbol x_k

KT – Krichevsky-Trofimov estimator

LPS – Least Probable Symbol

LUT – Look-Up Table

$\overline{L}$ – Mean length of codeword

$l(x_k)$ – Length of codeword assigned to symbol x_k

λ – Root of the context tree

MBAFF – Macroblock Adaptive Frame-Field Mode

MPS – Most Probable Symbol

MV – Motion vector

MVD – Motion vector data

pdf – Probability density function

p_{LPS} – Probability of least probable symbol (LPS)

p_{MPS} – Probability of most probable symbol (MPS)

$P_w^s(x_1^n)$ – Weighted probability of block of symbols $x_1^n = x_1, x_2, ..., x_n$ stored in node s of the context tree

P_e – Estimated probability in CTW technique

PPM – Prediction with Partial Matching

PPMA – "A" variant of Prediction with Partial Matching

QP – Quantization Parameter

SAC – Syntax-based Arithmetic Coding

UEG*k* – Unary/*k*-th order Exp-Golomb binarization scheme

UVLC – Universal Variable-length Coding

VCEG – Video Coding Experts Group

VC-1 – Video Coding 1

VLC – Variable Length Coding

VSW – Virtual Sliding Window

Y – Luma component

Y PSNR – Peak Signal to Noise Ratio for luminance

σ – Index of probability for LPS symbol

2D-DCT – 2-dimensional Discrete Cosine Transform

Chapter 1

Introduction

1.1. Scope of the dissertation

Digital video compression has already gained a great importance in many fields of communication and information technology, including digital television, recording films on CD and DVD, video surveillance, video databases, digital cinema, medical imaging etc. A large variety of video compression techniques has been presented in the references [Bovik00, Doma98, Flier04, Jack05, Mual02, Ohm04, Richa02, Richa03, Skarb93, Skarb98, Woot05]. Among them, hybrid coding schemes are mostly used in communication systems, including the most modern ones. Hybrid video technology is a cornerstone of all major contemporary international and commercial video coding standards [MPEG-1, MPEG-2, H263, AVC, VC-1, AVS]. Moreover, the hybrid video technology is still a subject of most research in video compression.

Therefore, the dissertation also deals with hybrid video compression. Hybrid video encoders remove video data redundancy by the use of motion-compensated prediction and transform coding [Doma98, Skarb93, Bovik00, Richa03, Ohm04]. Such coders produce three data streams that represent video in far more compact form relative to its original version. The three streams are: transform coefficients of residual signal, motion data and control information (see Figure 1.1).

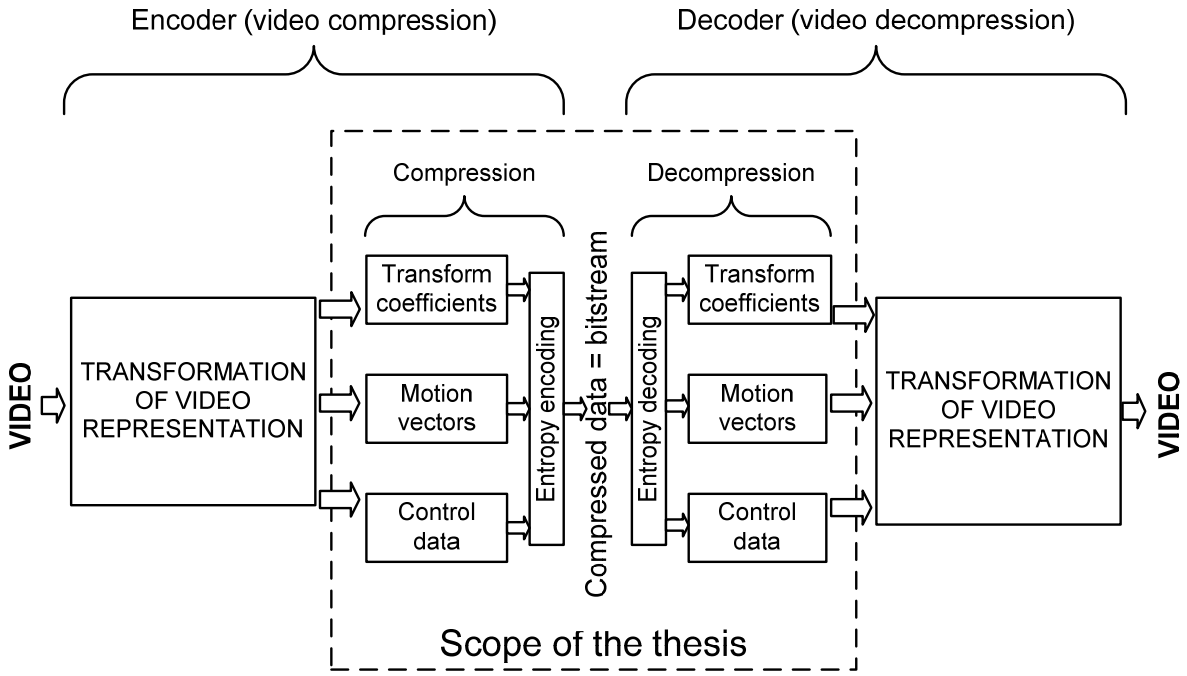


Figure 1.1. Scope of the thesis.

Nevertheless, these three data streams exhibit statistical redundancy. Therefore, in order to remove this redundancy entropy coding is always used at the output of a hybrid video encoder. The entropy coding is of great importance because it further reduces the bitrate of the compressed bitstream.

In particular, the dissertation deals with advanced entropy coding techniques in applications to hybrid compression of video. The dissertation focuses especially on advanced methods of data statistics estimation in entropy coding and their applications in order to increase compression performance of the newest generation of hybrid video encoders. The research has been done in the context of adaptive arithmetic coding recently applied in advanced video codecs. In order to make clear comparisons, the work refers to the state-of-the-art Context-based Adaptive Binary Arithmetic Coding (CABAC) [Marp02a, Marp02b, Marp03a] that became a part of the new worldwide Advanced Video Coding (AVC) standard (ISO MPEG-4 AVC and ITU-T Rec. H.264) [AVC, Schw02a, Wieg03a, Richa03, Oster04, Sull05, Marp05a, Marp05b, Marp06b, Wieg07].

The author takes into consideration the fact that application of more sophisticated techniques of data statistics modeling will increase complexity of both encoders and decoders. Therefore, in the dissertation, the influence of application of sophisticated techniques of data statistics estimation on complexity of both entropy encoder and entropy decoder is considered. Also the influence of application of more accurate techniques of data statistics

estimation on complexity of the whole AVC/H.264 encoder and decoder is taken into consideration.

1.2. Goals and thesis of the dissertation

The entropy encoder is an essential part of the video encoder that is used for removing correlation that exists within data. Initially, relatively simple non-adaptive techniques of Huffman coding [Huff52] have been used in video coders [MPEG-1, MPEG-2]. In video coders of more recent generation [H263, Rijk96, Gard98, AVC] more efficient techniques of arithmetic coding [Witt87, Said04] have been commonly applied. The state-of-the-art entropy coding technique used in video coders is Context-based Adaptive Binary Arithmetic Coding (CABAC) [Marp01, Marp02a, Marp02b, Marp03a] that is used in new Advanced Video Coding (AVC) standard (ISO MPEG-4 AVC and ITU-T Rec. H.264) [AVC]. CABAC technique uses arithmetic coding and far more sophisticated techniques of data statistics modeling in comparison to other entropy coders used in video compression.

The main goal of this dissertation is to study the ways of increasing compression of adaptive entropy coders used as the output stage of contemporary advanced video encoders. Such research is important in context of future new generation video coders. Works towards a new standard H.265 have been already initiated under auspices of ITU-T and its working group Video Coding Experts Group (VCEG, i.e. SG16/Q.6) [VCEG07].

The research is focused on improvement of adaptation of arithmetic encoders that are commonly used in video coders. In particular, the goal is to increase compression in adaptive arithmetic encoders by using more sophisticated schemes of the conditional probabilities estimation. Additionally, the relationship between the improvement of coding efficiency and the increase of complexity of entropy encoder and entropy decoder is to be explored.

The thesis of the dissertation is the following:
Improvement of adaptation of entropy coding that is used in contemporary advanced video coders leads to a reasonable increase of the compression of entropy coding at the cost of increase of the complexity of both video encoders and video decoders.

The thesis will be proved by application of proposed more exact techniques of the conditional probabilities estimation into the state-of-the-art Context-based Adaptive Binary

Arithmetic Coding (CABAC) algorithm [Marp03a] within the AVC video codec [AVC]. In order to obtain reliable experimental results, both encoder and decoder will be implemented. The compression efficiency of each of the modified AVC video codec with CABAC and more accurate data modeling technique will be tested and confronted with coding efficiency of the original AVC video codec with unmodified CABAC.

1.3. Methodology of experiments

The goal of the dissertation is to study whether it is possible to improve the coding efficiency of contemporary adaptive arithmetic coders using sophisticated techniques of data statistics estimation. For the reasons clearly presented in Chapter 5 the algorithm of Context-based Adaptive Binary Arithmetic Coding (CABAC) [Marp03a] has been used as the basis for further investigations.

The author is going to improve the coding efficiency of CABAC by proposal and application of even more accurate techniques of the conditional probabilities estimation in CABAC. The author is going to test the coding efficiency of such a modified CABAC within the framework of AVC [Richa03, AVC] video coder. The coding efficiency of the modified CABAC encoders will be confronted with the efficiency of the original CABAC.

The only way to assess of coding efficiency of the modified CABAC encoders is performing of series of experiments with the test video sequences. In order to do that, the author has implemented and embedded the proposed techniques of data statistics estimation into the structure of CABAC within the reference software JM 10.2 [AVCSoft] of AVC video codec. In order to obtain reliable experimental results both encoder and decoder have been implemented. In this way the modified AVC video codecs have been built. It must be stressed that implementation of the modified AVC video codecs was a very difficult and time-consuming task. The reference software JM 10.2 [AVCSoft] of AVC video codec contains about 90 000 lines of program code written in C programming language [Kern88] (about 58 000 lines of program code for AVC encoder and about 32 000 lines of program code for AVC decoder). Functions of CABAC entropy codec (encoder and decoder) contain approximately 6000 lines of program code in C, nevertheless this number does not take into consideration the program code that calls the individual functions of CABAC codec in many parts of AVC codec. In practice, the application of the proposed data modeling techniques

into CABAC needed modifying of significant part of AVC encoder and AVC decoder that can be counted in many thousands lines of program code written in C programming language.

The coding efficiency of the modified AVC encoders has been explored and compared against the coding efficiency of the original AVC. The test video sequences introduced in Annex F have been used. The video encoders have been investigated in series of experiments that have been done for several configurations of the encoders (see Section 6.6). In this way the efficiency of the proposed methods have been estimated for different encoder configurations as well as for different parameters of the new probability estimation techniques.

The complexity of the modified AVC encoder and decoder (with CABAC and CTW) has been examined (see Chapter 8). In order to do that, the execution times of the modified AVC video codec have been measured for a wide range of target bitrates. Obtained experimental results have been referred to the complexity of the original AVC video codec measured in the same way.

1.4. Overview of the dissertation

The dissertation is organized as follows. In Chapter 2 the main idea of hybrid video compression is presented.

In Chapter 3 entropy coding is discussed. The entropy coding techniques that have been applied in video coders are presented.

Chapter 4 contains a description of entropy coding methods used in video coders of successive generations. Entropy coding methods used in the AVC video coder are discussed in detail. Some aspects of compression performance and complexity of entropy codecs used in AVC are discussed.

Chapter 5 presents adaptation techniques used in CABAC entropy codec. Universal data modeling techniques of the Context-Tree Weighting (CTW) and the Prediction with Partial Matching (PPM) are presented. The author's method of joint application of CTW and PPM technique is discussed.

In Chapter 6 the research methodology is discussed in detail. The original method of embedding the proposed techniques of data statistics gathering into the structure of CABAC is presented. Experimental results on compression performance of three modified AVC video

codecs (with CTW, PPMA and joint application of CTW and PPMA) against the coding efficiency of the original AVC are also presented.

In Chapter 7 the impact of arithmetic codec core on compression performance of the original CABAC encoder is considered. Two different arithmetic codec cores are compared. These are M-codec core and the arithmetic codec core from the H.263 video coding standard.

Chapter 8 presents tests on complexity of the modified CABAC entropy codec with proposed more exact techniques of data statistics estimation. The complexity of each of the modified CABAC entropy codecs is compared to the complexity of the original CABAC entropy codec. Experimental results on influence of application of more accurate data statistics estimation techniques in CABAC on the complexity of whole the modified AVC video codec are presented.

In Chapter 9 the original architectures for software and hardware versions of CABAC codec have been presented. The chapter discusses in details the complexity of advanced entropy codecs.

In Chapter 10 conclusions of the dissertation are presented. The chapter lists the original results of the dissertation.

Chapter 2

Video compression

2.1. Introduction

A representation of uncompressed video signal needs huge amount of data. Therefore, transmission of original video signal is either too costly or even impossible in multimedia systems that exploit transmission channels with limited data rate (teleconference systems, video-on-demand services or internet protocol television (IPTV) systems).

In order to make possible or to reduce costs of transmission of a video signal, it is compressed before transmission. Compression of the original video signal is possible due to the fact that video data exhibits statistical redundancy [Bovik00, Doma98]. In practice, systems of video compression try to predict current content of video on the basis of video data that has been already encoded and sent to the decoder. For the reason that the prediction is usually not perfect, the prediction residual (which is the difference between the original image and its predicted version) must be sent to the decoder in order to reconstruct the encoded fragment of the image. Practically, the prediction residual has significantly lower energy than the original video signal, so it can be encoded with significantly smaller number of bits. In this way, compression is achieved.

2.2. Techniques of video coding

In the area of digital video compression, two groups of methods are of major interests. These are:

- Hybrid video coding using block-based motion-compensated prediction and the block-based Discrete Cosine Transform (DCT) [Bovik00];
- Wavelet video coding using motion-compensated filtering and the 3D wavelet analysis and synthesis [Ohm04].

Both groups of methods have been intensively developing and improving for years by the science community [Ohm04, Doma98, Bovik00]. Great hopes have been put in wavelet coding techniques due to the absence of blocking artifacts which are present in the case of block-based hybrid coding schemes [Tri02, Luo03]. Nevertheless, besides the application of the wavelet coding method in the state-of-the-art international still image coding standard JPEG 2000 [JPEG2000, Achar05a, Achar05b], it is not commonly used in practice. In opposite to wavelet coding, hybrid coding techniques have found commonly application in the industry and television e.g. MPEG-2 [MPEG-2], VC-1 [VC-1, Kalv07], AVS [AVS] and AVC [AVC], and in telecommunication e.g. H.263 [H263]. In any case, last comparisons of advanced video coding technologies based on the wavelet and the hybrid coding schemes have showed a better coding efficiency of the hybrid coding techniques by a greater complexity of the wavelet coding methods [Bar04]. Therefore, the author has limited the research to the hybrid compression of video. Nevertheless, more exact techniques of the data statistics modeling that will be worked out for advanced entropy coders will also be able to be used in the wavelet coders.

Individual compression techniques exhibit various compression performance. Higher compression performance means ability to obtain lower bitrate by a given quality of decoded video, or equivalently ability to obtain better quality of decoded video by a given bitrate. In many cases, compression performance is reached at the cost of higher complexity of encoder and often also decoder.

Measurements of complexity of video codecs is a complicated task. Contemporary video codecs perform not only the arithmetic operations. Contemporary video codecs also perform operations on bits together with conditional execution of fragment of program code. It obviously influences the complexity of video codecs. Therefore, in this dissertation the complexity is expressed as a time of processor that is needed to execute a given program code. The author is fully aware that this method of complexity measurement has some limitations: the time of processor strongly depends on the processor architecture and the way of implementing of program.

2.3. Hybrid compression of video

2.3.1. Input video signal

Digital video has many different representations [Wysz82, Doma98]. The YCbCr representation (with luma Y and two chroma Cb and Cr components [Doma98, Achar05a]) dominates in transmission systems to the end user. The human visual system (HVS) is more sensitive to changes of brightness than changes of color. For that reason, samples of chroma components (Cb and Cr) can be decimated without significant deterioration of video signal perception. In practice, in consumer multimedia systems the 4:2:0 format of chroma sampling is mostly used. In this format, all chroma components are decimated by a factor of 2 in both horizontal and vertical directions. It has been shown in Figure 2.1.

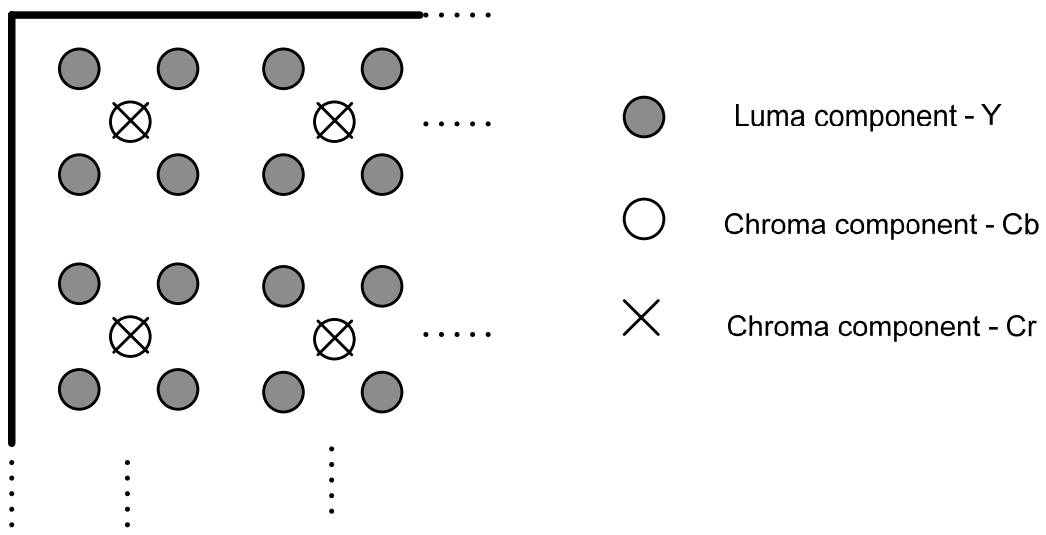


Figure 2.1. The 4:2:0 format of chroma sampling.

In this way, the number of samples of each chroma component is four times smaller than the number of samples of luma component. This is the first stage of video compression.

Hybrid compression of video uses block-based techniques of video coding. Therefore, each input image is split into non-overlapping blocks of 16x16 image samples called macroblocks as shown in Figure 2.2. Contemporary hybrid video coders process macroblocks one by one in the raster scan order beginning from the top-left macroblock of the image and ending on the bottom-right macroblock of the image. In newer hybrid video coders a macroblock can be further split into smaller non-overlapping blocks of samples for which block-based coding is realized [MPEG-2, H263, VC-1, AVS, AVC]. Advanced Video Coder (AVC) [AVC] allows for a possibility of splitting a macroblock into sixteen 4x4 blocks of samples.

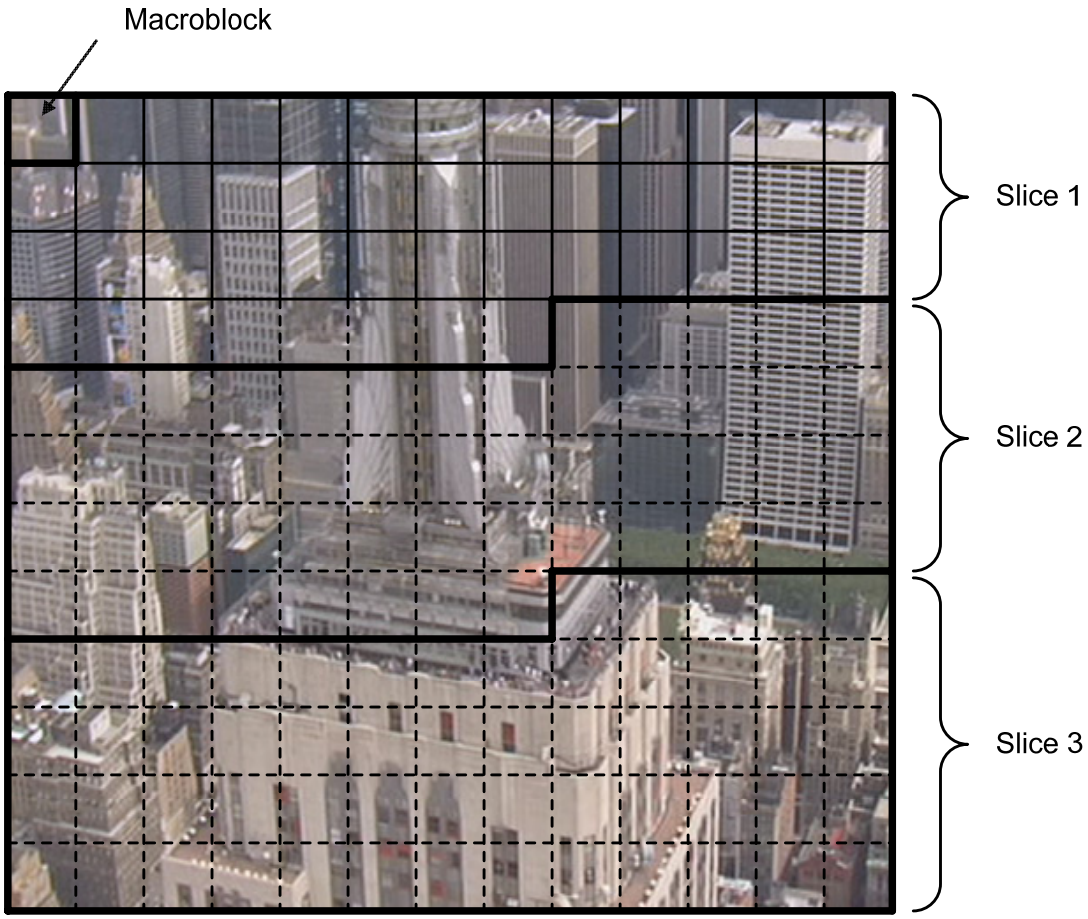


Figure 2.2. Macroblocks and slices in an image.

An image can be split into independent partitions called slices. In advanced hybrid coders a slice is formed from the raw of macroblocks or its part, as shown in Figure 2.2. A slice is self-contained unit within an image. It means that blocks of a given slice can be correctly encoded and decoded without referencing to the content of other slices.

2.3.2. Hybrid video coding algorithms

The idea of hybrid video coding has been well presented in the literature [Doma98, Skarb93, Bovik00, Richa03] and will not be presented in detail in the dissertation. Hybrid video compression exploits the fact, that a video signal contains spatial and temporal redundancy. These redundancies are eliminated in hybrid coders by the use of the following block-based techniques:

- Inter-frame prediction with block-based motion estimation and compensation.
- Intra-frame prediction.
- Transform coding technique.

- Entropy coding of residual data.

The structure of typical hybrid video encoder has been presented in Figure 2.3.

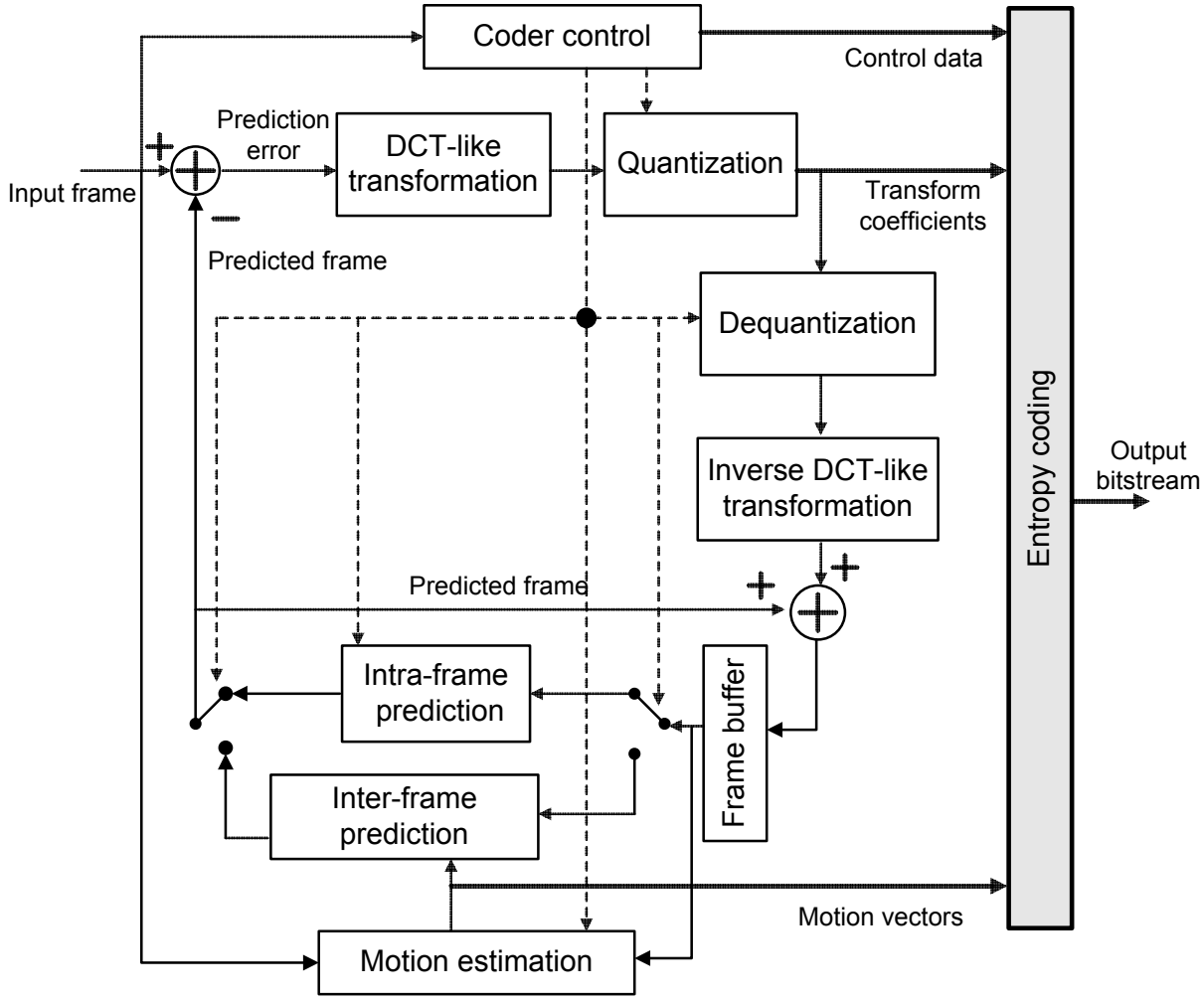


Figure 2.3. Structure of a hybrid video encoder.

By encoding a given block in the current frame, the hybrid encoder tries to predict its content on the basis of reference video data that has been already encoded and sent to the decoder. Hence, the encoder can use previously encoded frames or some parts of the current frame that has been already encoded as a reference. Obviously, the encoder is not able to faultlessly predict the content of current block with the use of the reference video data. The difference between the real content of current block and its prediction forms the prediction residual that has to be sent to the decoder. The prediction residual still shows some statistical redundancy that is reduced with the DCT-like transformation. In order to increase compression the resulted transform coefficients are quantized. The quantization of transform coefficients is a lossy operation that introduces an irreversible loss of information.

By predicting the content of the current block, the hybrid encoder must use only video data that is also known in the decoder. Therefore, a video encoder contains the reconstruction loop that is a part of the decoder. The reconstruction loop of the encoder is used to produce the reference video data that is used in prediction. In this way, both encoder and decoder use exactly the same reference video data. Thus, in the reconstruction loop of the encoder, quantized transform coefficients are dequantized and the inverse DCT-like transformation is performed. The obtained prediction residual is added to the predicted video signal, which results in the final reconstruction of the content of the current block.

The applied algorithms of prediction of the current frame block are of great importance. They influence the energy of the prediction residual and thus the compression performance of a hybrid video coder. One of the most powerful prediction techniques used in hybrid coders is motion-compensated prediction that has been proposed in the 1960s [Bovik00]. The basis of the technique is the observation that in most cases the consecutive frames (pictures) in a video signal differ between themselves insignificantly.

The process of motion-compensated prediction is reduced to motion estimation. Numerous different techniques of motion estimation have been presented [Tzir94]. In hybrid compression of video, block matching methods have found practical applications [Jain81]. With this method, by encoding of a given block of samples from the current frame, the encoder tries to find the best matching block of samples in the reference frames. Generally, three types of inter-frame prediction can be distinguished. When the current block is predicted with previous frames we have the forward prediction. If the current block is coded with reference to future frames we have the backward prediction. The current block can also be predicted with reference to previous and future frames which is called the bi-directional prediction.

For a given block of samples, the encoder must send to the decoder the index of reference frame and the co-ordinates of the best matched block of samples. The co-ordinates of the reference block are sent to the decoder in a form of a motion vector (MV). Motion vectors for each block of the current frame are calculated in motion estimation process. Motion estimation is one of the most computationally complex tasks of a hybrid encoder. Motion estimation needs about 30% - 60% of the whole encoding time [Doma98, Mual02]. Nevertheless, motion-compensated prediction strongly improves prediction and it is the basic prediction technique used in modern hybrid video coders.

Sometimes, there is a need to encode content of the current frame with no reference to neighboring frames. In such a case intra-frame prediction is used. In opposite to motion-

compensated prediction, intra-frame prediction predicts content of the current block on the basis of the content of neighboring blocks within the same frame that has been already encoded. Recently, advanced techniques of intra-frame prediction have been developed [Wieg03a, Richa03], and applied in Advanced Video Coding [AVC]. The efficiency of intra-frame prediction has been significantly increased by defining many different prediction modes that are adaptively chosen with respect to the local content of the frame [Richa03].

The data that is a result of intra- or inter-frame prediction still shows some correlation. This redundancy is reduced with 2-dimensional Discrete Cosine Transformation (2D-DCT) or its modifications [Doma98, Bovik00, Richa03]. The DCT-like transformation has an important feature of concentration of signal energy in a few low-frequency transform coefficients. Therefore, transform coefficients can be more efficiently encoded than equivalent signal before transformation. The data size of transform coefficients can be further reduced in the quantization process. As a result of that, many of high-frequency transform coefficients have small values and many of them are zero-valued. This step is connected with irreversible loss of video quality.

2.3.2.1. Entropy coding

Hybrid video coders produce three data streams that represent transform coefficients of prediction residual, motion vectors and control data (see Figure 2.3.). These three data streams still exhibit some statistical redundancy that negatively affects compression performance. In order to reduce this statistical redundancy, entropy coding is always used at the output of each contemporary hybrid video coder. Entropy coding is a lossless data compression technique and it represents input data in even more compact form.

Two groups of techniques of entropy coding have found common application in hybrid video coders. These are:

- Computationally simpler but less efficient techniques based on Variable-Length Coding (VLC) [Huff52, Gall75, Gall78, Golo66, Rice79, Przel05, Salom06, Salom07, Sayo00, Wan04];
- Computationally more complex but more efficient techniques that use arithmetic coding [Pas76, Riss76, Riss79, Witt87, Said04, Przel05, Salom06, Sayo00].

Generally speaking, both groups of entropy coding techniques reduce bitrate by encoding source symbols with respect to frequency of their occurrence in the video signal. The main idea is to assign a codeword to a single symbol or the whole block of symbols, whereupon the length of the codeword is dependent on probability of occurrence of a single symbol or a

block of symbols. Shorter codewords are assigned to symbols (or block of symbols) with higher probabilities of occurrence and longer codewords are assigned to symbols (or block of symbols) with lower probabilities of occurrence.

Entropy coding is of great importance in hybrid compression because it further reduces the size of compressed bitstream. Therefore, the scientific community has been doing research for many years on improvement of efficiency of entropy coding used in video coders. It has recently resulted in more advanced and more efficient techniques of entropy coding that have been applied in advanced video coding [Bobi02, Marp03a, Richa03, Karw04a, AVC].

2.3.2.2. I-, P- and B-frame types

Contemporary hybrid video coders use three main frame types: I-frame, P-frame and B-frame. All blocks of I-frames are coded with intra-frame prediction and none of blocks is coded with reference to neighboring frames. So, I-frames can be coded with no reference to other frames. Macroblocks of the P-frames can be coded with intra prediction as well as forward prediction. Macroblocks of the B-frames can be coded with all prediction types used in hybrid compression: intra prediction, forward prediction, backward prediction and bi-directional prediction. Macroblocks coded with intra prediction are called I-macroblocks. P-macroblocks are coded with inter-frame prediction using previously coded frames. B-macroblocks can be coded with forward prediction, backward prediction and bi-directional prediction. I-, P-, and B-frames consist of I-slice(s), P-slice(s), and B-slice(s) respectively.
The relationship between frames of different types has been shown in Figure 2.4.

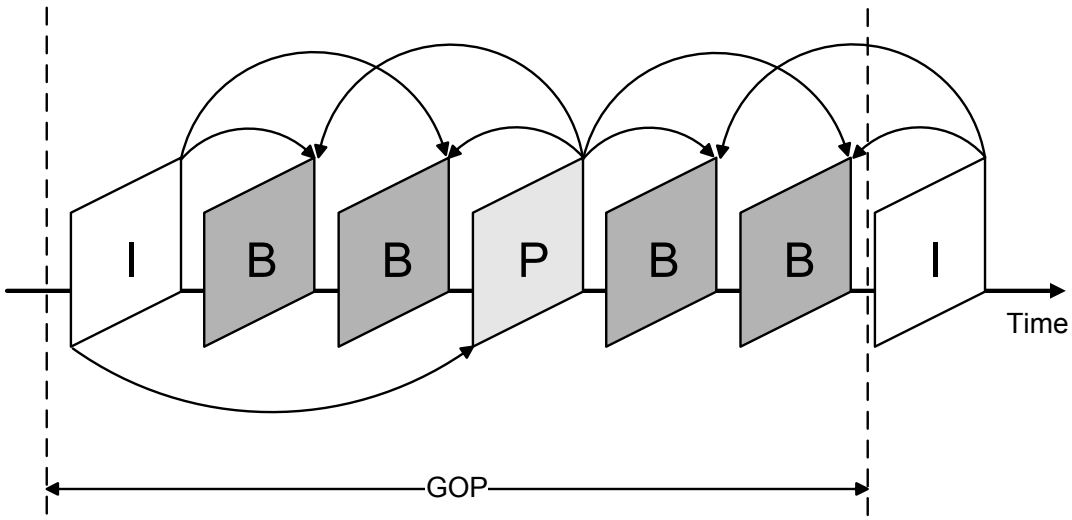


Figure 2.4. The relationship between frames (pictures) of different types within Group of Pictures (GOP).

Application of mechanisms of inter-frames prediction in P- and B-frame types yields significantly smaller bitstreams of encoded P- and B-frames relative to the bitstreams of I-frames. Therefore, on the one hand the hybrid encoder should avoid frequent usage of I-frames in order to achieve higher compression of video. But on the other hand, I-frames are self-contained and can be encoded and decoded with no reference to other frames. Therefore, in order to enable relatively fast access to encoded content of video I-frame type must be frequently used. In this way, a video signal is split into Group of Pictures (GOP). A GOP contains one I-frame and some number of P- and/or B-frames. A hypothetical GOP has been presented in Figure 2.4.

The above description of hybrid video coding is very brief due to the scope of the dissertation which is entropy coding and not hybrid video coding in general. More detailed description of such techniques may be found in [Doma98, Skarb98, Bovik00, Flier04, Jack05, Mual02, Ohm04, Richa02, Richa03, Woot05].

2.4. Advanced hybrid video coding

Dynamic development of multimedia services that exploit transmission channels with limited bandwidth (including video-on-demand, videoconference systems and IPTV) has created greater needs for higher compression performance of digital video. Therefore, for many years the scientific community has been doing intensive research on improvement of efficiency of hybrid compression techniques. That research resulted in more and more efficient hybrid video coders [Rijk96, Gard98, Côté98, Raja04, Kalv07, Fan04, Kam03, Lam06, Marp05a, Marp05b, Marp06b, Richa03, H263, AVS, VC-1, AVC].

There are three hybrid video coders of new generation:

- AVC (ISO/IEC MPEG-4 part 10, ITU-T H.264) [AVC];
- VC-1 [VC-1];
- AVS [AVS].

The state-of-the-art hybrid coder for digital video is H.264/MPEG-4 Advanced Video Coding standard [AVC]. From the point of view of compression efficiency it outperforms other hybrid video coders [Stock03, Wieg03b, Kam03, Schäf03, Raja04, Sull05]. Superior compression performance of the AVC coder has been achieved by a great number of improvements and many new video coding tools. These are:

- New techniques of spatial prediction for intra-frame coding.

In AVC, prediction of image samples is realized for 16x16 or 4x4 blocks of samples. In order to enhance the efficiency of samples prediction nine directional spatial prediction modes are used for 4x4 blocks and four prediction modes are supported for 16x16 blocks. At the given moment, the encoder chooses the best prediction mode taking into consideration a local content of coded image.

- Motion estimation and compensation with quarter-sample precision.

Most of the older video coding standards (MPEG-2, H.263) enable motion estimation and compensation with half-sample precision i.e., the motion vector components are expressed as multiplies of halves of a sampling period. In AVC, motion estimation can be done even with quarter-sample accuracy. Additionally, the interpolation process in AVC is realized in a more efficient way by applying more advanced 6-tap FIR filter.

- Motion estimation and compensation for blocks of variable size.

In the older MPEG-2 video coding standard, motion estimation and compensation can be done in 16x16 luma blocks. In order to increase the compression performance in the case of video sequences with high number of details, motion estimation and compensation can be optionally done in 8x8 luma blocks in the newer H.263 video coding standard. AVC video coding standard can realize even more advanced block-based motion estimation and compensation by further partitioning of each 8x8 luma block into smaller 8x4-, 4x8- or 4x4 luma blocks. In this way, the content of macroblock can be efficiently predicted with up to 16 motion vectors.

- Motion compensation with multiple reference frames.

Inter-frame prediction in AVC is realized in a more flexible way in comparison to the older video coding standards. In order to increase the compression performance, AVC supports the multi-frame motion-compensated prediction. The maximum number of reference frames that can be used for motion estimation and compensation is specified for each Level and can be equal to 2, 4, 5, 6, or 9 [Sull04].

- Direct motion compensation.

In AVC video coding standard, special modes of macroblock coding with skipping of partial data have been defined in B slice type. These are the temporal direct and the spatial direct modes. By encoding a macroblock in the direct mode, only quantized transform coefficients of prediction error are sent to the decoder. Motion vectors are predicted on the basis of motion vectors from the neighboring blocks. It allows to encode efficiently the content of a macroblock.

- Weighted prediction.

AVC is the first international video coding standard that exploits a special tool for efficient encoding of faded sequences. In contrast to the older standards of hybrid video compression, the signal of motion compensated prediction can be additionally weighted and offset in AVC with using of weighting and offset factors. For sequences with fade-to-black effect, the use of weighted prediction tool in AVC can even reduce the size of output bitrate by about 65% [Boy04].

- Macroblock adaptive frame-field coding mode.

In order to increase the coding efficiency of interlaced video sequences, macroblock adaptive frame-field coding mode (MBAFF) has been defined in AVC. MBAFF mode of AVC makes possible the use of two coding modes within a given image: frame mode and field mode. From the coding efficiency point of view, it is typically better to encode static parts of the image in the frame mode and moving parts of the image in the field mode. The choice between frame or field coding in MBAFF is done at the macroblock level, where two vertically adjacent macroblocks are encoded as two frames or two fields. The experimental results show that the use of MBAFF mode can lead to reduction of total bitrate by about 15% in comparison to the use of frame and field coding modes at the frame level only [Wieg03a].

- 4x4 and 8x8 block-size transforms.

The integer transform is applied on signal of prediction residual. Depending on the Profile of AVC the transformation can be performed on 4x4 and 8x8 block sizes. The encoder can choose the better solution based on the local structure of the image.

- In-loop deblocking filter.

The main disadvantage of hybrid coding schemes based on block transform coding is appearing of blocking effects for high compression. The blocking effect is visible in the form of sharp edges between blocks in which transform are calculated. This type of image distortion significantly decreases the visual quality of video. In order to improve the subjective quality of video, deblocking filter is applied to blocks of decoded image. It must be emphasized that the deblocking filter used in AVC is adaptive, so it smoothes edges between blocks with respect to the size of blocking artifacts.

- Advanced entropy coding techniques.

Two alternative techniques of entropy coding have been defined in AVC. The first one is simpler and is based on Variable-Length Coding (VLC). It uses Exp-Golomb coding and

Context-Adaptive Variable Length Coding (CAVLC). The second one is more efficient but at the same time more computationally complex and is called Context-based Adaptive Binary Arithmetic Coding. It uses efficient arithmetic coding. Both entropy coding techniques defined in AVC realize the sophisticated adaptation of entropy coding to the current signal statistics.

The following units of data are present in AVC:

- A coded video sequence that is a sequence of encoded frames (pictures);
- Each frame can be split into smaller partitions called slices – slice is a sequence of macroblocks within a frame;
- Macroblock that covers a block of 16x16 luma samples and certain number of chroma samples. (The number of chroma samples depends on the used format of chroma sampling);
- When inter frame prediction is used, a macroblock can be further partitioned into sub-macroblocks in which motion estimation and compensation is performed;
- Block that contains the quantized transform coefficients of prediction residual.

Similar techniques of video coding are used in other hybrid coders of new generation [VC-1, AVS]. Nevertheless, techniques used in VC-1 and AVS video coders are usually a certain simplification of techniques used in AVC. It is related especially to entropy coding technique which is crucial to the scope of this dissertation. Besides, AVC is the open international standard of video compression. Therefore, it will be considered as the fundamental technique in this dissertation.

Chapter 3

Entropy coding

3.1. Introduction

Entropy coding is a technique of lossless data compression that encodes source symbols $S = \{x_1, x_2, ..., x_N\}$ with respect to the probability of their occurrence $P = \{p_1, p_2, ..., p_N\}$. Generally speaking, to each source symbol or a chain of symbols entropy coder assigns a certain string of bits (codeword). The length of the codeword depends on the frequency of occurrence of coded symbol or block of symbols in data stream. Shorter codewords are assigned to symbols (or block of symbols) with higher probabilities of occurrence and longer codewords are assigned to symbols (or block of symbols) with lower probabilities of occurrence. In this way, the size of input data stream can be effectively reduced.

The smallest length of binary codeword that allows for encoding and decoding of a given symbol x_k is equal to $I(x_k)$ which results from Shannon's source coding theory [Shan48]. The quantity $I(x_k)$ is called the self-information of x_k symbol and in the case of binary codewords it is expressed by Equation 3.1

$$I(x_k) = \log_2\left(\frac{1}{p(x_k)}\right) = -\log_2\left(p(x_k)\right). \tag{3.1}$$

Mean self-information of source *S* is called entropy of source and is a function of probabilities $p(x_k)$ of all symbols generated by source *S*. The entropy of a source *S* is expressed by Equation 3.2

$$H(S) = \sum_{k=1}^{N} p(x_k) \cdot I(x_k) = \sum_{k=1}^{N} p(x_k) \cdot \log_2\left(\frac{1}{p(x_k)}\right). \quad (3.2)$$

Mean length of codeword $\overline{L}$ depends on probabilities $p(x_k)$ of source symbols and lengths $l(x_k)$ of codewords assigned to these symbols, where $k = 1, 2, ..., N$. It is expressed by Equation 3.3

$$\overline{L} = \sum_{k=1}^{N} l(x_k) \cdot p(x_k). \quad (3.3)$$

According to Shannon's source coding theory [Shan48], mean length of codeword $\overline{L}$ is always grater or equal to entropy $H(S)$ of source *S*. It means that $\overline{L} \geq H(S)$. Improvements of coding efficiency are aimed at approaching this limit. In this connection, the efficiency of entropy coding technique is defined as a function of entropy of source *S* and mean length of codeword $\overline{L}$ (see Equation 3.4.)

$$\eta = \frac{H(S)}{\overline{L}} \cdot 100\%. \quad (3.4)$$

Many entropy coding techniques have been developed since the precursor work of Shannon on information theory [Shan48]. In data compression, great popularity has been especially gained by Variable-Length Coding (VLC) techniques (such as Huffman coding [Huff52, Sayo00] and Exp-Golomb coding [Golo66]), arithmetic coding technique [Riss79, Witt87, Said04], and techniques of dictionary coding [Ziv77, Ziv78, Welch84]. This dissertation focuses only on VLC and arithmetic coding techniques since only they are applied in hybrid video coders.

3.2. Variable-length coding

The main idea behind the variable-length coding is very simple. It assigns a codeword of length $l(x_k)$ to each symbol x_k that is generated by the source *S*. In order to obtain compression, shorter codewords are ascribed to more probable symbols, whereas longer codewords are ascribed to less probable symbols. A great number of variable-length coding techniques have been proposed and well presented in the literature [Salom07]. In the context of hybrid compression of video only some of them are used.

3.2.1. Standard Huffman coding

Huffman coding is one of the most popular techniques of entropy coding used in data compression [Huff52, Przel05, Sayo00, Salom06, Salom07]. It is characterized by relatively high compression performance and low complexity of both encoder and decoder. Therefore, Huffman coding is a part of many well known data compression and archiving systems, like bzip2 [bzip2] and gzip [gzip]. It has also found common application in audio compression like MPEG-1 Layer 3 (MP3) [MPEG-1], and MPEG-2 AAC [MP2AAC], MPEG-4 AAC [MP4AAC], still image compression like JPEG [JPEG], PNG [PNG], TIFF [TIFF] and compression of digital video like MPEG-1 [MPEG-1], MPEG-2 [MPEG-2], H.263 [H263], VC-1 [VC-1], and AVC [AVC].

Let us assume that we have a source that generates *N* different symbols from alphabet $S = \{x_1, x_2, ..., x_N\}$ and a set of N probabilities $P = \{p_1, p_2, ..., p_N\}$ that correspond to individual symbols from alphabet *S*.

The basic algorithm of Huffman coding assigns one codeword to each source symbol with respect to the probability of its occurrence. For the sake of brevity, the detailed algorithm of Huffman codes creation will not be presented here. Generally speaking, Huffman codes can be characterized by the following properties:

- A variable-length codeword is assigned to each source symbol.
- Shorter codewords are assigned to symbols with higher probabilities of occurrence and longer codewords are assigned to symbols with lower probabilities of occurrence.
- Two symbols with the lowest probability of occurrence have the codewords of the same length.
- None of the codeword can be a prefix of other codewords.

Mean length $\overline{L}$ of Huffman code for symbols from source *S* is determined by Inequality 3.5:

$$H(S) \le \overline{L} \le H(S) + 1. \tag{3.5}$$

It must be pointed out, that Huffman coding is optimal only when all probabilities of symbols are a negative power of two. For that case, mean length $\overline{L}$ of Huffman code is equal to entropy $H(S)$ of source S. When probabilities of symbols are not a negative power of two, mean Huffman code length $\overline{L}$ is always greater than entropy $H(S)$.

In the reference [Gall78] also more accurate estimate of mean length $\overline{L}$ of Huffman code is described. This estimate depends on the maximum probability $p_{\max} = \max\{p(x_i)\}_{i=1}^{N}$ of source symbol. This estimate is given as:

$$\begin{aligned} p_{\max} < 0.5 \quad &\Rightarrow \quad \overline{L} \le H(S) + p_{\max}, \\ p_{\max} \ge 0.5 \quad &\Rightarrow \quad \overline{L} \le H(S) + p_{\max} + \sigma, \end{aligned} \tag{3.6}$$

where $\sigma = 1 - \log_2 e + \log_2(\log_2 e) \approx 0.086$.

Source symbol with probability greater than 0.5 can not be efficiently encoded with standard Huffman algorithm. For this case, the self-information of symbol is less than 1 while Huffman algorithm always assigns a codeword of at least 1 bit length to a given symbol. It affects negatively the compression performance of Huffman coding.

3.2.2. Block Huffman coding

The standard Huffman coding does not work efficiently when highest symbol probability $p_{\max}$ is greater than 0.5. This situation is more likely in the case of sources with small alphabet. Nevertheless, compression performance of standard Huffman coding can be increased by coding blocks of n symbols instead of coding of individual symbols independently [Doma98, Sayo00]. Such a modified Huffman algorithm will be called the block Huffman algorithm in this dissertation. It is possible to decrease mean length $\overline{L}$ of Huffman code when jointly coding blocks of *n* symbols. Mean length $\overline{L}'$ of block Huffman code in the case of coding of blocks of n symbols is determined by Equation 3.7 [Sayo00]

$$H(S) \le \overline{L}' \le H(S) + \frac{1}{n}. \tag{3.7}$$

It is clear that with the increase of the number n of symbols in a block, the efficiency of block Huffman coding also increases. But, let us assume that standard Huffman coding works on alphabet S that contains *N* different symbols. When coding blocks of n symbols from alphabet *S*, the number of all possible blocks of symbols is equal to N^n, so the size of Huffman codebook in the case of block Huffman coding is also equal to N^n. Therefore, with the increase of the number n of symbols in block, the exponential increase of the size of extended alphabet is observed relative to the size of standard alphabet S. This is serious limitation of block Huffman coding that can disqualify it from using in the case of sources with large size of alphabet.

3.2.3. Universal coding

Huffman coding requires storing variable-length codewords for all symbols from the alphabet in memory. Therefore, in the case of sources with large alphabet, application of Huffman coding may be too costly because of high demand of memory.

In this case, universal entropy coding techniques can be used. These techniques are characterized by regular algorithms of variable-length codewords construction and do not need to store the codetables in memory. Additionally, the complexity of both encoding and decoding for the universal coding is lower in comparison to the Huffman coding. In opposition to Huffman coding, universal coding techniques are not suitable for general purposes because the algorithm of codewords creation is adjusted to a certain assumed probability distribution of source symbols.

A great number of various universal coding techniques have been proposed [Salom07, Salom06]. One of the most popular universal coding techniques are Elias coding [Elias75], Exp-Golomb coding [Golo66], Fibonacci coding [Apos85], unary coding [Sayo00] and Rice coding [Rice79]. Some of them like Exp-Golomb coding or unary coding can be used in image (e.g. in JPEG-LS standard [JPEGLS]) and video compression.

Two techniques of universal coding are used in contemporary hybrid video coders. These are:

- Unary coding that is used in Advanced Video Coder (AVC) [AVC, Richa03];
- *k*-th order Exp-Golomb coding, that is used in AVC video coder [AVC, Richa03] as well as in Chinese Audio and Video Coding Standard (AVS) [AVS].

3.2.3.1. Unary coding

Unary coding [Richa03, Przel05, Salom06, Salom07] is applicable to sources that generate symbols that are integer numbers. The unary code of a given integer number $n \geq 0$ consists of *n* ones followed by a zero (or alternatively of *n* zeros followed by a 1). So, for example, the code 1110 corresponds to the integer number $n = 3$ and code 1111111110 corresponds to the integer number $n = 9$. Unary coding is efficient for sources that generate symbols of integer values *n* with probability expressed by Equation 3.8

$$p(n) \cong \frac{1}{2^n}. \tag{3.8}$$

If the source generates integer numbers *n* that are greater than 0, then the unary code can consists of $n-1$ ones followed by a zero (or alternatively of $n-1$ zeros followed by a 1). In that case, the code 110 corresponds to integer number $n = 3$.

Unary codes are characterized by simple encoding and decoding processes, which is their essential advantage.

3.2.3.2. *k*-th order Exp-Golomb coding

Exp-Golomb coding is also applicable to symbols that are integer numbers. Exp-Golomb codes were developed in 1978 by Teuhola [Teuh78]. Exp-Golomb code consists of two parts: the prefix part and the suffix part. For a given integer number $n \geq 0$ the *k*-th order Exp-Golomb code can be generated with the following algorithm [Teuh78, Marp03a]:

- Create the prefix part of *k*-th order Exp-Golomb code that is a unary code of integer value $y = \left\lfloor \log_2\left(\frac{n}{2^k}+1\right)\right\rfloor$;
- Create the suffix part of *k*-th order Exp-Golomb that is a binary representation of $z = n + 2^k\left(1-2^y\right)$ using $k + y$ significant bits.

Hence, the *k*-th order Exp-Golomb code has the length $l = k + 2 \cdot y + 1$. According to Shannon's theory, the minimal code length *l* that is needed to encode symbol of probability *p* is equal to $\log_2\left(\frac{1}{p}\right)$ bits. So *k*-th order Exp-Golomb coding is optimal for the following probability distribution (this equation has been derived by the author)

$$p(n) = \frac{1}{2^{k+1} \cdot \left(\frac{n}{2^k}+1\right)^2}. \tag{3.9}$$

Therefore, the Exp-Golomb coding can be effectively used for source symbols with geometrical probability distribution. The Exp-Golomb codes can be used to encode data of prediction residual in image (e.g. JPEG-LS standard [JPEGLS]) and video compression (e.g. AVC standard [AVC] and AVS standard [AVS]).

Exp-Golomb codes have an essential property that the number of codewords with a given length *l* grows exponentially with the code length *l*. Along with the increase of the value of the coded symbol *n* only a logarithmical increase of code length *l* exists. The key properties of both unary code and *k*-th order Exp-Golomb codes have been shown in Figure 3.1.

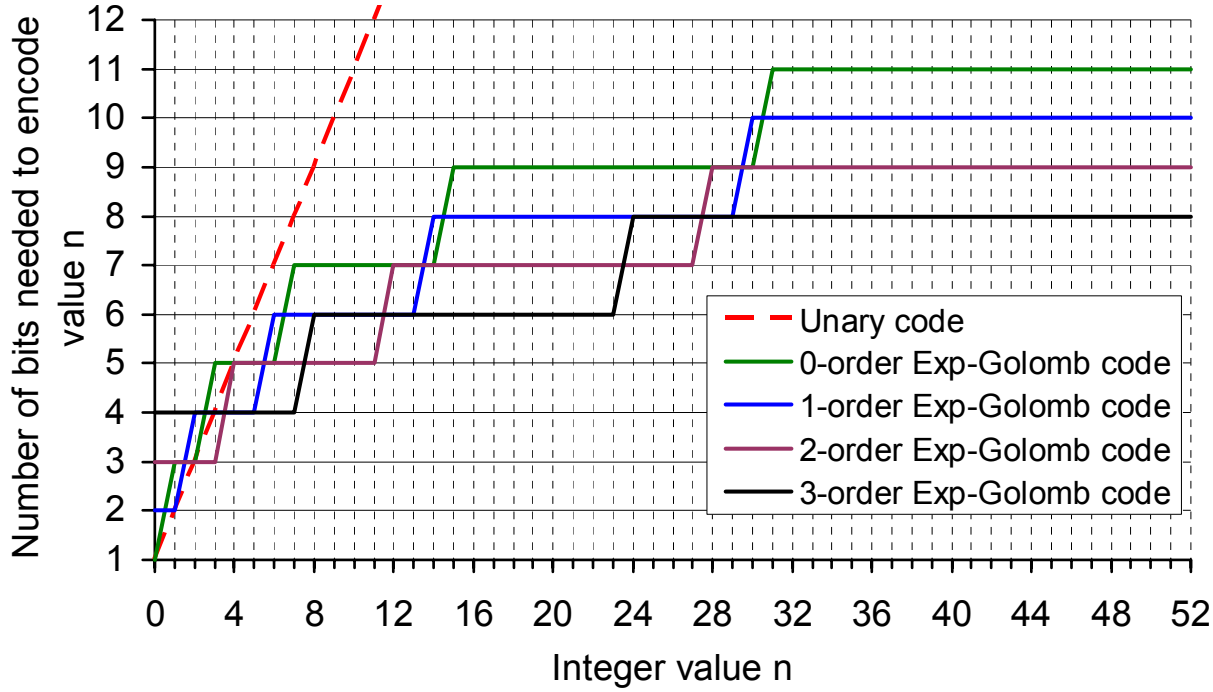


Figure 3.1. Length of universal codes.

From Figure 3.1 it is clear that the unary coding is very inefficient in the case of probability distribution of coded symbols expressed by Equation 3.9. In that case, the linear increase of the unary code length l along with the increase of the value of coded symbol n causes that a great number of bits would be needed to encode symbols of large values n.

3.3. Arithmetic coding

3.3.1. Main idea

The standard Huffman algorithm that works on individual source symbols can be inefficient when probabilities of symbols significantly differ between themselves. In this case, mean length $\overline{L}$ of Huffman code can be significantly higher than entropy $H(S)$ of source S. Compression performance may be increased by application of block Huffman coding that works on whole blocks of symbols. Unfortunately, memory requirement of block Huffman coding is significantly greater in comparison to standard Huffman coding. However, there is another technique of entropy coding that works well when probabilities of symbols are considerably differentiated. This is arithmetic coding.

The main idea behind arithmetic coding is similar to the idea of block Huffman coding and it assigns a codeword to whole blocks of symbols and not to individual symbols [Witt87, Riss79, Doma98, Sayo00, Przel05, Salom06]. In this sense arithmetic coding can also be understood as variable-length coding. Nevertheless, for the reason of different mechanism of codeword creation, arithmetic coding will not be classified as variable-length coding in this dissertation.

When coding whole blocks of symbols instead of individual symbols, it is possible to encode a given source symbol even with fractional number of bits. It is very important from the point of view of coding efficiency. In contrary to block Huffman coding, arithmetic coding encodes directly a given chain of symbols and do not need to create codewords for other blocks of symbols. This is an essential advantage of arithmetic coding that allows to omit a serious problem of high memory requirements that takes place in the case of block Huffman coding. Nevertheless, in comparison to computationally simple Huffman coding, both arithmetic encoding and arithmetic decoding are burdened with significantly higher complexity. Until quite lately it was a main reason (besides patents restrictions) of not using of complex arithmetic coding in systems of data compression. However, development of fast implementations of arithmetic coding [Penn88, Taub02, Marp03b] and recent increases of available computational power of digital processors have made more complex arithmetic coding become attractive for video compression systems. The above mentioned reasons yielded arithmetic coding to be applicable in highly efficient data compression and archiving systems [Mah05], still image coding (JPEG 2000 standard [JPEG2000]) and in contemporary hybrid video coders (standards H.263 and AVC [H263, AVC, Marpe03a, Richa03]).

Take a source S of N different symbols $S = \{x_1, x_2, ..., x_N\}$ and a set of N probabilities $P = \{p_1, p_2, ..., p_N\}$ assigned to these symbols. The working of N-ary arithmetic encoder can be divided into the following steps [Witt87, Riss79, Sayo00, Doma98, Sayo00, Przel05, Salom06]:

Step 1. Arithmetic encoder maps each symbol of alphabet into a certain sub-interval of the base interval $[0,1)$. The range of the given sub-interval is equal to the probability of occurrence of the given symbol. The way of mapping of symbols into sub-intervals is shown in Table 3.1.

Table 3.1. Mapping of symbols into sub-intervals.

Source symbol	Symbol probability	Assigned sub-interval
x_1	p_1	$[0, p_1)$
x_2	p_2	$[p_1, p_2)$
…	…	…
x_N	p_N	$\left[\sum_{k=1}^{N-1} p_k, 1\right)$

By encoding the first symbol x_m, the *m*-th sub-interval that has been associated with this symbol is chosen as the current (base) interval. Thus, the current interval is determined as $[a,b)$, with $a = \sum_{k=1}^{m-1} p_k$ and $b = \sum_{k=1}^{m} p_k$.

Step 2. When the next symbol x_n is read into arithmetic coder, the current interval $[a,b)$ is split into sub-intervals whose ranges depend on probabilities of individual symbols. Thus, the sub-interval associated with symbol x_n is determined as $[a+(b-a)\cdot c,\ a+(b-a)\cdot d)$, where *c* and *d* determine boundaries of the sub-interval that has been assigned to x_n symbol in the first step of arithmetic coding, so $c = \sum_{k=1}^{n-1} p_k$ and $d = \sum_{k=1}^{n} p_k$. The boundaries of the new current interval $[a,b)$ are calculated by substitution $a \Leftarrow a+(b-a)\cdot c$ and $b \Leftarrow a+(b-a)\cdot d$.

Step 3. Step 2 is performed until the last symbol will be encoded. Parameters of the current interval $[a,b)$ are calculated every time after encoding a new symbol.

Step 4. The final interval $[a,b)$ is the result of arithmetic coding for the whole sequence (block) of input symbols. Any binary number that lies within the final interval unambiguously represents the sequence of input symbols.

In Figure 3.2, the main idea of arithmetic coding has been presented by encoding of a hypothetical sequence of symbols $x_1x_2x_5$. Symbols have been generated by hypothetical source $S = \{x_1, x_2, x_3, x_4, x_5\}$ with assumed statistics $P = \{0.4, 0.2, 0.2, 0.1, 0.1\}$.

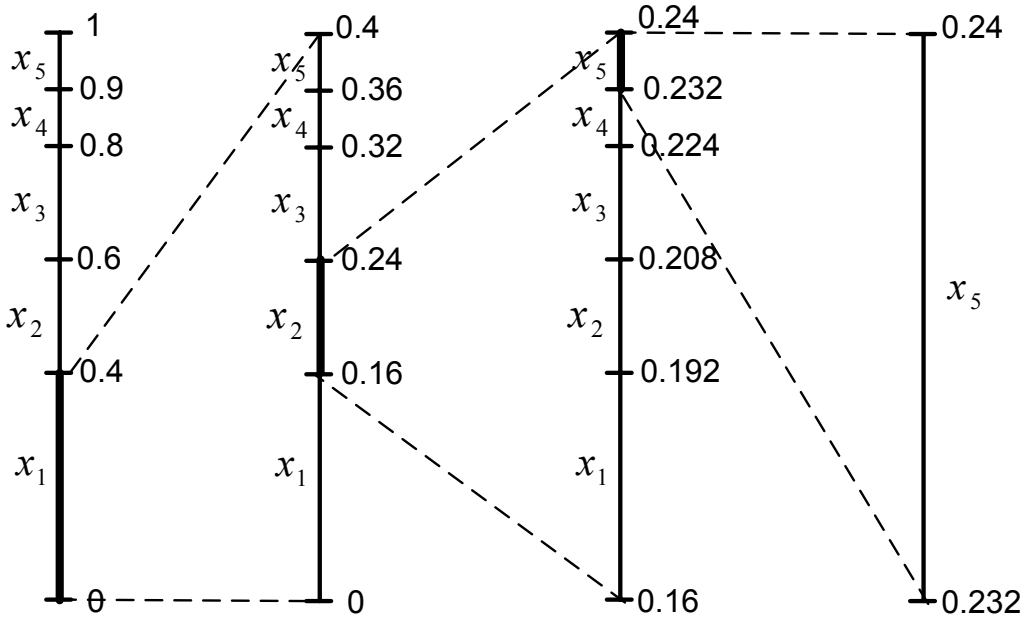


Figure 3.2. The main idea of arithmetic coding. (The idea of the drawing taken from [Sayo00]).

When coding a block of n symbols with arithmetic coding, mean length $\overline{L}$ of code per symbol is determined by Inequality 3.10 [Sayo00]

$$H(S) \le \overline{L} \le H(S) + \frac{2}{n}. \tag{3.10}$$

Inequality 3.5 and Inequality 3.10 yield that the maximum mean length of standard Huffman code is greater than the maximum mean length of arithmetic code when the number n of symbols in the coded block is greater than 2. However, comparing Inequality 3.7 with Inequality 3.10, the maximum mean length of a code for block Huffman coding is insignificantly smaller than the maximum mean length of code achieved when using arithmetic coding. Nevertheless, marginal theoretical superiority of block Huffman coding over arithmetic coding decreases with increasing of value n. The application of block Huffman coding for long blocks of symbols is practically impossible because of high memory requirements. In this situation, arithmetic coding is the best solution.

3.3.2. Practical realization of arithmetic coding

In order to represent exactly lower and upper boundaries of the current interval in arithmetic codec core for a long sequence of symbols, infinite precision of computations is needed. Such an arithmetic codec engine is in practice unrealizable. It was the main reason of existence of arithmetic coding method only in the area of theoretical considerations for a long

time. The problem of unlimited precision has been independently solved by Pasco [Pas76] and Rissanen [Riss76] in 1976 by developing an arithmetic codec engine where registers that represent the boundaries of the current interval exploit finite precision.

Implementations of arithmetic coders that are used in practice are based on proposals of Pasco and Rissanen and exploit fixed-point arithmetic. In these implementations, 16- or 32-bits precision for registers is mainly used to represent boundaries of the current interval $[L, H)$. In these registers, only fractional parts of interval boundaries are stored. For the reason that each sub-interval is included in interval $[0,1)$ the integer part of each number from any sub-interval is always the same and equal to 0 and do not have to be remembered.

Application of fixed precision for registers leads to serious limitation of the algorithm of arithmetic coding. Only a finite number of different blocks of symbols can be encoded with an interval that contains finite number of different numbers. In order to avoid this restriction, registers of arithmetic coders have to be renormalized during coding of source symbols. The idea behind renormalization of registers is simple. When the most significant bits of registers L and H are the same, it can be put to bitstream because the value of this bit will not change till the finish of coding. After that, registers L and H are modified by shifting their contents to the left; in this way their ranges are expanded. The least significant bit of register L is filled up with 0, and the least significant bit of register H is filled up with 1. It can occur that the algorithm can not produce the most significant bit and shift registers L and H. This problem (called underflow) may happen if the most significant bits in L and H do not match but differs by 1 and the 2-nd most significant bit in register H is 0 and the 2-nd most significant bit in register L is 1. In order to solve the underflow problem, contents of registers L and H have to be shifted left excluding the most significant bits, and the 2-nd most significant bits in L and H are overwritten with less significant bits. By shifting registers, the least significant bit of register L is filled up with 0 and the least significant bit of H is filled up with 1. After modifying the registers, coding of source symbols is continued.

More detailed description of arithmetic coders that exploit finite precision can be found in [Pas76, Riss76, Riss79, Sayoo00, Przel05].

Solution of the problem of infinite precision enabled arithmetic coding to be applicable in data compression and archiving systems. Nevertheless, high complexity of arithmetic coding was still a serious problem that limited its practical applications for a long time. A milestone in optimization of arithmetic coders was development of fast implementation of binary arithmetic codec, called Q-codec [Penn88] which was adapted to

work with binary alphabet. The increase of available computational power of microprocessors and discovery of fast implementations of arithmetic codec have caused that quite complicated arithmetic coding becomes attractive for data compression systems. Hence, modifications of Q-codec: QM-codec [Taub02] and MQ-codec [Taub02] were applied in JBIG [JBIG], JPEG [JPEG] and JBIG2 [JBIG2], JPEG2000 [JPEG2000, Taub02, Achar05b] image compression standards respectively. Arithmetic coding also started to be used in application to hybrid compression of video. Traditional multiplication- and division-based implementation of arithmetic codec is used in H.263 video coding standard [H263]. In Advanced Video Codec [AVC] fast implementation of binary arithmetic coding, the so-called M-codec [Marp03b, Marp06c] is used.

3.4. Data statistics modeling

Entropy coding techniques compress input data with respect to probabilities of occurrences of individual symbols. These probabilities are calculated by data statistics modeler. In the case of universal coding techniques (e.g. unary coding, *k*-th order Exp-Golomb coding), codewords are created in a regular way and a fixed probability density function (pdf) is assumed [Sayo00, Gall75]. Thus, in these methods there is no need to apply the data statistics modeler that estimates the statistics of coded data. However, these techniques can not be efficiently used when the real statistics of data differ from the assumed fixed probability density functions.

In contrast to universal coding techniques, Huffman and arithmetic coding can be more efficiently used for sources whose data statistics change in time. Before coding of source symbols, their probabilities have to be calculated. Thus, two stages of coding can be clearly distinguished in the case of Huffman and arithmetic coding. These are: data statistics estimation and proper coding of source symbols with respect to their probabilities.

Data statistics estimation is an essential part of entropy coder. It exploits mathematical model of source data. This model is used to describe the structure of source data. Thus, data statistics modeler estimates probabilities for source symbols on the basis of assumed model of source data. Therefore, the assumed mathematical model of source data (and the way of realizing it) has a great impact on values of probabilities of source symbols. Thus, this model has a crucial impact on coding efficiency of entropy coder. If the mathematical model corresponds well to the real model of source data, high compression performance of entropy

coding is achieved. If the mathematical model and the real model of source data are extremely different, entropy coding can even lead to expansion of data.

Hence, data statistics modeler is always built with respect to the type of source data that generates symbols. Generally, sources of data can be divided into two categories: memoryless sources and sources with memory. In the case of memoryless sources, each source symbol is independent from other symbols. So, the probability of the successive symbol does not depend on statistics of symbols that appeared earlier. In the case of sources with memory, probability of the successive symbol is strictly dependent on statistics of symbols that were generated earlier. Having knowledge about symbols that have been generated previously, it is possible to predict (with some probability) the value of the successive symbol.

One of the most popular and commonly used models of source with memory in data compression is Markov model that exploits *k*-th order discrete Markov chain [Sayo00]. *k*-th order Markov model has such a property that the value of the next source symbol is depended only on values of *k* past source symbols generated before. These past *k* symbols form the context. The conditional probability of the next symbol x_n that has been generated by *k*-th order Markov source fulfils the following Equation:

$$P\left(x_n / x_{n-1}, x_{n-2}, ..., x_{n-k}\right) = P\left(x_n / x_{n-1}, x_{n-2}, ..., x_{n-k}, ...\right). \tag{3.11}$$

Thus, data compression systems that use Markov model encode source symbols with respect to its context. The order of an assumed Markov model has usually big impact on efficiency of data modeling technique and its complexity. Generally speaking, parameters of an assumed Markov model should take into consideration the real structure of coded data.

The assumed model of source data can be realized in two ways: as a static model and as an adaptive model.

In simpler static model of source data, some predefined set of probabilities of coded symbols is used for the whole sequence of coded data. This set of predefined probabilities does not change during coding of symbols. Thus, the entropy coder does not calculate probabilities of symbols as they come and assumes that statistics of coded symbols does not change during coding. Therefore, if the real statistics of coded data significantly differs from the assumed one, the efficiency of entropy coding is reduced.

More sophisticated techniques of entropy coding exploits adaptive models. In this case, the algorithm of data statistics estimation also uses the predefined probability

distribution of data, but at the same time probabilities of symbols are updated during coding. In general, entropy coding techniques that exploit adaptive models are more computationally complex relative to techniques with static models. Nevertheless, the application of adaptive models allow for better adaptation to local statistics of coded data, which positively influences on compression performance of entropy coding.

Techniques of entropy coding used in contemporary hybrid video coders exploit both static and adaptive models. The non-stationary character of data that is read into entropy coder in video compression causes that statistics of coded data locally changes. Last comparisons of entropy coding techniques used in video coders have showed that techniques of entropy coding that exploit static models are characterized by significantly lower coding efficiency in comparison to techniques with adaptive models [Marp03a, Graj05]. Advanced hybrid video coders started to use entropy coding techniques with more efficient adaptive models [AVC]. Nevertheless, in order to trade off coding efficiency against complexity relatively simple adaptive models have been used in advanced video coders.

The goal of this dissertation is to explore whether more accurate and more complex adaptive models can be used to improve the coding efficiency of entropy coders used in advanced hybrid video coding.

Chapter 4

Entropy coding in hybrid compression of video

4.1. Entropy coding in the older hybrid coders – MPEG-1, MPEG-2 and H.263

In older hybrid video coders such as MPEG-1 [MPEG-1] and MPEG-2 [MPEG-2] simple non-adaptive techniques based on Huffman coding were used.

In MPEG-1 video coding standard only one predefined variable-length code table was used to encode quantized transform coefficients of prediction residual. The predefined variable-length code table was experimentally determined taking into consideration the data statistics from a set of tests video sequences. Such an approach has serious disadvantages that affect negatively on the compression performance:

- Statistics of coded data varies with time in a video sequence. Statistics also locally changes within frame (picture). So, application of one predefined variable-length code table can be very inefficient;
- Generally, data statistics of intra-frame is different from data statistics of inter-frame. So, the use of one variable-length code table for both types of prediction decreases coding efficiency of entropy coder;
- Huffman based entropy coding techniques can not efficiently encode symbols with probability of occurrence p greater than 0.5.

In MPEG-2 [MPEG-2] video coding standard, one of the above drawbacks was partially eliminated by the application of additional variable-length code table for more efficient encoding of transform coefficients in I-macroblocks. Nevertheless, entropy coder used in MPEG-2 was still not able to adapt efficiently to varying statistics of coded data.

In newer H.263 [H263] video coding standard, two different techniques of entropy coding were proposed. The first one is based on Huffman coding and the second one uses more efficient arithmetic coding. Variable-length coding (VLC) used in H.263 was a simple and non-adaptive variety of Huffman coding, similarly as in MPEG-1 [MPEG-1] and MPEG-2 [MPEG-2] video coding standards. Generally, it used one variable-length code table to encode motion vector data (MVD) and one variable-length code table to encode quantized transform coefficients. In order to increase compression performance of H.263, an optional, more efficient entropy coding technique called Syntax-based Arithmetic Coding (SAC) was defined. In comparison to simpler VLC coding, SAC technique can be distinguished by the following features that enhance the coding efficiency [Ran95]:

- SAC technique in a larger extent adapts coding to different data statistics in intra- and inter-macroblocks. Separate probability models have been defined for intra- and inter-macroblocks;
- The probability distribution of non-zero transform coefficients depends on theirs position in zig-zag ordered array. So, coding of transform coefficients has become dependent on this position in the ordered array. In order to do that four different probability models have been defined;
- SAC can efficiently encode symbols with probability p greater than 0.5 by application of arithmetic coding technique.

All these improvements give about 5% bitstream reduction in contrast to simpler VLC technique within H.263 video coding standard [Côté98, Erol98].

4.2. Entropy coding in advanced hybrid video coders – VC-1, AVS, AVC

Along with development of video compression algorithms, more sophisticated techniques of entropy coding were applied in advanced hybrid video coders, such as Video Coding 1 (VC-1) [VC-1, Kalv07], Audio and Video Coding Standard of China (AVS) [AVS], and Advanced Video Coding (AVC) [AVC] (see Section 2.4).

4.2.1. Entropy coding in VC-1 and AVS video coders

In both VC-1 and AVS video coders, only simpler variable-length coding techniques were used. They do not use techniques that are based on more efficient arithmetic coding. Nevertheless, the video coders employ techniques of VLC coding that adapt much better to the current signal statistics as compared to VLC techniques applied in the older hybrid video coders (MPEG-1, MPEG-2 and H.263).

VLC technique applied in VC-1 realizes multilevel adaptation to the statistics of coded data [VC-1]:

- Encoding of motion vectors (MV) is different for progressive- and interlace-coding. In order to adapt coding of motion vectors to the current data statistics, four separate adaptively chosen VLC tables have been defined;
- In order to efficiently encode data that represents the quantized transform coefficients, separate sets of VLC tables for luma and chroma components have been prepared. The statistics of quantized transform coefficients also significantly differs for different size of output bitstream. For that reason different sets of VLC codes have been defined in VC-1 to efficiently encode transform coefficients for low-, medium- and high bitrates.

AVS video coder encodes quantized transform coefficients in a reverse zig-zag scan order by ascribing one VLC code to a pair (run, level). In order to track changing statistics of coded data nineteen separate two-dimensional 2D-VLC tables that contains 0-th, 1-th, 2-nd and 3-rd order Exp-Golomb codes have been defined [Wan04]. 2D-VLC codes have been ascribed to (run, level) pair while taking into consideration the fact that the level of non-zero transform coefficients increases and the run of zeros decreases while moving from higher frequency part to lower frequency part. A proper 2D-VLC table is chosen according to used prediction mode (intra- or inter- prediction), and type of coded data (luma component or chroma component), similarly as it takes place in VC-1 video coder. Additionally, 2D-VLC tables are switched based on the value of previously coded non-zero transform coefficient. Coding of motion vectors is simpler than used in VC-1 and exploits 0-th order signed Exp-Golomb codes.

4.2.2. Entropy coding in Advanced Video Coder AVC

Two alternative techniques of entropy coding have been defined in Advanced Video Coder (AVC). These are: simpler Variable-Length Coding (VLC) and more computationally complex but more efficient Context-based Adaptive Binary Arithmetic Coding (CABAC).

Variable-Length Coding of AVC will be called Universal Variable-Length Coding (UVLC) in this dissertation. The methods that have been used in UVLC and CABAC entropy coding techniques have been introduced in Table 4.1.

Table 4.1. Entropy coding techniques used in AVC video coder.

Entropy coding technique	**Used methods**	**Features of the entropy coding technique**	
		Complexity	Compression performance
Universal Variable-Length Coding (UVLC).	1) Context-Adaptive Variable Length Coding (CAVLC) used for coding of quantized transform coefficients. 2) 0-th order Exp-Golomb coding used for coding of motion vectors and other syntax elements.	Lower	Lower
Context-based Adaptive Binary Arithmetic Coding (CABAC).	Binary arithmetic coding and adaptive models of source data.	Higher	Higher

4.2.2.1. Universal Variable-Length Coding (UVLC) in AVC

Universal Variable-Length Coding (UVLC) is a simpler (as compared to CABAC) technique of entropy coding used in AVC [Richa03, AVC]. It is used in the *Baseline Profile* and exploits two methods of entropy coding:

- Non adaptive 0-th order Exp-Golomb coding with computationally simple both encoding and decoding procedures [Golo66];
- More sophisticated but more complex Context-Adaptive Variable Length Coding (CAVLC).

The probability distribution of motion vector prediction residuals is similar to geometrical distribution [Lange06]. For that reason, 0-th order Exp-Golomb coding is used. Additionally, the control data that indicates the prediction mode of macroblock is also coded with 0-th order

Exp-Golomb codes. The structure of 0-th order Exp-Golomb codes has been introduced in Figure 4.1.

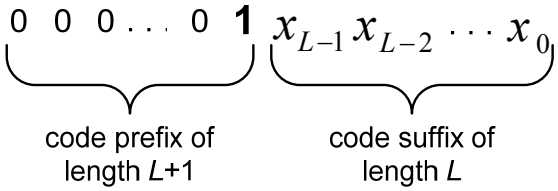


Figure 4.1. Structure of 0-th order Exp-Golomb codes.

0-th order Exp-Golomb codes have an essential feature that the length L of suffix of code is always shorter by 1 than the length L+1 of prefix of code. It significantly reduces the complexity of both encoding and decoding of 0-th order Exp-Golomb codes. Thus, 0-th order Exp-Golomb codes used in AVC trades off both complexity and coding efficiency.

In encoded bitstream, data of quantized transform coefficients usually constitutes more than a half of the whole bitstream of encoded data. Therefore, entropy coding technique used for quantized transform coefficients of prediction residual has a key significance on compression performance. Thus, more efficient but also more time consuming Context-Adaptive Variable Length Coding (CAVLC) technique has been applied in AVC to encode quantized transform coefficients of prediction residual [AVC, Richa03].

CAVLC entropy coding technique is based on Huffman coding and has been adapted to the statistics of data that represents quantized transform coefficients of prediction residual:

- The number of non-zero valued transform coefficients in neighboring blocks of frame is highly correlated;
- After the zig-zag scanning of quantized transform coefficients, many coefficients in the highest frequency part have zero value or have values of +1 or -1. Additionally, the non-zero coefficients in the scanned sequence are often separated from each other by a sequence of zero-valued transform coefficients. In the lower frequency part the quantized transform coefficients have usually large values. So, the magnitude of the non-zero valued transform coefficients usually decreases with the increase of frequency.

Taking into consideration the statistics of quantized transform coefficients of prediction residual, there is defined a set of parameters that allows for efficient representation of quantized transform coefficients of prediction residual. These are:

- The number of non-zero valued transform coefficients *TotalCoeff* and the number of coefficients with amplitude equal to 1 *TrailingOnes*;

- The amplitude of non-zero transform coefficient *Level*;
- The number of zero-valued transform coefficients *TotalZeros* that occur before the last non-zero transform coefficient in scanned sequence of transform coefficients;
- The number of zero-valued coefficients *RunBefore* that precede a given non-zero transform coefficient.

The number of non-zero transform coefficients *TotalCoeff* and the number of coefficients with amplitude equal to 1 *TrailingOnes* are coded jointly as one symbol called *coeff_token*, one codeword is assigned to *coeff_token* parameter. In neighboring 4x4 blocks both the number of non-zero transform coefficients *TotalCoeff* and the number of coefficients with amplitude equal to 1 *TrailingOnes* are correlated. Therefore, four different tables of codes have been defined to encode efficiently the *coeff_token* parameter. These are: three variable-length code tables and one fixed-length table with 6-bits codes which are adaptively chosen based on the number of *TotalCoeff* in the left and the upper 4x4 blocks relative to the current block. In order to increase the compression performance of CAVLC, the separate table of codes has been defined for chroma blocks for the reason of different data statistics in comparison to statistics in luma blocks. In this way, CAVLC technique adapts to the current signal statistics by switching between different tables of codes.

Because of the fact that the amplitude of non-zero coefficients usually tends to decrease with the increase of frequency, the non-zero coefficients are coded starting from the highest frequency part of spectrum and ending on the lowest frequency part. In this way coefficients are usually coded in order of increasing amplitude. There exists a correlation between the amplitude of a given non-zero coefficient and the amplitude of non-zero coefficients coded earlier. The amplitude of non-zero coefficient is coded by determining two parameters: a prefix of coefficient's amplitude (*level_prefix*) and a suffix of coefficient's amplitude (*level_suffix*). The *level_prefix* is coded with one VLC table of codes. The *level_suffix* is coded adaptively with taking into consideration the amplitude of previously coded non-zero transform coefficient. A sign of each non-zero coefficients is additionally coded using 1 bit.

In order to encode efficiently the position of non-zero coefficients in the scanned sequence, two parameters are finally coded. These are: the number of zero-valued coefficients *TotalZero* before the last non-zero coefficient and the parameter *RunBefore* that describes the distribution of zero-valued coefficients among non-zero coefficients. The quantity *TotalZeros* is strictly dependent on the number of non-zero coefficients. Therefore, sixteen VLC tables of

codes that are adaptively chosen based on the value of *TotalCoeff* have been defined. Coding of parameter *RunBefore* is dependent on the number of zero-valued coefficients before the previously coded non-zero coefficient. So, it also depends indirectly on the *TotalZeros* parameter.

4.2.2.2. Context-based Adaptive Binary Arithmetic Coding (CABAC) in AVC

The state-of-the-art entropy coding technique used in advanced hybrid compression of digital video is Context-based Adaptive Binary Arithmetic Coding (CABAC) [Marp03a, Marp04, Richa03]. CABAC is the other entropy coding technique that can be used in AVC [Richa03, Wieg03a, AVC] in *Main* and *High* profiles. Three elementary functional blocks can be distinguished in CABAC: the binarizer of input symbols, the context modeler that estimates conditional probabilities of binary symbols and the binary arithmetic codec that encodes each binary symbol with respect to the conditional probability of its occurrence (see Figure 4.2).

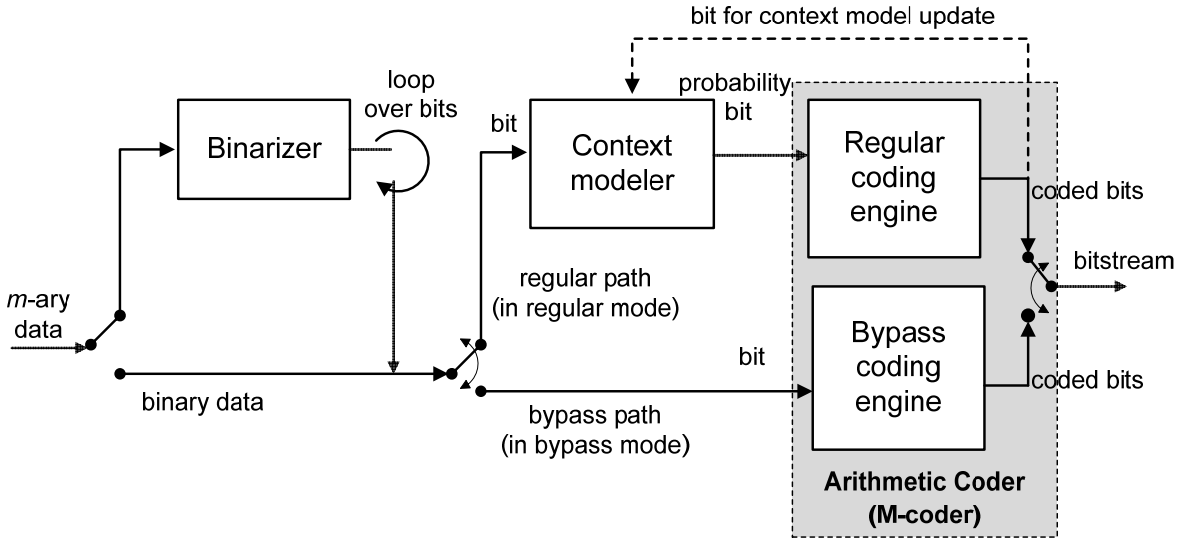


Figure 4.2. CABAC encoder block diagram. The idea of the drawing taken from [Marp03a].

In Advanced Video Coder, CABAC technique is optionally used to reduce statistical redundancy in quantized transform coefficients of prediction residual, motion vectors of prediction residual and control data.

4.2.2.2.1. Binarization process in CABAC

In CABAC, due to the application of binary arithmetic coder, all non-binary valued syntax elements have to be mapped into a string of binary symbols. This is realized by the

binarizer block at the first stage of coding. The binarizer block has a huge impact on the number of binary symbols that are fed to binary arithmetic codec, which influences on the size of bitstream at the output of entropy codec. For that reason, in order to reduce the number of binary symbols at the output of the binarizer block, the binarization in CABAC has been adapted to statistics of non-binary valued syntax elements by application of five different basic binarization schemes: unary, truncated unary, *k*-th order Exp-Golomb, fixed-length and Huffman-based binarization. In order to encode more efficiently data that represent quantized transform coefficients of prediction residual and motion vectors of prediction residual, two more binarization schemes have been defined which are concatenation of unary binarization and *k*-th order Exp-Golomb binarization (UEG*k*). These are:

- Concatenation of unary binarization and 0-th order Exp-Golomb binarization (UEG0) that is used for transform coefficients;
- Concatenation of unary binarization and 3-rd order Exp-Golomb binarization (UEG3) that is used for motion vectors.

The binarization schemes used in CABAC have been listed in Table 4.2.

Table 4.2. Binarization schemes used in CABAC [Marp03a].

	Binarization scheme				
	Huffman-based	Unary	Truncated unary	UEG0, UEG3	Fixed-length
Application	Type of macroblock and type of sub-macroblock	Index of reference frame, QP parameter for macroblock	Type of prediction for chroma in I-macroblocks, indication of chroma blocks with non-zero transform coefficients	Motion vectors and transform coefficients	Indication of luma blocks with non-zero transform coefficients, other syntax elements coded in macroblock and block layer

Individual binarization schemes exploit the following entropy coding algorithms:

- Unary binarization is based on unary coding;

- In contrast to unary binarization, the truncated unary binarization has been defined for a finite set of integer valued-syntax elements with maximum value S ($0 \le x \le S$). When $x < S$ the truncated binarization corresponds to the unary binarization. For $x = S$ the last 0 bit of unary code is omitting;
- Fixed-length binarization encodes syntax elements with a fixed number of bits;
- *k*-th order Exp-Golomb binarization exploits *k*-th order Exp-Golomb coding (e.g. $k = 0$ or $k = 3$);
- Huffman-based binarization exploits predefined Huffman codes.

Thus, the binarization in CABAC works similar as variable-length coding (e.g. Huffman coding) but in contrast to variable-length coding the inter-symbols redundancy is extra reduced with arithmetic coding.

4.2.2.2.2. Context modeling in CABAC

Binary arithmetic encoder encodes the input binary symbols with respect to the conditional probabilities of their occurrence in the video data stream. The conditional probabilities of binary symbols are estimated by the context modeler. The way of calculating these probabilities has also a great influence on compression performance of entropy coder. So, in order to obtain an accurate adaptation to the current signal statistics, the total number of 399 individual finite-state machines (FSM) are used by the context modeler block (this is only for the case of a transform calculated in 4x4 blocks). The individual FSM calculates probabilities of symbols for selected context. Such finite-state machines will be referred as statistical models. It has been shown on Figure 4.3.

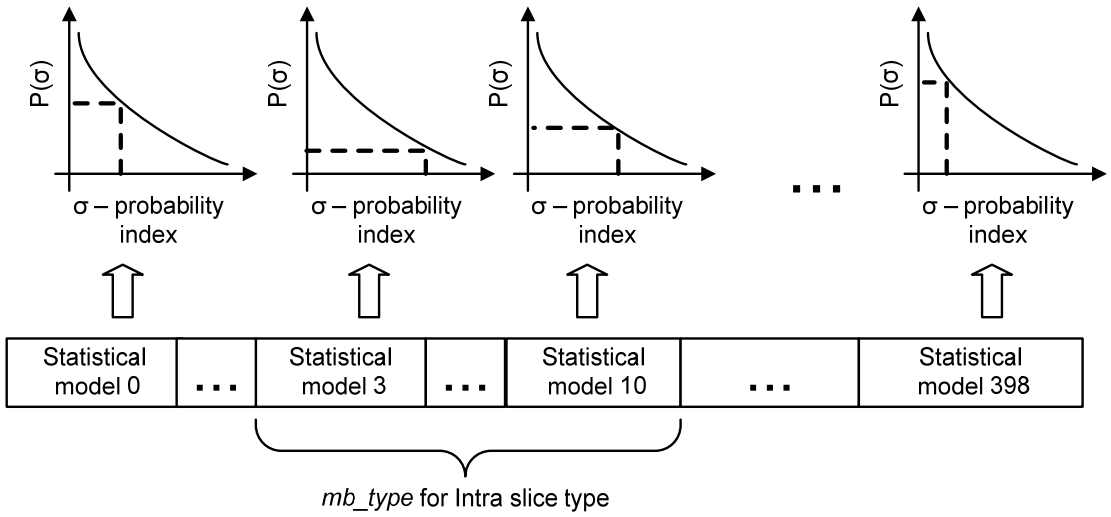


Figure 4.3. Definition of probability models in CABAC.

Each of defined probability models estimates independently the statistics of coded data. The statistical distribution of data is characterized by two variables: the value of most probable symbol MPS (equal to 0 or 1) and the probability index σ that is explicitly related to the probability of least probable symbol LPS. By encoding of successive source symbols both value of MPS symbol and probability index σ change according to the algorithm of probability estimation described in Section 4.2.2.2.3. In CABAC, a given syntax element uses some sub-set of all possible probability models. For a given syntax element, the sub-set of probability models has been defined with respect to the statistic of the binary symbols in the binarized word. By coding of the binary symbol, one proper probability model has to be chosen. In order to do that, the adaptation algorithm exploits the statistics of coded syntax element from neighboring blocks (and more precisely from the left block and the upper block relative to the current block). Based on the values of syntax element in neighboring adjacent blocks (usually 4x4 blocks) the proper probability distribution is chosen. It allows the adaptation algorithm to adapt rapidly to the current statistics of two-dimensional signal. For example, for the syntax element *mb_type* that is sent in header of macroblock and means the type of macroblock the selection of statistical model is introduced on Figure 4.4.

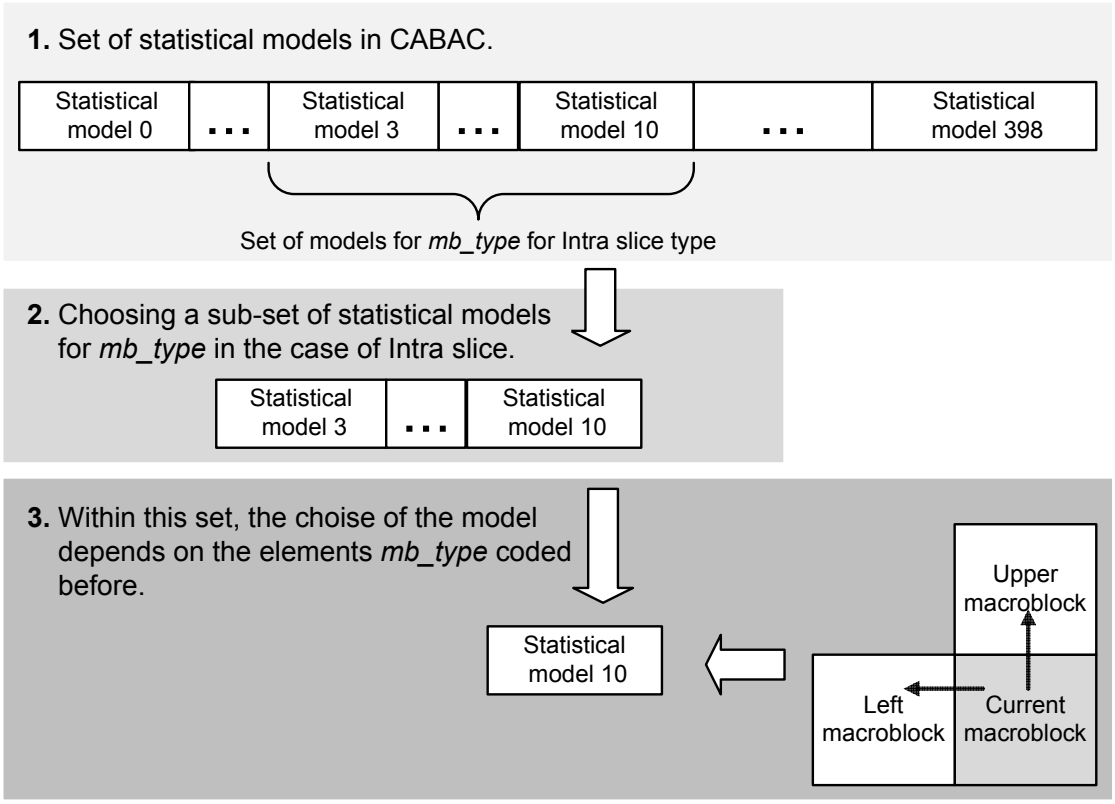


Figure 4.4. Selection of statistical model in CABAC.

In this way, two levels of adaptation to the current signal statistics are realized in CABAC technique. The algorithm of data statistics estimation used in CABAC belongs to most advanced used in entropy coders in hybrid compression of digital video.

4.2.2.2.3. Probability estimation and binary arithmetic coding in CABAC

The binary symbol together with its conditional probability is finally read into arithmetic encoder. In order to decrease the complexity of both encoder and decoder, fast implementation of multiplication-free and division-free binary arithmetic codec (the so-called modulo-codec or M-codec) has been used in CABAC [Marp03b]. In M-codec, the input binary symbol is considered as *most probable symbol* (MPS) or *least probable symbol* (LPS). The value of MPS (0 or 1) depends on the number of currently used statistical model and is updated every time during the coding process of binary symbol. In arithmetic codec core, by encoding a successive symbol, the current interval $[L, L+R)$ is divided into two sub-intervals, one associated with MPS and the other associated with LPS. The ranges of intervals and probabilities assigned to MPS and LPS have been expressed by Equation 4.1. and Equation 4.2

$$R_{LPS} = R \cdot p_{LPS}, \tag{4.1}$$

$$R_{MPS} = R - R_{LPS}, \;\; p_{MPS} = 1 - p_{LPS}. \tag{4.2}$$

The probability p_{LPS} of LPS is estimated by the context modeler block with respect to the number of actually used statistical model.

Division of the current interval into two sub-intervals (Equation 4.1) is the most time-consuming operation of each binary arithmetic codec. In M-codec, the complex multiply operation from Equation 4.1 has been replaced by fast memory access LUT (Look-Up Table). In order to do that, all the values of interval ranges R and probabilities p_{LPS} have been quantized to a limited set of possible values $Q = \{Q_0, Q_1, ..., Q_{K-1}\}$ and $P = \{p_0, p_1, ..., p_{N-1}\}$ respectively. The number of elements in sets Q and P have a huge impact on coding efficiency as well as memory complexity of binary arithmetic codec. Since CABAC trades-off computational complexity, memory complexity and coding efficiency, the new range R_{LPS} is approximated with a set of $K = 4$ quantized values of interval range and $N = 64$ predefined quantized values of conditional probabilities for LPS. In M-codec, the pre-computed values $Q_\rho \cdot P_\sigma$ of 4x64 products (for $0 \le \rho \le K-1$ and $0 \le \sigma \le N-1$) are stored in memory.

The limitation of both computational and memory complexity of M-codec has been mainly possible by significant simplification of the conditional probabilities estimation technique. A limited set of only 128 different quantized values of conditional probabilities ranging in the interval $P_\sigma \in [0.01875, 0.98125]$ has been defined in CABAC; 64 values of probabilities for LPS $P_{LPS} \in [0.01875, 0.5]$ and 64 equivalent probabilities for MPS with values $P_{MPS} = 1 - P_{LPS}$. In order to accelerate both probability estimation and probability update processes, they have been realized with a finite-state machine (FSM) in CABAC. Each state of the FSM defines the conditional probability of least probable symbol LPS and the transition rule between states of FSM that is depended on the value of currently coded symbol (0 or 1) (see Figure 4.5). The used transition rules between states of FSM from time unit *t* to time unit *t*+1 are based on the method of Howard and Vitter [Howa92] and can be described by Equation 4.3.

$$P_{LPS}^{(t+1)} = \begin{cases} \alpha \cdot P_{LPS}^{(t)} & \text{if an MPS previously occured} \\ \alpha \cdot P_{LPS}^{(t)} + (1-\alpha) & \text{if an LPS previously occured} \end{cases} \tag{4.3}$$

Based on Equation 4.3, the scaling factor α it determined as:

$$P_{LPS}^{\min} = P_{LPS}^{\max} \cdot \alpha^{N-1}, \quad N = \frac{128}{2} = 64, \tag{4.4}$$

$$\alpha = \left(\frac{0.01875}{0.5} \right)^{\frac{1}{63}} \approx 0.95. \tag{4.5}$$

Above equations are citations from [Marp03a].

The estimation process of the conditional probabilities used in CABAC assumes the "exponential aging" model of coded data [Howa92].

In order to decrease the complexity of M-codec, all 128 conditional probabilities used in CABAC have been pre-computed and together with the procedure of probability update (procedure of probability update is depended on the value of currently coded symbol, LPS or MPS) encapsulated into a finite-state machine (FSM). For each of 399 used probability models (called contexts) in CABAC (given number of contexts concerns only the transform calculated in 4x4 blocks) the independent FSM has been ascribed. By coding a new symbol with a given probability model, the FSM tracks the statistics of coded data and modifies the parameters of the currently used probability model (context). The conditional probability of symbol that is currently coded is unambiguously determined by the current state σ of the given FSM, the number of current state σ of the FSM together with the quantized value $Q(R)$

of the current interval R of M-codec are used to calculate the new ranges R_{LPS} and R_{MPS} of LPS and MPS symbols respectively.

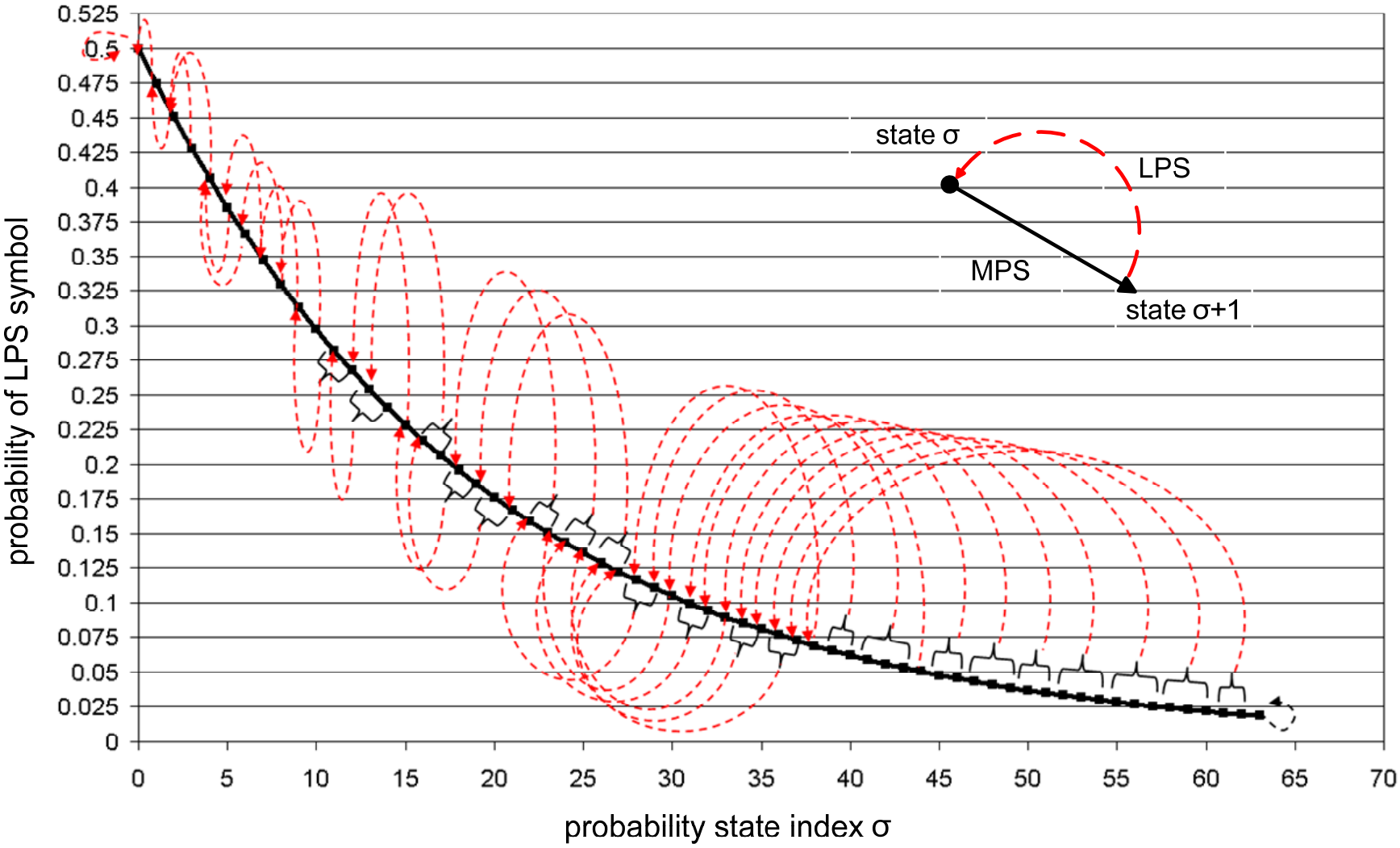


Figure 4.5. Probability estimation in CABAC algorithm for LPS symbol. The idea of the drawing taken from [Marp03a].

Complexity of M-codec is lower than all commonly known and used low-complexity binary arithmetic codecs such as QM codec [Taub02] (used in JBIG [JBIG] and JPEG [JPEG] image compression standards) and MQ codec [Taub02] (used in JBIG2 [JBIG2] and JPEG2000 [JPEG2000, Taub02, Achar05b] image compression standards). In comparison to MQ-codec that has been considered the state-of-the-art fast algorithm of binary arithmetic coding so far, the throughput rate of M-codec is even 5%-18% higher [Marp04]. Additionally, the bitrate at output of M-codec is 2%-4% smaller than the bitrate at output of MQ-codec [Marp04]. In application to video compression, the coding efficiency of M-codec is virtually the same as the coding efficiency of traditional arithmetic codec with time-consuming multiplication- and division-operations [Marp06a].

Highly optimized M-codec is still more complex than a traditional variable-length codec. In order to additionally limit the computational power needed to encode and decode a binary symbol, a bypass mode of arithmetic codec is used in CABAC. The bypass arithmetic coding is a simplified mode of arithmetic coding used for certain binary symbols with approximately uniform probability distribution. In the bypass mode, coding of binary symbols has been significantly accelerated by omitting complex probability estimation and probability update procedures.

4.2.2.2.4. Procedure of contexts initialization in CABAC

Each probability model defined in CABAC tracks the statistics of coded data by modifying two variables that correspond to the value of most probable symbol MPS and the probability of least probable symbol LPS. By encoding source symbols the algorithm of probability estimation adapts better and better to the statistics of coded signal, it means that successive symbols are encoded more and more efficiently.

In order to assure the entropy decodeability of the next slice without the entropy decoding of the preceding slice, data statistics gathered in a given slice are not directly exploited in the next slice in CABAC. On the other hand, the lack of knowledge of source symbols statistics negatively affects the compression ratio of arithmetic encoder engine at the beginning of a new slice and this significantly influences the size of resulted bitstream. This problem occurs especially in the case of small slices (P and B slice types).

Because of that, in order to make possible fast adaptation of the modeling block to the current signal statistics, some a priori knowledge of symbols probability distribution has to be exploited. In CABAC, this is realized by contexts initialization procedure that is invoked at the beginning of each new slice. The context initialization of CABAC sets two variables at the

beginning of each new slice with some pre-defined values: the value of the most probable symbol MPS (equal to 0 or 1) and the state σ of the FSM that corresponds to the probability of least probable symbol LPS. The initialization of values of MPS and the state σ is controlled by certain *m* and *n* parameters that are determined by H.264 recommendation [AVC] for each of 399 defined contexts.

The contexts initialization in CABAC has been determined taking into consideration the fact that the statistics of coded video data is depended on:

- The type of slice (I-slice type, P-slice type or B-slice type);
- The nature of video sequence (sequence with dynamic motion or sequence with slow motion) and the content of video sequence;
- The quality of coded video sequence expressed as the size of resulted bitstream.

Thus, *m* and *n* parameters have been independently defined for each of three slice types. In order to take into account different nature of video signal for sequences with dynamic and slow motion as well as relationship of video data statistics with content of video sequence, three different sets of *m* and *n* parameters have been calculated for P- and B-slices. Based on the signal statistics in the current inter-slice, the best set of initialization parameters out of three different sets (in terms of compression efficiency) is chosen for the successive inter-slice.

Statistics of coded data is strongly dependent on the quality of coded sequence. In order to include it in a reckoning, parameters that describe probability model (value of MPS and state σ) are derived from *m* and *n* parameters with taking into consideration the value of quantization parameter (QP) for slice and luma.

The used slice- and QP-depended initialization of context models allows for additional improvement of compression performance of CABAC. Experimental results showed that the bitrate savings of 0-3% are possible when using context initialization in the case of interlaced television sequences at low bitrates [Schw02b].

4.2.2.2.5. CABAC technique – conclusions

CABAC algorithm makes a huge progress in the development of techniques of entropy coding used in hybrid compression of video. It uses efficient binary arithmetic coding that works with sophisticated technique of data statistics estimation. The technique of data statistics estimation used in CABAC exploits adaptive models of source data and it surely belongs to most efficient data modeling methods that has been ever applied in digital video

coders. Therefore, CABAC algorithm provides considerably better compression performance than any other entropy coder commonly used in video compression [Marp03a].

4.3. Coding efficiency of entropy coders within hybrid video coding

With respect to algorithms of data statistics estimation, entropy coding techniques defined in advanced video coders (VC-1, AVS, and AVC) are superior to techniques used in the older video coders (MPEG-1, MPEG-2 and H.263). Therefore, entropy coding techniques that have been developed recently are surely much more efficient than techniques used in older hybrid video coders (MPEG-1, MPEG-2 and H.263). Unfortunately, the direct comparison of efficiency of different entropy coders that work within different video coders is extremely difficult. In order to increase compression performance, entropy coders have been adjusted to the statistics of data at their inputs. These data statistics are usually different for different video coders for the reason of different techniques of video compression used in individual video coders. Nevertheless, the level of complexity of VLC-based entropy coders in VC-1, AVS and AVC allows for claiming that the coding efficiency of these techniques is comparable.

The most advanced entropy coding technique that has ever found the common application in hybrid video coding is CABAC technique [Marp03a]. It exploits sophisticated mechanism of adaptation to the current signal statistics and efficient arithmetic coding. CABAC algorithm is characterized by very high coding efficiency. It is a milestone in arithmetic coding techniques used in digital video compression. In hybrid compression of video, the state-of-the-art entropy coding technique that is based on variable-length coding is UVLC technique used in AVC video coder. In order to compare efficiency of CABAC with coding efficiency of UVLC technique within AVC, series of experiments have been done.

4.3.1. Coding efficiency of entropy coders within AVC

Coding efficiency of CABAC to coding efficiency of UVLC within AVC video coder has been already compared and experimental results have been well presented in the literature [Marp03a, Graj05]. According to these experimental results the application of CABAC algorithm within AVC leads to 6%-23% bitrate savings relative to simpler UVLC entropy coding technique. Unfortunately, those experiments were done with different video sequences relative to test sequences used in this dissertation.

In order to compare the coding efficiency of UVLC and CABAC in the same conditions as evaluation of own research, the author has done his own experiments on coding efficiency of CABAC and UVLC within AVC video coder. Experiments have been done with the CITY, CREW, ICE and HARBOUR test sequences, each in 704x576 spatial resolution and 60 frames per second (see Annex F). Tests have been done with intra- and inter-prediction modes by setting the structure of GOP on I29P. Experiments have been done for a wide range of QP parameter values with both rate-distortion optimization and rate control switched off. For a given QP parameter value 600 frames of each of the CITY, CREW and HARBOUR test sequences and 480 frames of the ICE video sequence have been encoded and decoded with UVLC and CABAC.

Coding efficiency of CABAC has been compared against coding efficiency of UVLC within AVC. Hence, coding efficiency of CABAC has been expressed as a percentage bitrate reduction in comparison to the bitrate obtained with UVLC. The detailed experimental results for test sequences have been presented in Annex E in Figure E.1 to Figure E.4. Averaged results for 4 test sequences have been presented in Figure 4.6. Averaged results have been presented for the typical in digital television range of useful bitrates.

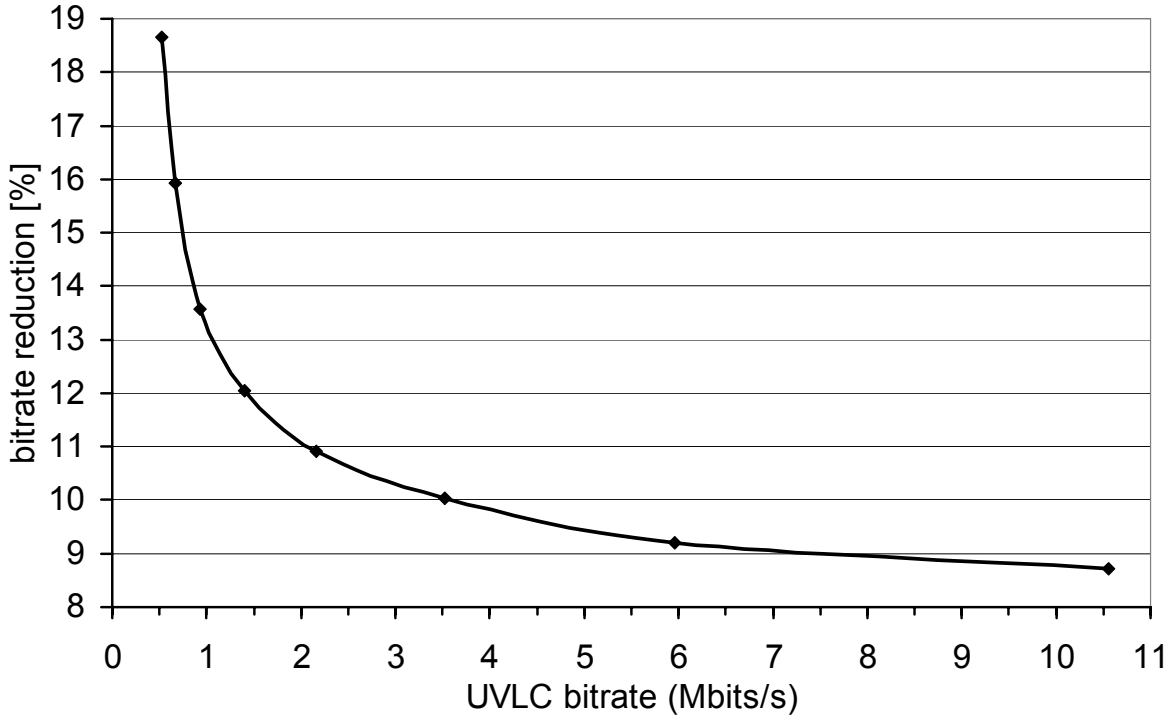


Figure 4.6. Average compression gain due to application of CABAC instead of UVLC (average for 4 test sequences: CITY, CREW, HARBOUR, ICE).

The author's experimental results prove that coding efficiency of CABAC is significantly higher relative to coding efficiency of UVLC. CABAC algorithm significantly outperforms UVLC technique by 6% to even above 20%. The obtained results are quite compliant to those from [Marp03a] where the bitrate reductions of 5%-21% were obtained due to application of CABAC instead of UVLC. It is indication to that the methodology of comparison used in the dissertation is compatible with those applied in [Marp03a].

The higher coding efficiency of CABAC relative to UVLC within AVC is mainly a result of:

- Using the technique of arithmetic coding which is generally more efficient that techniques of variable-length coding;
- Application of much more advanced techniques of data statistics estimation in the case of data that represents motion vectors of prediction residual and control information relative to techniques used in the UVLC method.

Bitrate reduction obtained when using CABAC strictly depends on the value of QP parameter and the content of test sequence. The smaller size of output bitrate of the test sequence (so, the bigger value of QP parameter) the higher coding efficiency of CABAC relative to UVLC technique. Such a result has been obtained for all test sequences. The same observation has been noticed in experimental results from [Marp03a]. The variable-length codes of UVLC have been determined with assumption of certain statistics of coded data. It means that assumed data statistics differs from the real statistics in the case of lower bitrates, which results with poorer coding efficiency of UVLC in that cases.

4.3.2. Complexity of CABAC decoder relative to UVLC decoder

CABAC technique exhibits extraordinary coding efficiency relative to advanced entropy coding methods based on the variable-length coding (like UVLC technique). Very high coding efficiency of CABAC has been achieved by a significant increase of the complexity of entropy encoding and entropy decoding. In order to accurately test the complexity of CABAC relative to UVLC technique, the author has done experiments.

CABAC entropy encoder as well as CABAC entropy decoder perform almost the same arithmetic operations in order to encode or decode a bitstream. Therefore, the complexity of both CABAC encoder and CABAC decoder is very comparable. In contrast to entropy coding techniques based on arithmetic coding (like CABAC) the techniques based on variable-length coding (like UVLC) are marked by asymmetry in the complexity of entropy encoder and entropy decoder. In general, the variable-length code decoder is much more time-

consuming in comparison to the variable-length code encoder. Therefore, the author has compared the complexity of CABAC decoder relative to the complexity of UVLC decoder. Author's comparison of total decoding times for CABAC decoder and UVLC decoder is reliable for the reason that experiments have been done with optimized for speed both CABAC and UVLC decoders.

Experiments have been done on the same set of test sequences as experiments presented in Section 4.3.1. The reference implementation of AVC video decoder has been used [AVCSoft].

Total entropy decoding times for UVLC and CABAC within the JM 10.2 reference software of AVC video coding standard have been measured for all test sequences. The optimized implementation of CABAC decoder that exists within JM 10.2 reference software has been used. For the reason that the reference UVLC decoder (from JM 10.2) was not optimized for speed, and author's optimized implementation of UVLC decoder has been used. The optimized implementation of UVLC decoder has been based on author's method of efficient search of binary trees with variable-length codewords as presented in [Karw04b]. During tests, total decoding times of CABAC and UVLC have been measured with *QuerryPerformanceCounter*() function for all test sequences. The *QuerryPerformanceCounter*() function comes from the Win32API library and counts the number of processor ticks needed to execute a given fragment of program code. The decoding times of CABAC decoder have been compared to decoding times of UVLC decoder. In Figure E.5. to Figure E.8. the experimental results on increase of the total decoding time for CABAC decoder relative to total decoding time for UVLC decoder have been presented. Tests have been done on Intel Core 2 Duo E6600 platform (2.4 GHz, 4MB of memory cache of Level 2) with 2 GB of RAM under the 32-bit Windows XP with Service Pack 2 operation system. The source code of AVC video decoders with CABAC and UVLC have been compiled in the release mode with Intel C++ Compiler (in version 10.0.025) for 32-bit Intel Architecture (IA-32) of microprocessors [IntelComp].

The detailed experimental results for test sequences have been presented in Annex E in Figure E.5 to Figure E.8. The averaged experimental results obtained for CITY, CREW, HARBOUR and ICE test sequences have been presented in Figure 4.7. Averaged results have been introduced for the typical (in digital television) range of useful bitrates.

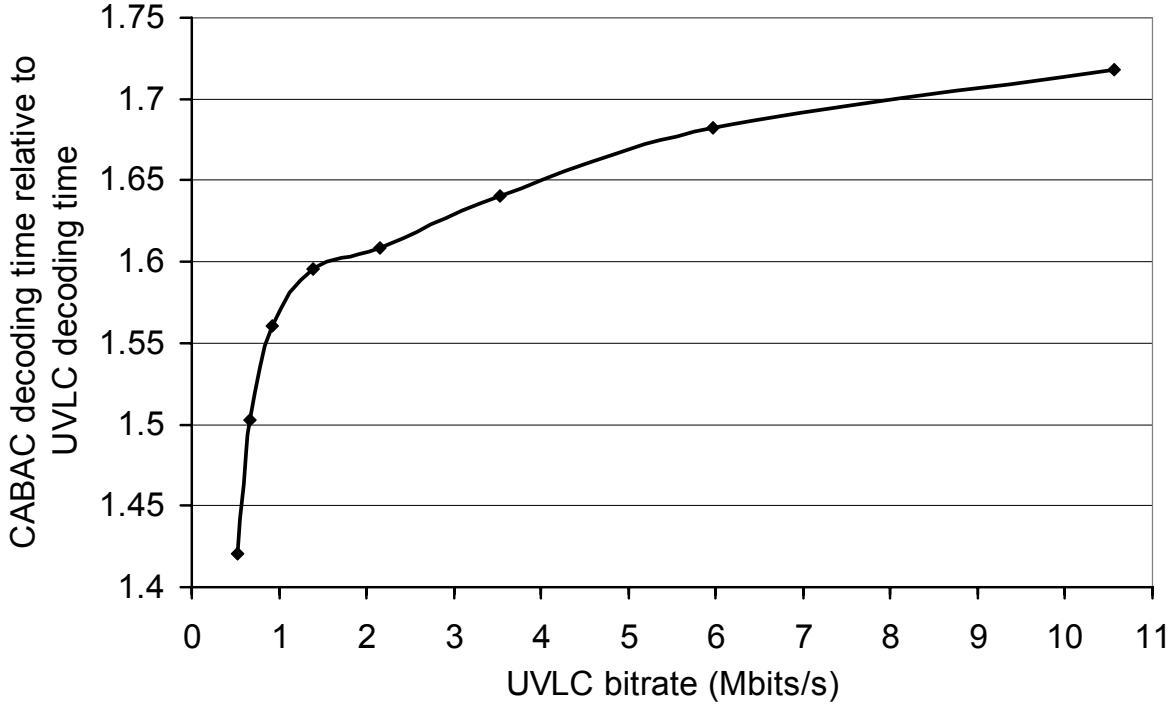


Figure 4.7. Average increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC (average for 4 test sequences: CITY, CREW, HARBOUR, ICE).

The obtained experimental results proved that higher coding efficiency of CABAC technique (relative to UVLC method) has been achieved by significant increase of total entropy decoding time, and what also involved with it, total entropy encoding time. The optimized CABAC decoder is approximately 1.5 times slower than the optimized UVLC decoder for bitrates of order of a few mega bits. Additionally, the relative increase of total decoding time for CABAC decoder (in comparison to UVLC decoder) further grows for higher bitrates. In the case of extremely high bitrates (greater than 50 Mbits/s) the CABAC decoder is more than two times slower that simpler UVLC decoder.

4.3.3. Efficiency and complexity of CABAC – conclusions

CABAC algorithm provides considerably better coding efficiency than any other entropy coder that has found application in digital video compression. The obtained experimental results prove that CABAC significantly outperforms another advanced UVLC entropy coder that is considered as the state-of-the-art entropy coder among variable-length coders used in video compression (see Section 4.3.1.). The experimental results also prove that the increase of compression performance of CABAC was possible by significant increase

of the complexity of entropy coder (see Section 4.3.2). In author's opinion, improvement of compression performance of the state-of-the-art entropy coders is possible when using even more sophisticated adaptive models of source data. Nevertheless, it will involve further increase of the complexity of the modified entropy coder.

Chapter 5

Advanced adaptation techniques of entropy coders

5.1. The starting point to research

The state-of-the-art hybrid coder for digital video is AVC, the new worldwide video coding standard [AVC]. In the coding efficiency respect, AVC video coder clearly outperforms the older hybrid video coders such as MPEG-1, MPEG-2 and H.263 [Sull05, Wieg03a] as well as recent video coders VC-1 [Lam06] and AVS [Fan04]. The extremely high coding efficiency of AVC has been achieved by a great number of new tools and improvements [Wieg03a] and highly advanced entropy coding.

As revealed earlier, the state-of-the-art entropy coding technique applied in hybrid compression of digital video is Context-based Adaptive Binary Arithmetic Coding (CABAC) that has been used in AVC video coder. Mechanisms of data statistics estimation that are used in CABAC exploit adaptive models of source data and belong to the most advanced and most efficient ones that have been ever applied in hybrid video coders. Therefore, the compression performance of CABAC is much higher as compared to other entropy coding techniques used within the hybrid compression of digital video [Marp03a, Graj05].

The goal of the thesis is to increase compression of advanced adaptive entropy coders used in contemporary video coders by applying even more sophisticated (than currently used) mechanisms of data statistics modeling. For that reason, the state-of-the-art CABAC entropy coder and the state-of-the-art AVC video coder have been chosen as the base to research.

The author's research is up-to-date and very important in context of intensive works towards future generation video codecs. Works towards a new standard H.265 [VCEG07] have been already started. There are proposals of new more advanced and more efficient techniques of video coding, but review of all of them is out of scope of this dissertation. Nevertheless, the current activities relatively weakly concern the adaptive entropy coding.

5.2. Advantages of adaptation technique in CABAC

CABAC is the most powerful entropy coding technique that has ever found common use in digital video compression [Marp03a]. The extremely high compression performance of CABAC is mainly a result of using a complex and sophisticated methods of adaptation to the local statistics of video data. These advanced adaptation methods are:

1) Matching of binarization process to the statistics of non-binary valued syntax elements by applying of five different basic binarization schemes: unary, truncated unary, *k*-th order Exp-Golomb, fixed-length and Huffman-based. The application of adaptive binarization significantly reduces the number of binary symbols that are finally put to arithmetic encoder core;
2) Statistics of individual syntax elements that are coded in video encoder significantly differ between themselves. Therefore, the use of one probability model that is common to all coded syntax elements would be inefficient from the point of view of compression performance. In CABAC, a total number of 399 different probability models have been defined for all coded syntax elements (this is only for the case of transform calculated in 4x4 blocks). A given syntax element uses some subset from set of 399 probability models. This is the first level of adaptation to the current signal statistics. Each of defined probability models independently estimates the statistics of binary symbol or group of binary symbols that is a part of given syntax element;
3) Statistics of a given syntax element is closely related to the local content of the video sequence. Hence, by encoding a given syntax element in CABAC the number of the currently used probability model is dependent on the statistics of coded element in the neighboring blocks (left and upper block relative to the currently encoded block). It constitutes the second level of adaptation to the current signal statistics;
4) The algorithm of the conditional probabilities estimation, that are finally fed to arithmetic codec core, should take into account the real probability distribution of coded data. In

hybrid compression of digital video, data that represents the prediction residuals are mainly encoded by entropy encoder. The character of signal of prediction residuals is similar to geometrical distribution. Therefore, the estimation process of the conditional probabilities used in CABAC assumes the "exponential aging" model of coded data [Howa92] that well correspond to the real statistics of coded data.

5.3. Proposals of improvements of CABAC adaptation – review of references

The techniques of adaptation used in CABAC (presented in Section 5.2) have crucial importance to the compression performance of entropy coder. In order to noticeably increase the coding efficiency of advanced adaptive entropy coders, efforts should be put into further improving of introduced adaptation techniques.

In order to improve compression performance of advanced entropy coders used in digital video compression, improvements of the following adaptation techniques are intensively investigated:

- Even more sophisticated schemes of coding of the quantized transform coefficients and motion vectors;
- Even more complex context pattern for coded video data;
- Applying of even more accurate techniques of the conditional probabilities estimation that are fed to arithmetic codec.

5.3.1. More complex context pattern in CABAC

The statistics of signal that represents an individual image from video sequence is strictly dependent on its content. For images of natural scenes, this content locally changes. Hence, the statistics of data that is coded by entropy coder also locally changes. Therefore, in order to achieve high coding efficiency, it is very important to adapt entropy coding to the local statistics of coded data. The context modeler of CABAC estimates the statistics of currently encoded data based on data that has been already encoded in adjacent neighboring blocks. The data of neighboring blocks form a context in which the data from the current block is encoded. The context pattern determines which neighboring blocks are taken into consideration in forming the context. In general, the statistics of image data locally changes and it is very difficult to state which one of neighboring blocks should be taken into

consideration to estimate the data statistics from the current block. In CABAC, the data statistics of the current block is always estimated with respect to data from the left and the upper block relative to the current block. The context pattern of CABAC algorithm has been shown in Figure 5.1.

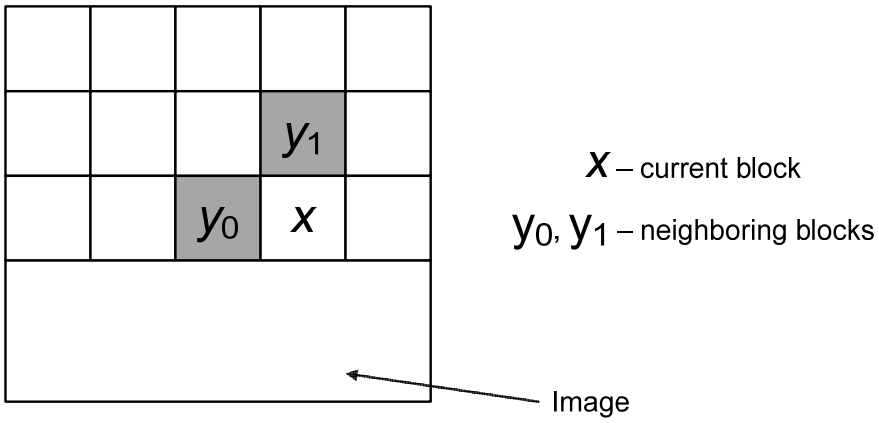


Figure 5.1. The context pattern in CABAC algorithm.

It is obvious that the context pattern defined in CABAC is not always appropriate to track efficiently the statistics of data from the current block. Nevertheless, the context pattern of CABAC is a compromise between coding efficiency and complexity.

In [Mrak03a, Mrak03b, Mrak03c] the authors have proposed more sophisticated technique of context modeling that is used to optimized selection of the context for currently coded symbol. The proposed method is called Growing, Reordering and Selection by Pruning (GRASP). GRASP algorithm takes advantage of the extended context pattern, in which data from more neighboring blocks are taken into consideration in the process of statistics estimation for the currently coded symbol. The context pattern proposed in GRASP algorithm has been presented in Figure 5.2.

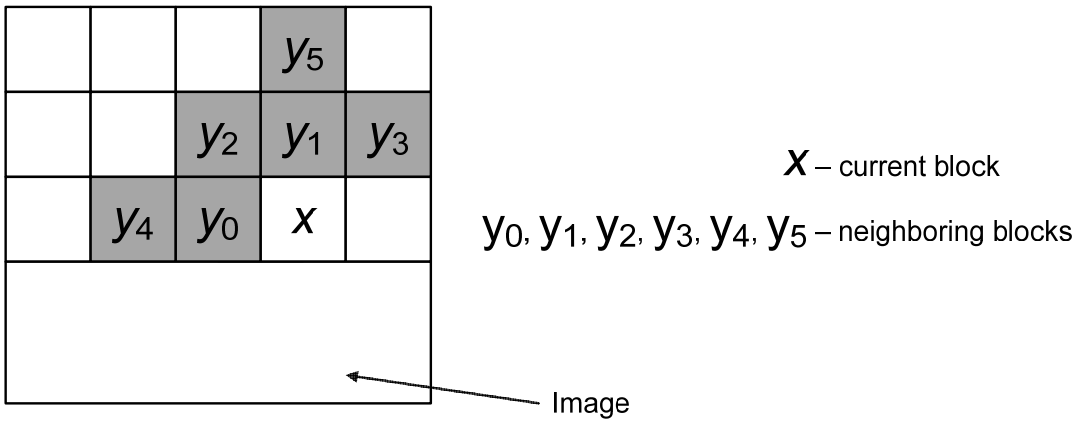


Figure 5.2. The context pattern proposed in GRASP [Mrak03a, Mrak03b, Mrak03c].

In GRASP algorithm, data of neighboring blocks are inserted into the binary context tree [Mrak03b, Mrak03c]. The binary context tree estimates the probability distribution of currently coded symbol by gathering the conditional statistics of data in each node of the context tree. Based on the statistics estimated in the individual nodes of the context tree GRASP algorithm calculates the adaptive code length that would be needed to encode the current symbol in a given context. For each coded symbol, the GRASP algorithm selects the best context from the context tree that allows encoding a new symbol with the smallest number of bits. In this way, the technique of context selection has been optimized in respect to output code length.

The authors have compared coding efficiency of CABAC that exploits GRASP context modeling with efficiency of the original CABAC within AVC for selected syntax elements [Mrak03a, Mrak03b, Mrak03c]. The experimental results achieved by authors showed that the bitrate reduction of up to 3% is possible when using GRASP technique within CABAC [Mrak03c]. However, the higher coding efficiency of the modified CABAC with GRASP algorithm has been achieved by a significantly increase of both encoder and decoder complexity [Mrak03c]. Unfortunately, the authors have not presented any numbers on the complexity of CABAC with GRASP.

5.3.2. Advanced entropy coding of transform coefficients and motion vectors

In hybrid video coding data that represents the transform coefficients and the motion vectors makes fundamental part of data stream that is finally passed to entropy coding. Therefore, the way in which data of transform coefficients and motion vectors is encoded has a great influence on the size of the resulted bitstream.

CABAC algorithm realizes very advanced coding of both transform coefficients and motion vectors within hybrid compression of digital video. In order to better adapt the coding to the current signal statistics, the context modeler of CABAC estimates the probabilities of successive symbols with taking into consideration the statistics of symbols that have been already encoded in neighboring blocks.

In the case of motion vectors, data from neighboring blocks are used to choose the context model for the data of the current block. In the case of the transform coefficients the recently coded non-zero transform coefficient determines the context model for next non zero-valued transform coefficient.

One of the ideas of improving the compression performance of CABAC is to exploit more sophisticated knowledge of history to encode more efficiently of the transform coefficients and the motion vectors. In [Ghan04] the authors have proposed a modified scheme for context modeling and arithmetic coding of the motion vectors data. In particular, the authors' method of the statistics estimation for *y* motion vector component based on the value of *x* motion vector component has been presented in [Ghan04]. The authors have showed that coding efficiency of the motion vectors data in CABAC within AVC can be further enhanced when using the proposed modification. Numbers on total bitrate reduction have not been presented in [Ghan04].

The original idea of increasing of coding efficiency of the transform coefficients in CABAC framework has been presented in [Mila06]. In CABAC, the probability distribution of the currently coded transform coefficient is estimated based on the statistics of only one previously coded transform coefficient. Because of the fact that the DCT-like transform used in AVC is sub-optimal in the sense of de-correlation task, the transform coefficients of both a single block and a macroblock are still partially correlated between each other [Mila06]. In [Mila06] the authors have proposed more advanced technique of the data statistics modeling for transform coefficients within CABAC. The proposed technique exploits some correlation that exists between transform coefficients in a single block and between blocks within a macroblock. The statistical dependences of transform coefficients are determined with the Directed Acyclic Graph (DAG) [Mila06]. In the DAG, statistics of a given transform coefficient is estimated on the basis of statistics of two neighboring coefficients that are located on the left and on the upper relative to a given coefficient in the two dimensional block of transform coefficients. The experimental results showed that the proposed more advanced technique of the transform coefficients statistics estimation leads to bitstream reduction of even 10% in comparison to original CABAC [Mila06]. It must be stated that authors of [Mila06] have proposed new scheme of coding of transform coefficients. In contrast to that, the author of the dissertation is going to improve the algorithm of the conditional probabilities estimation in CABAC and in this way increase coding efficiency of CABAC.

5.3.3. More accurate data modeling techniques

Estimation of the conditional probabilities of coded symbols is one of the most computationally and memory complex part of both entropy encoder and entropy decoder. In

order to accelerate entropy coding in video coders, simplified mechanisms of probabilities estimation are used in practice [MPEG-1, MPEG-2, H263, VC-1, AVS, AVC].

CABAC algorithm exploits the most advanced technique of the data statistics estimation within hybrid video coders. However, in order to keep the computational and memory cost of CABAC in reasonable boundaries the technique of data modeling has been strongly simplified [Marp03a]. First of all, a limited set of only 128 quantized values of probabilities have been defined for coded symbols. Secondly, estimation of the conditional probabilities of symbols has been realized with a finite-state machine (FSM) and only one transition rule between probabilities has been applied for all probability models defined in CABAC. These simplifications lead to significant speeding up of entropy coding but of course it affects negatively the compression performance of CABAC. Therefore, for some time now there has being done research on further improving of coding efficiency of CABAC by applying of more accurate methods of the conditional probabilities estimation.

In [Bely06] the authors have proposed the improvement of algorithm of the conditional probabilities estimation in CABAC by applying of Virtual Sliding Window (VSW) algorithm. The VSW is an adaptive mechanism of probabilities estimation that is based on the idea of Imaginary Sliding Window (ISW) proposed in [Ryab96]. In VSW algorithm, probabilities of successive symbols are estimated with respect to statistics of W previously encoded symbols that are placed in virtual window. The statistics of symbols from virtual window is updated every time after encoding the new symbol. The authors proved, that the application of the VSW technique within CABAC in AVC allows to improve the compression efficiency of original CABAC by about 0.1% - 1.7% for QP parameter ranging from 10 to 40 [Bely06].

Another idea of improving estimation of probabilities in CABAC has been proposed in [Hong04]. The authors have replaced the simplified algorithm of the conditional probabilities estimation with adaptive method of the probabilities estimation based on Context-Tree Weighting (CTW) [Will95, Will98a, Begl04] well known in data compression. The authors have proposed and tested a relatively simple way of applying CTW method into CABAC in AVC; only one independent context tree has been defined for each of 8 syntax elements coded in AVC video coder [Hong04]. In spite of a relatively simple application of CTW method into CABAC the authors have achieved very promising experimental results. They obtained a bitstream reduction of 1% - 3% in comparison to original CABAC within AVC. The dissertation continuous the idea of application of CTW technique within CABAC.

Nevertheless, more sophisticated method of application of CTW within CABAC is considered in the dissertation.

In parallel to research for this thesis, the application of CTW method into CABAC within AVC has been also considered in [Firo06]. However, the authors of reference [Firo06] have done experiments only for two selected syntax elements. Besides, in the opinion of author of the dissertation, the methodology of experiments in [Firo06] is not clear.

In this dissertation, the author has also investigated possibilities of further improving of coding efficiency of CABAC by application of advanced methods of data statistics estimation. The following more exact techniques of data statistics gathering have been taken into consideration:

- More sophisticated author's method of application of CTW technique in CABAC algorithm [Karw06, Karw07a, Karw07b], in contrast to proposal from [Hong04];
- Prediction with Partial Matching (PPM) [Karw07a];
- Author's method of joint application of both CTW and the PPM techniques in CABAC algorithm [Karw07a].

The three more exact techniques of the conditional probabilities estimation have been applied by the author into the state-of-the-art CABAC algorithm [Marp03a] within AVC video codec [AVC]. In this way, three modified AVC video codecs have been obtained. The compression performance of each of the modified AVC video codec has been tested and confronted with the coding efficiency of the original AVC with unmodified CABAC. In order to obtain reliable experimental results, both encoder and decoder have been implemented.

5.4. Universal data modeling techniques

5.4.1. Context-Tree Weighting technique

Context-Tree Weighting (CTW) is a universal method of data statistics estimation and it calculates the conditional probabilities of source symbols [Will95, Will98a, Begl04]. CTW method is well-known and commonly used in data compression and archiving systems [Åberg97]. Recently, CTW technique also started to be used in lossless image compression [Ekstr96, Bonc06, Xiao06] and in digital video compression [Hong04]. The CTW estimates probability of symbol x_n with taking into consideration the symbols that have been coded earlier. These previously coded symbols form the context in which the new symbol x_n has

been observed. In order to store information about symbols statistics for maximum context length D that is generated by n-ary source data, n-ary context tree of depth D is used. So, the depth D determines the number of previously coded symbols that are taken into consideration in estimation the probability for the next symbol. A special case of n-ary context tree is a binary context tree that is used to gather data statistics of binary source data.

The context tree used in CTW method is a collection of nodes connected by branches. The structure of binary context tree of depth $D = 3$ has been shown in Figure 5.3.

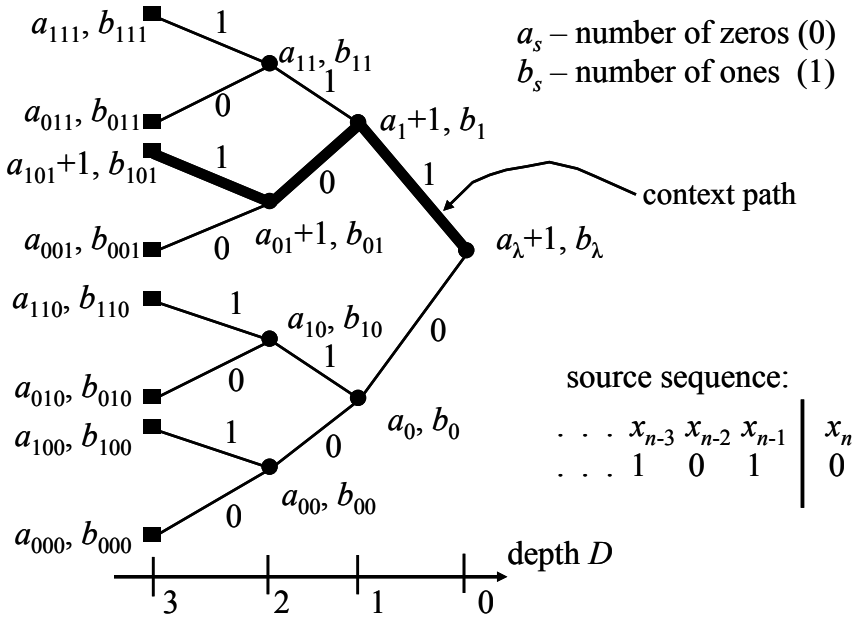


Figure 5.3. Binary tree of contexts. The idea of the drawing taken from [Volf02].

In a given node s of the context tree, information about the number of zeros a_s and the number of ones b_s that follow individual context c_s in the source sequence is kept. The context c_s makes a path on the context tree that is determined by branches of context tree between root λ and node s of the context tree.

At depth 0, …, $D - 1$ each node s has its successor *0s* (associated with context *0*) and successor *1s* (associated with context *1*) as shown in Figure 5.4.

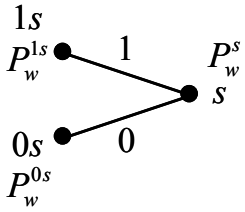


Figure 5.4. Node *s* and associated with it descendant nodes *0s* and *1s*.

By encoding of a new symbol x_n, the information about *D* previous symbols is used. These *D* past symbols determine the context for x_n, this context specifies the *context path* on the context tree (see Figure 5.3.). In order to update the information about symbol statistics for a given context c_s, all counters (a_s and b_s) stored in each node *s* on the *context path* have to be updated (e.g. when x_n = 0 counter a_s is incremented by 1 in each node *s* on the *context path*). In this way CTW method determines adaptively the source symbols statistics.

Based on the values of counters a_s and b_s, CTW method calculates recursively the value of weighted probability $P_w^s(x_1^n)$ (w means weighted probability) of block (sequence) of symbols $x_1^n = x_1, x_2, ..., x_n$ in each node *s* on the *context path* beginning from the leaf (node *s* at the maximum depth *D*) to the root λ of the context tree (see Equation 5.1)

$$P_w^s(x_1^n) = \begin{cases} P_e(a_s, b_s), & \text{for} \quad |c_s| = D \\ \frac{1}{2} P_e(a_s, b_s) + \frac{1}{2} P_w^{0s}(x_1^n) P_w^{1s}(x_1^n), & \text{for} \quad |c_s| < D \end{cases}, \tag{5.1}$$

where

- $|c_s|$ means the length of the context c_s;
- $P_w^s(x_1^n)$ means the weighted probability for sequence of symbols $x_1^n = x_1, x_2, ..., x_n$ that is stored in node *s*;
- $P_w^{0s}(x_1^n)$ and $P_w^{1s}(x_1^n)$ means the weighted probabilities for sequence of symbols $x_1^n = x_1, x_2, ..., x_n$ which are stored in nodes 0*s* and 1*s* respectively;
- $P_e(a_s, b_s)$ means the estimated probability.

The node *s* that is a leaf of the context tree has no successors, so memoryless estimator of the probability is only calculated. In [Krich81] it has been proved that Krichevsky–Trofimov (KT) estimator is a good modeling algorithm for binary memoryless sources. KT estimator

calculates the conditional estimated probability of the next symbol x_n based on the symbols generated by a source so far.

$$P_e^s(x_n = 0 / x_1^{n-1} \text{ contain } a_s \text{ zeros and } b_s \text{ ones}) = \frac{a_s + \frac{1}{2}}{a_s + b_s + 1}, \tag{5.2}$$

$$P_e^s(x_n = 1 / x_1^{n-1} \text{ contain } a_s \text{ zeros and } b_s \text{ ones}) = \frac{b_s + \frac{1}{2}}{a_s + b_s + 1}. \tag{5.3}$$

In Equation 5.2 and Equation 5.3, *e* means estimated probability.

In the case of estimated probability P_e for a sequence of symbols x_1^{n-1}, x_n Equations 5.2 and 5.3 take form of Equation 5.4 and Equation 5.5 respectively.

$$P_e^s(a_s + 1, b_s) = P_e^s(a_s, b_s) \frac{a_s + \frac{1}{2}}{a_s + b_s + 1}, \qquad \text{for sequence } x_1^{n-1}, 0, \tag{5.4}$$

$$P_e^s(a_s, b_s + 1) = P_e^s(a_s, b_s) \frac{b_s + \frac{1}{2}}{a_s + b_s + 1}, \qquad \text{for sequence } x_1^{n-1}, 1. \tag{5.5}$$

In Equation 5.4 and Equation 5.5 $P_e^s(a_s, b_s)$ is the estimated probability of sequence of symbols x_1^{n-1}.

Each node *s* on the *context path* that is not a leaf of the context tree has its successors *0s* and successor *1s*. In these nodes *s* CTW method weights between two alternatives: a model of memoryless source and a model of source with memory. In the case of the model of memoryless source, KT estimator $P_e^s(a_s, b_s)$ is calculated for a sequence of symbols x_1^{n-1}, x_n seen in node *s*. In the case of the model of source with memory, the sequence of symbols x_1^{n-1}, x_n seen in node *s* is a concatenation of two subsequences: one seen in node *0s* and the other seen in node *1s*. The estimated probability of the sequence of symbols x_1^{n-1}, x_n seen in node *s* is a product of the weighted probabilities calculated in nodes *0s* and *1s* and is equal to $P_w^{0s}(x_1^n) \cdot P_w^{1s}(x_1^n)$. For the reason of the fact that CTW method has no a priori knowledge about the real model of source data, two probabilities derived from memoryless and memory model of source data are weighted together with a factor of $\frac{1}{2}$. The final conditional weighted probability $P_w^{\lambda}(x_n / x_1^{n-1})$ estimated in the root λ of the context tree is calculated based on the weighted probabilities of blocks of symbols x_1^n and x_1^{n-1} (see Equation5.6)

$$P_w^{\lambda}(x_n / x_1^{n-1}) = \frac{P_w^{\lambda}(x_1^n)}{P_w^{\lambda}(x_1^{n-1})}. \tag{5.6}$$

This probability is finally used by entropy coder.

(Equations presented in this section have been formulated on the basis of the reference [Volf02]).

5.4.2. Prediction with Partial Matching technique

Prediction with Partial Matching (PPM) is another technique of data statistics modeling used for conditional probability estimation of the successor symbol [Clear84, Begl04]. PPM algorithm has been worked out by Cleary and Witten in 1984 and in conjunction with Huffman or arithmetic coding it is distinguished by high compression performance. Currently, PPM technique is commonly used in text compression systems [Fere03, Shkar02] which are characterized by high coding efficiency.

The main idea of PPM technique is to gather symbols statistics for contexts (past symbols) of different lengths d ($d = D_{PPM}, D_{PPM} - 1, ..., 0, -1$). PPM method assumes a priori the memory source data and tries to use the longest possible context to estimate the conditional probability of the successor symbol. If the new symbol has not occurred yet in this context, the algorithm encodes *ESCAPE* symbol and goes to a shorter context. Then it tries to encode the new symbol in a shorter context. If the new symbol has not occurred in none of the possible contexts, PPM method uses -1 order model in which each of symbols from alphabet A have the same probability equal to $P_e^{-,-1}(x_n) = \frac{1}{|A|}$, where $|A|$ is the size of alphabet A. *ESCAPE* symbol is not taken into consideration in determining the size $|A|$ of alphabet A.

ESCAPE symbol is treated as an additional symbol of the original alphabet A and is used to inform the decoder to use the shorter context in the process of the conditional probability estimation of the successive symbol. In this way, the number of different symbols that are encoded by entropy encoder is increased by 1 in contrast to the size of the original alphabet A. So, in the case of binary source, three different symbols can be encoded.

For that reason, the binary arithmetic codec core can not be directly used with the original PPM method in the case of binary sources. Therefore, slightly different algorithm of PPM technique has been considered.

In each node s on depth d on the context path ($d = D_{PPM}, D_{PPM} - 1, ..., 0, -1$), PPM method estimates the conditional estimated probability $P_e^{s,d}(x_n / x_1^{n-1})$ and the probability of

the *ESCAPE* symbol $P_{esc}^{s,d}(x_1^{n-1})$ (*e* means estimated, *esc* means escape). Based on the data statistics of symbols estimated for different contexts, the conditional estimated probabilities $P_e^{s,d}(x_n/x_1^{n-1})$ calculated for different contexts of different lengths are blended together and the resulted probability $P_{PPM}(x_n/x_1^{n-1})$ is used by entropy codec. The way of blending of the conditional estimated probabilities $P_e^{s,d}(x_n/x_1^{n-1})$ calculated for different context lengths depends on the values of probabilities $P_{esc}^{s,d}(x_1^{n-1})$ of *ESCAPE* symbol and can be determined by Equation 5.7 and Equation 5.8 (these equations are citation from [Volf02])

$$\omega_{s,d^*}(x_1^{n-1}) = (1 - P_{esc}^{s,d^*}(x_1^{n-1})) \prod_{d=d^*+1}^{D_{PPM}} P_{esc}^{s,d}(x_1^{n-1}), \tag{5.7}$$

$$P_{PPM}(x_n/x_1^{n-1}) = \sum_{d=-1}^{D_{PPM}} \omega_{s,d}(x_1^{n-1}) P_e^{s,d}(x_n/x_1^{n-1}), \tag{5.8}$$

where:

- $P_{PPM}(x_n/x_1^{n-1})$ is the resulted conditional probability of symbol x_n estimated with PPM technique;
- $P_e^{s,d}(x_n/x_1^{n-1})$ is the conditional estimated probability calculated in node *s* on the depth *d* of the context path;
- $\omega_{s,d}(x_1^{n-1})$ is the factor that weights the conditional estimated probability $P_e^{s,d}(x_n/x_1^{n-1})$ stored in node *s* on the depth *d*.

Several different variants of the PPM method have been worked out. The most popular are: PPMA [Clear84], PPMB [Clear84], PPMC [Moff90], PPMD [Howa93], PPMZ [Bloom98], PPMII [Shkar02], and PPM* [Clear93]. Some of them have been presented in Table 5.1.

Table 5.1. Popular variants of PPM method.

		Variant of PPM method			
		PPMA	**PPMB**	**PPMC**	**PPMD**
PROBABILITIES	$P_e^{s,d}(x_n/x_1^{n-1})$	$\frac{c_s(\sigma)}{C_s+1}$	$\frac{c_s(\sigma)-1}{C_s}$	$\frac{c_s(\sigma)}{C_s+q_s}$	$\frac{2c_s(\sigma)-1}{2C_s}$
	$P_{esc}^{s,d}(x_1^{n-1})$	$\frac{1}{C_s+1}$	$\frac{q_s}{C_s}$	$\frac{q_s}{C_s+q_s}$	$\frac{q_s}{2C_s}$

The meaning of parameters presented in Table 5.1. is as follows:

C_s - The number of occurrences of context *s* so far in the source sequence. It is equal to the sum of occurrences of all source symbols that appeared in the context *s*.

$c_s(\sigma)$ - The total number of times that symbol $\sigma \in A$ occurred in the context *s* so far in the source sequence.

q_s - The number of different symbols that occurred in the context *s* so far in the source sequence.

There is a relationship between the conditional estimated probabilities $P_e^{s,d}(\sigma/x_1^{n-1})$ (for $\sigma \in A$) and the probability $P_{esc}^{s,d}(x_1^{n-1})$ of *ESCAPE* symbol that is expressed with the following equation:

$$P_{esc}^{s,d}(x_1^{n-1}) + \sum_{\substack{\sigma \\ \sigma \in A}} P_e^{s,d}\left(\sigma/x_1^{n-1}\right) = 1. \tag{5.9}$$

The individual variant of PPM technique differs from the way of calculating of the conditional estimated probability $P_e^{s,d}(x_n/x_1^{n-1})$ and the probability of *ESCAPE* symbol $P_{esc}^{s,d}(x_1^{n-1})$. Nevertheless, the main idea of data statistics estimation is the same for all variants of the PPM. The author has tested the "A" variant of Prediction with Partial Matching and obtained experimental results are not satisfactory. Therefore, other variants of PPM technique have not been considered in the dissertation since results similar to those achieved for PPMA technique were expected.

5.4.3. Joint application of Context-Tree Weighting and Prediction with Partial Matching

Both of the presented techniques of the data statistics modeling (CTW and PPM) can be used for general-purposes. However CTW as well PPM method assumes some features of coded data in order to estimate the conditional probabilities.

PPM technique tries to encode the new symbol in the longest possible context in which the new symbol has appeared earlier in the source sequence so far. Therefore, a model of memory source data is assumed.

CTW technique seems to be more universal than PPM method. Generally speaking, it gathers the symbol statistics in each possible context and weights the resulted conditional probabilities that have been calculated for different contexts. Additionally, in CTW method the conditional probability calculated in a given context is a result of two estimates: in the assumption of memory source model and in the assumption of memoryless source model. Undoubtedly, the disadvantage of such an approach lies in the fact of mixing the conditional probabilities estimated with "good" and "bad" model of the source data.

In hybrid compression of digital video, data that represents the quantized transform coefficients of prediction residual, motion vectors of prediction residual and control data show the feature of non-stationarity. It means that the probability distribution of data changes in time. So, it is extremely difficult to adapt to the changing statistics of coded data. It is also hard to state whether CTW or PPM method will give better estimate for a given symbol.

In order to exploit the features of both techniques, the author has proposed his own method of joint application of CTW and PPM. The idea of joint using of CTW and PPM has been already presented in the literature in [Volf98, Volf02], but it concerned data archiving systems and not video compression. Besides, completely different idea of joint application of CTW and PPM is propose in this dissertation.

The author's method of joint application of CTW and PPM works within CABAC in the following way:

1) By encoding a new symbol, the algorithm estimates two separate conditional probabilities: P_{CTW} calculated with CTW technique and P_{PPM} calculated with PPM technique;
2) After encoding a new symbol in a given context, the algorithm checks which one of the two conditional probabilities (P_{CTW} or P_{PPM}) allow for obtaining the smallest number of bits. This information is stored in the context tree. In order to do that, dual

context tree has been created and ascribed for each of the statistical model defined in CABAC;

3) By encoding a new symbol in a given context, the algorithm estimates the probability γ_{CTW} of the fact that CTW technique will give better estimate in the current context. The probability γ_{PPM} of the fact that PPM technique will give better estimate in the current context is equal to $\gamma_{PPM} = 1 - \gamma_{CTW}$. Both γ_{CTW} and γ_{PPM} probabilities are calculated with CTW technique using the information about signal statistics stored in dual context trees;
4) For a new symbol, the mixed conditional probability is calculated:
$P_{CTW+PPM} = \gamma_{CTW} \cdot P_{CTW} + \gamma_{PPM} \cdot P_{PPM}$;
5) Finally, for CTW and CTW+PPM method the algorithm accumulates the codeword lengths achieved in a given context so far and better solution is finally chosen
$P_{FINAL} = P_{CTW}$ or $P_{FINAL} = P_{CTW+PPM}$.

The P_{FINAL} is the conditional probability of source symbol that is used by arithmetic codec.

5.5. Conclusions

The application of the more sophisticated techniques of data statistics estimation in CABAC leads to a reasonable increase of his coding efficiency [Mrak03a, Mrak03b, Mrak03c, Hong04, Bely06]. This dissertation is a continuation of the research on improvement of coding efficiency of CABAC. The author is going to realize this goal by application in CABAC of advanced techniques of data statistics estimation based on CTW and/or PPM.

The coding efficiency of CABAC with CTW has been already investigated in the literature [Hong04]. Nevertheless, a relatively simple mechanism of embedding CTW into CABAC has been used in [Hong04]. But, the authors have obtained promising experimental results. This dissertation proposes a novel more sophisticated method of incorporating the CTW into CABAC and it is the topic of the successive chapters.

In the author's knowledge, both PPM technique and technique of joint application of CTW and PPM have not been considered in the context of CABAC within AVC. Therefore, this topic is worth investigating which is done in the next chapter.

Chapter 6

Improvement of entropy coding in AVC video codec

6.1. Main idea

The goal of the dissertation is to improve compression performance of advanced adaptive arithmetic coders by using more sophisticated techniques of conditional probabilities estimation. For the reasons clearly presented in Chapter 5 the algorithm of Context-based Adaptive Binary Arithmetic Coding (CABAC) that works within AVC video coder has been considered. A relatively simple technique of the data statistics estimation from CABAC [Marp03a] has been replaced with more accurate techniques of the data statistics gathering. These more exact techniques are:

- Context-Tree Weighting (CTW);
- The "A" variant of Prediction with Partial Matching (PPMA);
- Author's method of joint application of CTW and PPMA.

The author has proposed a novel method of incorporating of more sophisticated data modeling techniques into CABAC within AVC. Compression performance of the modified CABAC coder (with CTW and/or PPMA) has been thoroughly tested and compared with the coding efficiency of original CABAC coder. Both modified and original CABAC coders have been tested within the framework of AVC video coder.

6.2. General structure of the new entropy codec

The simplified method of the data statistics estimation that is based on a finite-state machine (FSM) in CABAC [Marp03a] has been replaced with more sophisticated techniques of the conditional probabilities estimation based on CTW and/or PPMA. In this way, three new (modified) entropy codecs based on CABAC have been built:

- CABAC codec with CTW technique;
- CABAC codec with PPMA technique;
- CABAC codec with joint application of CTW and PPMA.

The general structure of the new entropy codec has been presented in Figure 6.1.

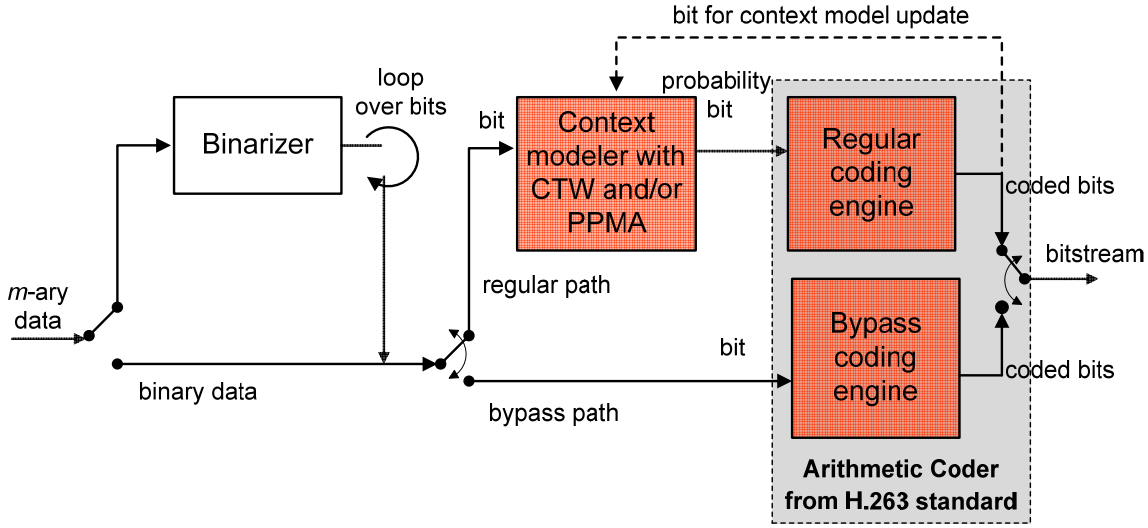


Figure 6.1. The block diagram of the new entropy codec.

In more detail, more exact techniques of the conditional probabilities estimation have been adopted for CABAC codec in the following way:

1) Both encoder and decoder have been implemented in order to obtain reliable experimental results;
2) Binarization schemes, definition of probability models and the method of selection of the proper probability model have been left unchanged with respect to CABAC;
3) The simple technique of the data statistics estimation from CABAC has been replaced with more sophisticated ones based on CTW and/or PPMA;
4) For the reason that the M-codec core (the core of binary arithmetic codec) from CABAC has been adopted to operate properly with a limited set of only 128 predefined quantized values of probabilities [Marp03a, Marp03b], it has been replaced with a traditional multiplication- and division-based implementation of an *m*-ary

arithmetic codec core defined in H.263 video coding standard [H263]. The *m*-ary arithmetic codec core from H.263 standard has been working as a binary arithmetic codec within the modified AVC video codecs. In this way, the modified and the original AVC video codecs have been working with different arithmetic codec cores. It obviously could influence on experimental results on coding efficiency of the modified AVC video codec relative to the original AVC. Therefore, compression performance of H.263 arithmetic codec core has been tested and confronted with coding efficiency of M-codec core from CABAC algorithm with several test sequences within AVC video codec (see Chapter 7);

5) A simplified mode of arithmetic coding (the so-called bypass mode) has been left unchanged (like in CABAC algorithm);
6) The data statistics gathered with CTW and/or PPMA are initialized to zeros each time before an I-slice and a slice containing one or more consecutive pictures of the same type. In this way, the author's idea of extended slice has been considered for P- and B-slices. Nevertheless, the author has experimentally investigated how it influences the coding efficiency of the modified and the original entropy coders.

Each of the three modified entropy codec is not a standard CABAC codec. Nevertheless, each of the modified entropy coder produces bitstream of the same syntax as the original CABAC. Therefore, the author's proposals can be used as an extension in AVC or can be applied in a new video coding standard.

The three modified CABAC entropy codecs have been applied to AVC video codecs which results in three modified AVC video codecs with CABAC and more accurate techniques of conditional probabilities estimation. The three modified AVC codecs have been implemented as modifications of reference software JM 10.2 [AVCSoft] of AVC. Detailed information on implementation of the modified AVC video codecs is presented in the next sections.

6.3. Modified AVC Video Codec with CTW technique

6.3.1. Implementation of CTW technique

The original scheme of CTW technique (as described in Section 5.4.1) estimates probabilities of block of symbols that occurred in different contexts. These probabilities are

finally appropriately weighted to form the resulted conditional probability of the new symbol. In order to prevent from the overflow during computations of probabilities for block of symbols, extremely high precision of calculations must be ensured, which in consequence requires using the floating point representation for probabilities. The direct application of the original scheme of CTW technique is also very inefficient from the point of view of memory usage. This is the main disadvantage of CTW technique. In order to track the statistics of coded data, besides the number of zeros a_s and the number of ones b_s the original CTW algorithm must store the floating-point representations of the estimated probability P_e and the weighted probability P_w in each node of the context tree. The number of nodes of the context tree grows exponentially as the context tree depth D increases. Therefore, the number of memory bytes needed to store a single node s is an important factor that influences the storage complexity of CTW algorithm.

6.3.1.1. The optimized scheme of CTW technique

In order to decrease the high demand for memory by CTW technique, the optimized (in contrast to the original scheme) scheme of the conditional weighted probabilities estimation is used in practice [Will97a, Will97b, Will98b, Volf99, Volf02, Will06]. The optimized scheme of the conditional weighted probabilities estimation in CTW has been proposed by Willems and Tjalkens [Will98b] in which, instead of estimated and weighted probabilities for block of symbols, the conditional estimated and conditional weighted probabilities are calculated in each node s on the context path. In order to explain the working of the optimized CTW technique, the equations presented in [Volf02, Will06] will be citied here.

Consider a successive source symbol $x_n = 1$. Assuming that the node $0s$ is this child of node s that is located on the context path (thus the node $1s$ is the child of node s that is not located on the context path) the conditional weighted probability for a symbol x_n can be calculated with the following equation:

$$P_w^s\left(x_n = 1 / x_1^{n-1}\right) = \frac{P_e^s\left(x_1^{n-1}, x_n = 1\right) + P_w^{0s}\left(x_1^{n-1}, x_n = 1\right) \cdot P_w^{1s}\left(x_1^{n-1}, x_n = 1\right)}{P_e^s\left(x_1^{n-1}\right) + P_w^{0s}\left(x_1^{n-1}\right) \cdot P_w^{1s}\left(x_1^{n-1}\right)}. \tag{6.1}$$

The above equation results from Equation 5.1 and the relationship $P\left(x_n / x_1^{n-1}\right) = \frac{P\left(x_1^{n-1}, x_n\right)}{P\left(x_1^{n-1}\right)}$.

Because of the fact that the child node $1s$ does not belong to the context path in this consideration, the weighted probability of block of symbols $x_1^{n-1}, x_n = 1$ in node $1s$ is equal to $P_w^{1s}\left(x_1^{n-1}, x_n = 1\right) = P_w^{1s}\left(x_1^{n-1}\right)$, so the above equation can be rewritten to:

$$P_w^s\left(x_n = 1/x_1^{n-1}\right) = \frac{P_e^s\left(x_1^{n-1}, x_n = 1\right) + P_w^{0s}\left(x_1^{n-1}, x_n = 1\right) \cdot P_w^{1s}\left(x_1^{n-1}\right)}{P_e^s\left(x_1^{n-1}\right) + P_w^{0s}\left(x_1^{n-1}\right) \cdot P_w^{1s}\left(x_1^{n-1}\right)}. \tag{6.2}$$

Putting $\frac{P_e^s\left(x_1^{n-1}\right)}{P_w^{0s}\left(x_1^{n-1}\right) \cdot P_w^{1s}\left(x_1^{n-1}\right)} = \beta^s\left(x_1^{n-1}\right)$ the Equation 6.2 takes a form:

$$P_w^s\left(x_n = 1/x_1^{n-1}\right) = \frac{\beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(x_n = 1/x_1^{n-1}\right) + P_w^{0s}\left(x_n = 1/x_1^{n-1}\right)}{\beta^s\left(x_1^{n-1}\right) + 1}. \tag{6.3}$$

In this work, the optimized scheme of CTW technique as presented in [Volf02] has been used. In order to estimate the conditional weighted probability $P_w^s\left(x_n/x_1^{n-1}\right)$ for the successive source symbol x_n, the optimized scheme of CTW technique realizes the following steps of calculations in every node *s* that belongs to the context path:

1. Calculating of $\beta_n^s\left(x_1^{n-1}\right)$ ratio based on the value of $\beta^s\left(x_1^{n-1}\right)$ ratio

$$\beta_n^s\left(x_1^{n-1}\right) = \frac{\beta^s\left(x_1^{n-1}\right)}{a_s + b_s + \frac{2}{\alpha}}. \tag{6.4}$$

2. Calculating of the weighted conditional estimated probabilities $\beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(0/x_1^{n-1}\right)$ and $\beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(1/x_1^{n-1}\right)$ for the successive source symbol equal to 0 and equal to 1 respectively

$$\beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(0/x_1^{n-1}\right) = \beta_n^s\left(x_1^{n-1}\right) \cdot \left(a_s + \frac{1}{\alpha}\right), \tag{6.5}$$

$$\beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(1/x_1^{n-1}\right) = \beta_n^s\left(x_1^{n-1}\right) \cdot \left(b_s + \frac{1}{\alpha}\right). \tag{6.6}$$

3. Calculating of the $\eta^s\left(x_1^{n-1}, x_n\right)$ factor, based on the results from previous step and the conditional weighted probabilities $P_w^{t's}\left(x_n/x_1^{n-1}\right)$ estimated in the child node $t's$ of node *s* on the context path

$$\eta^s\left(x_1^{n-1}, 0\right) = \beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(0/x_1^{n-1}\right) + P_w^{t's}\left(0/x_1^{n-1}\right), \tag{6.7}$$

$$\eta^s\left(x_1^{n-1}, 1\right) = \beta^s\left(x_1^{n-1}\right) \cdot P_e^s\left(1/x_1^{n-1}\right) + P_w^{t's}\left(1/x_1^{n-1}\right). \tag{6.8}$$

4. Calculating of the conditional weighted probability $P_w^s\left(x_n/x_1^{n-1}\right)$ for the successive source symbol x_n equal to 0 and equal to 1 based on the results from the previous step

$$P_w^s\left(0/x_1^{n-1}\right)=\frac{\eta^s\left(x_1^{n-1},0\right)}{\eta^s\left(x_1^{n-1},0\right)+\eta^s\left(x_1^{n-1},1\right)}, \tag{6.9}$$

$$P_w^s\left(1/x_1^{n-1}\right)=\frac{\eta^s\left(x_1^{n-1},1\right)}{\eta^s\left(x_1^{n-1},0\right)+\eta^s\left(x_1^{n-1},1\right)}. \tag{6.10}$$

5. Updating of the β^s ratio with the conditional estimated probability $P_e^s\left(x_n/x_1^{n-1}\right)$ of the successive source symbol x_n and its conditional weighted probability $P_w^{t's}\left(x_n/x_1^{n-1}\right)$ estimated in the child node $t's$ of node s (on the context path)

$$\beta^s\left(x_1^n\right)=\frac{\beta^s\left(x_1^{n-1}\right)\cdot P_e^s\left(x_n/x_1^{n-1}\right)}{P_w^{t's}\left(x_n/x_1^{n-1}\right)}. \tag{6.11}$$

6. Incrementing of the a_s or b_s counter depending on the value of the successive symbol x_n (if $x_n=0$ the a_s counter is incremented, when $x_n=1$ the b_s counter is incremented). In the author's implementation of CTW technique, both counters (a_s and b_s) are halved and rounded up each time when one of the counters reaches the assumed maximum value of 96. Experimental results proved that the assumed maximum value of the number of zeros a_s and the number of ones b_s is sufficient in the case of sources with binary alphabet. The use of higher threshold does not lead to improve compression ratio.

The main idea of the optimized scheme of CTW technique has been additionally presented in Figure 6.2.

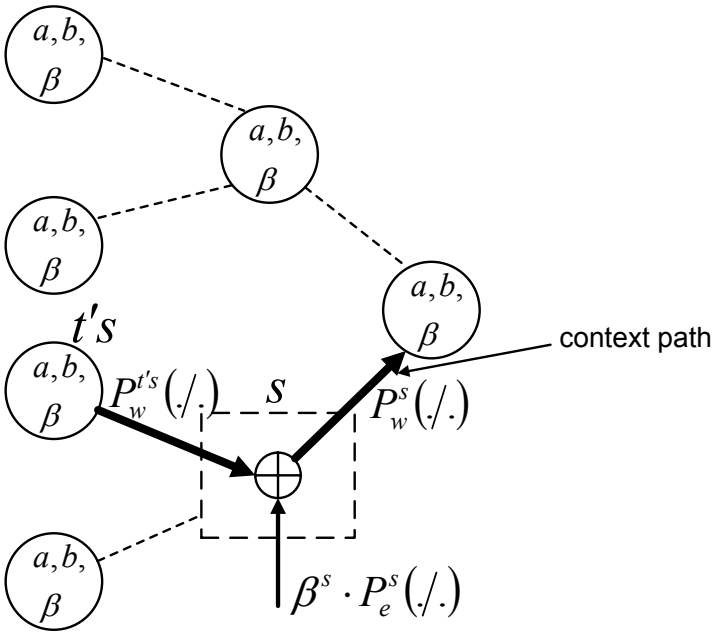


Figure 6.2. The optimized scheme of CTW technique.

In each node *s* on the context path, there is a balance switch β^s that weights two conditional probabilities: $P_e^s(./.)$ that has been estimated in the assumption of memoryless source and $P_w^{t's}(./.)$ that has been estimated in the assumption of a source with memory. The values of these two probabilities modify the state of the balance switch β^s.

In contrast to the original scheme of CTW algorithm, the optimized CTW technique must store only the β^s ratio as well as a_s and b_s counters within a single node *s* of the context tree. Therefore, the optimized CTW technique is characterized by significantly lower storage complexity in comparison to the original scheme of CTW technique. It is of great importance in the case of applying CTW technique for sources with long term memory where context trees of greater depth *D* are needed. Additionally, a given node *s* exploits only the weighted probability estimated in node $t's$ that belongs to the context path and not from the other child node of node *s*. In this way, the number of calls to memory is reduced.

6.3.1.2. Representation of probabilities

Unquestionably, the main drawback of the optimized as well as the original scheme of CTW technique is the necessity to use the floating-point representation during computations. It is connected with serious limitations of the compression system. Firstly, the floating-point operations can be realized in slightly different way in different platforms, which can lead to different calculation results in these platforms. In this way, the necessity for exactly the same operation results in both CTW encoder and CTW decoder can not always be fulfilled. Secondly, the floating-point operations are usually much more time-consuming in contrast to equivalent integer operations, especially in the case of multiplication and division operations. It decreases significantly the throughput of compression system that is based on CTW.

It is possible to limit the above mentioned drawbacks of the floating point representation when doing all computations in the logarithmic domain. Any floating point number *n* with a fixed number of bits *p* after the decimal point can be represented in the logarithmic domain by an integer value $\lfloor 2^p \log_2 n \rfloor$ (taking advantage of conclusions presented in [Volf02] $p = 8$ bits were used to represent the fractional part of logarithm in this dissertation). Addition, multiplication and division operations are used in Equations from 6.4 to 6.11. Relationships between these arithmetic operations in the linear and the logarithmic domain are as follows (equations are citation from [Volf02]):

- Time-consuming multiplication and division operations are replaced with computationally simple addition and subtraction operations respectively, hence:

$$2^p \log_2(a \cdot b) = 2^p \log_2 a + 2^p \log_2 b, \tag{6.12}$$

$$2^p \log_2\left(\frac{a}{b}\right) = 2^p \log_2 a - 2^p \log_2 b, \tag{6.13}$$

- Addition of two numbers a and b, with $a \geq b$ is slightly more complex in the logarithmic domain and can be realized with the following equation:

$$\begin{aligned} 2^p \log_2(a+b) &= 2^p \log_2 a + 2^p \log_2\left(1+\frac{b}{a}\right) = \\ &= 2^p \log_2 a + 2^p \log_2\left(1 + 2^{\left(2^p \log_2 b - 2^p \log_2 a\right)/2^p}\right) \end{aligned} \tag{6.14}$$

From the complexity point of view, the main problem is to convert a given floating point number to its logarithmic representation. But it can be efficiently realized with a look-up table that contains logarithmic representations of all possible floating point numbers that can occur in Equations from 6.4 to 6.11. In order to do that, the following pre-computed values must be stored in the memory: $\left\lfloor 2^p \log_2\left(i+\frac{1}{\alpha}\right)\right\rfloor$, $\left\lfloor 2^p \log_2\left(i+\frac{2}{\alpha}\right)\right\rfloor$, and $\left\lfloor 2^p \log_2\left(1+2^{i/2^p}\right)\right\rfloor$ for all possible values of integer i. In this way, the computationally complex logarithm operation is reduced to a single reference to memory. It significantly speeds up the working of CTW algorithm.

6.3.2. Embedding CTW technique into CABAC algorithm

CTW technique has been originally embedded in the structure of CABAC entropy codec as it has been presented in detail in Section 6.2. In order to realize the author's method of application of CTW in CABAC, a separate context tree of depth D has been ascribed to each of 399 probability models defined for the 4x4 transform (see Figure 6.3).

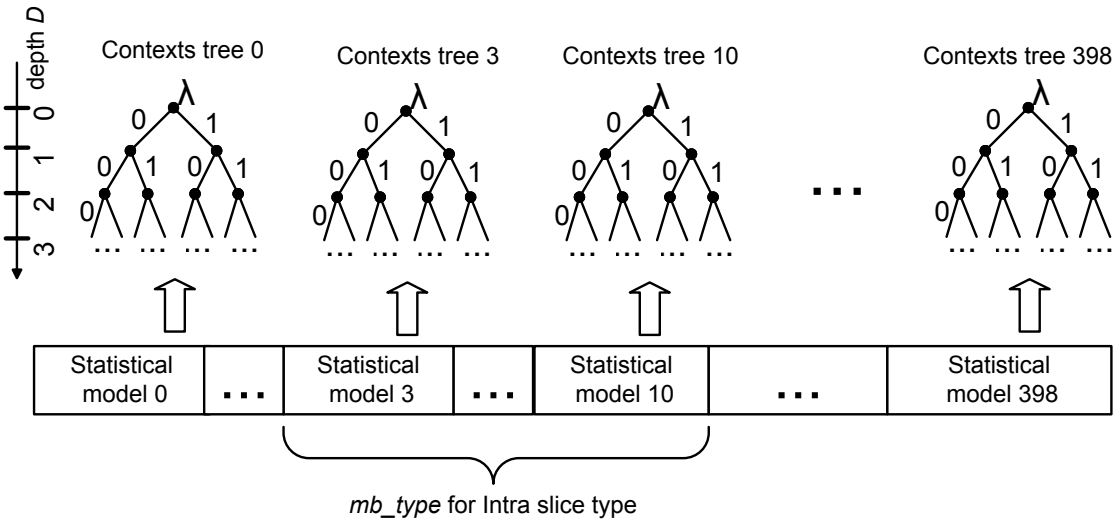


Figure 6.3. Context trees in CABAC.

In author's implementation, context trees are re-initialized each time before an I-slice and a slice of a new type. Two schemes of context trees initialization have been used in tests. In the first one (the so-called simple context initialization) the counters of the number of zeros a_s and the number of ones b_s are initialized to 0 and the parameter β^s is initialized to 1. In the second scheme, CABAC context initialization has been used. Furthermore, the context (D previously coded symbols) consists of all binary symbols related to a given syntax element, and not only the same probability model. Such an approach allows for removing the statistical redundancy of all symbols that represent the same syntax element.

6.4. Modified AVC Video Codec with PPMA

The "A" variant of Prediction with Partial Matching technique (PPMA) has been implemented within AVC reference software according to equations presented in Section 5.4.2. The most computationally complex operations that are performed to estimate the conditional probability of successive symbol x_n in PPMA are dividing and multiplying. Therefore, in order to optimize for speed the process of the conditional probabilities estimation in PPMA all calculations have been performed in the logarithmic domain, similarly as in the case of CTW technique. So, the same look-up tables as defined for CTW technique have been used in PPMA technique implementation.

The implementation of PPMA technique uses counters of the number of zeros a_s and the number of ones b_s stored in the context trees of CTW technique. Based on the values of

counters a_s and b_s in nodes s on the context path the escape probability $P_{esc}^{s,d}(x_1^{n-1})$, the conditional estimated probability $P_e^{s,d}(x_n/x_1^{n-1})$, and the probability $\omega_{s,d^*}(x_1^{n-1})$ are calculated in each node s on the context path. It has been presented in Figure 6.4.

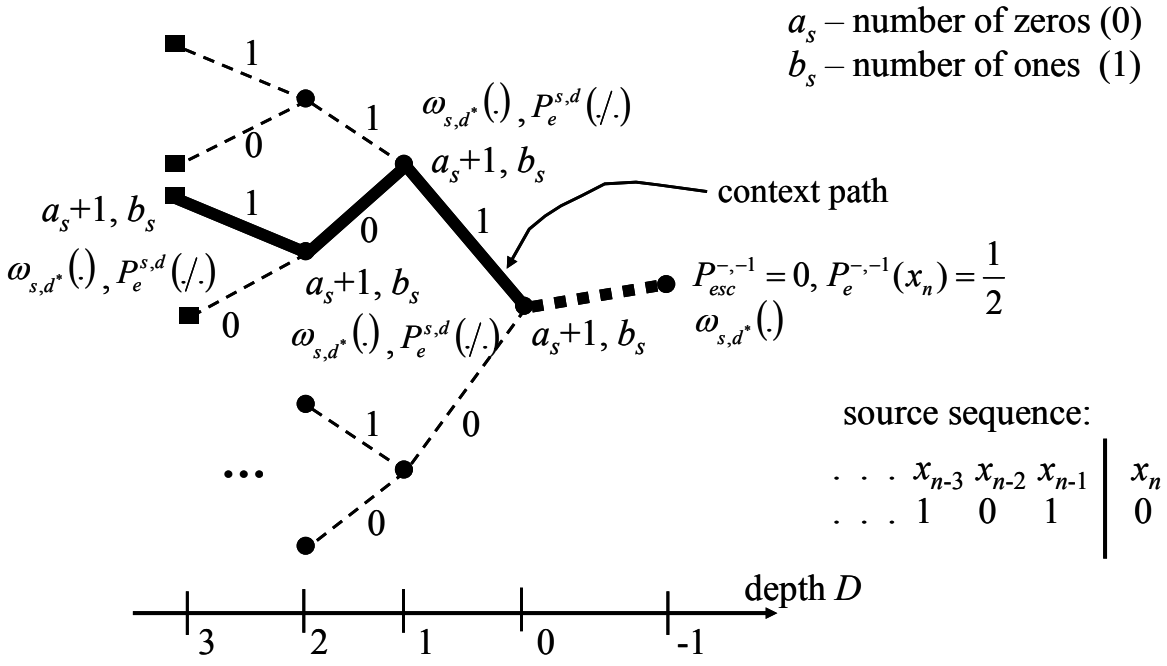


Figure 6.4. Estimating conditional probability with PPMA technique.

Additionally, the probability $\omega_{s,d^*}(x_1^{n-1})$ is calculated at depth $D=-1$ and finally all estimated probabilities are mixing to form the conditional probability $P_{PPM}(x_n/x_1^{n-1})$.

6.5. Modified AVC Video Codec with joint application of CTW and PPMA

The new method of joint application of CTW and PPMA has been implemented and embedded into CABAC entropy codec within the reference software of AVC video codec [AVCSoft]. In more detail, the technique of joint application of CTW and the PPMA has been implemented within CABAC in the following way:

- The logarithmic-domain based implementation of CTW technique has been used as described in Section 6.3.1;
- The logarithmic-domain based implementation of the "A" variant of PPM technique has been used as presented in Section 6.4;

- These two probabilities are weighted together with taking into consideration the performance of both CTW and PPMA technique in a given context for source symbols that have been already encoded. The information on performance of both CTW and PPMA technique is stored in dual context trees. In order to realize that, a set of dual context trees has been defined, and the number of dual context trees is equal to the number of different probability models in CABAC. In the weighting procedure, the weighted coefficients for the two probabilities estimated with CTW and PPMA are calculated with CTW technique that works with dual context trees. In this step also the logarithmic-domain based implementation of CTW technique has been used.

The experimental results on compression performance of the modified AVC with CABAC and the PPMA (see Section 6.7.2) proved that the efficiency of PPMA method within CABAC is strongly dependent on the depth D of context trees. The relationship between depth D of context trees and compression performance of the modified AVC with CABAC and PPMA is different for different values of QP parameter (the proper experimental results have been presented in Section 6.7.2). Taking it into consideration, the following depths D of context trees have been assumed for PPMA technique within the modified AVC with CABAC and joint application of CTW and PPMA:

- If the QP parameter is less or equal to 27 then depth $D = 3$, if the QP parameter is greater than 27 and less than 32 then $D = 2$, if the QP parameter is greater or equal to 32 then the depth $D = 1$. This configuration of depth D in relation to value of the QP parameter has been used for I-frames only;
- If the QP parameter is less or equal to 26 then the depth $D = 4$, if the QP parameter is greater than 26 and less than 34 then $D = 3$, if the QP parameter is greater or equal to 34 then the depth $D = 2$. This configuration of depth D in relative to value of the QP parameter has been used for P- and B-frames.

In this way, depending on the value of QP parameter, the depth D of context tree has been modified for PPMA technique in order to obtain better coding efficiency.

6.6. Methodology of experiments

Compression performance of each of the modified AVC video codec has been tested with several test sequences and confronted against coding efficiency of the original AVC

video codec with unmodified CABAC. Coding efficiency of the modified AVC encoders has been expressed as a percentage reduction of bitrate relative to the size of bitstream obtained for the same test sequence encoded with the original AVC with CABAC in the same configuration of the encoder. The bitrate reduction has been calculated with the following formula:

$$\text{bitrate reduction}[\%] = \left(1 - \frac{\text{bitstream size}(\text{modified H.264/AVC})}{\text{bitstream size}(\text{original H.264/AVC})}\right) \cdot 100\%, \qquad (6.15)$$

where:

$\text{bitstream size}(\text{modified H.264/AVC})$ - Size of bitstream of encoded video sequence obtained for the modified AVC encoder with CABAC that exploits CTW and/or PPMA.

$\text{bitstream size}(\text{original H.264/AVC})$ - Size of bitstream of encoded video sequence obtained for the original AVC encoder with unmodified CABAC.

The well-known and commonly used in digital video compression CITY, CREW, ICE and HARBOUR progressive test video sequences have been used to test the compression performance of the modified and the original AVC video encoders. The test sequences used in experiments have been presented in Annex F. Parameters of the test sequences were as follows:

- 704x576 spatial resolution (4CIF format);
- 352x288 spatial resolution (CIF format);
- 60 frames per second;
- Each of the progressive CITY, CREW and HARBOUR test sequences consisted of 600 frames and ICE sequence consisted of 480 frames.

In course of experiments, large set experimental results have been produced. For the concise of the text, these results have been gathered in Annexes A-E. In the main text, mostly the averaged results have been reported only. These averaged results are the overall indication of the tendencies and are calculated as averages in set of test sequences with the following formula:

$$\text{bitrate reduction}[\%] = \left(1 - \frac{\sum_i \text{bitstream size}(\text{modified H.264/AVC})}{\sum_i \text{bitstream size}(\text{original H.264/AVC})}\right) \cdot 100\%, \qquad (6.16)$$

where *i* represents the number of test sequence.

The test sequences in CIF format (352x288) have been created by downsampling the basis 4CIF test sequences (704x576). It has been done with the *DownConvertStatic* function from Joint Scalable Video Model (JSVM) reference software version 9.1 [JSVM07]. The *DownConvertStatic* function realizes downsamling of the input video sequence by filtering of each frame in both horizontal and vertical directions. The frames are filtered with MPEG-4 downsampling filter with coefficients {2, 0, -4, -3, 5, 19, 26, 19, 5, -3, -4, 0, 2}/64. The resulted frames (in CIF format) are finally created by taking every second sample (in both horizontal and vertical directions) of filtered 4CIF frames.

Tests on compression performance of the modified AVC video encoders as well as the original AVC video encoder have been done in the following configurations of the encoders:

a) Experiments have been done with I-, P-, and B-frame types. Two structures of GOP have been considered: the I29P structure with I-frame inserted every 30-th frame and the IBBPBBP… structure with I-frame inserted every 30-th frame. In this way two GOP structures have been presented in every one second of test video sequence.
b) Two reference frames have been used for motion estimation and motion compensation in both the modified and the original AVC video encoders.
c) In the original AVC video encoder CABAC entropy encoder has been working in its most efficient coding mode by setting to use the adaptive technique of contexts initialization at the beginning of each new slice. In the adaptive technique of contexts initialization, the way of contexts initialization for inter-predicted frames (P-frame and B-frame) is adaptive and is dependent on the data statistics of previously coded frame of the same type (P- or B-frame). Based on the data statistics of previously coded frame CABAC algorithm chooses one out of three sets of contexts parameters that allow for initializing contexts in the way that best corresponds to real statistics of previously coded frame. Relative to non-adaptive fixed method of contexts initialization (in which only one set of context parameters is used for contexts initialization with regardless of data statistics in the previously coded frame), the adaptive technique of contexts initialization in CABAC leads to increase the compression performance of CABAC. In some experiments this arrangement has been also used for the modified CABAC with CTW (see Section 6.8). For other experiments, the modified CABAC has been used the simpler technique of

context initialization and parameters of context trees have been initialized to 0 each time before an I-slice and a slice of a new type.

d) In the experiments, in both the modified and the original AVC encoders the rate-distortion optimization has been left switched off. Experiments have been done for a wide range of the QP parameter values (from QP=8 to QP=44 with step 3) from excellent subjective quality of decoded video sequence (QP=8) to very poor subjective quality of decoded video sequence (QP=44), where the QP parameter is the quantization index that is an encoding parameter of AVC. The bitrate control has been switched off in both the modified and the original encoders. In this configuration, the encoding paths for the modified AVC encoders and the original AVC encoder were really the same. Therefore, for a given QP parameter value a pair of video sequences decoded with the modified and the original AVC decoders were exactly the same with identical PSNR measures and only the sizes of encoded bitstreams were different. In this way, the compression performance of the modified and the original AVC video encoders could be directly compared. (Peak Signal to Noise Ratio (PSNR) is the measure of quality of reconstructed video signal that is commonly used in video compression [Doma98, Richa03]).

e) In all experiments, only the 4x4 integer transform has been used. Experiments have not been made with the 8x8 integer transform.

The compression performance of the modified AVC video encoders has been tested for different context lengths used to estimate the conditional probability of a new symbol. In order to do that, different depths D of context trees have been considered in the modified AVC video encoders. The goal of these experiments was estimation of the optimum depth D of context trees from the point of view of coding efficiency and complexity of the modified AVC encoders.

In all experiments, for both original and modified AVC codecs, the slices were not shorter than a picture. In many cases mentioned in the text, the codecs have been tested with long slices containing consecutive pictures of the same type. For example, for the IBBPBBP… sequences, the individual slice were single I and P pictures, or pairs of B pictures. Such a long slices are not allowed in standard AVC but the author has introduced it in some experiments with original CABAC and modified CABAC. The type of used slice has been marked in the description of the respective experiments.

The coding efficiency of the modified AVC encoders has been confronted with the coding efficiency of the original AVC with CABAC in the following scenarios of experiments (in each scenario experiments have been done with 4 mentioned test video sequences):

Scenario 1:

- Video sequences in 4CIF format have been used;
- The I29P structure of GOP has been assumed;
- Experiments have been done for a wide range of QP parameter values, from QP=8 to QP=44 with step 3. For a given QP parameter value, experiments have been done by fully encoding and decoding of each test sequence;
- In both the modified and the original AVC video encoders rate-distortion optimization has been switched off. The bitrate control has been also switched off;
- Different depths *D* of context trees have been considered in the modified AVC video codec.

Scenario 2:

- Conditions of experiments were the same as in **Scenario 1** with one exception that test video sequences in CIF format were used.

Scenario 3:

- Conditions of experiments were the same as in **Scenario 1** with one exception: the IBBPBBP… structure of GOP has been considered.

The goal of experiments was to show how the compression performance of CABAC may be improved after application of accurate data statistics estimation techniques. The gain of compression performance of the modified CABAC encoders relative to the original CABAC has been calculated with Equation 6.15 and Equation 6.16 and presented in the function of QP parameter value. The QP parameter is the index to the proper quantizer step size Qstep that is used in the original AVC encoder. The bitrates for I, P and B pictures obtained with the original and the modified AVC encoders (for different values of QP parameter) have been also presented in tables. The bitrate for a given type of picture determines the number of bits of all pictures of a given type that present in one second of the sequence. Thus, the overall bitrate is the sum of bitrates for I, P and B pictures.

The averaged experimental results presented in this dissertation give the indication of coding efficiency improvement for the modified AVC encoders relative to the original AVC with CABAC. Experiments have been done for a wide range of bitrates and a large number of

pictures. The goal of the dissertation was not to show the concrete values of compression gain and the confidence intervals were not calculated. Besides, the way of presentation of experimental results used in the dissertation is commonly used in works in the field of video compression.

6.7. Compression performance of the modified AVC video encoders

In this section the author's experimental results on the coding efficiency of the three modified AVC video encoders relative to the efficiency of the original AVC encoder have been presented. The three modified AVC video encoders are:

- AVC video encoder with CABAC that exploits more exact technique of the data statistics estimation based on Context-Tree Weighting (CTW);
- AVC encoder with CABAC that uses conditional probabilities estimation technique based on "A" variant of Prediction with Partial Matching (PPMA);
- AVC video encoder with CABAC in which the simpler method of the data statistics gathering has been replaced with more accurate technique based on joint application of CTW and PPMA.

6.7.1. Compression performance of the modified AVC with CABAC and CTW in contrast to the original AVC with CABAC

In order to unambiguously answer the question how the application of CTW technique influences the compression performance of CABAC within AVC, series of experiments have been done. The compression performance of the modified AVC with CABAC and CTW has been compared to the coding efficiency of the original AVC with CABAC. Experiments have been done in three scenarios presented in Section 6.6.

When using CTW technique, the accuracy of the conditional probabilities estimation for coded symbols is strictly dependent on the number of previously coded symbols that are taken into consideration in the process of data statistics estimation. For that reason, compression performance of the modified AVC video encoder with CABAC and CTW technique has been investigated for different context lengths by defining of context trees of different depths D.

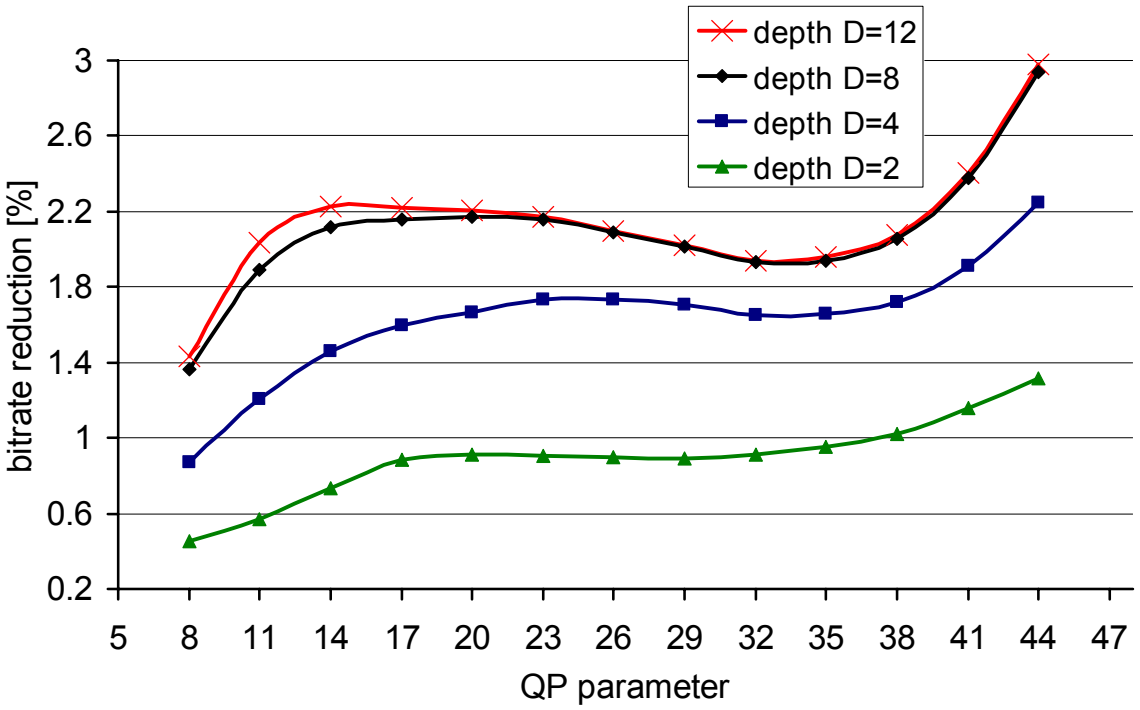
6.7.1.1. Experimental results on compression performance of the modified AVC with CABAC and CTW – 4CIF test sequences, I29P GOP structure

In the first series of experiments, the compression performance of the modified AVC with CABAC and CTW has been compared against the coding efficiency of the original AVC with CABAC. Experiments have been done according to **Scenario 1** (see Section 6.6).

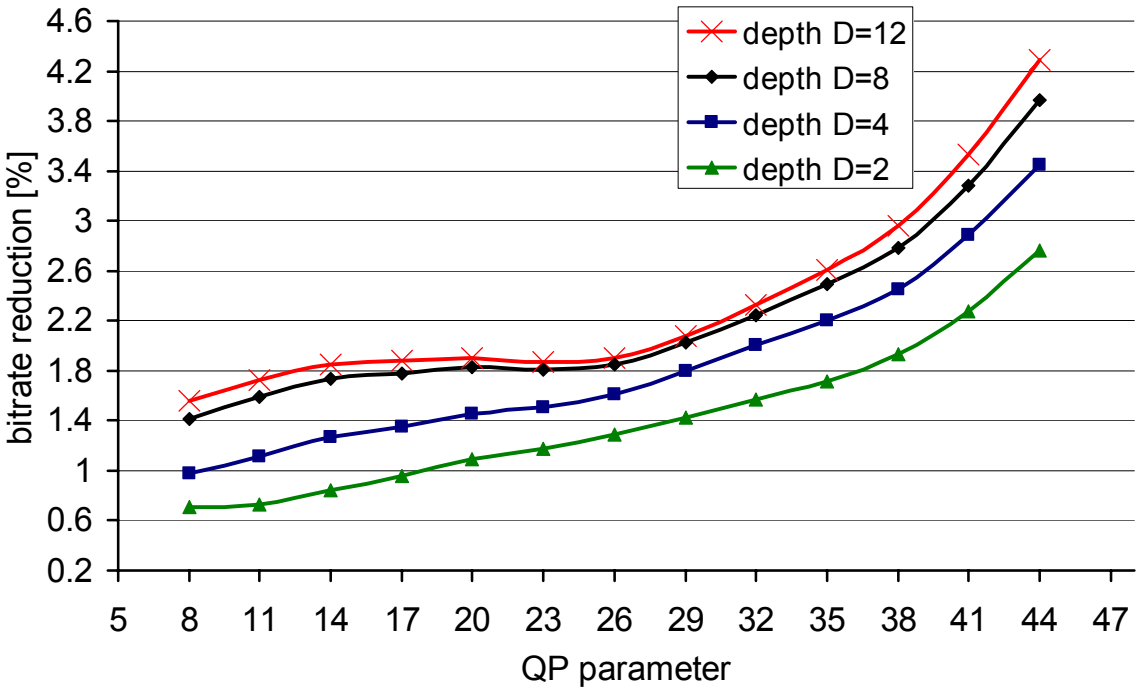
Detailed experimental results obtained for each of the test sequence have been presented in Annex A. Results achieved for I-frames only, for P-frames only and for the whole test sequence have been presented there. In this section, the averaged experimental results on compression performance of the modified and the original AVC encoders that have been obtained for CITY, CREW, ICE and HARBOUR test sequences have been presented for I-frames and P-frames in Table 6.1 and Figure 6.5. The averaged experimental results for the whole test sequences have been also presented in Figure 6.5.

Table 6.1. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I- and P-frames. The bitrate reduction is a result of application of CTW technique within CABAC algorithm.

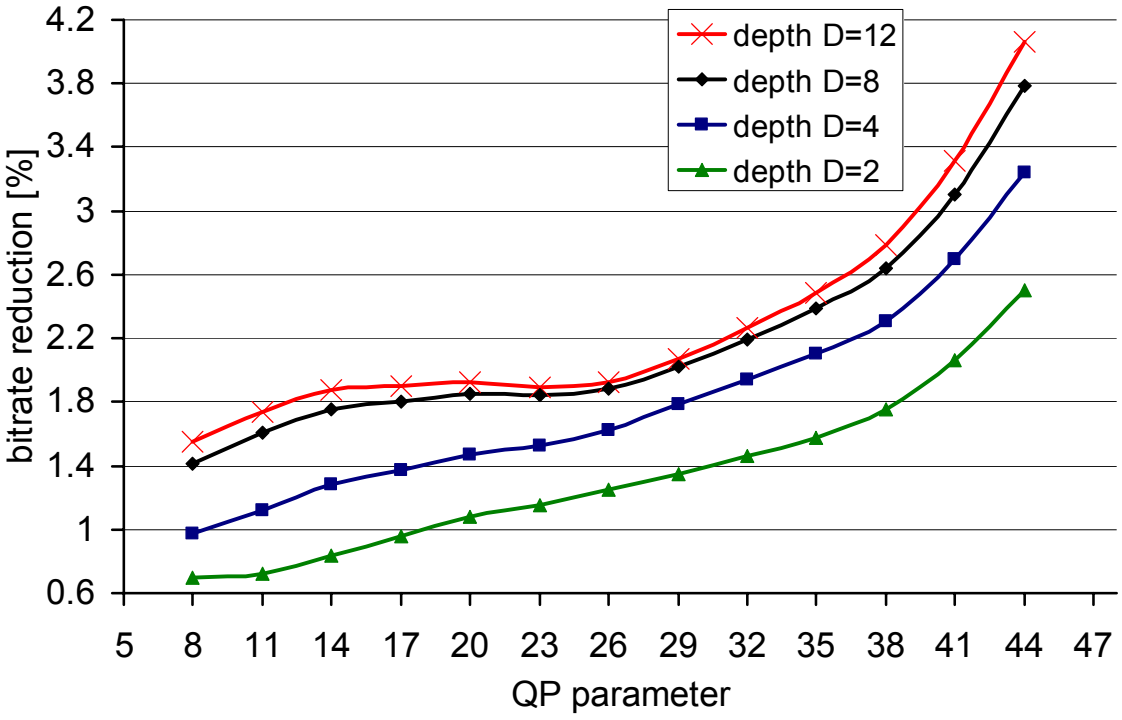
		averaged bitrate after using CABAC and CTW and different depths D of context trees [Mbits/s]				CABAC with CTW gain relative to original CABAC [%]			
QP parameter	Averaged bitrate for CABAC [Mbits/s]	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for I frames									
8	2.9467	2.9333	2.9211	2.9065	2.9045	0.4547	0.8689	1.3620	1.4313
11	2.3597	2.3462	2.3311	2.3150	2.3116	0.5702	1.2097	1.8940	2.0381
14	1.8928	1.8790	1.8652	1.8527	1.8507	0.7320	1.4581	2.1184	2.2279
17	1.4371	1.4244	1.4142	1.4061	1.4052	0.8827	1.5935	2.1576	2.2220
20	1.1017	1.0916	1.0833	1.0777	1.0774	0.9114	1.6684	2.1742	2.2060
23	0.8076	0.8003	0.7936	0.7902	0.7901	0.9079	1.7340	2.1581	2.1717
26	0.5965	0.5911	0.5861	0.5840	0.5839	0.8961	1.7306	2.0902	2.0986
29	0.4328	0.4289	0.4254	0.4241	0.4240	0.8907	1.7075	2.0160	2.0235
32	0.3171	0.3142	0.3119	0.3110	0.3109	0.9154	1.6525	1.9293	1.9387
35	0.2268	0.2246	0.2230	0.2224	0.2223	0.9524	1.6557	1.9423	1.9610
38	0.1594	0.1578	0.1567	0.1561	0.1561	1.0225	1.7172	2.0559	2.0763
41	0.1103	0.1090	0.1082	0.1077	0.1077	1.1604	1.9084	2.3753	2.4025
44	0.0747	0.0737	0.0730	0.0725	0.0724	1.3127	2.2503	2.9367	2.9803
Results for P frames									
8	69.1166	68.6258	68.4392	68.1384	68.0423	0.7102	0.9800	1.4153	1.5543
11	53.3168	52.9287	52.7212	52.4685	52.3974	0.7279	1.1171	1.5911	1.7244
14	38.6177	38.2938	38.1260	37.9467	37.9014	0.8387	1.2733	1.7375	1.8548
17	25.1254	24.8837	24.7846	24.6783	24.6526	0.9621	1.3563	1.7795	1.8819
20	15.3664	15.1987	15.1431	15.0857	15.0739	1.0914	1.4534	1.8268	1.9037
23	8.8265	8.7227	8.6932	8.6668	8.6618	1.1769	1.5109	1.8101	1.8662
26	4.8190	4.7566	4.7412	4.7296	4.7271	1.2945	1.6144	1.8546	1.9051
29	2.7390	2.7000	2.6896	2.6836	2.6819	1.4234	1.8019	2.0224	2.0816
32	1.6077	1.5824	1.5755	1.5715	1.5702	1.5712	2.0006	2.2488	2.3323
35	1.0000	0.9829	0.9780	0.9752	0.9740	1.7119	2.2032	2.4889	2.6061
38	0.6411	0.6287	0.6254	0.6232	0.6221	1.9318	2.4570	2.7885	2.9612
41	0.4555	0.4451	0.4423	0.4405	0.4394	2.2795	2.8844	3.2845	3.5337
44	0.3498	0.3402	0.3378	0.3360	0.3348	2.7614	3.4496	3.9628	4.2922


depth D=12
depth D=8
depth D=4
depth D=2
bitrate reduction [%]
3
2.6
2.2
1.8
1.4
1
0.6
0.2
5 8 11 14 17 20 23 26 29 32 35 38 41 44 47
QP parameter

(a)


depth D=12
depth D=8
depth D=4
depth D=2
bitrate reduction [%]
4.6
4.2
3.8
3.4
3
2.6
2.2
1.8
1.4
1
0.6
0.2
5 8 11 14 17 20 23 26 29 32 35 38 41 44 47
QP parameter

(b)



(c)

Figure 6.5. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I-frames (a), P-frames (b) and whole test sequences (c). Bitrate reduction is a result of using the modified AVC with CABAC and CTW technique in contrast to the original AVC with unmodified CABAC. The structure of GOP has been set on I29P.

Analysis of the achieved experimental results yields a conclusion that the gain of the coding efficiency of the modified CABAC with CTW technique relative to the compression performance of the original CABAC algorithm is strictly dependent on:

- The content of test video sequence (see Annex A);
- The type of frames;
- The value of the QP parameter;
- The depths *D* of context trees used in the modified AVC video codec.

The content of a video sequence influences the statistics of data that is finally coded with entropy encoder. These statistics correspond in minor or greater extent to the pre-defined "exponential aging" model (see Section 4.2.2.2.3) of source data that is used in the context modeler block of the original CABAC. It significantly influences the compression performance of the original CABAC. If the real statistics of data being encoded significantly

differs from the assumed one, the coding efficiency of the original CABAC decreases. In the modified CABAC, the application of CTW technique allows for much more accurate adaptation to the current statistics of coded data which leads to gain of compression performance in comparison to the original CABAC.

The same situation takes place in the case of different frame types (I- and P-frames in these experiments). Since I- and P-frames are usually distinguished by different statistical properties it influences on the coding efficiency of both the modified and the original CABAC entropy encoders. The author puts the thesis that another two elements that significantly influences on the compression performance of the modified AVC video encoder are:

- The algorithm of the context trees initialization;
- The size of the data set after which the context trees initialization is being done.

In the modified CABAC, conditional probabilities of source symbols are estimated with taking into consideration the statistics of already encoded symbols that have been gathered in the context trees. Therefore, CTW technique will work well if the statistics of future source symbols is estimated on the basis of bigger data set of previously encoded symbols. If this data set is too small, CTW technique will have incomplete information on the probability distribution of the source data and the data modeling algorithm will not be able to adjust properly to the current signal statistics. Generally, considering the coding efficiency of CTW technique the best solution would be to reset statistics gathered on the context trees as rarely as it is possible. But, taking into account practical applications of frames of different types, the context trees used in CTW technique have been reset to its default statistics each time before an I-slice or a slice of a new type. So, in this scenario of experiments context trees have been initialized to its default values each time before an I-frame and the first P-frame in GOP. It means a relatively poorer efficiency of CTW algorithm at the beginning of I-frames and at the beginning of the first P-frame in GOP. This problem is more and more visible in the case of higher values of QP parameter that corresponds to sequences of lower and lower bitrates. In author's opinion, for the reason that the size of I-frames is generally significantly greater than the size of P-frames the context trees resetting much more influences on the compression performance for the P-frames. But, the data statistics gathered in a given P-frame have been used in coding process of the next neighboring P-frame. It has significantly increased the compression performance of the modified AVC encoder for P-frames.

Compression performance of the modified AVC video encoder relative to coding efficiency of the original AVC is higher in the case of higher values of QP parameter (see

Table 6.1 and Figure 6.5). Depending on the value of QP parameter and the depth D of the context trees, 0.5% to 3% bitrate reduction has been obtained for I-frames and 0.7% to 4.3% bitrate reduction has been achieved for P-frames. Generally, higher gains have been observed in the case of higher QP parameter values. The usage of the data statistics from the previous P-frames in estimation of statistics in the successive P-frames has a big impact on obtained experimental results. Therefore, higher gains of coding efficiency have been obtained for P-frames.

The depth D of context trees used to track the statistics of coded data strongly influences on the efficiency of CTW technique. In order to test the influence of depths D of context trees on the compression performance of the modified AVC, experiments have been done for depths D=2, D=4, D=8 and D=12 for each of the test sequence. Obtained experimental results have showed that the bigger depth D of context trees the better efficiency of CTW technique and the greater gain of compression performance of the modified AVC video encoder relative to the original AVC encoder. In the case of I-frames the bitrate reductions of 0.5%-1.3%, 0.9%-2.3%, 1.4%-3.0% have been obtained for the depth D=2, D=4, and D=8 respectively. The experimental results for D=12 were nearly the same as for D=8. In the case of P-frames the bitrate reductions of 0.7%-2.8%, 1%-3.5%, 1.4%-4%, 1.5%-4.3% have been observed for the depth D=2, D=4, D=8 and D=12 respectively. The achieved experimental results are in agreement with the main idea of CTW technique. CTW technique assumes a certain maximum context length $D_{\max}$ and estimates probabilities of source symbols in each possible context from $D = 0$ to $D = D_{\max}$. Additionally, CTW method estimates probabilities in the assumption of memory source model and memoryless source model. For the reason of the fact that the real structure of data being coded is generally unknown for video signal it can not be said which model of source data will give the best estimate in a given moment. CTW method solves this problem by weighting probabilities calculated in different contexts with both the memory and the memoryless model of source data. It is obvious that the bigger depth D of the context tree the bigger number of different memory models of source data that can be exploit in CTW technique. It influences directly on the improvement of efficiency of CTW method. The obtained experimental results have showed that above a certain depth D of the context trees there is no point to further increase the depth D because it will not lead to the further increase of efficiency of CTW technique. In these experiments such a situation have took place for the depth D=8. The application of the depth D=12 in CTW technique only marginally improves the performance of data modeling

algorithm by significantly increasing of total estimation time for CTW method within the modified CABAC encoder and decoder. Better compression efficiency of the modified CABAC encoder for higher depths *D* of the context trees indicates that neighboring binary symbols that are fed to the arithmetic encoder are mutually correlated. Experimental results have showed that a given binary symbol is mainly correlated with 8 to 12 previous binary symbols. The statistics of "older" symbols practically does not influence on the statistics of the current symbol. This is very important conclusion of the experiment.

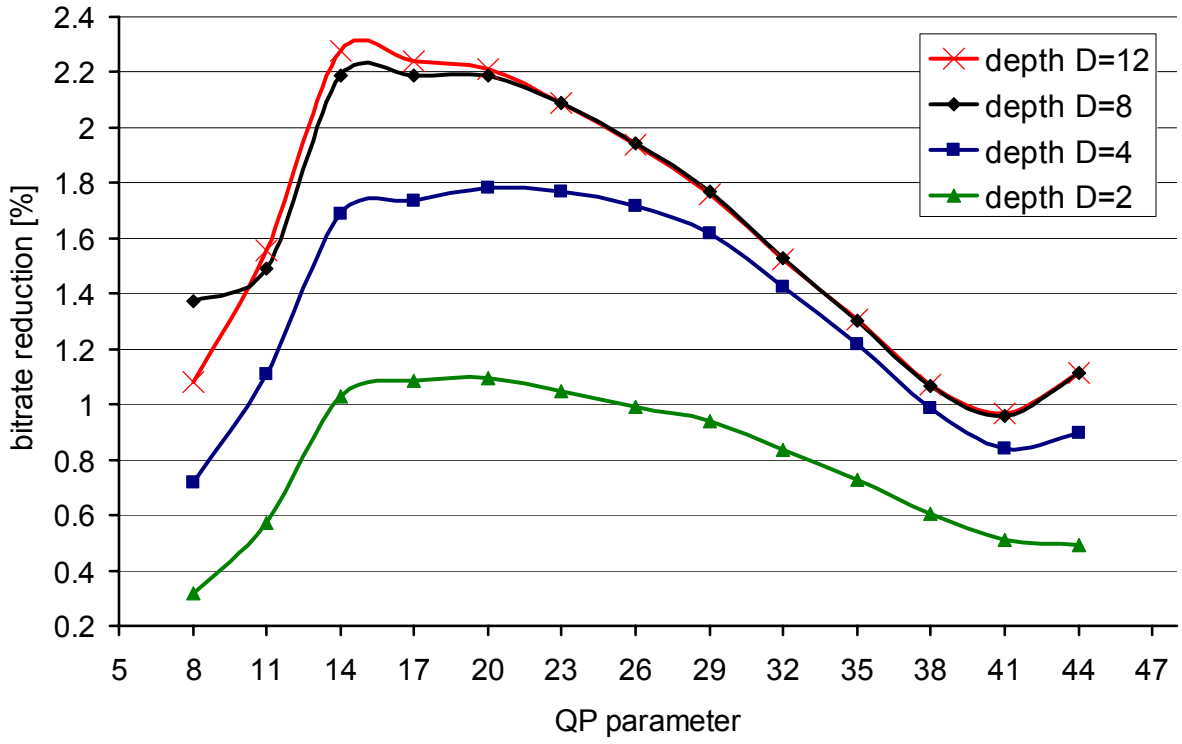
6.7.1.2. Experimental results on compression performance of the modified AVC with CABAC and CTW – CIF test sequences, I29P GOP structure

In the previous section the author has put the thesis that the size of the data set on which basis the context trees estimate the statistics of coded symbols significantly affects the compression performance of the modified CABAC with CTW method. In order to experimentally check correctness of this thesis experiments according to **Scenario 2** (see Section 6.6) have been done.

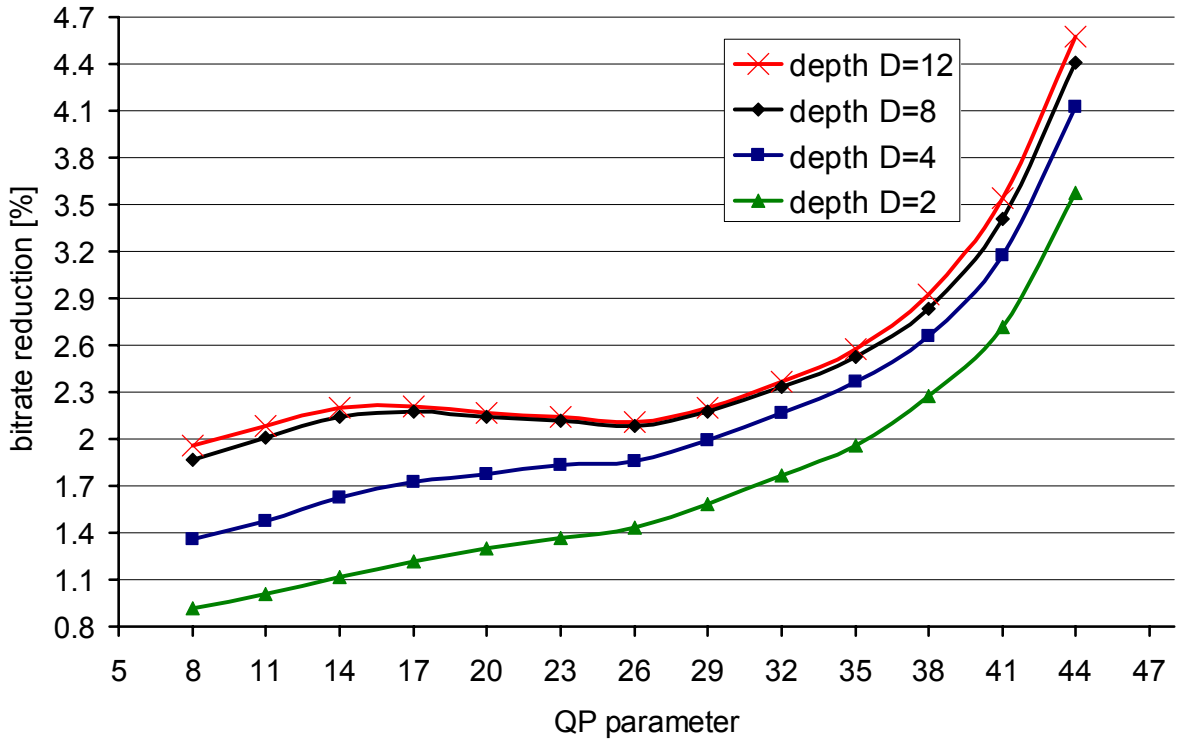
The detailed experimental results achieved for CITY, CREW, ICE and HARBOUR test sequences (in CIF format) have been presented in Annex A. The results on the compression performance of the modified CABAC with CTW obtained for I-frames only, P-frames only and the whole sequences have been presented there. The averaged experimental results achieved for the test sequences have been presented in Table 6.2 and in Figure 6.6.

Table 6.2. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR CIF test sequences for I- and P-frames. The bitrate reduction is a result of application of the CTW technique within CABAC algorithm.

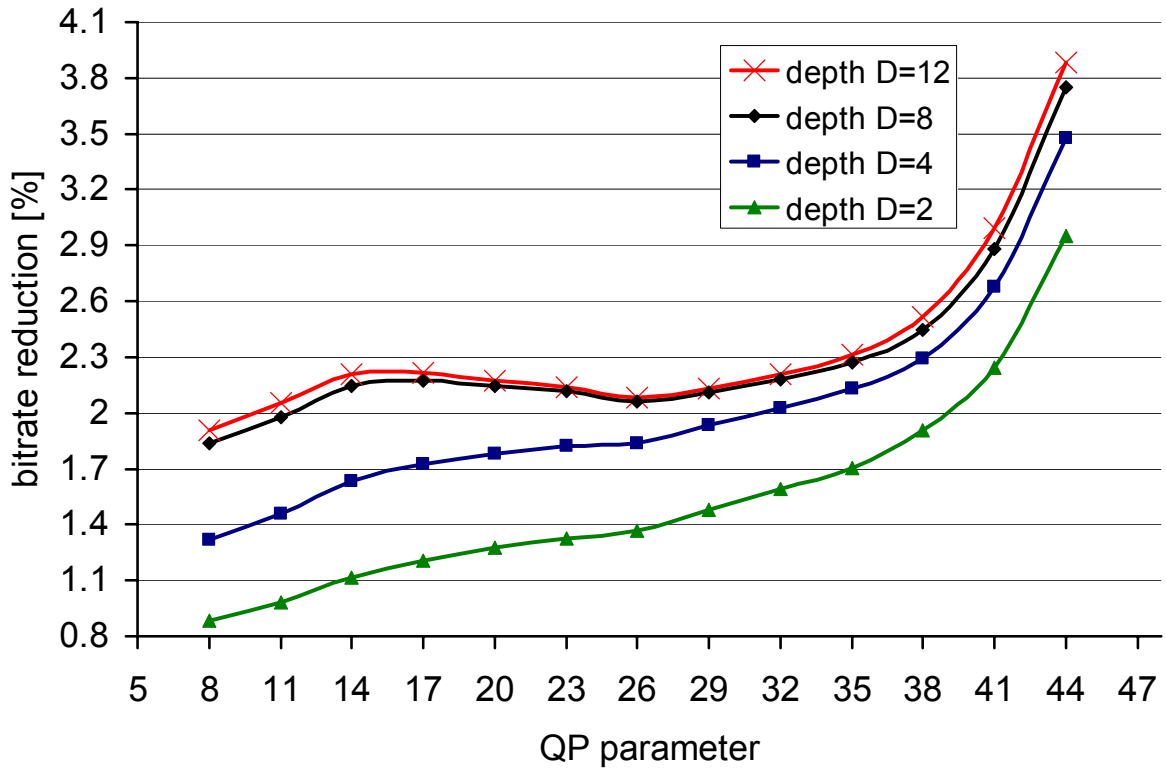
		averaged bitrate after using CABAC and CTW and different depths D of context trees [Mbits/s]				CABAC with CTW gain relative to original CABAC [%]			
QP parameter	Averaged bitrate for CABAC [Mbits/s]	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for I frames									
8	0.8615	0.8588	0.8553	0.8497	0.8522	0.3175	0.7182	1.3714	1.0807
11	0.6856	0.6817	0.6780	0.6754	0.6749	0.5710	1.1078	1.4928	1.5588
14	0.5618	0.5561	0.5523	0.5495	0.5490	1.0275	1.6878	2.1884	2.2765
17	0.4443	0.4395	0.4366	0.4346	0.4343	1.0860	1.7364	2.1871	2.2417
20	0.3543	0.3504	0.3480	0.3466	0.3465	1.0951	1.7852	2.1888	2.2114
23	0.2723	0.2694	0.2675	0.2666	0.2666	1.0458	1.7684	2.0880	2.0880
26	0.2087	0.2067	0.2052	0.2047	0.2047	0.9928	1.7150	1.9425	1.9366
29	0.1547	0.1532	0.1522	0.1519	0.1520	0.9407	1.6179	1.7666	1.7601
32	0.1141	0.1131	0.1124	0.1123	0.1123	0.8350	1.4268	1.5298	1.5232
35	0.0808	0.0802	0.0798	0.0798	0.0798	0.7300	1.2187	1.3022	1.3053
38	0.0558	0.0554	0.0552	0.0552	0.0552	0.6053	0.9864	1.0671	1.0716
41	0.0378	0.0376	0.0375	0.0374	0.0374	0.5097	0.8406	0.9598	0.9664
44	0.0253	0.0252	0.0251	0.0251	0.0251	0.4933	0.8979	1.1149	1.1149
Results for P frames									
8	14.2341	14.1035	14.0413	13.9682	13.9551	0.9174	1.3546	1.8678	1.9601
11	10.5634	10.4567	10.4073	10.3513	10.3432	1.0100	1.4778	2.0077	2.0847
14	7.5312	7.4468	7.4085	7.3698	7.3655	1.1206	1.6286	2.1428	2.1995
17	4.9992	4.9384	4.9130	4.8906	4.8887	1.2161	1.7232	2.1709	2.2094
20	3.2035	3.1620	3.1466	3.1349	3.1341	1.2963	1.7779	2.1420	2.1665
23	2.0818	2.0534	2.0437	2.0377	2.0373	1.3636	1.8302	2.1190	2.1413
26	1.2979	1.2793	1.2737	1.2708	1.2705	1.4320	1.8585	2.0821	2.1042
29	0.8028	0.7900	0.7868	0.7853	0.7851	1.5833	1.9938	2.1725	2.1984
32	0.4922	0.4835	0.4816	0.4807	0.4806	1.7640	2.1667	2.3328	2.3673
35	0.3081	0.3021	0.3008	0.3004	0.3002	1.9619	2.3700	2.5225	2.5769
38	0.1977	0.1932	0.1925	0.1921	0.1919	2.2733	2.6615	2.8360	2.9270
41	0.1383	0.1345	0.1339	0.1336	0.1334	2.7177	3.1787	3.4048	3.5404
44	0.1006	0.0970	0.0965	0.0962	0.0960	3.5721	4.1236	4.4093	4.5757


bitrate reduction [%]
QP parameter
depth D=12
depth D=8
depth D=4
depth D=2

(a)


bitrate reduction [%]
QP parameter
depth D=12
depth D=8
depth D=4
depth D=2

(b)



(c)

Figure 6.6. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR CIF test sequences for I-frames (a), P-frames (b) and whole test sequences (c). The bitrate reduction is a result of using the modified AVC with CABAC and CTW technique in contrast to the original AVC with unmodified CABAC. The structure of GOP has been set on I29P.

The experimental results achieved for sequences in CIF format allow forming the same conclusions as it took place for experiments done with test sequences in 4CIF format:

- The content of the test sequence influences on the compression performance of both the modified and the original AVC;
- The coding efficiency of the modified and the original AVC were different for I- and P-frames. Depending on the value of QP parameter and the depth *D* of context trees, 0.3% to 2.2% bitrate reduction has been obtained for I-frames and 0.9%-4.6% bitrate reduction has been observed for P-frames;
- The value of QP parameter affected the compression performance of the modified and the original AVC video encoders;
- The compression performance of the modified AVC with CABAC and the CTW was different for different depths *D* of the context trees.

The fundamental difference between experimental results obtained for test sequences in CIF and 4CIF formats concerns I-frames. In the case of I-frames the gain of the compression performance of the modified AVC relative to the original AVC clearly decreases with the increase of the value of QP parameter (see Figure 6.6). The averaged bitrate reductions (after using the modified AVC with depth D=8) obtained for I-frames for sequences in CIF and 4CIF formats have been presented in Figure 6.7.

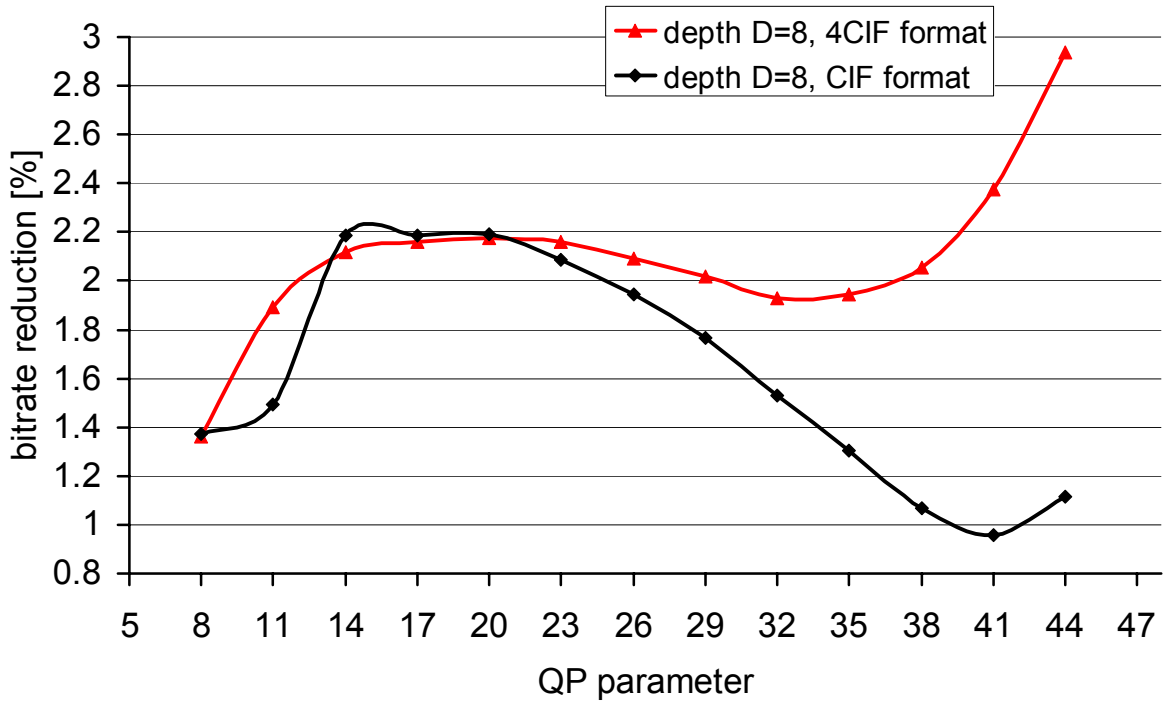


Figure 6.7. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF and CIF test sequences for I-frames. The bitrate reduction is a result of the use of the modified AVC encoder with CABAC and CTW technique (for D=8) in contrast to the original AVC.

In the case of sequences in 4CIF format the gain of the compression performance of the modified AVC (relative to the original AVC for I-frames) did not decrease with the increase of the value of QP parameter. Since the test sequences in CIF format have been achieved by downsampling of the original 4CIF test sequences it can be said that statistical properties of corresponding test sequences in both CIF and 4CIF formats are similar. Thus, the only parameter that is significantly different for sequences in CIF and 4CIF formats is the spatial resolution which is two times smaller in each direction for CIF format relative to 4CIF format. So, the number of image samples is four times smaller for CIF format in comparison to 4CIF format which directly influences on the size of the data encoded within a single image. Thus, size of the data set after which the context trees have been reset was significantly smaller in

the case of sequences in CIF format. It has caused the poorer compression performance of the modified AVC in contrast to results obtained for sequences in 4CIF format. The problem of too small set of already encoded symbols on the basis of which CTW technique estimates the data statistics has not occurred in the case of P-frames since the gathered data statistics have been reset only before the first P-frame in the GOP. It has been presented in Figure 6.8.

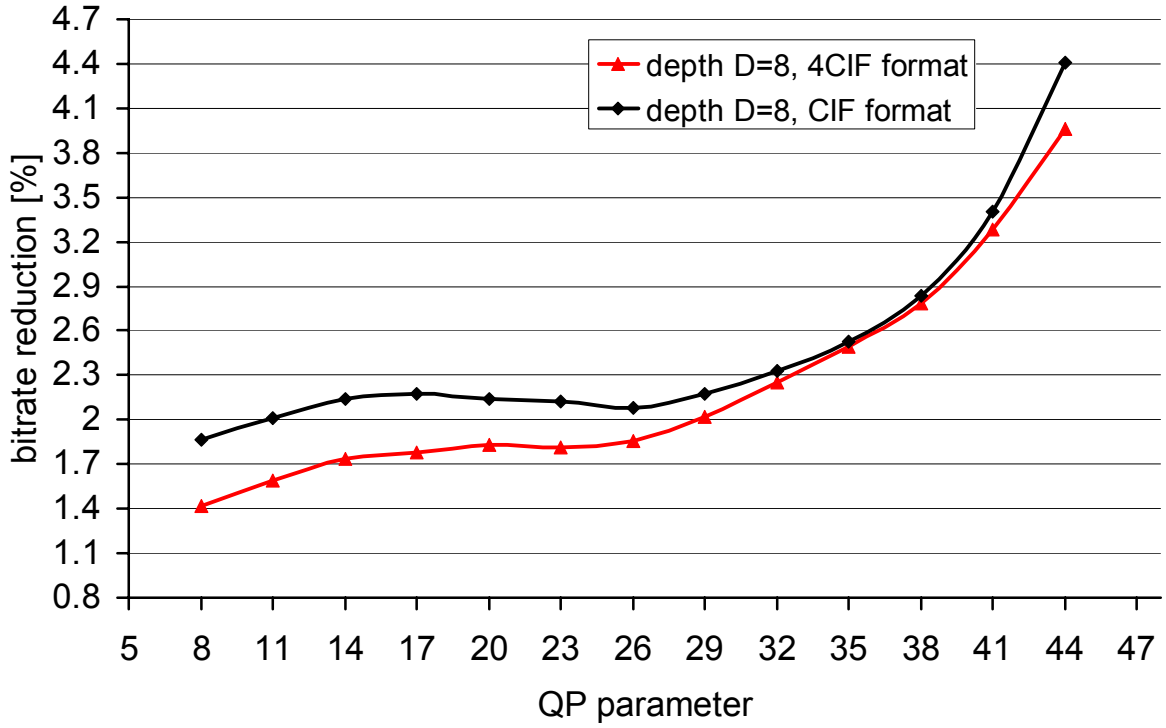


Figure 6.8. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF and CIF test sequences for P-frames. The bitrate reduction is a result of the use of the modified AVC encoder with CABAC and CTW technique (for *D*=8) in contrast to the original AVC with unmodified CABAC.

This experiment has proved the thesis formulated in the previous section that the size of the data set after which the context trees are initialized to its default values has significant impact on the efficiency of CTW technique.

6.7.1.3. Experimental results on compression performance of the modified AVC with CABAC and CTW – 4CIF test sequences, IBBPBBP… structure of GOP

The second series of experiments on the compression performance of the modified AVC video encoder (with CABAC and CTW) have been done according to **Scenario 3** (see Section 6.6).

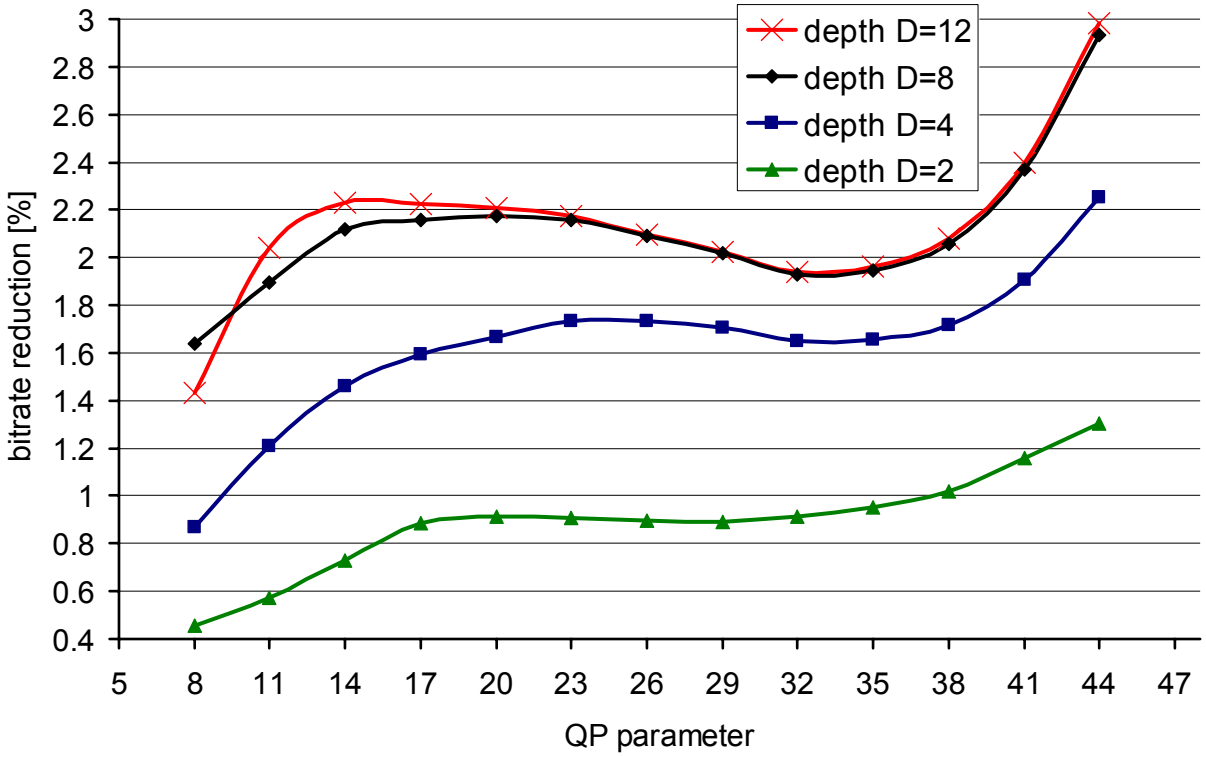
The detailed experimental results obtained for the test sequences for I-frames only, P-frames only, B-frames only and the whole video sequences have been presented in Annex A. The averaged experimental results obtained for CITY, CREW, ICE and HARBOUR test sequences have been presented in Table 6.3, Table 6.4 and Figure 6.9.

Table 6.3. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I- and P-frames. The bitrate reduction is a result of application of CTW technique within CABAC algorithm. GOP structure has been set on IBBP…

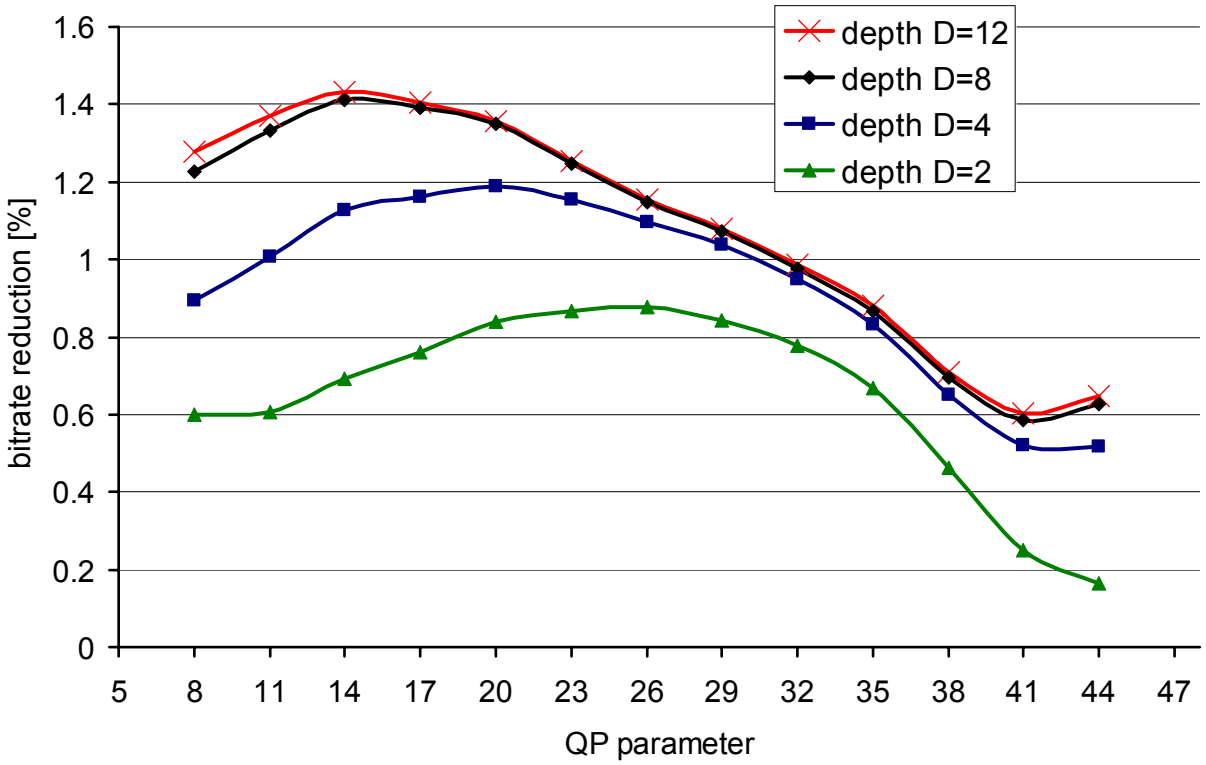
		averaged bitrate after using CABAC and CTW for different depths D of context trees [Mbits/s]				CABAC with CTW gain relative to original CABAC [%]			
QP parameter	Averaged bitrate for CABAC [Mbits/s]	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for I frames									
8	2.9569	2.9435	2.9312	2.9085	2.9146	0.4548	0.8689	1.6386	1.4315
11	2.3678	2.3543	2.3392	2.3230	2.3196	0.5702	1.2099	1.8942	2.0383
14	1.8994	1.8855	1.8717	1.8591	1.8571	0.7317	1.4581	2.1185	2.2280
17	1.4421	1.4293	1.4191	1.4109	1.4100	0.8828	1.5936	2.1579	2.2222
20	1.1055	1.0954	1.0870	1.0814	1.0811	0.9118	1.6688	2.1747	2.2066
23	0.8104	0.8031	0.7964	0.7929	0.7928	0.9082	1.7340	2.1584	2.1726
26	0.5985	0.5931	0.5882	0.5860	0.5860	0.8964	1.7310	2.0902	2.0985
29	0.4343	0.4304	0.4269	0.4255	0.4255	0.8917	1.7075	2.0160	2.0235
32	0.3182	0.3153	0.3129	0.3120	0.3120	0.9154	1.6524	1.9297	1.9384
35	0.2276	0.2254	0.2238	0.2232	0.2231	0.9513	1.6566	1.9433	1.9620
38	0.1600	0.1583	0.1572	0.1567	0.1566	1.0221	1.7191	2.0567	2.0785
41	0.1107	0.1094	0.1086	0.1081	0.1080	1.1565	1.9064	2.3717	2.3988
44	0.0749	0.0739	0.0732	0.0727	0.0727	1.3048	2.2526	2.9333	2.9834
Results for P frames									
8	22.9633	22.8255	22.7582	22.6820	22.6697	0.6001	0.8933	1.2250	1.2788
11	18.0085	17.8991	17.8269	17.7683	17.7617	0.6074	1.0087	1.3336	1.3706
14	13.3964	13.3038	13.2456	13.2075	13.2047	0.6910	1.1257	1.4101	1.4309
17	9.0468	8.9780	8.9418	8.9208	8.9196	0.7610	1.1612	1.3926	1.4056
20	5.8634	5.8142	5.7936	5.7842	5.7838	0.8378	1.1893	1.3503	1.3568
23	3.6775	3.6456	3.6350	3.6315	3.6314	0.8663	1.1558	1.2488	1.2532
26	2.2145	2.1951	2.1902	2.1890	2.1889	0.8764	1.0967	1.1492	1.1559
29	1.3428	1.3315	1.3289	1.3284	1.3283	0.8439	1.0387	1.0726	1.0806
32	0.8133	0.8070	0.8056	0.8054	0.8053	0.7792	0.9504	0.9759	0.9855
35	0.5071	0.5037	0.5029	0.5027	0.5026	0.6695	0.8342	0.8682	0.8790
38	0.3176	0.3161	0.3155	0.3154	0.3153	0.4637	0.6503	0.6951	0.7085
41	0.2131	0.2126	0.2120	0.2119	0.2118	0.2510	0.5220	0.5865	0.6029
44	0.1546	0.1543	0.1538	0.1536	0.1536	0.1650	0.5159	0.6259	0.6485

Table 6.4. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for B-frames. The bitrate reduction is a result of application the CTW technique within CABAC algorithm. GOP structure has been set on IBBP…

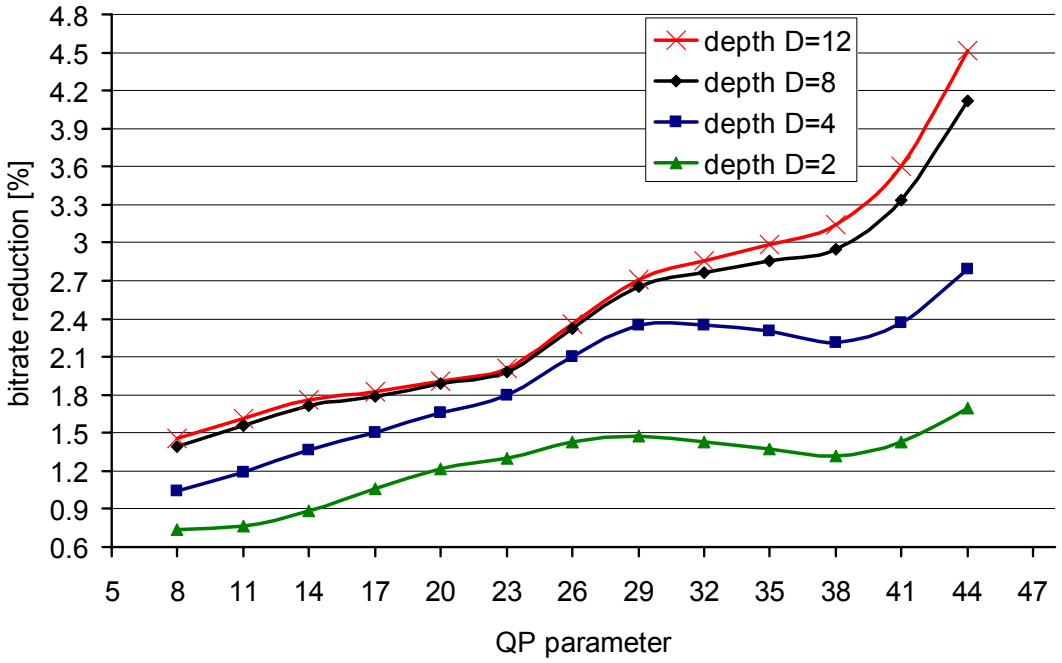
Results for B frames									
		averaged bitrate after using CABAC and CTW for different depths D of context trees [Mbits/s]				CABAC with CTW gain relative to original CABAC [%]			
QP parameter	averaged bitrate for CABAC [Mbits/s]	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
8	46.4451	46.1002	45.9593	45.7982	45.7677	0.7426	1.0458	1.3929	1.4583
11	35.6832	35.4100	35.2582	35.1268	35.1081	0.7655	1.1908	1.5592	1.6115
14	25.8443	25.6155	25.4916	25.4002	25.3885	0.8854	1.3648	1.7187	1.7636
17	16.3150	16.1416	16.0695	16.0230	16.0176	1.0634	1.5053	1.7901	1.8234
20	9.7006	9.5830	9.5393	9.5175	9.5155	1.2127	1.6629	1.8875	1.9083
23	5.4216	5.3511	5.3241	5.3141	5.3129	1.3017	1.7981	1.9841	2.0056
26	2.7965	2.7566	2.7376	2.7314	2.7304	1.4257	2.1034	2.3265	2.3628
29	1.4936	1.4715	1.4585	1.4540	1.4532	1.4795	2.3479	2.6502	2.7059
32	0.8452	0.8330	0.8253	0.8218	0.8210	1.4326	2.3496	2.7690	2.8524
35	0.5237	0.5165	0.5116	0.5088	0.5081	1.3762	2.3066	2.8560	2.9849
38	0.3356	0.3311	0.3281	0.3257	0.3250	1.3202	2.2135	2.9459	3.1441
41	0.2348	0.2315	0.2293	0.2270	0.2264	1.4287	2.3655	3.3311	3.6058
44	0.1629	0.1602	0.1584	0.1562	0.1556	1.7001	2.7879	4.1197	4.5125


depth D=12
depth D=8
depth D=4
depth D=2
bitrate reduction [%]
QP parameter
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
5
8
11
14
17
20
23
26
29
32
35
38
41
44
47

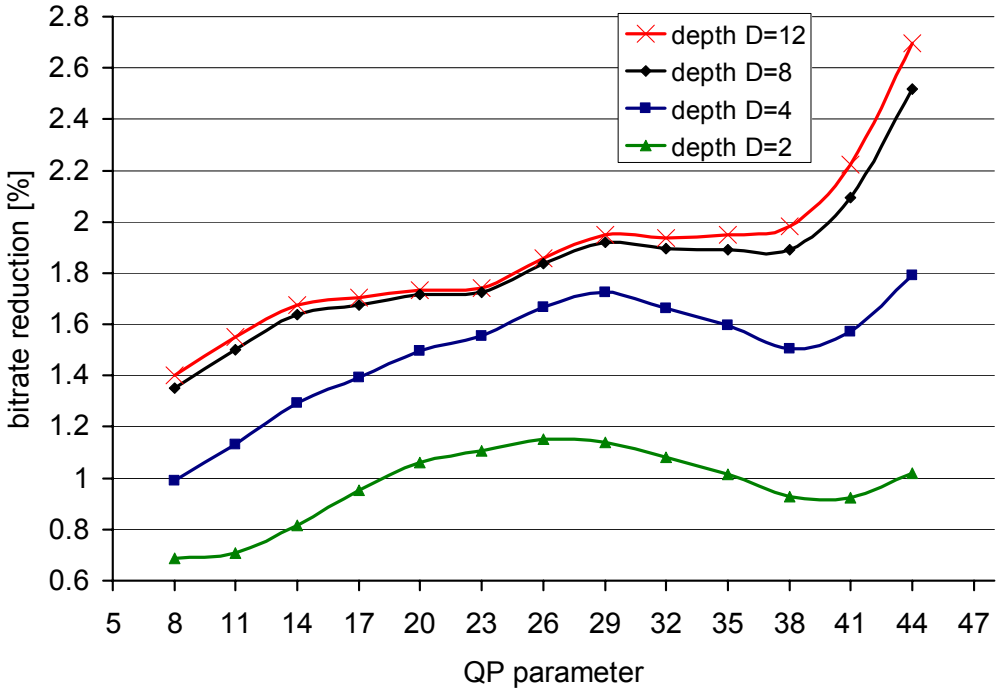
(a)


depth D=12
depth D=8
depth D=4
depth D=2
bitrate reduction [%]
QP parameter
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
5
8
11
14
17
20
23
26
29
32
35
38
41
44
47

(b)



(c)



(d)

Figure 6.9. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I-frames (a), P-frames (b), B-frames (c) and whole test sequences (d). The bitrate reduction is a result of the use of the modified AVC with CABAC and CTW technique in contrast to the original AVC with unmodified CABAC.

Experimental results obtained for I-frames are similar to those presented in Section 6.7.1.1, where I29P GOP structure has been used. Depending on the value of QP parameter and the depth D of the context trees, 0.4% to 3% bitrate reduction has been obtained for I-frames. However, different results have been obtained in the case of P-frames in comparison to results achieved in Section 6.7.1.1. In this case, 0.2% to 1.4% bitrate reduction has been obtained and the coding efficiency decreases with the increasing of QP parameter value. It mainly results from different structures of GOP that have been used in the first and the second series of experiments (I29P and IBBP… structures of GOP). In this case, different structures of GOP determine different ways of working of the context trees initialization method in CTW technique. The applied algorithm of the context trees initialization strongly influences the efficiency of CTW technique. In author's implementation of CTW technique the data statistics gathered in the context trees have been reset each time before an I-slice or a slice of a new type. In the case of I29P structure of GOP there were 29 consecutive P-frames within a GOP. Therefore, the context trees in CTW method have been reset to its default values only in the case of the first P-frame in a GOP. In each successive P-frame the data statistics gathered in the preceding P-frames has been used to estimate the probability distribution of the data in the successive P-frame. It obviously positively affects the efficiency of CTW technique in the successive P-frames within a GOP. So, the last 28 P-frames in a GOP have been encoded with "good" statistics gathered in the context trees. In the case of IBBPBBP… structure of a GOP the P-frames have been alternated with B-frames. Therefore, the context trees have been initialized before each P-frame in author's implementation. It must be stated that simple context trees initialization has been used in experiments, in which the counters of the number of zeros a_s and the number of ones b_s have been initialized to 0. So, at the beginning of each P-frame CTW method had not a "good" knowledge about the character of coded data, because context trees had been earlier initialized to 0. Additionally, taking into consideration the fact of relatively small size of the data that represents P-frames, CTW technique has not been able to exactly estimate the probabilities of symbols. Therefore, the efficiency of the modified AVC video encoder significantly falls down with the increasing of the value of QP parameter for P-frames.

According to experimental results, the compression performance of the modified AVC with CABAC and CTW increases with the increase of the value of QP parameter for B-frames. It has been presented in Figure 6.9. From 0.7% to 4.5% bitrate reduction has been achieved that is dependent on the depth D of the context trees. Generally, from the size of data

point of view the B-frames are significantly smaller than I- and P-frames. Therefore, the algorithm of the context trees initialization has even a greater importance on the efficiency of CTW method in the case of B-frames (in contrast to I- and P- frames). In the analyzed structure of GOP, the B-frames have been grouped in pairs. Thus, the context trees have been reset only before the first B-frame from a pair. The second B-frame from a pair has used the data statistics gathered in the previous B-frame. It positively influences on the compression performance of the modified AVC. Such an approach turned out to work better than the adaptive method of the context initialization used in the original CABAC. It is clearly visible for higher values of the QP parameter when the distance between the coding efficiency of the modified and the original AVC video encoders is higher.

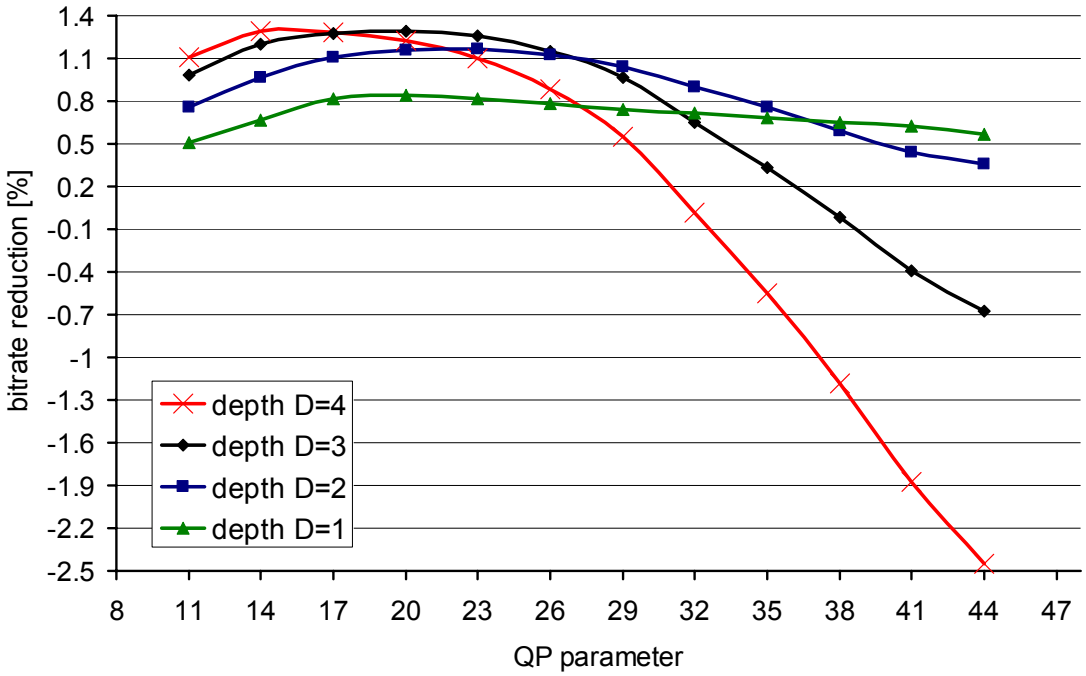
6.7.2. Compression performance of the modified AVC with CABAC and PPMA in contrast to the original AVC with CABAC

In order to test the influence on the compression performance of the application of PPMA technique in CABAC algorithm within AVC experiments have been done. The compression performance of the modified AVC encoder with CABAC that exploits PPMA has been confronted with the coding efficiency of the original AVC with unmodified CABAC. Experiments have been done according to **Scenario 1** (see Section 6.6).

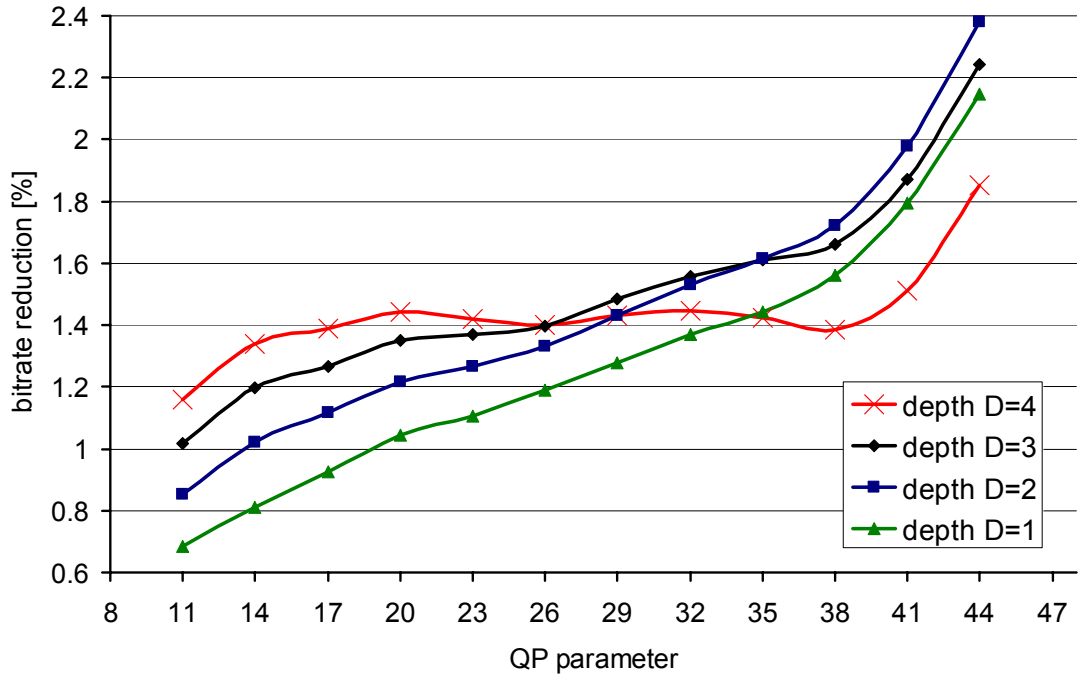
Test video sequences in 4CIF format have been used and I29P structure of GOP has been assumed. Experiments on the coding efficiency of the modified AVC with CABAC and the CTW have been done for different depths D of the context trees equal to D=1, D=2, D=3 and D=4. The experimental results obtained for individual test sequences have been presented in Annex B. In Table 6.5 and in Figure 6.10 the averaged experimental results obtained for I-frames and P-frames have been presented.

Table 6.5. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I- and P-frames. The bitrate reduction is a result of application PPMA technique within CABAC algorithm.

		averaged bitrate after using CABAC and PPMA and different depths D of context trees [Mbits/s]				CABAC with PPMA gain relative to original CABAC [%]			
QP parameter	averaged bitrate for CABAC [Mbits/s]	D=1	D=2	D=3	D=4	D=1	D=2	D=3	D=4
Results for I frames									
11	2.3597	2.3477	2.3418	2.3365	2.3335	0.5049	0.7555	0.9828	1.1086
14	1.8928	1.8802	1.8745	1.8701	1.8684	0.6670	0.9696	1.2027	1.2916
17	1.4371	1.4253	1.4212	1.4187	1.4186	0.8178	1.1054	1.2774	1.2845
20	1.1017	1.0924	1.0889	1.0875	1.0881	0.8415	1.1553	1.2883	1.2270
23	0.8076	0.8010	0.7983	0.7975	0.7988	0.8203	1.1633	1.2574	1.1001
26	0.5965	0.5918	0.5897	0.5896	0.5912	0.7859	1.1266	1.1489	0.8844
29	0.4328	0.4296	0.4283	0.4286	0.4304	0.7434	1.0432	0.9653	0.5470
32	0.3171	0.3148	0.3142	0.3150	0.3170	0.7151	0.8996	0.6497	0.0197
35	0.2268	0.2252	0.2251	0.2260	0.2280	0.6856	0.7584	0.3373	-0.5523
38	0.1594	0.1584	0.1585	0.1594	0.1613	0.6508	0.5896	-0.0157	-1.1793
41	0.1103	0.1096	0.1098	0.1107	0.1124	0.6210	0.4442	-0.3898	-1.8789
44	0.0747	0.0742	0.0744	0.0752	0.0765	0.5693	0.3550	-0.6731	-2.4478
Results for P frames									
		averaged bitrate after using CABAC and PPMA and different depths D of context trees [Mbits/s]				CABAC with PPMA gain relative to original CABAC [%]			
QP parameter	averaged bitrate for CABAC [Mbits/s]	D=1	D=2	D=3	D=4	D=1	D=2	D=3	D=4
11	53.3168	52.9515	52.8617	52.7735	52.6980	0.6852	0.8537	1.0191	1.1607
14	38.6177	38.3050	38.2229	38.1546	38.1005	0.8096	1.0223	1.1991	1.3392
17	25.1254	24.8925	24.8452	24.8069	24.7767	0.9270	1.1151	1.2679	1.3879
20	15.3664	15.2062	15.1793	15.1589	15.1449	1.0424	1.2177	1.3503	1.4413
23	8.8265	8.7288	8.7147	8.7057	8.7011	1.1074	1.2674	1.3695	1.4206
26	4.8190	4.7616	4.7548	4.7516	4.7514	1.1901	1.3308	1.3985	1.4013
29	2.7390	2.7040	2.6997	2.6983	2.6997	1.2762	1.4326	1.4831	1.4321
32	1.6077	1.5857	1.5831	1.5826	1.5844	1.3683	1.5299	1.5591	1.4474
35	1.0000	0.9856	0.9839	0.9839	0.9858	1.4424	1.6134	1.6117	1.4227
38	0.6411	0.6311	0.6301	0.6305	0.6322	1.5617	1.7216	1.6608	1.3839
41	0.4555	0.4473	0.4465	0.4469	0.4486	1.7932	1.9793	1.8711	1.5127
44	0.3498	0.3423	0.3415	0.3420	0.3433	2.1461	2.3812	2.2412	1.8517



(a)



(b)

Figure 6.10. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I-frames (a) and P-frames (b). The bitrate reduction is a result of using the modified AVC with CABAC and PPMA technique in contrast to the original AVC.

The obtained experimental results proved that the compression performance of the modified AVC encoder with CABAC and PPMA is strictly dependent on:

- The length D of the context (past symbols) that is used for estimation of the probability of the next symbol;
- The value of QP parameter for macroblock well, the size of output bitrate of encoded video sequence;
- The content of the video sequence (the amount of details in each frame) and a kind of motion in sequence (slow motion or fast motion) that directly influence the statistics of prediction residual data that is finally encoded by entropy encoder;
- The prediction mode (intra prediction or inter prediction) that significantly affects the probability distribution of coded data that is fed to entropy encoder.

Based on achieved experimental results it is clear that the efficiency of PPMA technique is different for different depths D of the context trees that are used to the data statistics gathering. In the case of smaller depths D of the context trees PPM technique can not track the long-term dependences between source symbols which not allow for accurate estimation of the conditional probabilities of source symbols. It causes significant decrease of the coding efficiency of PPM data modeling technique. The greater depth D of the context trees the greater knowledge of PPMA modeling technique on the statistics of previously encoded source symbols and the better coding efficiency of PPMA within CABAC. However, in order to inform the decoder on the context length in which the statistics of new symbol will be estimated, the encoder must send to the decoder some sequence of ESCAPE symbols. In PPMA technique, this sequence of ESCAPE symbols can be treated as side information that finally influences on the coding efficiency of the modified AVC with CABAC and PPMA. So, when depth D of context trees exceeds a certain value $D_{\max}$ cost of sending side information is greater than gain of the coding efficiency that is a result of using the longer contexts in the probability estimation for the next symbol. Therefore, the efficiency of PPMA modeling technique decreases when using context trees of depth $D > D_{\max}$. According to the literature, the optimal value of $D_{\max}$ for PPM family of techniques used in text compression systems is equal to 5 or 6 [Salom06]. In the application of PPMA technique within CABAC, the coding efficiency of PPMA method fast falls down for depths D of context trees greater than 3 or 4. This trend is especially visible for I-frames encoded with high value of QP parameter. The difference in optimal depth $D_{\max}$ of the context trees between PPMA within

CABAC and PPM in text compression systems results from the fact that PPMA within CABAC works with binary source data whereas PPM within a given text compression system usually works with 256-ary source data. So, the coding efficiency of the compression system with PPMA is dependent on the nature of coded data (its statistics) and also the size of alphabet of source data.

The size of already encoded data set on the basis of which entropy encoder estimates the statistics of successive symbols strongly influences on the compression performance of entropy encoder. Therefore, the compression performance of the modified AVC with CABAC and PPMA is different for different values of QP parameter. The smaller value of QP parameter, the higher bitrate of encoded video sequence. In this situation, the efficiency of PPMA technique is higher, hence greater bitrate reduction of the modified AVC in comparison to the original AVC with unmodified CABAC is achieved. In the case of bigger values of QP parameter the size of resulted bitrate of encoded video sequence is significantly smaller. In the modified AVC encoder, the gathered data statistics are reset to zero each time before an I-slice or a slice of a new type. For smaller bitrates PPMA is not able to adjust to the current statistics of coded data within a single frame. Due to the context trees are reset before each I-frame, the poor coding efficiency of the modified CABAC with PPMA has been observed (see Figure 6.10). Within a GOP, the successive P-frames have used the data statistics of the previous P-frames so better compression performance of the modified AVC encoder has been obtained (see Figure 6.10). Depending on the value of QP parameter and the depth D of the context trees, from 0.5% to 1.3% bitrate reduction has been obtained for I-frames and from 0.7% to 2.4% bitrate reduction has been achieved for P-frames. In the case of I-frames and higher values of QP parameter, the compression performance of the modified CABAC with PPMA is worse in comparison to the coding efficiency of the original CABAC. The experiment has proved extremely high significance of the applied algorithm of the context trees initialization to the compression performance of the modified AVC video encoder.

The difference in the compression performance between the modified and the original AVC video encoder is also dependent on the content of the test video sequence. In the experiments, different results have been achieved for each of the test sequences (see detailed experimental results in Annex B). It results from two facts:

- The content of the video sequence influences on the probability distribution of the data that is finally encoded with entropy encoder. For some sequences, the character of coded

data can differ from the long-term memory model of source assumed in PPM technique which influences on the coding efficiency of CABAC with PPMA;

- The pre-defined statistical model that is used in the original CABAC assumes the "exponential aging" model of source data [Howa92]. If the real statistics of coded data meaningfully differs from the assumed one, the compression performance of CABAC decreases. In these cases, the compression performance of the modified and the original AVC video encoders can significantly differ between themselves;

The same situation takes place in the case of different frame types (I- and P-frames in these experiments), since I- and P-frames are usually distinguished by different statistical properties. Additionally, different coding efficiency of the modified AVC with CABAC and PPMA for I- and P-frame types results from:

- The size of I-frames is usually significantly greater than the size of P-frames. So, working on a greater data set in the case of I-frames PPMA method is able to adjust more precisely to the current signal statistics;
- However, the data statistics gathered in previous P-frames is used in the successive P-frames which positively influences on the compression performance of the modified AVC in the case of P-frames.

6.7.3. Compression performance of the modified AVC with CABAC and PPMA – summary and conclusions

Experimental results have proved that for some depths D of the context trees the coding efficiency of the modified AVC video encoder with PPMA is visibly better in comparison to the compression performance of the original AVC with unmodified CABAC. Nevertheless, the compression performance of the modified CABAC with PPMA is poor in comparison to the coding efficiency of the modified CABAC that estimates the conditional probabilities of symbols with CTW technique. Therefore, the compression performance of other variants of PPM data modeling method has not been explored.

6.7.4. Compression performance of the modified AVC with CABAC and joint application of CTW and PPMA in contrast to the original AVC with CABAC

The coding efficiency of the modified AVC with CABAC that exploits sophisticated technique of the data statistics modeling based on new proposal of joint application of CTW

and PPMA has been tested. The experimental results have been compared to the coding efficiency of the original AVC with CABAC achieved for the same test video sequences. Experiments have been done according to **Scenario 3** (see Section 6.6). The compression performance of the modified AVC encoder with CABAC and CTW and PPMA has been investigated for depths D of the context trees equal to 2, 4, and 8. Four 4CIF test sequences have been used in experiments. The IBBPBBP… structure of GOP has been considered.

In Annex C the detailed experimental results on the compression performance of the modified and the original AVC obtained for CITY, CREW, ICE and HARBOUR test sequences have been presented. These experimental results concern I-, P- and B-frame types. Additionally, the experimental results for the whole test sequences have been shown.

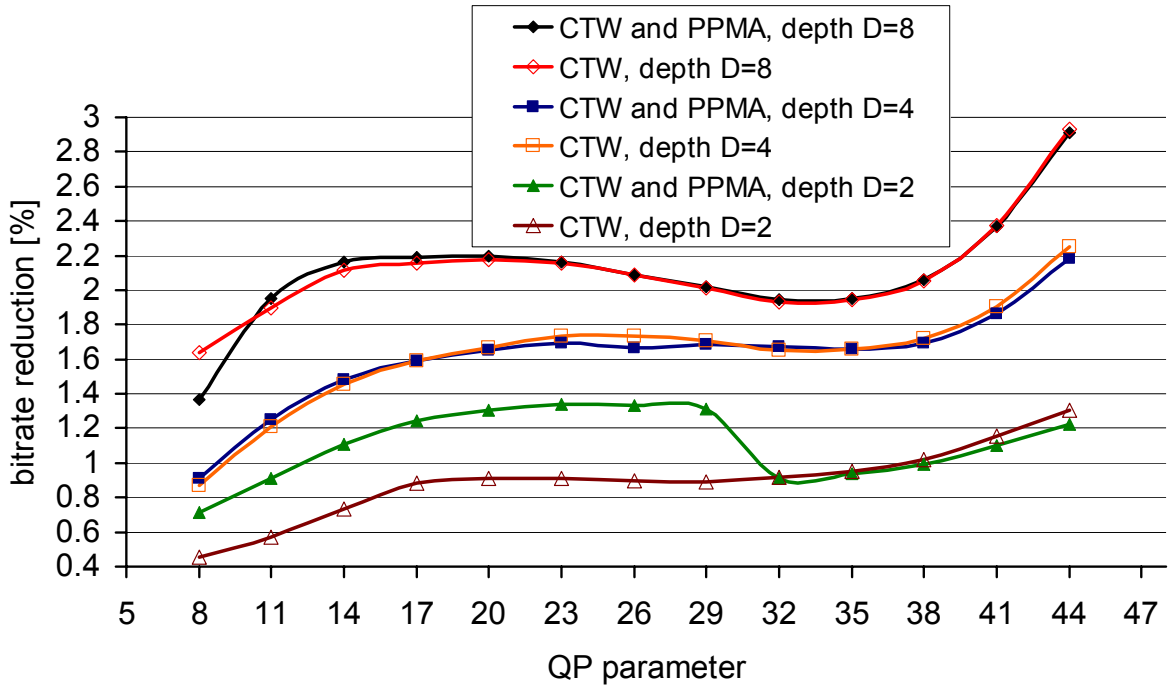
The averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR test sequences has been presented in Table 6.6, Table 6.7, and Figure 6.11 for I-, P- and B-frames. All experimental results have been referenced to results obtained for the modified AVC with CABAC and CTW.

Table 6.6. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I- and P-frames. The bitrate reduction is a result of application of CTW+PPMA technique within CABAC algorithm.

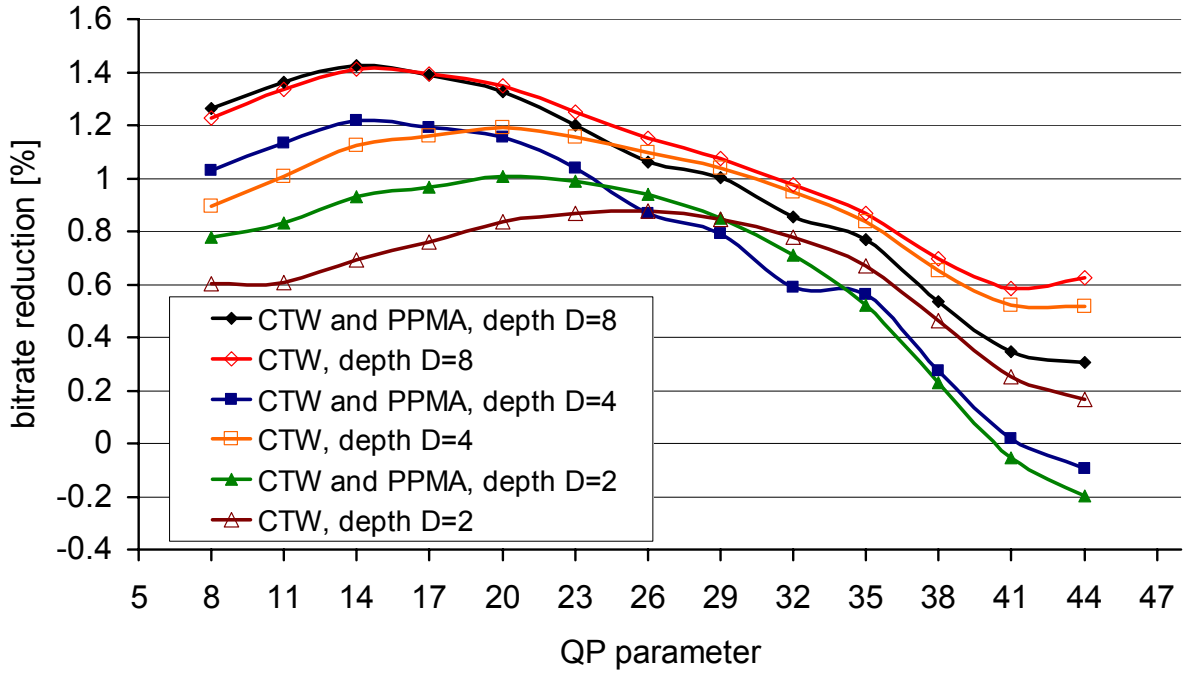
		averaged bitrate after using CABAC and CTW+PPMA for different depths D of context trees [Mbits/s]			CABAC with CTW+PPMA gain relative to original CABAC [%]		
QP parameter	averaged bitrate for CABAC [Mbits/s]	D=2	D=4	D=8	D=2	D=4	D=8
Results for I frames							
8	2.9569	2.9358	2.9301	2.9164	0.7132	0.9086	1.3692
11	2.3678	2.3462	2.3383	2.3217	0.9133	1.2494	1.9500
14	1.8994	1.8783	1.8713	1.8583	1.1105	1.4805	2.1651
17	1.4421	1.4241	1.4191	1.4105	1.2451	1.5934	2.1894
20	1.1055	1.0910	1.0872	1.0812	1.3044	1.6500	2.1955
23	0.8104	0.7996	0.7967	0.7929	1.3379	1.6917	2.1655
26	0.5985	0.5905	0.5885	0.5860	1.3337	1.6658	2.0877
29	0.4343	0.4286	0.4270	0.4255	1.3137	1.6839	2.0212
32	0.3182	0.3153	0.3129	0.3120	0.9122	1.6736	1.9423
35	0.2276	0.2254	0.2238	0.2231	0.9393	1.6566	1.9510
38	0.1600	0.1584	0.1573	0.1567	0.9940	1.6957	2.0598
41	0.1107	0.1095	0.1086	0.1081	1.1000	1.8612	2.3649
44	0.0749	0.0740	0.0733	0.0727	1.2247	2.1825	2.9133
Results for P frames							
8	22.9633	22.7843	22.7271	22.6735	0.7797	1.0287	1.2623
11	18.0085	17.8585	17.8045	17.7631	0.8331	1.1331	1.3629
14	13.3964	13.2717	13.2334	13.2054	0.9305	1.2166	1.4256
17	9.0468	8.9595	8.9389	8.9212	0.9654	1.1930	1.3879
20	5.8634	5.8044	5.7956	5.7856	1.0056	1.1557	1.3264
23	3.6775	3.6411	3.6393	3.6333	0.9892	1.0375	1.2020
26	2.2145	2.1937	2.1953	2.1910	0.9375	0.8677	1.0621
29	1.3428	1.3314	1.3321	1.3293	0.8501	0.7931	1.0026
32	0.8133	0.8076	0.8085	0.8064	0.7103	0.5908	0.8548
35	0.5071	0.5044	0.5042	0.5032	0.5226	0.5620	0.7691
38	0.3176	0.3168	0.3167	0.3159	0.2291	0.2724	0.5353
41	0.2131	0.2132	0.2131	0.2124	-0.0551	0.0176	0.3449
44	0.1546	0.1549	0.1547	0.1541	-0.1973	-0.0938	0.3057

Table 6.7. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for B-frames. The bitrate reduction is a result of application of CTW+PPMA technique within CABAC algorithm.

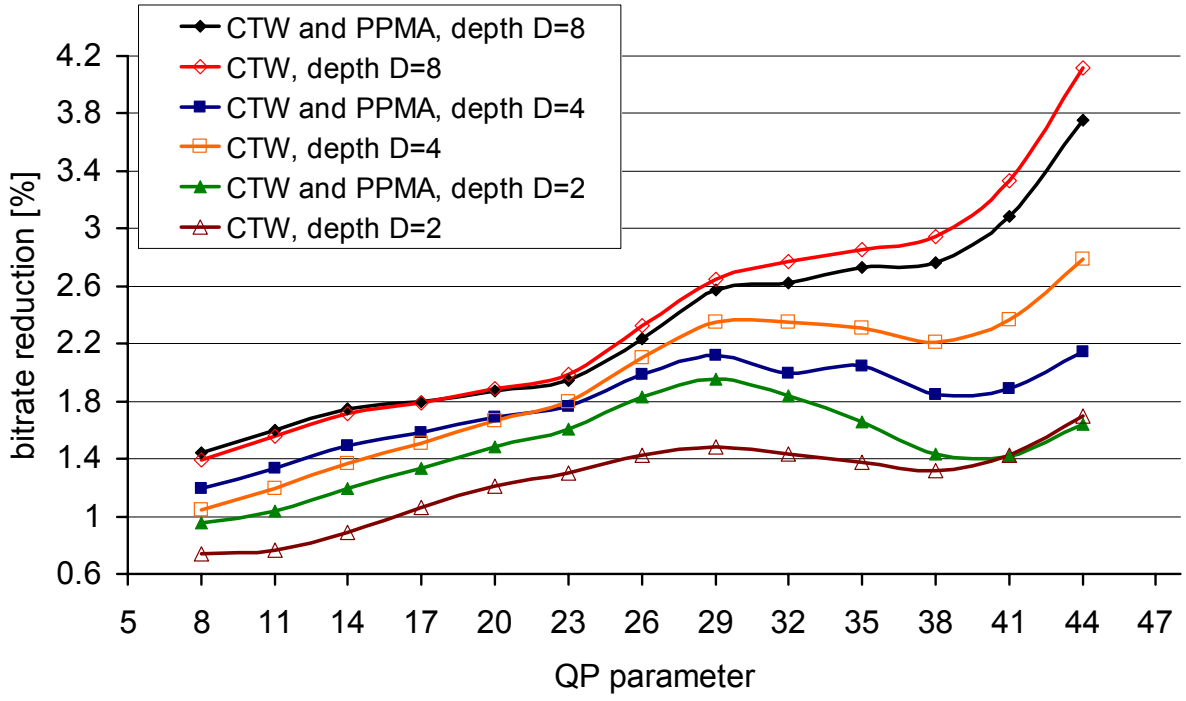
Results for B frames							
		averaged bitrate after using CABAC and CTW+PPMA for different depths D of context trees [Mbits/s]			CABAC with CTW+PPMA gain relative to original CABAC [%]		
QP parameter	averaged bitrate for CABAC [Mbits/s]	D=2	D=4	D=8	D=2	D=4	D=8
8	46.4451	46.0015	45.8907	45.7768	0.9551	1.1935	1.4388
11	35.6832	35.3122	35.2055	35.1125	1.0396	1.3386	1.5994
14	25.8443	25.5359	25.4587	25.3927	1.1933	1.4921	1.7473
17	16.3150	16.0971	16.0573	16.0218	1.3360	1.5798	1.7972
20	9.7006	9.5565	9.5368	9.5187	1.4860	1.6888	1.8751
23	5.4216	5.3347	5.3262	5.3161	1.6043	1.7604	1.9462
26	2.7965	2.7453	2.7408	2.7339	1.8303	1.9889	2.2384
29	1.4936	1.4645	1.4619	1.4552	1.9505	2.1199	2.5696
32	0.8452	0.8296	0.8283	0.8230	1.8402	1.9934	2.6205
35	0.5237	0.5150	0.5130	0.5094	1.6593	2.0455	2.7324
38	0.3356	0.3307	0.3294	0.3263	1.4320	1.8462	2.7671
41	0.2348	0.2315	0.2304	0.2276	1.4159	1.8854	3.0820
44	0.1629	0.1603	0.1594	0.1568	1.6372	2.1481	3.7546



(a)



(b)



(c)

Figure 6.11. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I-frames (a), P-frames (b) and B-frames (c). The bitrate reduction is a result of using the modified AVC encoder with CABAC and joint application of CTW and PPMA technique in contrast to the original AVC with unmodified CABAC.

General conclusions from experimental results achieved for the modified AVC with CABAC and CTW and PPMA are the same as it took place for the coding efficiency of the modified AVC with CABAC and CTW and for the modified AVC with CABAC and PPMA. The gain of the coding efficiency of the modified AVC (with CABAC and CTW and PPMA) relative to the efficiency of the original AVC is strictly dependent on the content of the video sequence, on the value of QP parameter and the depth *D* of context trees. These aspects have been exactly discussed in the case of tests for the modified AVC with CTW (Section 6.7.1) and the modified AVC with PPMA (Section 6.7.2) and will not be quoted here again.

The obtained experimental results have clearly showed that the application of both CTW and PPMA techniques in CABAC can additionally improve the coding efficiency of entropy encoder in comparison to the modified CABAC with CTW. The gain of the compression performance strongly depends on the context length *D* that is used to estimate the probability of the successive source symbol. The smaller depth *D* of the context trees the greater difference in the coding efficiency between CABAC with CTW and CABAC with both CTW and PPMA techniques. In the cases of greater values of depth *D* of context trees the difference between the compression performances of two entropy encoders clearly diminishes. In the case of the context trees of depth *D* equal to 8 the efficiency of both CABAC with CTW and PPMA techniques and CABAC with CTW technique are comparable.

The compression performance of the modified AVC with CABAC and joint application of CTW and PPMA is strongly dependent on QP parameter value. As it has been stated earlier, the value of QP parameter determines the size of the data set within a single slice and a single frame. The method of joint application of CTW and PPMA estimates probabilities of coded symbols with taking into consideration the statistics of data that has been already encoded. The statistics of encoded data are stored in the context trees of depth *D*. The accuracy of these statistics has a great influence on values of probabilities estimated with the method of joint application of CTW and PPMA in CABAC because:

- PPMA technique estimates P_{PPMA} probability on the basis of information that is stored in the context trees;
- CTW technique calculates P_{CTW} probability with respect to the data statistics from the context trees;
- After encoding the current symbol, the estimation algorithm checks which one of two probabilities (P_{CTW} or P_{PPMA}) has allowed for obtaining the smallest number of bits in the current context. This additional information is stored in dual context trees that

have been created for each of the probability model defined in CABAC. Based on the information of dual context trees two probabilities, one estimated with CTW technique and second estimated with PPMA technique are appropriately weighted. The result of weighting is strongly dependent on the number of binary symbols that have been stored in dual context trees.

Therefore, the accuracy of the probabilities calculated with the data modeling technique based on CTW and PPMA is strictly dependent on the size of the data set on the basis of which the context trees "learn" the statistics of coded data. In author's implementation, all context trees have been re-initialized to default values each time before an I-slice and a slice of a new type. So, in this experiment all context trees have been re-initialized before each I-frame, P-frame and the first B-frame from a pair. As it has been stated earlier, after the context trees initialization the estimation algorithm begins its working with usually "bad" statistics, so at the beginning the data modeling algorithm is not able to estimate precisely the probabilities of 0 and 1 symbols. It has a great influence on poorer coding efficiency of the modified AVC just after context trees initialization. The accuracy of the estimated probabilities increases with encoding successive source symbols because the data statistics stored in the context trees are updated each time. Unfortunately, the size of data within P- and B-frames is too small and the estimation algorithm is not able to estimate precisely the real statistics of coded data before the next initialization process of the context trees. A consequence of that is poorer and poorer compression performance of the modified AVC for sequences encoded with greater and greater QP parameter values that correspond to smaller and smaller bitrate of encoded video sequence. This problem is especially visible in the case of P-frames, because the context trees have been re-initialized each time before P-frame (see Figure 6.11). In this experiment, the compression performance of the modified AVC video encoder is higher for I- and B-frames because of the following facts:

- The size of the data that represents I-frames is usually several times greater than the size of the data of P- or B-frames. So, the data modeling algorithm has sufficient amount of data to estimate precisely the real statistics of coded symbols. In this case, the influence of applied context trees initialization method on the compression performance of the modified AVC is considerably limited (see Figure 6.11);
- B-frames have been grouped in pairs, so in author's implementation by encoding of the second B-frame (from the pair) the data statistics in the first B-frame has been

used. It has significantly improved the compression performance of each second B-frame from pairs. So, the total coding efficiency for B-frames has been also improved.

The experimental results have clearly showed that the difference between the coding efficiency of the modified AVC with CABAC and joint application of CTW and PPMA and the coding efficiency of the modified AVC with CABAC and CTW visibly decreases with the increasing of QP parameter value. Above a certain value QP_t of QP parameter, the efficiency of CTW and PPMA starts to be worse in comparison to the efficiency of CTW. In author's opinion it results directly from the idea of CTW and the idea of CTW and PPMA. In the case of CTW technique only one probability P_{CTW}, which accuracy depends on the statistics stored in the context trees, is estimated. In the case of the method of joint application of CTW and PPMA three probabilities are estimated: P_{CTW}, P_{PPMA} and a mixed probability $P_{CTW+PPMA}$ whose accuracy also depends on the content of the context tree. So, the influence of incorrect data statistics saved in the context trees on estimated probabilities is far greater in the case of method of joint application of CTW and PPMA technique. The way of weighting of probabilities estimated with CTW and PPMA techniques is especially of great importance here, and in the case of wrong data statistics in the context trees these two probabilities are improperly mixing, which results with "bad" probability.

The value of QP_t is strictly dependent on used depth D of the context trees. The experimental results proved that the greater depth D of the context trees the smaller value of QP_t from which the coding efficiency of AVC with CABAC and joint application of CTW and PPMA starts to be worse than the coding efficiency of the modified AVC with CABAC and CTW.

6.8. Influence of algorithm of contexts initialization on compression performance of entropy encoders

The algorithm of contexts initialization and the frequency of resetting of contexts to the default values influence on the compression performance of adaptive entropy encoders. The original AVC with CABAC and the modified AVC encoders have been working with different contexts initialization methods. What is more, the frequency of contexts initialization is different for the original and the modified AVC video encoders. The interesting question is how it influences on the coding efficiency of both original and modified AVC encoders.

6.8.1. Influence of the frequency of contexts initialization on the coding efficiency of entropy coders

The original CABAC algorithm performs the contexts initialization each time before a new slice. The modified CABAC coders (with CTW and/or PPMA) take advantage of long inter slices and the context trees initialization is performed each time before an I-slice and a slice of a new type. Thus, when the slices of the same type occur one after another in GOP, data statistics of the previous slices are used in the successive slices. As a matter of fact the original CABAC also takes into consideration the statistics of the previous slice in contexts initialization for the successive slice (in the so-called adaptive contexts initialization), but data statistics of the previous slice are not used explicitly in CABAC.

In order to investigate how the application of long inter slices influences on the compression performance of entropy encoder, a slightly modified version of the original CABAC has been prepared. In this version of CABAC, the algorithm of the contexts initialization has been changed and the contexts are reset to default values in really the same moments as it takes place in the modified CABAC with CTW and/or PPMA. Thus, the modified version of the original CABAC also takes advantage of long inter slices. The coding efficiency of CABAC with modified algorithm of contexts initialization has been tested and confronted with the coding efficiency of the original CABAC and with the modified CABAC with CTW. Experiments have been done according to **Scenario 1** (see Section 6.6).

The averaged experimental results obtained for the test sequences for P-frames have been presented in Figure 6.12.

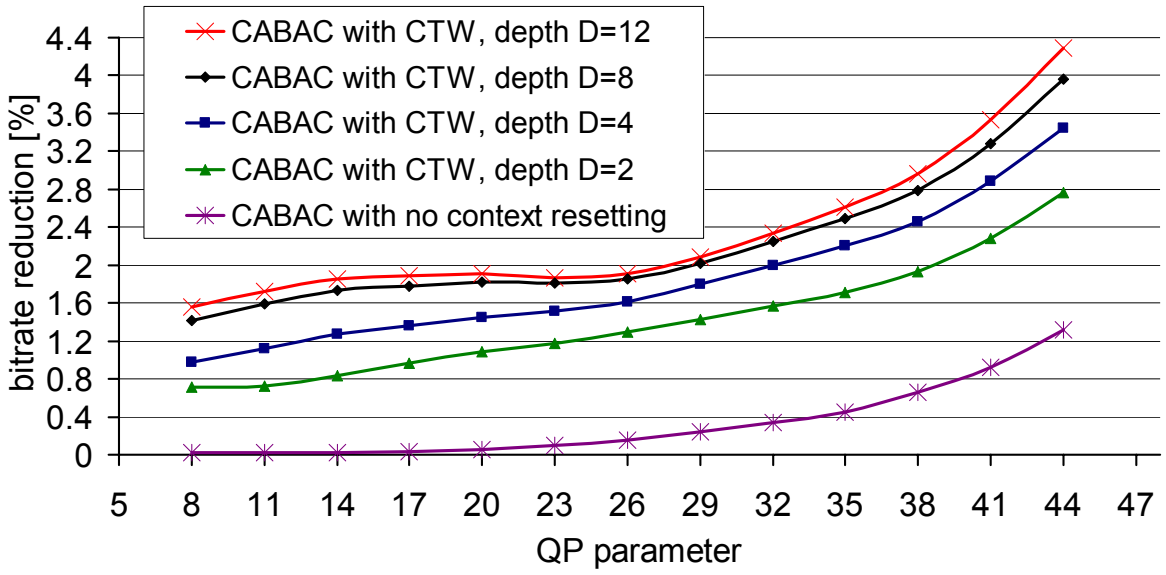


Figure 6.12. Influence of the frequency of the contexts initialization on the compression performance of CABAC entropy encoder for P-frames.

The obtained experimental results clearly show that the direct exploiting of data statistics gathered in the previous slices improves the compression performance of entropy encoder for the successive slices. In experiments CABAC with modified contexts initialization leads to 0%-1.3% bitrate reduction in comparison to the original CABAC. Nevertheless, the coding efficiency of CABAC with modified contexts initialization is clearly lower that the compression performance of the modified CABAC with CTW for all considered depths *D* of context trees. It means that better compression performance of the modified CABAC encoders results mainly from the fact of application of sophisticated techniques of data statistics estimation in CABAC and not only the idea of long inter slices. It must be emphasized that better compression performance of the modified CABAC with CTW (relative to the original CABAC and the original CABAC with modified contexts initialization) has been obtained even with much simpler technique of context trees initialization in comparison to the original CABAC algorithm.

6.8.2. Influence of the more sophisticated method of context trees initialization on compression performance of the modified CABAC with CTW

A very simple method of context trees initialization has been used in the modified CABAC with CTW and/or PPMA. It surely influences on the compression performance of

entropy encoders. In order to test this influence, the advanced mechanism of contexts initialization from CABAC has been adopted to the modified CABAC with CTW. The slice- and QP-dependent contexts initialization of CABAC sets the initial probabilities for 0 and 1 symbols for each of 399 defined contexts. The author has modified the simple method of context trees initialization in the modified CABAC with CTW in the way that the counters of the number of zeros a_s and the number of ones b_s in roots of 399 context trees are set to values that allow for obtaining of CABAC initial probabilities for 0 and 1 symbols. The counters of remaining nodes s of all 399 context trees are initialized to zero.

The compression performance of the modified CABAC with CTW and the more sophisticated method of context trees initialization has been tested and confronted to the coding efficiency of the modified CABAC with CTW and simple method of context trees initialization. Experiments have been done according to **Scenario 3** (see Section 6.6). The depth $D = 8$ of the context trees have been used in the modified CABAC with CTW and advanced context trees initialization method. For the reason of assumed structure of GOP and the features of I-, P-, and B-slices the method of context trees initialization mostly influences on the coding efficiency of entropy encoder for P-frames. The averaged experimental results obtained for P-frames for four test sequences have been presented in Figure 6.13.

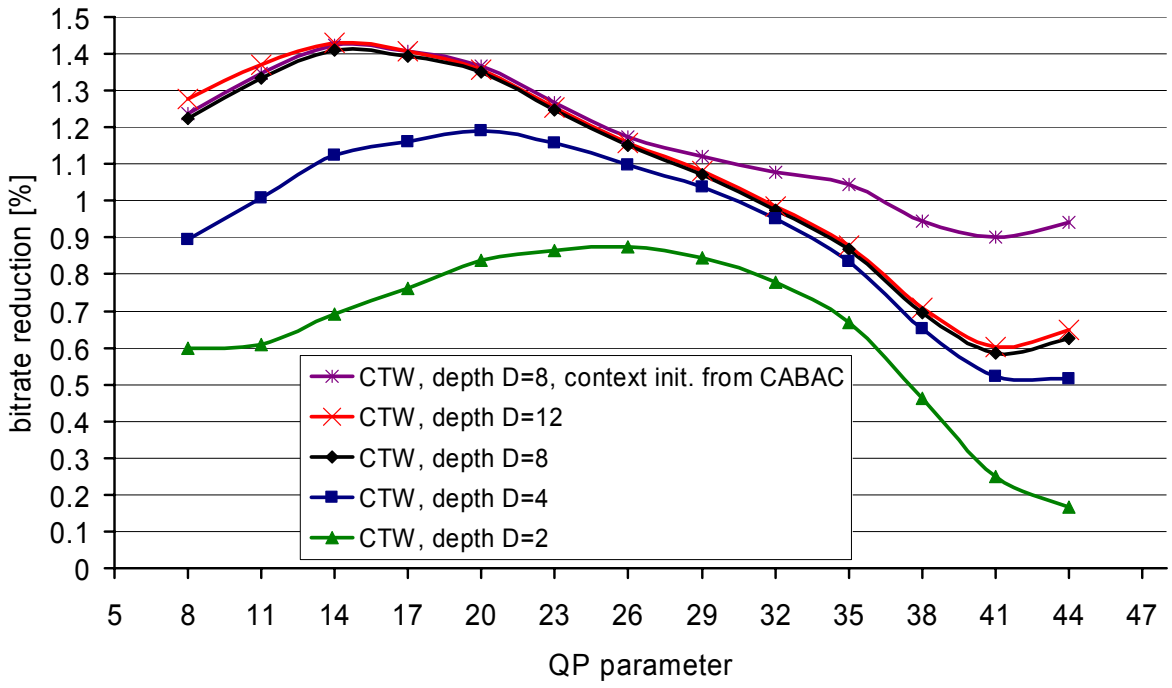


Figure 6.13. Influence of the method of context trees initialization on the compression performance of the modified CABAC with CTW. The experimental results concern the P-frames.

Experimental results clearly present that the application of the more sophisticated method of context trees initialization leads to the increase of the coding efficiency of the entropy encoder. Significant increase of the compression performance has been observed for lower transmission bitrates that correspond to higher values of QP parameter. For lower transmission bitrates the modified entropy encoder is not able to adjust to the current signal statistics accurately for the reason of small data set in P-slice. Experiments unambiguously showed a great significance of the method of contexts initialization on the coding efficiency of advanced entropy encoders in the case of lower bitrates.

6.9. Conclusions

The author has proposed the original method of application of sophisticated techniques of the conditional probabilities estimation of symbols within the state-of-the-art CABAC entropy coder [Marp03a] that works within Advanced Video Coder AVC [AVC]. The proposed techniques of the data statistics estimation have been based on Context-Tree Weighting [Will95, Will98a] and/or Prediction with Partial Matching [Clear84]. Moreover, the author has built three modified AVC codecs that use the proposed techniques of data statistics estimation and in series of experiments the author has tested their coding efficiency.

Application of the more accurate techniques of the data statistics estimation in advanced adaptive arithmetic coders leads to a reasonable increase of the compression performance of the contemporary advanced video encoders. The author's experimental results have unambiguously proved that the modified AVC (with CABAC that exploits CTW or CTW and PPMA) clearly outperforms the original AVC with CABAC. Bitrate reduction of 1.5% to 4.6% has been obtained. According to experimental results, the compression performance of the modified AVC with CABAC and PPMA is poorer in comparison to the coding efficiency of other modified AVC encoders. Application of PPMA within CABAC can decrease the bitrate by 0.5% to 2.4%. However, in some conditions (see Section 6.7.2) the application of PPMA technique within CABAC can increase the size of bitstream even by 2.5% in comparison to the original CABAC. In the three modified AVC encoders, their compression performance is strictly dependent on:

- The value of QP parameter that determines the size of encoded bitstream and the quality of decoded video sequence;

- The depth *D* of the context trees that are used to estimate the conditional probabilities of coded symbols;
- The content of the video sequence that influences the statistics of data coded with entropy encoder.

The experimental results have also proved that the algorithm of the context trees initialization is of great importance on the compression performance of the modified AVC video encoders. The coding efficiency of CABAC entropy encoder with CTW and/or PPMA may be additionally increased if more sophisticated technique of the context trees initialization is used.

Chapter 7

Impact of arithmetic encoder core on compression performance

7.1. Arithmetic encoder cores

The core of the binary arithmetic codec used in CABAC (M-codec, also called modulo-codec) has been highly optimized for speed. In order to do that, M-codec has been adopted to work properly with a limited set of only 128 predefined quantized values of probabilities [Marp03a, Marp03b]. In the modified AVC video codecs (those proposed by the author in Chapter 6) more sophisticated techniques of conditional probabilities estimation have been used. In general, these techniques produce values of conditional probabilities in a significantly greater set of numbers as compared to that defined in CABAC. Therefore, in the modified AVC video codecs the core of M-codec from CABAC has been replaced with a traditional multiplication- and division-based arithmetic codec core that is able to work with values of probabilities from a larger set of numbers. In the experimental implementation of the modified AVC video codecs, the *m*-ary arithmetic codec core has been used. In fact, in the experimental implementation, the *m*-ary arithmetic codec core was the same as defined for H.263 video coding standard [H263]. In the modified AVC video codecs, this *m*-ary arithmetic codec core works as a binary arithmetic codec.

7.2. The problem

The original AVC with CABAC and the modified AVC video codecs work with different cores of arithmetic codec. It may obviously influence the compression performance of the respective entropy encoders. There arises a question about the impact of the applied core of arithmetic codec on the coding efficiency of the whole entropy encoder. In order to unambiguously answer this question the author has done a set of experiments in which the coding efficiency of both M-codec core and H.263 arithmetic codec core has been compared.

Unfortunately, this question has not been clearly answered in the references. Some tests have been done in cause of AVC standardization activities. There is known the general conclusion "… the M-coder provides virtually the same coding efficiency as a conventional multiplication- and division-based implementation of binary arithmetic coding …" [Marp06a]. But the references did not compare directly the coding efficiency of the M-coder relative to the efficiency of H.263 arithmetic coder.

7.3. Test platform for coding efficiency of arithmetic codec cores

In order to investigate the influence of applied arithmetic codec core on the compression performance of the modified AVC video codecs, the coding efficiency of two versions of AVC video encoder with CABAC has been compared. Both encoders have standard AVC mechanisms of probability estimation and initialization. These are:

- The original AVC encoder with unmodified CABAC that works with M-codec core;
- The modified AVC encoder with unmodified CABAC that works with H.263 arithmetic codec core.

Thus, both original and modified AVC encoders only differ from the point of the core of arithmetic codec. In the modified AVC with CABAC and H.263 arithmetic codec core, the conditional probabilities determined by the finite-state machine originally used in CABAC have been fed to the H.263 arithmetic codec core. In this way, both M-codec core and H.263 arithmetic codec core have been working with the same values of the conditional probabilities of coded symbols. The software for the original AVC codec was the publicly available version JM 10.2 [AVCSoft] of the Advanced Video Codec (AVC) implementation. The software for the modified AVC codec with H.263 arithmetic codec core has been prepared by

the author by embedding the H.263 arithmetic codec core into the reference AVC implementation version JM 10.2. In order to avoid influence of programming bugs, both encoder and decoder have been implemented and carefully tested before experiments.

There has been done a comparison of the coding efficiency of the M-codec core and the H.263 arithmetic codec core with several test sequences. The difference in the coding efficiency between M-codec core and H.263 arithmetic codec core has been expressed as a percentage bitrate reduction calculated with the following formula:

$$\text{bitrate reduction}[\%] = \left(1 - \frac{\text{bitstream size}(\text{H.263 arithmetic codec core})}{\text{bitstream size}(\text{modulo} - \text{codec core})}\right) \cdot 100\%, \qquad (7.1)$$

where:

$\text{bitstream size}(\text{H.263 arithmetic codec core})$ - size of bitstream obtained for AVC encoder with CABAC that works with H.263 arithmetic encoder core.

$\text{bitstream size}(\text{modulo - codec core})$ - size of bitstream obtained for AVC encoder with CABAC that works with the original M-encoder core.

7.4. Experimental results on coding efficiency of arithmetic codec cores

The compression performance of both M-codec core and H.263 arithmetic codec core has been tested in the following conditions:

- CITY, CREW, ICE and HARBOUR test sequences in 4CIF format have been used;
- The experiments have been done for both intra and inter prediction modes by setting the structure of GOP on I29P;
- Tests have been done for a wide range of QP parameter that corresponds to the range from excellent to bad subjective quality of video.

The detailed experimental results achieved for individual test sequences have been presented in Annex D. In Figure 7.1 and Figure 7.2, and Table 7.1 and Table 7.2 the averaged experimental results obtained for CITY, CREW, ICE and HARBOUR test sequences have been presented for I-frames and P-frames respectively.

Table 7.1. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I-frames only. The bitrate reduction is a result of application in CABAC the H.263 arithmetic codec core instead of the M-codec core.

QP parameter	bitrate for CABAC with M-codec core [Mbits/s]	bitrate for CABAC with H.263 arithmetic codec core [Mbits/s]	bitrate reduction due to application H.263 arithmetic codec core [%] as defined in Eq. 7.1
8	2.946675	2.944493	0.0741
11	2.359655	2.357753	0.0806
14	1.892825	1.891238	0.0839
17	1.437100	1.435853	0.0868
20	1.101658	1.100698	0.0871
23	0.807648	0.806953	0.0861
26	0.596465	0.595948	0.0868
29	0.432785	0.432428	0.0826
32	0.317090	0.316840	0.0788
35	0.226795	0.226658	0.0606
38	0.159418	0.159348	0.0439
41	0.110303	0.110293	0.0091
44	0.074658	0.074693	-0.0469

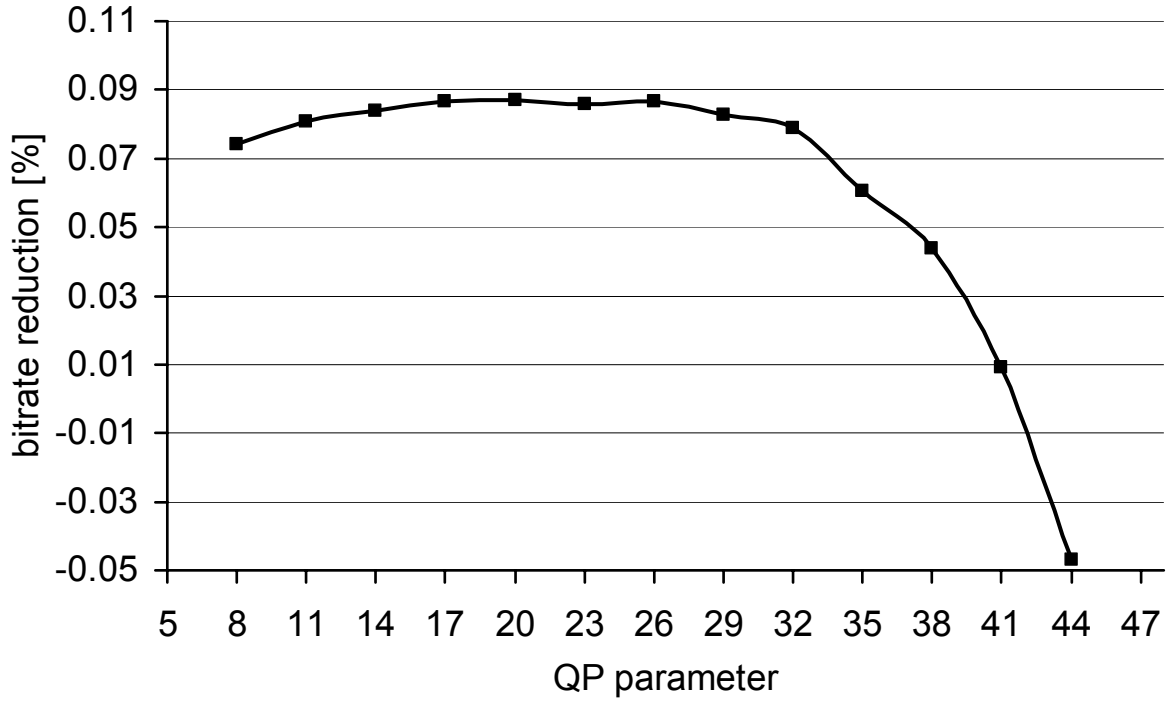


Figure 7.1. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for I-frames only. The presented bitrate reduction is a result of application in CABAC the H.263 arithmetic codec core instead of the M-codec core.

Table 7.2. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for P-frames only. The bitrate reduction is a result of application in CABAC the H.263 arithmetic codec core instead of the M-codec core.

QP parameter	bitrate for CABAC with M-codec core [Mbits/s]	bitrate for CABAC with H.263 arithmetic codec core [Mbits/s]	bitrate reduction due to application H.263 arithmetic codec core [%] as defined in Eq. 7.1
8	69.116600	69.059105	0.0832
11	53.316830	53.272720	0.0827
14	38.617665	38.585875	0.0823
17	25.125410	25.105213	0.0804
20	15.366410	15.355005	0.0742
23	8.826533	8.820478	0.0686
26	4.818950	4.816378	0.0534
29	2.738960	2.738150	0.0296
32	1.607658	1.607855	-0.0123
35	1.000045	1.000815	-0.0770
38	0.641118	0.642245	-0.1759
41	0.455470	0.456823	-0.2969
44	0.349820	0.351318	-0.4281

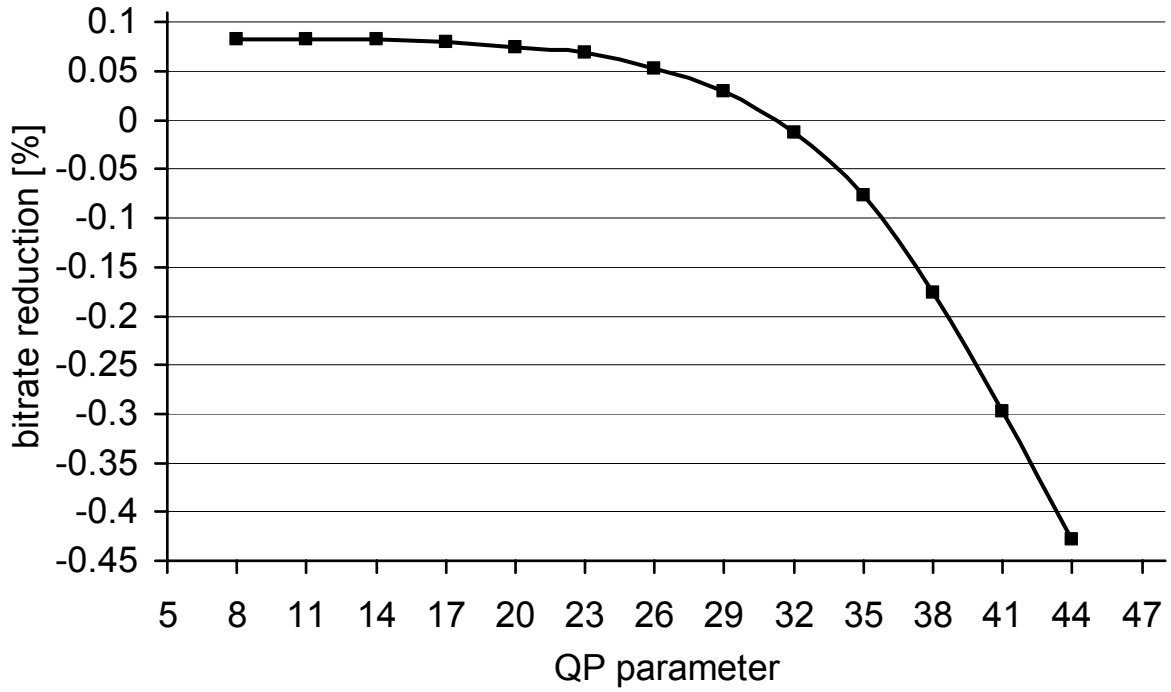


Figure 7.2. Averaged bitrate reduction achieved for CITY, CREW, ICE and HARBOUR 4CIF test sequences for P-frames only. The presented bitrate reduction is a result of application in CABAC the H.263 arithmetic codec core instead of the M-codec core.

The obtained experimental results have proved that application of H.263 arithmetic codec core within CABAC only marginally influences the coding efficiency of entropy encoder. For a wide range of values of QP parameter, H.263 arithmetic codec core insignificantly outperforms the fast binary M-codec core. For both I- and P-frames the maximum bitrate reduction after using H.263 arithmetic codec core is below 0.1% for a wide range of QP parameter values. For high values of QP parameter, CABAC with M-codec core is characterized by even higher coding efficiency in comparison to CABAC with H.263 arithmetic codec core. Greater differences of the compression performance between two tested entropy encoders have been observed for P-frames in the case of lower bitrates. The presented experimental results well correspond to those from [Marp06a], where it is said that the coding efficiency of M-codec core (from CABAC) and a traditional arithmetic codec core with multiplication and division operations are virtually the same.

The difference in the compression performance between the two tested cores of arithmetic codec results mainly from two facts. Firstly, the considered arithmetic codec cores differ among themselves in the field of precision of the registers that are used for storing of state of the arithmetic codec core. The M-encoder core uses 10-bits and-9 bits registers to store the information about the lower endpoint L of the current interval and the range R of the current interval respectively [Marp03a, Marp03b]. H.263 arithmetic encoder core uses 16-bits registers to store the information about lower as well as higher endpoints of the current interval [H263]. Secondly, in the procedure of the current interval subdivision the M-codec core makes firstly an approximation of the current interval range R by quantizing it to a limited set Q of $K = 4$ quantized range values. The quantized interval range $Q(R)$ and the probability state index σ are finally used by the M-codec core in determining the new interval. It is obvious that the quantization process of the current interval influences the range R of resulted interval. Therefore, taking apart the fact of different precision for registers used in the M-codec core and H.263 arithmetic codec core, both arithmetic codec cores may not work identically even with the same values of the conditional probabilities in inputs.

7.5. Conclusions

Application of different arithmetic codec core in the modified AVC encoders relative to the original AVC (M-codec core and H.263 arithmetic codec core) only marginally influences the coding efficiency of entropy encoders in the modified AVC video encoders.

For a wide range of tested bitrates the gain of the compression performance of entropy encoder after using H.263 arithmetic codec core is below 0.1%. The experimental results on the coding efficiency of two tested cores of arithmetic encoder have unambiguously confirmed that better compression performance of the modified AVC video encoders in comparison to the coding efficiency of the original AVC is only a result of using of more sophisticated techniques of the conditional probabilities estimation in CABAC.

Chapter 8

Complexity of advanced adaptation techniques in arithmetic coding

8.1. The goal of the work

The higher compression performance of the modified AVC video encoders with CTW and/or PPMA relative to the original AVC is burdened with higher complexity of both modified encoder and modified decoder in comparison to the original AVC video codec. The higher complexity of the modified AVC video codec is a result of application of more sophisticated data modeling techniques in CABAC. The secondary goal of the work is to test the relationship between the improvement of the compression performance and the increase of complexity of entropy encoder and entropy decoder. In order to test the influence of application of the more accurate techniques of data statistics estimation on complexity of entropy codec, respective experiments have been done.

8.2. Methodology

In the dissertation, complexity of the modified CABAC codecs is measured by the effort of the processor. Results are referred to complexity of the original CABAC codec measured in the same way. The author knows that results of such experiment strongly depend on program implementation and the processor architecture. Nevertheless, the experiment is an attempt of estimation of the modified CABAC codec complexity.

For the reason of poor coding efficiency of CABAC with PPMA relative to the compression performance of CABAC with CTW, only the complexity of CABAC with CTW entropy codec has been measured. Its complexity has been referenced to the complexity of the original CABAC.

Tests have been done under the following conditions:

- CITY and CREW video sequences in 4CIF format have been used. Complexity of entropy codecs mainly depends on the target bitrate and less on the content of video sequence. Therefore, the set of two video sequences is sufficient in complexity experiments;
- The I29P structure of GOP has been considered. Thus, experiments have been done for intra- and inter-prediction modes;
- Experiments have been done for a wide range of QP parameter values.

Experiments have been done for both entropy encoder and entropy decoder. The total entropy encoding times for the modified CABAC with CTW and the original CABAC have been measured by encoding of 200 frames of each of the test sequences with QP parameter values changing from 11 to 44 with step equal to 3. Total entropy decoding times for the modified CABAC with CTW and the original CABAC have been measured by decoding of 600 frames of each of the test sequences for QP parameter values changing from 8 to 44 with step equal to 3. In this way, experiments on the complexity of entropy codecs have been done for a wide range of bitrates from excellent subjective quality (QP=8 or QP=11) to very poor subjective quality (QP=44). The complexity of the modified CABAC entropy codec with CTW has been measured for various depths D of context trees. Implementations of the original AVC codec with unmodified CABAC and the modified AVC codec with CABAC that uses CTW have been used. Implementations of both video codecs have been based on the JM 10.2 reference software of AVC video codec [AVCSoft]. The total encoding/decoding times for the modified and the original entropy codecs have been measured with *QuerryPerformanceCounter*() function described in Section 4.3.2. Experiments have been done on Intel Core 2 Duo E6600 platform (2.4 GHz, 4MB of Level 2 memory cache) with 2 GB of RAM under 32-bit Windows XP with Service Pack 2 operation system. The optimized for speed the modified AVC and the original AVC video codec have been prepared from source code with Intel C++ Compiler (in version 10.0.025) for 32-bit Intel Architecture (IA-32) of microprocessors.

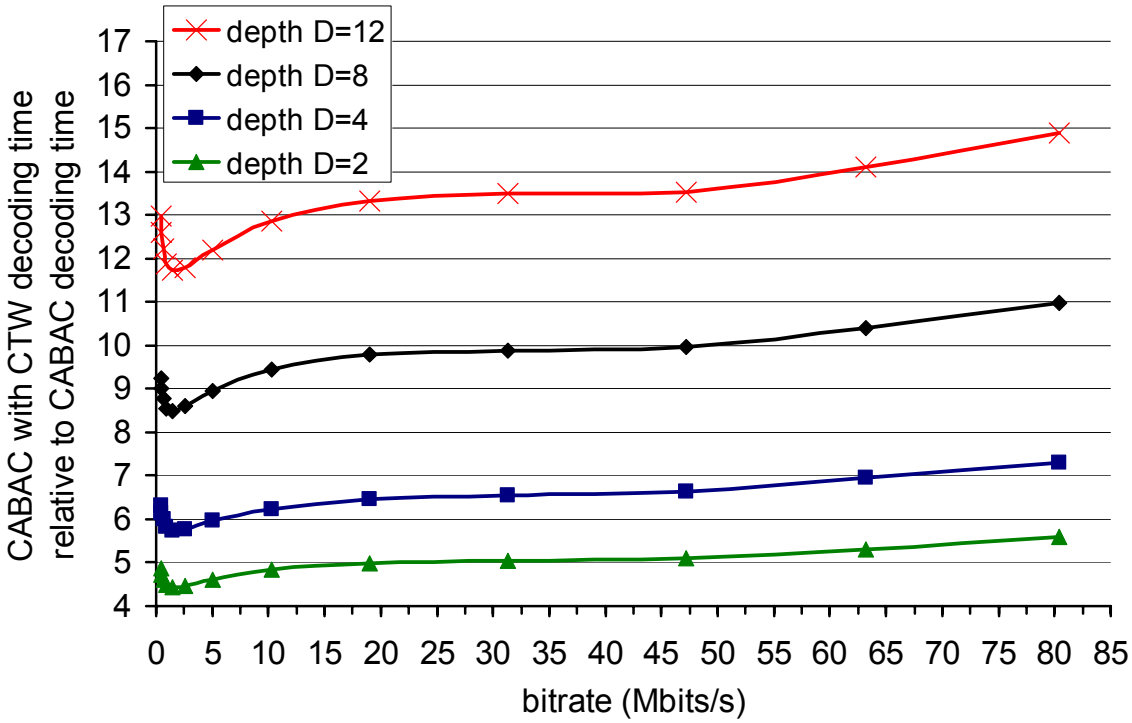
8.3. Experimental results on the complexity of entropy codecs

The experimental results on the increase of the total decoding time of the modified entropy decoder (CABAC with CTW) relative to the total decoding time of the original entropy decoder (CABAC algorithm) have been presented in Figure 8.1. Analogous experimental results for the modified and the original entropy encoders have been presented in Figure 8.2.

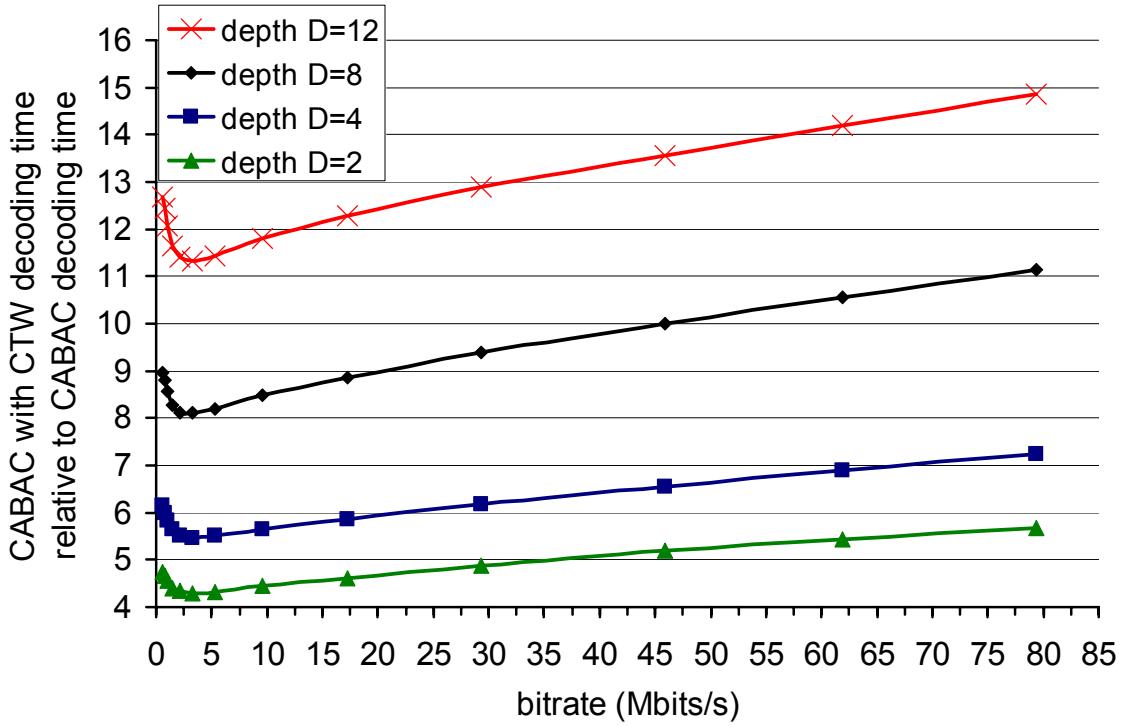
Additionally, the detailed experimental results have been gathered in Table 8.1 and Table 8.2.

Table 8.1. Increase of the total decoding time of CABAC with CTW and H.263 arithmetic decoder core relative to the total decoding time of the original CABAC with M-codec core.

		entropy decoding times [processor ticks]					increase of CABAC decoding time due to application of CTW (for depth D of CTW)			
			CABAC with CTW and H.263 arithmetic decoder core for different depths D of CTW							
QP parameter	bitrate for CABAC [Mbits/s]	CABAC with M-codec core	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
8	80.3902	9.1391E+10	5.0986E+11	6.6850E+11	1.0025E+12	1.3600E+12	5.5790	7.3148	10.9698	14.8812
11	63.1403	7.5966E+10	4.0365E+11	5.2752E+11	7.9020E+11	1.0709E+12	5.3136	6.9442	10.4021	14.0968
14	47.1895	5.9757E+10	3.0554E+11	3.9641E+11	5.9481E+11	8.0840E+11	5.1130	6.6336	9.9538	13.5280
17	31.3045	4.1352E+10	2.0900E+11	2.7109E+11	4.0897E+11	5.5756E+11	5.0543	6.5555	9.8899	13.4833
20	18.9680	2.6296E+10	1.3122E+11	1.7003E+11	2.5736E+11	3.5022E+11	4.9901	6.4659	9.7871	13.3185
23	10.2831	1.5323E+10	7.4050E+10	9.5617E+10	1.4462E+11	1.9706E+11	4.8327	6.2403	9.4381	12.8604
26	5.0630	8.1347E+09	3.7469E+10	4.8484E+10	7.2768E+10	9.9297E+10	4.6060	5.9602	8.9454	12.2066
29	2.6073	4.4584E+09	1.9856E+10	2.5734E+10	3.8301E+10	5.2606E+10	4.4536	5.7720	8.5909	11.7993
32	1.4660	2.6125E+09	1.1558E+10	1.4958E+10	2.2188E+10	3.0638E+10	4.4241	5.7255	8.4928	11.7273
35	0.9072	1.6709E+09	7.5100E+09	9.7461E+09	1.4299E+10	1.9862E+10	4.4945	5.8328	8.5576	11.8867
38	0.6172	1.1752E+09	5.4118E+09	7.0440E+09	1.0310E+10	1.4353E+10	4.6051	5.9940	8.7729	12.2138
41	0.4838	9.5141E+08	4.4981E+09	5.8403E+09	8.5700E+09	1.1984E+10	4.7279	6.1385	9.0077	12.5958
44	0.4171	8.2589E+08	4.0103E+09	5.2149E+09	7.6430E+09	1.0714E+10	4.8558	6.3143	9.2543	12.9728
Results for CREW test sequence										
8	79.3321	9.0711E+10	5.1534E+11	6.5752E+11	1.0109E+12	1.3470E+12	5.6811	7.2484	11.1443	14.8493
11	61.8917	7.4898E+10	4.0697E+11	5.1681E+11	7.9019E+11	1.0627E+12	5.4337	6.9002	10.5503	14.1888
14	45.9035	5.8844E+10	3.0509E+11	3.8553E+11	5.8800E+11	7.9810E+11	5.1847	6.5517	9.9925	13.5630
17	29.3435	4.0817E+10	1.9850E+11	2.5190E+11	3.8300E+11	5.2662E+11	4.8631	6.1714	9.3833	12.9019
20	17.2745	2.5784E+10	1.1901E+11	1.5135E+11	2.2855E+11	3.1668E+11	4.6156	5.8697	8.8637	12.2818
23	9.5826	1.5162E+10	6.7496E+10	8.5468E+10	1.2867E+11	1.7886E+11	4.4515	5.6368	8.4859	11.7963
26	5.3341	8.9253E+09	3.8623E+10	4.9106E+10	7.3106E+10	1.0209E+11	4.3274	5.5019	8.1909	11.4386
29	3.2550	5.7442E+09	2.4712E+10	3.1379E+10	4.6543E+10	6.5010E+10	4.3021	5.4626	8.1025	11.3175
32	2.1080	3.8902E+09	1.6855E+10	2.1432E+10	3.1579E+10	4.4370E+10	4.3328	5.5092	8.1176	11.4056
35	1.4310	2.7283E+09	1.2016E+10	1.5408E+10	2.2547E+10	3.1755E+10	4.4043	5.6477	8.2643	11.6393
38	0.9793	1.9093E+09	8.7025E+09	1.1153E+10	1.6340E+10	2.3060E+10	4.5580	5.8413	8.5583	12.0778
41	0.7332	1.4757E+09	6.8834E+09	8.8452E+09	1.3003E+10	1.8372E+10	4.6645	5.9939	8.8112	12.4493
44	0.5744	1.2078E+09	5.7357E+09	7.4224E+09	1.0814E+10	1.5312E+10	4.7487	6.1452	8.9530	12.6773



(a)

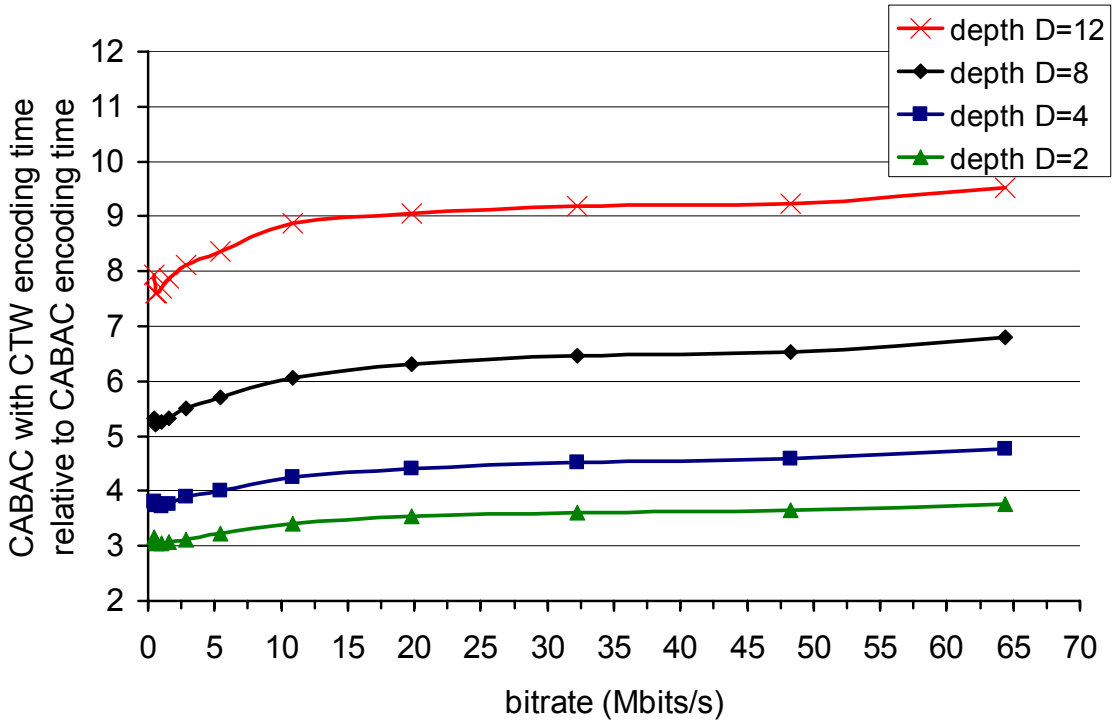


(b)

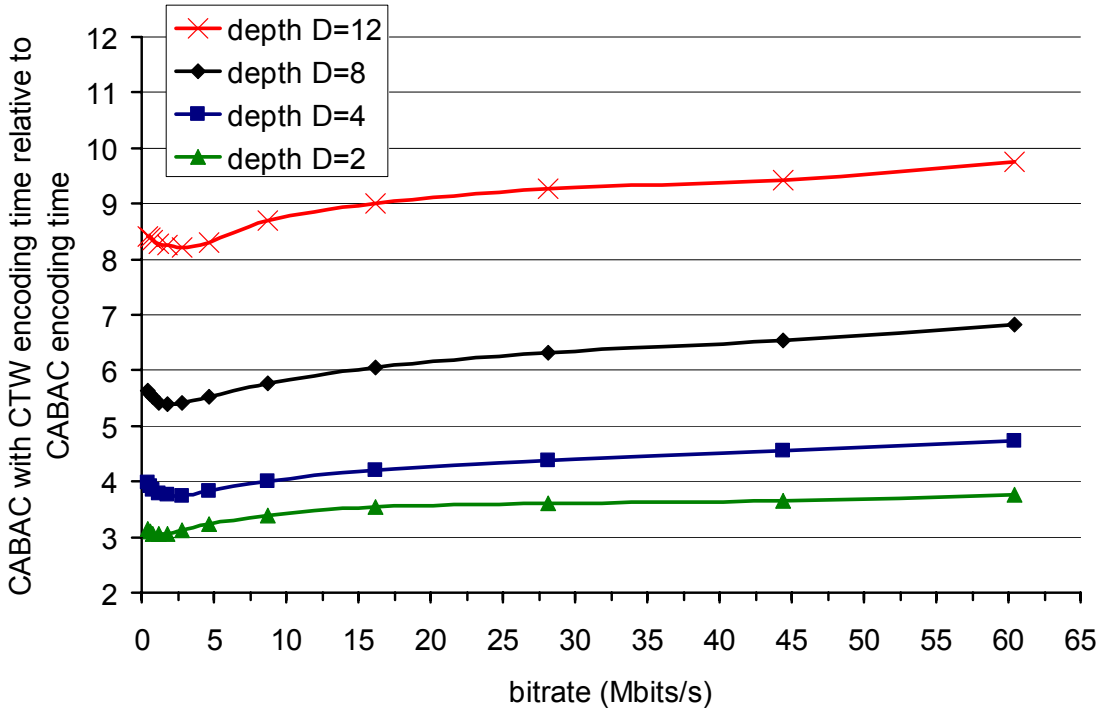
Figure 8.1. Increase of the total decoding time of CABAC with CTW and H.263 arithmetic decoder core relative to the total decoding time of CABAC with M-decoder core within AVC for (a) CITY and (b) CREW test sequences.

Table 8.2. Increase of the total encoding time of CABAC with CTW and H.263 arithmetic encoder core relative to the total encoding time of the original CABAC with M-codec core.

		entropy encoding times [processor ticks]					increase of CABAC encoding time due to application of CTW (for depth D of CTW)			
			CABAC with CTW and H.263 arithmetic encoder core for different depths D of CTW							
QP parameter	bitrate after using CABAC [Mbits/s]	CABAC with M-codec core	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
11	64.3358	2.9741E+10	1.1177E+11	1.4152E+11	2.0201E+11	2.8350E+11	3.7581	4.7586	6.7923	9.5322
14	48.2767	2.3323E+10	8.5221E+10	1.0710E+11	1.5253E+11	2.1543E+11	3.6540	4.5921	6.5399	9.2366
17	32.2474	1.6302E+10	5.8726E+10	7.3787E+10	1.0540E+11	1.4962E+11	3.6023	4.5262	6.4651	9.1780
20	19.7570	1.0607E+10	3.7486E+10	4.6877E+10	6.7014E+10	9.6078E+10	3.5340	4.4193	6.3177	9.0577
23	10.8929	6.3937E+09	2.1708E+10	2.7191E+10	3.8750E+10	5.6690E+10	3.3952	4.2528	6.0606	8.8665
26	5.4540	3.5343E+09	1.1417E+10	1.4184E+10	2.0160E+10	2.9584E+10	3.2303	4.0131	5.7040	8.3705
29	2.8382	2.0165E+09	6.3021E+09	7.8534E+09	1.1085E+10	1.6362E+10	3.1252	3.8945	5.4972	8.1139
32	1.6020	1.2444E+09	3.8138E+09	4.6778E+09	6.6350E+09	9.7906E+09	3.0648	3.7591	5.3318	7.8676
35	0.9932	8.3851E+08	2.5636E+09	3.1211E+09	4.4183E+09	6.4506E+09	3.0573	3.7221	5.2692	7.6929
38	0.6690	6.1710E+08	1.8868E+09	2.3080E+09	3.2268E+09	4.6927E+09	3.0575	3.7400	5.2290	7.6044
41	0.5125	5.1526E+08	1.5956E+09	1.9417E+09	2.6847E+09	3.9222E+09	3.0967	3.7684	5.2104	7.6120
44	0.4359	4.6206E+08	1.4571E+09	1.7578E+09	2.4618E+09	3.6700E+09	3.1535	3.8043	5.3279	7.9427
Results for CREW test sequence										
11	60.3912	2.8135E+10	1.0617E+11	1.3320E+11	1.9188E+11	2.7461E+11	3.7737	4.7343	6.8201	9.7604
14	44.4406	2.1777E+10	7.9480E+10	9.9108E+10	1.4246E+11	2.0517E+11	3.6497	4.5510	6.5417	9.4211
17	28.1322	1.4816E+10	5.2053E+10	6.4952E+10	9.3609E+10	1.3741E+11	3.5134	4.3840	6.3182	9.2746
20	16.2148	9.2466E+09	3.1306E+10	3.8819E+10	5.5969E+10	8.3173E+10	3.3856	4.1982	6.0530	8.9950
23	8.7148	5.4336E+09	1.7677E+10	2.1731E+10	3.1362E+10	4.7205E+10	3.2533	3.9993	5.7718	8.6877
26	4.6610	3.2071E+09	1.0036E+10	1.2277E+10	1.7690E+10	2.6599E+10	3.1293	3.8280	5.5157	8.2936
29	2.7675	2.0842E+09	6.4318E+09	7.8047E+09	1.1284E+10	1.7100E+10	3.0860	3.7447	5.4138	8.2043
32	1.7590	1.4341E+09	4.4387E+09	5.3856E+09	7.7281E+09	1.1845E+10	3.0951	3.7553	5.3888	8.2593
35	1.1842	1.0374E+09	3.2560E+09	3.9215E+09	5.6242E+09	8.5971E+09	3.1385	3.7800	5.4212	8.2868
38	0.8009	7.4939E+08	2.3821E+09	2.8782E+09	4.1460E+09	6.2459E+09	3.1787	3.8407	5.5325	8.3346
41	0.5895	5.9437E+08	1.8904E+09	2.3334E+09	3.3038E+09	4.9831E+09	3.1805	3.9258	5.5585	8.3839
44	0.4515	4.9186E+08	1.6065E+09	1.9554E+09	2.7670E+09	4.1335E+09	3.2662	3.9755	5.6256	8.4038



(a)



(b)

Figure 8.2. Increase of the total encoding time of CABAC with CTW and H.263 arithmetic encoder core relative to the total encoding time of CABAC with M-encoder core within AVC for (a) CITY and (b) CREW test sequences.

Experimental results show higher total encoding and total decoding times for the modified CABAC with CTW entropy codec in comparison to the original CABAC entropy codec. The increase of the total encoding and the total decoding times for the modified CABAC with CTW relative to the original CABAC entropy codec results from two facts:

- Relative to the original CABAC, the modified CABAC entropy codec uses more efficient but also algorithmically more complex technique of the conditional probabilities estimation based on CTW;
- The modified CABAC entropy codec with CTW works with more complex multiplication- and division-based core of the arithmetic codec from H.263 video coding standard, whereas the original CABAC entropy codec works with highly optimized for speed M-codec core with no time-consuming multiplication and division operations. It significantly influences complexity of the modified CABAC entropy codec with CTW technique.

Complexity of the modified CABAC entropy codec is strongly dependent on the depth D of the context trees used to data statistics gathering. The depth D of the context trees directly influences on the length of the context path and the number of nodes s in which CTW technique estimates the conditional weighted probabilities. Therefore, the greater depth D of the context trees the greater number of the conditional weighted probabilities that have to be estimated and the higher total encoding and total decoding times for the modified CABAC entropy codec. The content of the video sequence only marginally influences on the total encoding and the total decoding times for entropy codec.

Depending on the depth D of context trees, the modified CABAC decoder is 4 to 15 times slower than the original CABAC decoder and the modified CABAC encoder is 3 to 10 more time-consuming in comparison to the original CABAC encoder. It must be stated again that the modified CABAC entropy codec works with the more algorithmically complex multiplication- and division-based core of arithmetic codec from H.263 video coding standard. It has a significant impact on the complexity of the modified CABAC with CTW.

In comparison to the original CABAC, the increase of the total decoding times for the modified CABAC with CTW entropy decoder is visibly higher than the increase of the total encoding times for the modified CABAC with CTW entropy encoder. The value of the currently encoded symbol x_n is known in the modified entropy encoder, so entropy encoder at once knows how to update data statistics in nodes s on the context path. In contrast to it, entropy decoder does not know the value of the new symbol when estimating the conditional

weighted probabilities for symbol equal to 0 and symbol equal to 1. Therefore, entropy decoder does not know how to update data statistics in nodes *s* on the context path. In author's implementation of the modified CABAC with CTW entropy decoder, data statistics in nodes *s* on the context path are updated in the assumption that the new symbol is equal to 0. Additionally, the entropy decoder updates the data statistics in nodes *s* from the context path in the case when the new symbol is equal to 1. The updated data statistics in the case when the new symbol is equal to 1 are stored in temporal array. If the value of the new symbol decoded with arithmetic decoder is equal to 1 (so, not equal to 0) the context path on the context tree must be overwritten with data statistics stored in temporal array that contains the updated context path for the case when the new symbol is equal to 1. Therefore, the procedure of data statistics updating in the decoder is more time-consuming than data statistics updating in the encoder. It influences on the higher total entropy decoding times in comparison to total entropy encoding times.

8.4. Impact of arithmetic codec core type on the complexity of entropy codec

8.4.1. Problem

Both the original and the modified CABAC entropy codec differ between themselves from the technique of the conditional probabilities estimation and the core of arithmetic codec. Unquestionably, both these elements influence on the complexity of entropy codec. There arises a question about what the influence of the technique of data statistics estimation on the complexity of entropy codec is.

8.4.2. Methodology

In order to test the impact of the application of a more sophisticated technique of conditional probabilities estimation (based on CTW) on the complexity of entropy codec in CABAC, an experimental platform of the original AVC video codec has been prepared. In the experimental platform of the original AVC video codec CABAC entropy codec has been adjusted to work with H.263 arithmetic codec core. The complexity of the modified CABAC with CTW that works with H.263 arithmetic codec core has been compared to the complexity of the original CABAC entropy codec that also works with H.263 arithmetic codec core.

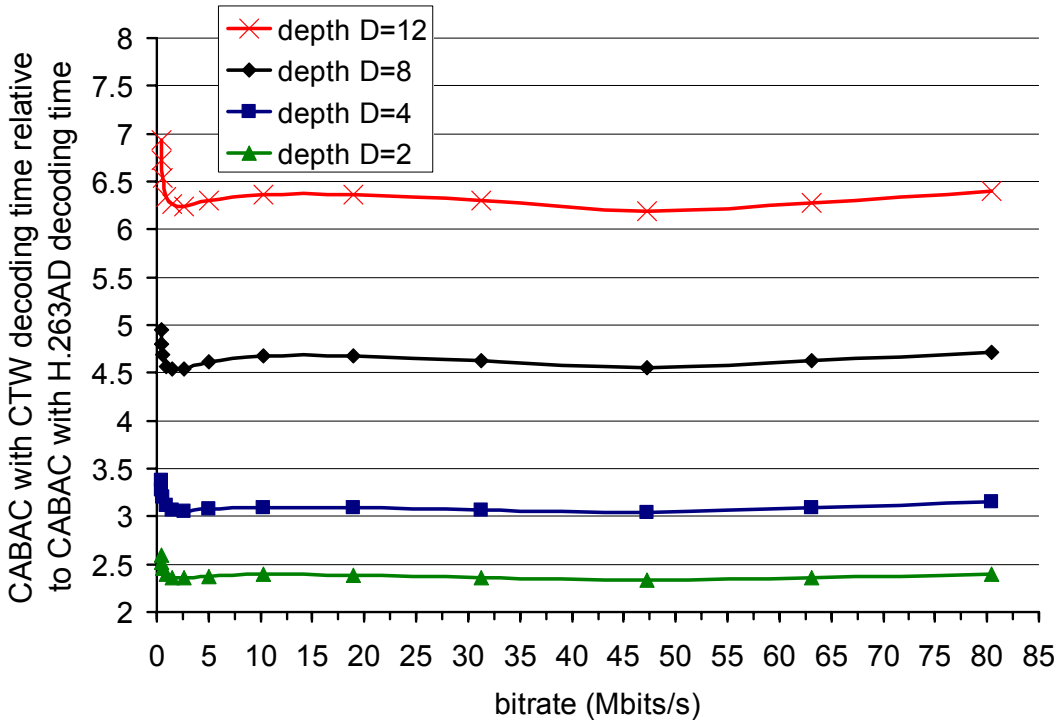
Thus, the only element that is different in the modified CABAC (with CTW) and the experimental platform of the original CABAC is the technique of data statistics gathering. The modified CABAC with CTW and the experimental platform of the original CABAC have been working within the modified and the original AVC video codec respectively. Experiments have been done in the same conditions as presented in the previous section.

8.4.3. Experimental results

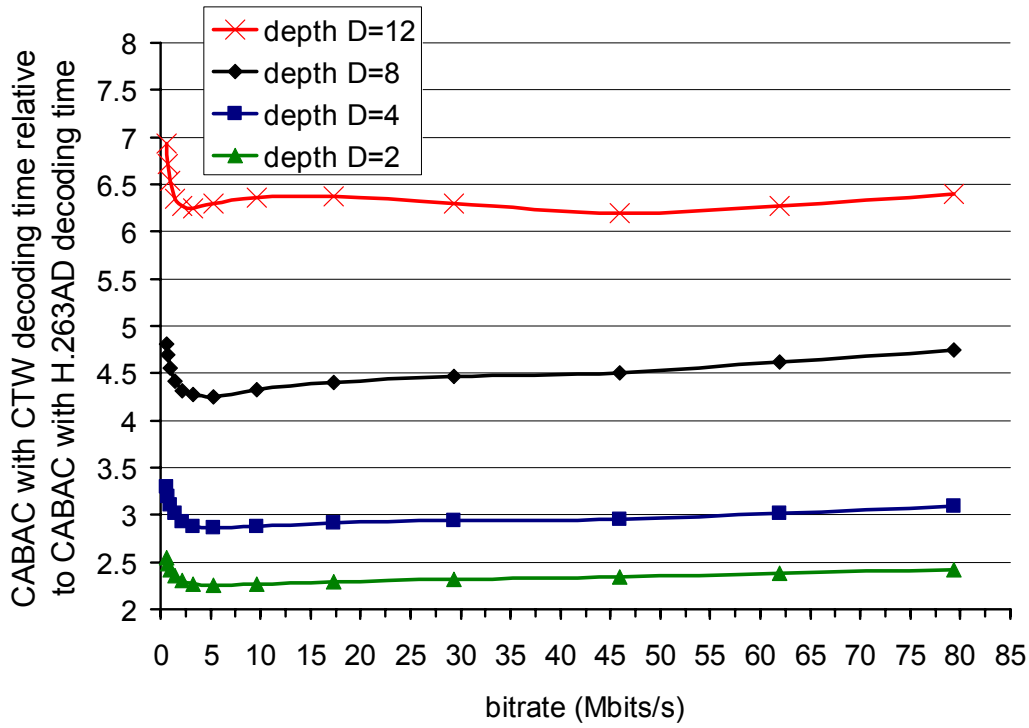
Experimental results on the increase of the total encoding and the total decoding times of the modified entropy codec (CABAC with CTW and H.263 arithmetic codec core) relative to the total encoding and the total decoding times of the original entropy codec (CABAC with H.263 arithmetic codec core) have been presented in Table 8.3 and Table 8.4. Experimental results have been also illustrated in Figure 8.3 and Figure 8.4.

Table 8.3. Increase of the total decoding time of CABAC with CTW relative to the total decoding time of CABAC (with H.263 AD).

		entropy decoding times [processor ticks]								
			CABAC with CTW and H.263 arithmetic decoder core for different depths D of CTW				CABAC with CTW (and H.263 AD) decoding time relative to CABAC (with H.263 AD) decoding time			
QP parameter	bitrate for original CABAC [Mbits/s]	CABAC with H.263 AD core	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
8	80.3902	2.1237E+11	5.0986E+11	6.6850E+11	1.0025E+12	1.3600E+12	2.4009	3.1478	4.7208	6.4040
11	63.1403	1.7076E+11	4.0365E+11	5.2752E+11	7.9020E+11	1.0709E+12	2.3639	3.0893	4.6277	6.2714
14	47.1895	1.3055E+11	3.0554E+11	3.9641E+11	5.9481E+11	8.0840E+11	2.3404	3.0365	4.5563	6.1923
17	31.3045	8.8480E+10	2.0900E+11	2.7109E+11	4.0897E+11	5.5756E+11	2.3622	3.0638	4.6222	6.3016
20	18.9680	5.5002E+10	1.3122E+11	1.7003E+11	2.5736E+11	3.5022E+11	2.3858	3.0913	4.6791	6.3675
23	10.2831	3.0956E+10	7.4050E+10	9.5617E+10	1.4462E+11	1.9706E+11	2.3921	3.0888	4.6717	6.3656
26	5.0630	1.5765E+10	3.7469E+10	4.8484E+10	7.2768E+10	9.9297E+10	2.3767	3.0755	4.6159	6.2986
29	2.6073	8.4245E+09	1.9856E+10	2.5734E+10	3.8301E+10	5.2606E+10	2.3569	3.0546	4.5464	6.2444
32	1.4660	4.8878E+09	1.1558E+10	1.4958E+10	2.2188E+10	3.0638E+10	2.3647	3.0603	4.5394	6.2683
35	0.9072	3.1318E+09	7.5100E+09	9.7461E+09	1.4299E+10	1.9862E+10	2.3980	3.1120	4.5659	6.3420
38	0.6172	2.1958E+09	5.4118E+09	7.0440E+09	1.0310E+10	1.4353E+10	2.4647	3.2080	4.6952	6.5367
41	0.4838	1.7831E+09	4.4981E+09	5.8403E+09	8.5700E+09	1.1984E+10	2.5227	3.2754	4.8063	6.7209
44	0.4171	1.5449E+09	4.0103E+09	5.2149E+09	7.6430E+09	1.0714E+10	2.5959	3.3756	4.9473	6.9352
Results for CREW test sequence										
8	79.3321	2.1299E+11	5.1534E+11	6.5752E+11	1.0109E+12	1.3470E+12	2.4195	3.0870	4.7462	6.3241
11	61.8917	1.7095E+11	4.0697E+11	5.1681E+11	7.9019E+11	1.0627E+12	2.3807	3.0232	4.6225	6.2167
14	45.9035	1.3046E+11	3.0509E+11	3.8553E+11	5.8800E+11	7.9810E+11	2.3385	2.9551	4.5071	6.1175
17	29.3435	8.5703E+10	1.9850E+11	2.5190E+11	3.8300E+11	5.2662E+11	2.3162	2.9392	4.4690	6.1447
20	17.2745	5.1974E+10	1.1901E+11	1.5135E+11	2.2855E+11	3.1668E+11	2.2898	2.9120	4.3973	6.0930
23	9.5826	2.9753E+10	6.7496E+10	8.5468E+10	1.2867E+11	1.7886E+11	2.2685	2.8725	4.3244	6.0114
26	5.3341	1.7177E+10	3.8623E+10	4.9106E+10	7.3106E+10	1.0209E+11	2.2486	2.8589	4.2561	5.9437
29	3.2550	1.0896E+10	2.4712E+10	3.1379E+10	4.6543E+10	6.5010E+10	2.2681	2.8799	4.2717	5.9667
32	2.1080	7.3106E+09	1.6855E+10	2.1432E+10	3.1579E+10	4.4370E+10	2.3056	2.9316	4.3196	6.0692
35	1.4310	5.1098E+09	1.2016E+10	1.5408E+10	2.2547E+10	3.1755E+10	2.3516	3.0154	4.4125	6.2145
38	0.9793	3.5883E+09	8.7025E+09	1.1153E+10	1.6340E+10	2.3060E+10	2.4253	3.1081	4.5538	6.4264
41	0.7332	2.7667E+09	6.8834E+09	8.8452E+09	1.3003E+10	1.8372E+10	2.4880	3.1970	4.6998	6.6403
44	0.5744	2.2479E+09	5.7357E+09	7.4224E+09	1.0814E+10	1.5312E+10	2.5516	3.3019	4.8106	6.8117



(a)

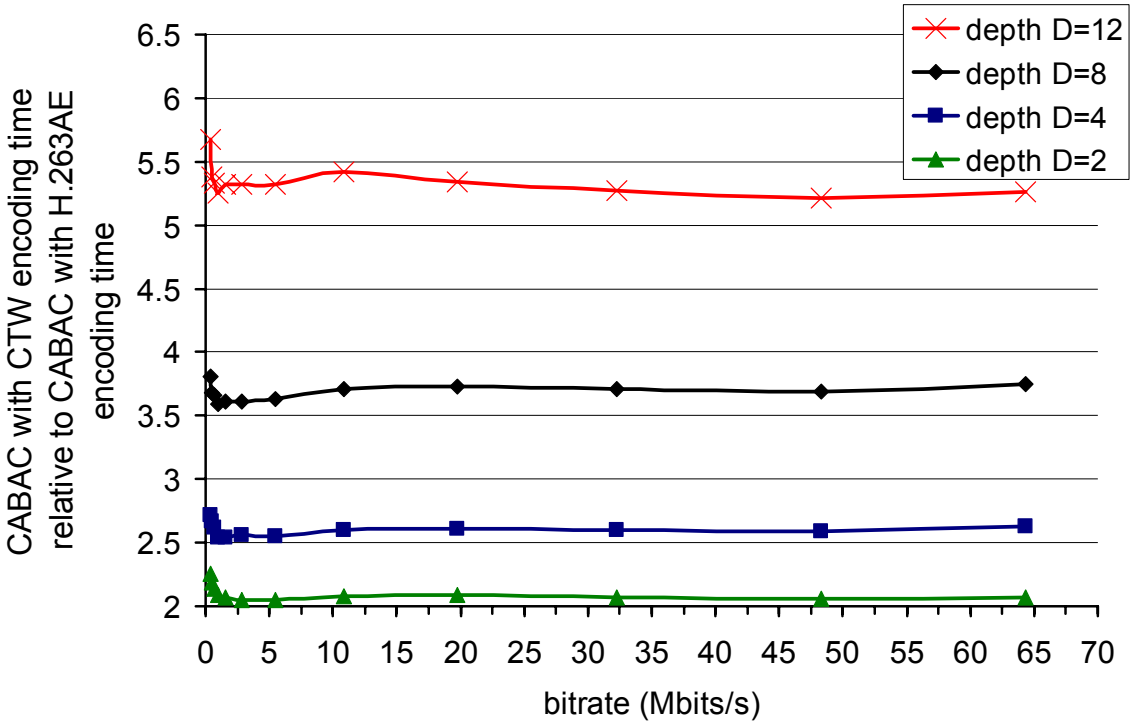


(b)

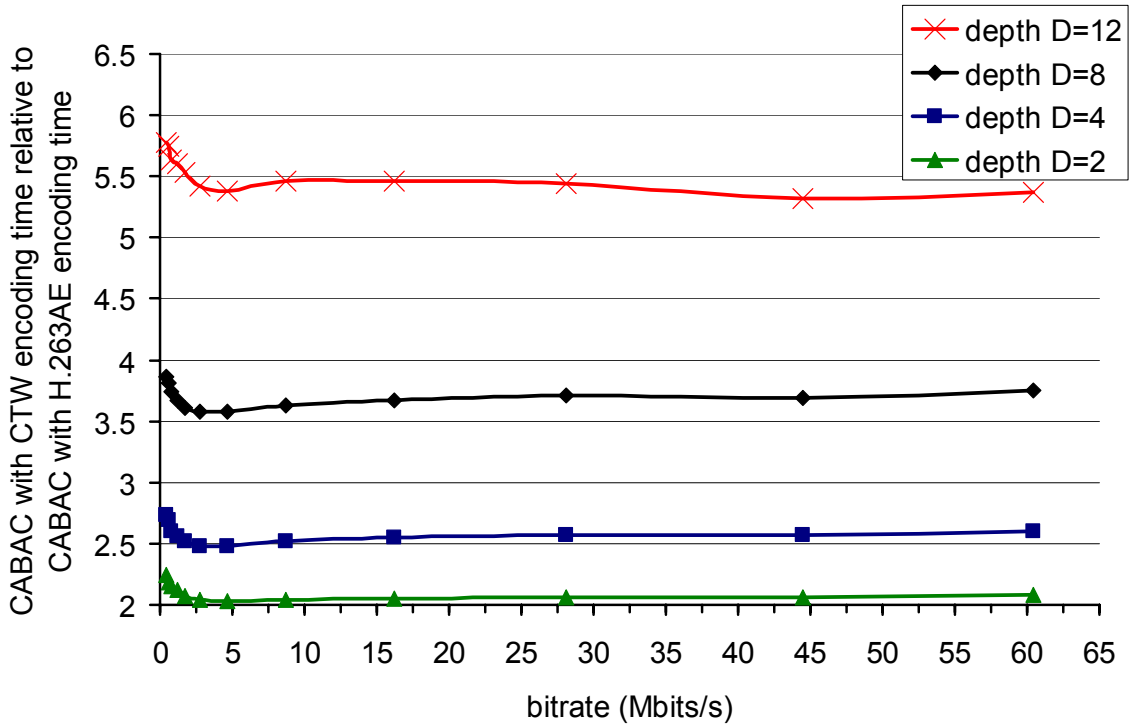
Figure 8.3. Increase of the total decoding time of CABAC with CTW and H.263 arithmetic decoder core relative to the total decoding time of CABAC with H.263 arithmetic decoder core within AVC for (a) CITY and (b) CREW test sequences.

Table 8.4. Increase of the total encoding time of CABAC with CTW relative to the total encoding time of CABAC (with H.263 AE).

		entropy encoding times [processor ticks]					CABAC with CTW (and H.263 AE core) encoding time relative to CABAC (with H.263 AE core) encoding time			
			CABAC with CTW (and H.263 AE core) for different depths D of CTW							
QP parameter	bitrate for original CABAC [Mbits/s]	CABAC with H.263 AE core	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
11	64.3358	5.3917E+10	1.1177E+11	1.4152E+11	2.0201E+11	2.8350E+11	2.0730	2.6248	3.7466	5.2580
14	48.2767	4.1356E+10	8.5221E+10	1.0710E+11	1.5253E+11	2.1543E+11	2.0607	2.5898	3.6882	5.2091
17	32.2474	2.8397E+10	5.8726E+10	7.3787E+10	1.0540E+11	1.4962E+11	2.0680	2.5984	3.7115	5.2689
20	19.7570	1.7984E+10	3.7486E+10	4.6877E+10	6.7014E+10	9.6078E+10	2.0844	2.6066	3.7262	5.3423
23	10.8929	1.0453E+10	2.1708E+10	2.7191E+10	3.8750E+10	5.6690E+10	2.0766	2.6012	3.7069	5.4230
26	5.4540	5.5593E+09	1.1417E+10	1.4184E+10	2.0160E+10	2.9584E+10	2.0537	2.5514	3.6263	5.3216
29	2.8382	3.0734E+09	6.3021E+09	7.8534E+09	1.1085E+10	1.6362E+10	2.0505	2.5553	3.6068	5.3237
32	1.6020	1.8393E+09	3.8138E+09	4.6778E+09	6.6350E+09	9.7906E+09	2.0735	2.5433	3.6074	5.3230
35	0.9932	1.2293E+09	2.5636E+09	3.1211E+09	4.4183E+09	6.4506E+09	2.0854	2.5389	3.5942	5.2475
38	0.6690	8.8092E+08	1.8868E+09	2.3080E+09	3.2268E+09	4.6927E+09	2.1418	2.6200	3.6630	5.3270
41	0.5125	7.2875E+08	1.5956E+09	1.9417E+09	2.6847E+09	3.9222E+09	2.1895	2.6644	3.6840	5.3821
44	0.4359	6.4635E+08	1.4571E+09	1.7578E+09	2.4618E+09	3.6700E+09	2.2543	2.7196	3.8088	5.6781
Results for CREW test sequence										
11	60.3912	5.1117E+10	1.0617E+11	1.3320E+11	1.9188E+11	2.7461E+11	2.0770	2.6058	3.7538	5.3721
14	44.4406	3.8604E+10	7.9480E+10	9.9108E+10	1.4246E+11	2.0517E+11	2.0589	2.5673	3.6903	5.3146
17	28.1322	2.5258E+10	5.2053E+10	6.4952E+10	9.3609E+10	1.3741E+11	2.0608	2.5715	3.7061	5.4401
20	16.2148	1.5236E+10	3.1306E+10	3.8819E+10	5.5969E+10	8.3173E+10	2.0547	2.5479	3.6735	5.4591
23	8.7148	8.6426E+09	1.7677E+10	2.1731E+10	3.1362E+10	4.7205E+10	2.0453	2.5144	3.6288	5.4619
26	4.6610	4.9484E+09	1.0036E+10	1.2277E+10	1.7690E+10	2.6599E+10	2.0282	2.4810	3.5749	5.3753
29	2.7675	3.1527E+09	6.4318E+09	7.8047E+09	1.1284E+10	1.7100E+10	2.0401	2.4756	3.5790	5.4238
32	1.7590	2.1415E+09	4.4387E+09	5.3856E+09	7.7281E+09	1.1845E+10	2.0727	2.5149	3.6088	5.5312
35	1.1842	1.5338E+09	3.2560E+09	3.9215E+09	5.6242E+09	8.5971E+09	2.1228	2.5568	3.6669	5.6052
38	0.8009	1.1077E+09	2.3821E+09	2.8782E+09	4.1460E+09	6.2459E+09	2.1505	2.5984	3.7430	5.6387
41	0.5895	8.6702E+08	1.8904E+09	2.3334E+09	3.3038E+09	4.9831E+09	2.1804	2.6912	3.8105	5.7474
44	0.4515	7.1554E+08	1.6065E+09	1.9554E+09	2.7670E+09	4.1335E+09	2.2452	2.7327	3.8670	5.7768



(a)



(b)

Figure 8.4. Increase of the total encoding time of CABAC with CTW and H.263 arithmetic encoder core relative to the total encoding time of CABAC with H.263 arithmetic encoder core within AVC for (a) CITY and (b) CREW test sequences.

Experimental results show that depending on the depth D of context trees the application in CABAC of the more accurate technique of data statistics estimation based on CTW extends the decoding time of entropy decoder from 2.5 to 6.5 times in comparison to the original entropy decoder. The encoding time for the modified entropy encoder (CABAC with CTW) is approximately 2 to 5.5 times longer in comparison to the original CABAC entropy encoder for the depths D of context trees changing from 2 to 12. Experimental results proved that a good compromise between the compression performance and the complexity of the modified entropy codec relative to the original entropy codec is using of CTW with context trees of depth $D = 8$. Experimental results on the coding efficiency of the modified AVC video codec with CABAC and CTW presented in Section 6.7.1 have proved that the use of context trees with depth D greater than 8 only marginally increases the compression performance by a huge increase of the complexity of the modified entropy codec. Thus, relative to the original CABAC entropy codec, the usage of context trees of depth $D = 8$ in the modified AVC video codec leads to the increase of total decoding times by 4.5 to 5 times for the modified CABAC with CTW entropy decoder and the increase of the total encoding times by 3.5 to 4 for the modified CABAC with CTW entropy encoder. It must be emphasized one more time that the increase of the total encoding and the total decoding times concerns only the block of the entropy codec. The influence of the application of CTW technique in CABAC on the increase of the complexity of the whole modified AVC video codec is obviously significantly smaller.

8.5. Complexity of the modified and the original entropy codec – conclusions

Better coding efficiency of the modified AVC video codec (with CABAC and the CTW technique) is thus burdened with much higher complexity of both modified entropy encoder and modified entropy decoder. Nevertheless, the results of author's research are similar to those obtained for other contemporary compression improvements. When comparing the two state-of-the-art entropy coding techniques commonly used in hybrid compression of digital video (CABAC algorithm and UVLC method) it is clear that better compression performance of CABAC relative to UVLC has been also achieved by a significant increase of the complexity of entropy codec. The author's experimental results on the complexity of CABAC decoder relative to UVLC decoder within AVC (see Section 4.3.2) have shown that the optimized CABAC entropy decoder works from 1.3 to even 2.3 times

slower that the optimized UVLC entropy decoder. As a matter of fact, the compression performance improvement of CABAC relative to UVLC is higher than the coding efficiency improvement of the modified CABAC with CTW relative to the original CABAC. However, better coding efficiency of CABAC relative to UVLC results also from application (in CABAC) of the more efficient arithmetic coding in contrast to the simpler variable-length coding used in UVLC, whereas both the modified CABAC with CTW and the original CABAC use the efficient technique of arithmetic coding. Moreover, further improvement of the compression performance of more and more advanced techniques of entropy coding is more and more difficult.

The used implementation of CTW technique within the modified AVC video codec assumes sequential estimation of the conditional probabilities in nodes *s* of the context path. Nevertheless, in the encoder it is possible to implement CTW technique in a way that exploits many microprocessors that work simultaneously [Volf02]. When estimating the probabilities at depth *d* of the context tree for the current symbol x_n probabilities at depth $d+1$ can be simultaneously estimated for the successive source symbol x_{n+1}. In this way, probabilities for $D+1$ successive source symbols can be estimated in parallel when the context tree of the depth *D* is used. Thus, CTW technique can be significantly accelerated in the encoder in the case of platforms with many microprocessors. The implementation of CTW oriented towards multi-processor platforms has not been considered in this dissertation.

8.6. Complexity of the modified AVC relative to the original AVC

8.6.1. Goal and methodology

Sections 8.3 and 8.4 present experimental results on the complexity of the modified entropy codec (CABAC with CTW) relative to the original entropy codec (original CABAC). But, what is the influence of the application of CTW technique in CABAC on the complexity of the whole AVC video codec? This question can not be unambiguously answered because it depends on the percentage contribution of entropy coding in total AVC video coding. For a given video coder, the percentage contribution of entropy coding is mainly dependent on:

- The method of implementation and optimization of entropy codec and all other functional blocks of video coder;

- Features of the target platform that video codec is designed for. The architecture and parameters of microprocessor, the size of cache memory and system memory and their efficiency strongly influence on the percentage contribution of individual functional blocks of video codec.

Thus, the influence of the complexity of entropy codec on the complexity of the whole video codec will be different for different video codecs implementations.

The author has tested the influence of the application of CTW technique in CABAC on the complexity of the whole modified AVC video codec that has been built on the basis of the JM 10.2 reference implementation of AVC. The complexity of the modified AVC has been referenced to the complexity of two AVC video codecs:

- The original AVC with CABAC that works with a M-codec core highly optimized for speed;
- The experimental AVC with CABAC that works with an un-optimized core of arithmetic codec from H.263 video coding standard.

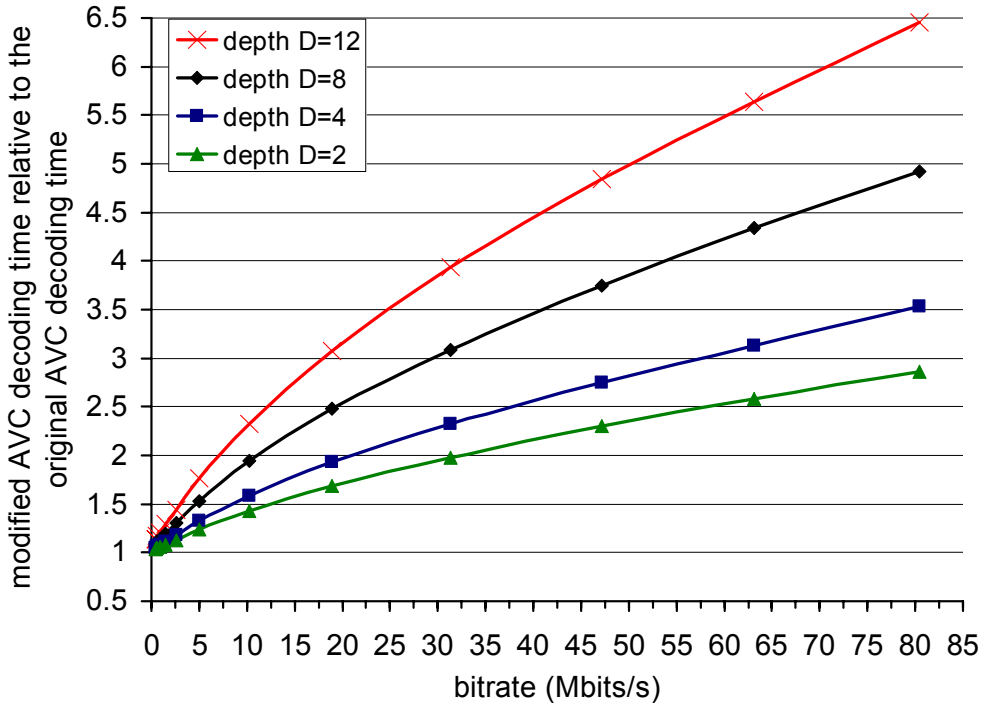
Experiments have been done on the same platform as indicated in Section 8.2.

8.6.2. Experimental results

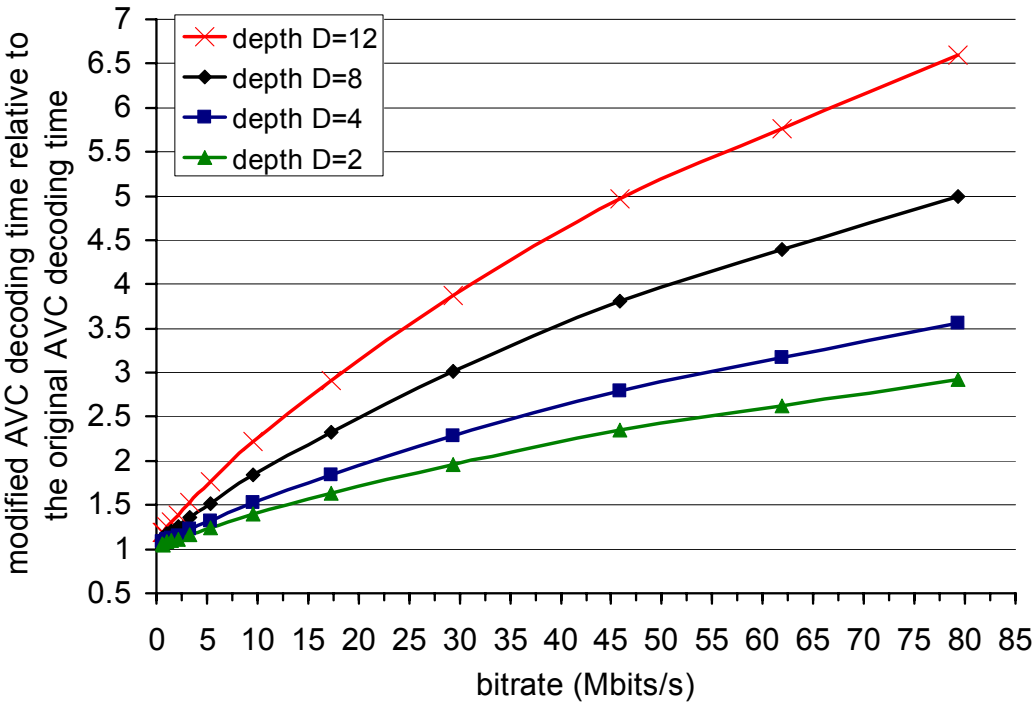
In this section experimental results on the complexity of the modified AVC video codec (with CABAC and CTW) relative to the original AVC (with CABAC and M-codec core) as well as the experimental AVC (with CABAC and H.263 arithmetic codec core) have been presented. In Table 8.5 and Figure 8.5 experimental results on the complexity of the modified AVC decoder (with CABAC and the CTW) relative to the complexity of the original AVC decoder (with CABAC and M-codec core) have been presented. Further, in Table 8.6 and Figure 8.6 the complexity of the modified AVC encoder has been confronted with the complexity of the original AVC encoder.

Table 8.5. Increase of the total decoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic decoder core) relative to the total decoding time of AVC with original CABAC (with M-codec core).

		AVC decoding times [seconds]					modified AVC with CABAC and CTW decoding time relative to AVC with original CABAC decoding time			
			modified AVC with CABAC and CTW (with H.263 AD core) for different depths D							
QP parameter	bitrate for CABAC [Mbits/s]	AVC with original CABAC	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
8	80.3902	9.6880E+01	2.7752E+02	3.4287E+02	4.7691E+02	6.2570E+02	2.8646	3.5391	4.9227	6.4585
11	63.1403	8.9050E+01	2.2973E+02	2.7894E+02	3.8669E+02	5.0230E+02	2.5798	3.1324	4.3424	5.6407
14	47.1895	8.0799E+01	1.8583E+02	2.2234E+02	3.0301E+02	3.9135E+02	2.2999	2.7517	3.7501	4.8435
17	31.3045	7.2580E+01	1.4393E+02	1.6863E+02	2.2431E+02	2.8582E+02	1.9831	2.3234	3.0905	3.9379
20	18.9680	6.4361E+01	1.0875E+02	1.2435E+02	1.5940E+02	1.9820E+02	1.6896	1.9320	2.4766	3.0795
23	10.2831	5.6689E+01	8.1324E+01	9.0039E+01	1.0990E+02	1.3181E+02	1.4346	1.5883	1.9386	2.3251
26	5.0630	4.9439E+01	6.1309E+01	6.5751E+01	7.5689E+01	8.7004E+01	1.2401	1.3299	1.5310	1.7598
29	2.6073	4.4160E+01	4.9748E+01	5.2225E+01	5.7595E+01	6.3751E+01	1.1265	1.1826	1.3042	1.4436
32	1.4660	4.0017E+01	4.3295E+01	4.4835E+01	4.7969E+01	5.1799E+01	1.0819	1.1204	1.1987	1.2944
35	0.9072	3.7813E+01	3.9968E+01	4.1007E+01	4.3001E+01	4.5674E+01	1.0570	1.0845	1.1372	1.2079
38	0.6172	3.6275E+01	3.8203E+01	3.8914E+01	4.0579E+01	4.2486E+01	1.0531	1.0727	1.1186	1.1712
41	0.4838	3.5848E+01	3.7217E+01	3.7773E+01	3.9219E+01	4.0939E+01	1.0382	1.0537	1.0940	1.1420
44	0.4171	3.4725E+01	3.5999E+01	3.6542E+01	3.7907E+01	3.9439E+01	1.0367	1.0523	1.0916	1.1358
Results for CREW test sequence										
8	79.3321	9.3818E+01	2.7459E+02	3.3392E+02	4.6875E+02	6.1923E+02	2.9269	3.5592	4.9964	6.6003
11	61.8917	8.6507E+01	2.2733E+02	2.7375E+02	3.7974E+02	4.9847E+02	2.6279	3.1645	4.3897	5.7622
14	45.9035	7.7883E+01	1.8318E+02	2.1757E+02	2.9619E+02	3.8695E+02	2.3520	2.7935	3.8029	4.9684
17	29.3435	7.0200E+01	1.3741E+02	1.6004E+02	2.1146E+02	2.7220E+02	1.9575	2.2797	3.0122	3.8775
20	17.2745	6.2842E+01	1.0243E+02	1.1588E+02	1.4637E+02	1.8318E+02	1.6300	1.8441	2.3291	2.9149
23	9.5826	5.5498E+01	7.7495E+01	8.5089E+01	1.0215E+02	1.2319E+02	1.3964	1.5332	1.8406	2.2198
26	5.3341	4.9921E+01	6.1980E+01	6.6244E+01	7.5870E+01	8.8034E+01	1.2416	1.3270	1.5198	1.7635
29	3.2550	4.5796E+01	5.3293E+01	5.6075E+01	6.2183E+01	7.0080E+01	1.1637	1.2245	1.3578	1.5303
32	2.1080	4.2859E+01	4.7528E+01	4.9512E+01	5.3684E+01	5.9222E+01	1.1089	1.1552	1.2526	1.3818
35	1.4310	3.9829E+01	4.3637E+01	4.5044E+01	4.8185E+01	5.2221E+01	1.0956	1.1309	1.2098	1.3111
38	0.9793	3.7404E+01	4.0498E+01	4.1576E+01	4.3889E+01	4.6987E+01	1.0827	1.1115	1.1734	1.2562
41	0.7332	3.6546E+01	3.8419E+01	3.9309E+01	4.1138E+01	4.3737E+01	1.0513	1.0756	1.1256	1.1968
44	0.5744	3.4422E+01	3.6607E+01	3.7310E+01	3.8888E+01	4.1112E+01	1.0635	1.0839	1.1297	1.1944



(a)

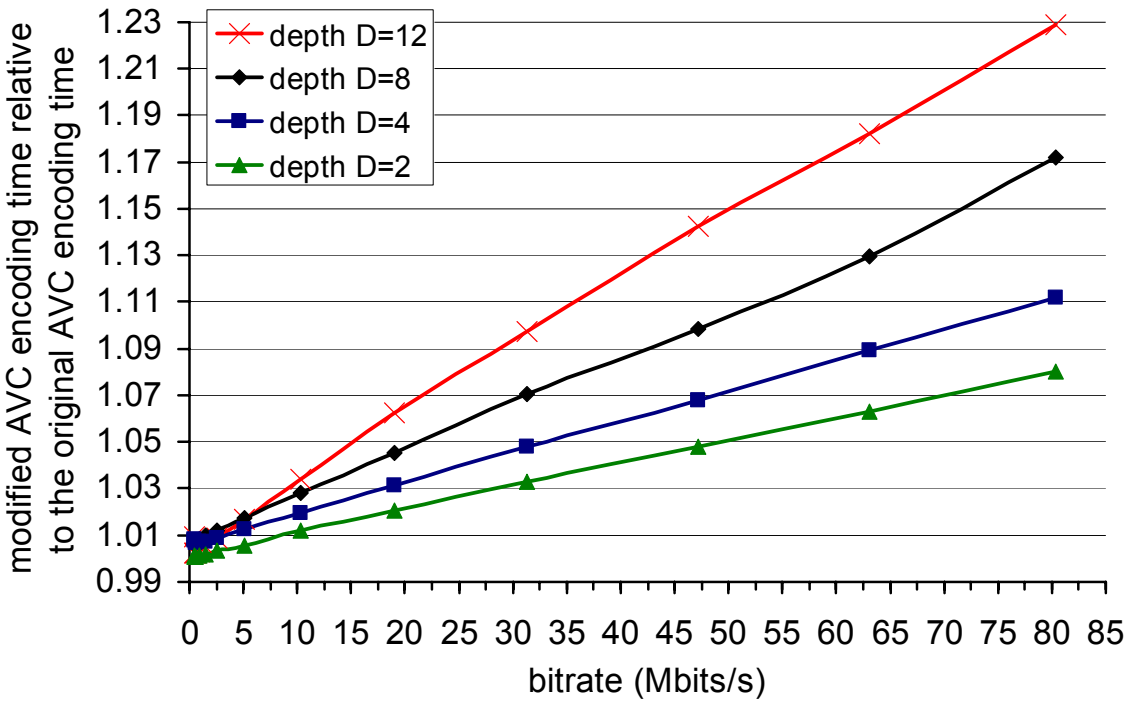


(b)

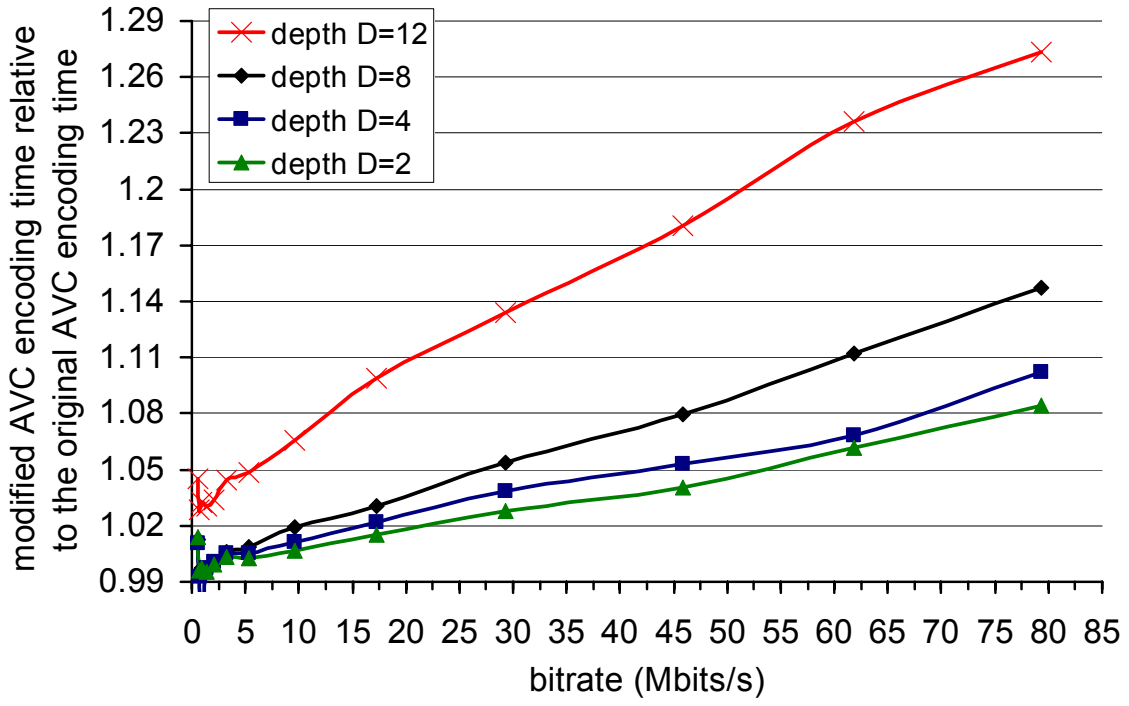
Figure 8.5. Increase of the total decoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic decoder core) relative to the total decoding time of AVC with original CABAC (with M-codec core) for (a) CITY and (b) CREW test sequences.

Table 8.6. Increase of the total encoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic encoder core) relative to the total encoding time of AVC with original CABAC (with M-codec core).

		AVC encoding times [seconds]					modified AVC with CABAC and CTW encoding time relative to AVC with original CABAC encoding time			
			modified AVC with CABAC and CTW (with H.263 AE core) for different depths D							
QP parameter	bitrate for CABAC [Mbits/s]	AVC with original CABAC	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
8	80.3902	1.7106E+03	1.8474E+03	1.9018E+03	2.0050E+03	2.1019E+03	1.0799	1.1117	1.1721	1.2287
11	63.1403	1.7000E+03	1.8075E+03	1.8517E+03	1.9204E+03	2.0096E+03	1.0632	1.0893	1.1296	1.1821
14	47.1895	1.6902E+03	1.7716E+03	1.8045E+03	1.8568E+03	1.9310E+03	1.0481	1.0676	1.0986	1.1425
17	31.3045	1.6885E+03	1.7445E+03	1.7696E+03	1.8077E+03	1.8532E+03	1.0332	1.0481	1.0706	1.0976
20	18.9680	1.6898E+03	1.7242E+03	1.7425E+03	1.7666E+03	1.7950E+03	1.0204	1.0312	1.0455	1.0623
23	10.2831	1.6935E+03	1.7134E+03	1.7265E+03	1.7415E+03	1.7509E+03	1.0118	1.0195	1.0284	1.0339
26	5.0630	1.7030E+03	1.7129E+03	1.7240E+03	1.7329E+03	1.7315E+03	1.0058	1.0124	1.0176	1.0167
29	2.6073	1.7261E+03	1.7316E+03	1.7409E+03	1.7472E+03	1.7416E+03	1.0032	1.0086	1.0122	1.0090
32	1.4660	1.7620E+03	1.7654E+03	1.7749E+03	1.7790E+03	1.7711E+03	1.0020	1.0073	1.0097	1.0052
35	0.9072	1.8035E+03	1.8054E+03	1.8161E+03	1.8182E+03	1.8089E+03	1.0011	1.0070	1.0082	1.0030
38	0.6172	1.8516E+03	1.8533E+03	1.8643E+03	1.8645E+03	1.8563E+03	1.0009	1.0069	1.0070	1.0026
41	0.4838	1.8944E+03	1.8960E+03	1.9091E+03	1.9065E+03	1.9117E+03	1.0008	1.0078	1.0064	1.0091
44	0.4171	1.9203E+03	1.9230E+03	1.9361E+03	1.9322E+03	1.9246E+03	1.0014	1.0082	1.0062	1.0022
Results for CREW test sequence										
8	79.3321	1.7920E+03	1.9436E+03	1.9752E+03	2.0556E+03	2.2820E+03	1.0846	1.1022	1.1471	1.2734
11	61.8917	1.7796E+03	1.8896E+03	1.9006E+03	1.9787E+03	2.1999E+03	1.0618	1.0680	1.1119	1.2362
14	45.9035	1.7803E+03	1.8525E+03	1.8745E+03	1.9224E+03	2.1013E+03	1.0405	1.0529	1.0798	1.1803
17	29.3435	1.7984E+03	1.8487E+03	1.8676E+03	1.8950E+03	2.0391E+03	1.0280	1.0385	1.0537	1.1338
20	17.2745	1.8121E+03	1.8397E+03	1.8520E+03	1.8671E+03	1.9914E+03	1.0152	1.0220	1.0303	1.0990
23	9.5826	1.8228E+03	1.8351E+03	1.8434E+03	1.8573E+03	1.9426E+03	1.0067	1.0113	1.0189	1.0657
26	5.3341	1.8349E+03	1.8397E+03	1.8448E+03	1.8509E+03	1.9239E+03	1.0026	1.0054	1.0087	1.0486
29	3.2550	1.8522E+03	1.8577E+03	1.8620E+03	1.8629E+03	1.9340E+03	1.0029	1.0053	1.0058	1.0441
32	2.1080	1.8818E+03	1.8800E+03	1.8833E+03	1.8824E+03	1.9456E+03	0.9990	1.0008	1.0003	1.0339
35	1.4310	1.9069E+03	1.8981E+03	1.9017E+03	1.8993E+03	1.9650E+03	0.9954	0.9973	0.9960	1.0305
38	0.9793	1.9183E+03	1.9128E+03	1.8660E+03	1.9126E+03	1.9806E+03	0.9971	0.9727	0.9970	1.0325
41	0.7332	1.9227E+03	1.9155E+03	1.9085E+03	1.9140E+03	1.9772E+03	0.9963	0.9926	0.9955	1.0283
44	0.5744	1.8862E+03	1.9129E+03	1.9057E+03	1.9099E+03	1.9708E+03	1.0142	1.0103	1.0126	1.0448



(a)



(b)

Figure 8.6. Increase of the total encoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic encoder core) relative to the total encoding time of AVC with original CABAC (with M-codec core) for (a) CITY and (b) CREW test sequences.

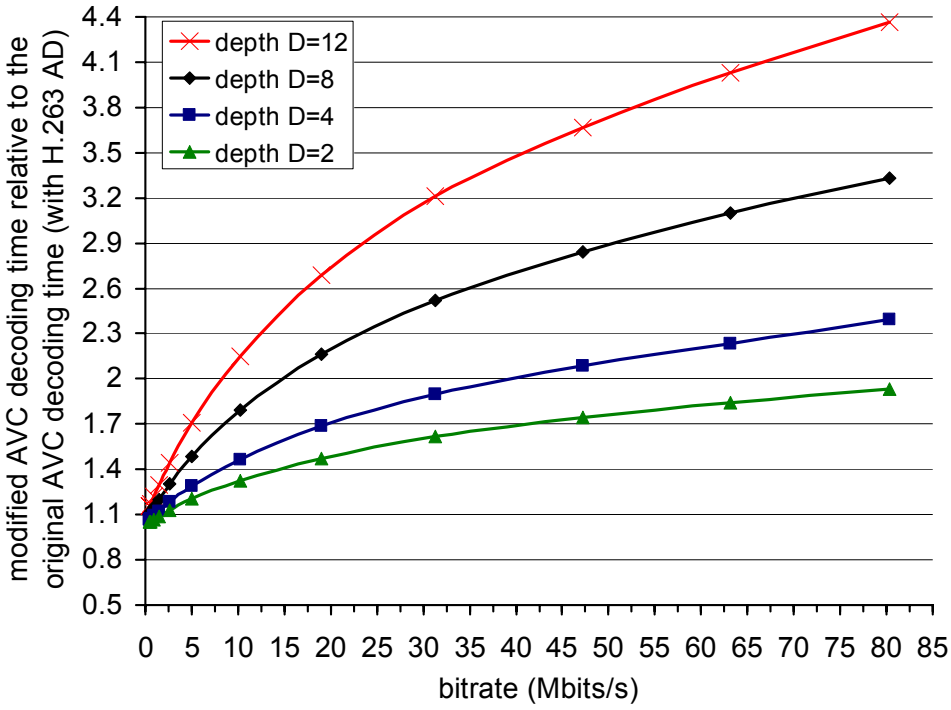
The total encoding and total decoding times of the modified AVC relative to the original AVC are dependent on the depth *D* of context trees and the target bitrate. The depth *D* of context trees determines the number of nodes *s* in which CTW technique estimates the conditional probabilities of symbols in individual contexts from the context path. The bigger depth *D* of context trees the greater number of estimated probabilities and the greater total encoding and total decoding of entropy codec. Thus, the complexity of the whole modified AVC video codec also increases. The target bitrate of encoded video sequence has also a significant impact on the total encoding and the total decoding times of the modified AVC video codec. With the increase of the bitrate entropy coding contribution in the whole video coding also increases. Therefore, the greater differences between the total encoding and the total decoding times for both modified and the original AVC video codecs have been observed for higher bitrates. Depending on the depth *D* of context trees and the value of the target bitrate the modified AVC video decoder is 1 to 6.5 times slower in comparison to the original AVC with CABAC that works with M-codec core. However, the modified AVC video encoder is 1 to 1.27 times slower relative to the original AVC video encoder.

From experimental results it is clear that the impact of application of CTW in CABAC on the complexity of video codec is significantly smaller in the case of the encoder. It results from the asymmetry of the complexity of a hybrid video encoder and a hybrid video decoder. A hybrid video encoder is much more time-consuming in comparison to a hybrid video decoder for the reason of motion estimation and encoder control units that are not present in video decoder. Therefore, the entropy coding contribution in the total coding time is far smaller for a video encoder than for a video decoder.

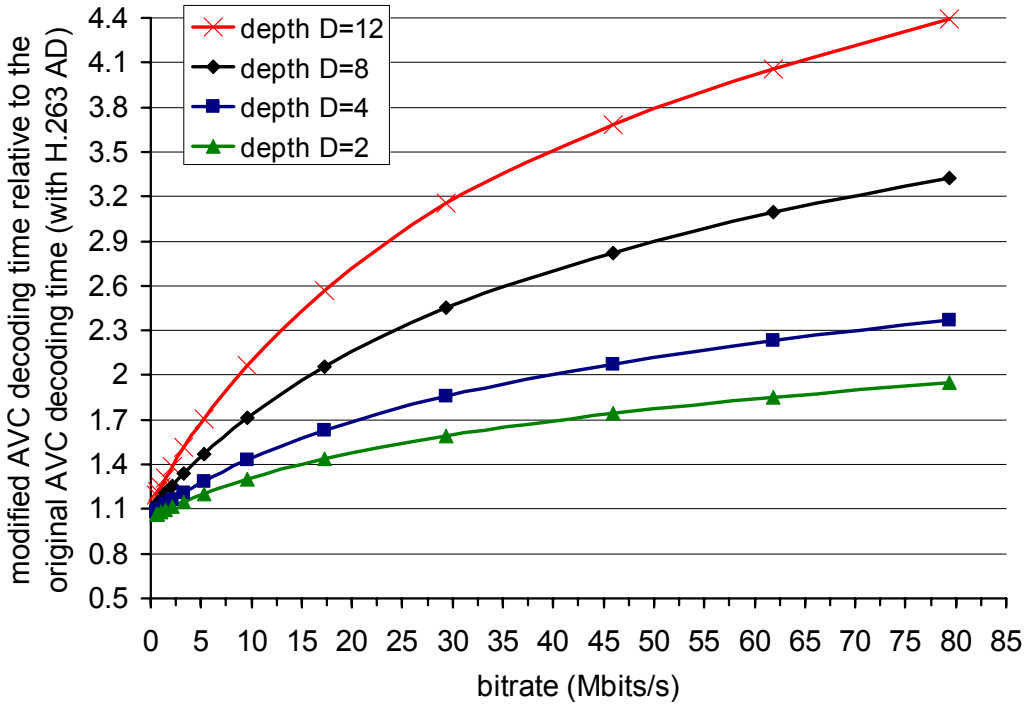
In the described experiments both the modified and the original AVC have been working with different cores of arithmetic codec, which surely influences on the complexity of the modified and the original AVC. In order to eliminate the influence of different arithmetic codec cores in the modified and the original AVC, the complexity of the modified AVC has been confronted with the complexity of experimental AVC with CABAC that works with H.263 arithmetic codec core. In this way, both the modified and the experimental AVC differed from the technique of the conditional probabilities estimation. Experimental results for video decoders have been presented in Table 8.7 and Figure 8.7. Experimental results for video encoders have been presented in Table 8.8 and Figure 8.8.

Table 8.7. Increase of the total decoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic decoder core) relative to the total decoding time of AVC with CABAC and H.263 arithmetic decoder core.

		AVC decoding times [seconds]					modified AVC with CABAC and CTW decoding time relative to AVC with CABAC and H.263 AD decoding time			
			modified AVC with CABAC and CTW (with H.263 AD core) for different depths D							
QP parameter	bitrate for CABAC [Mbits/s]	AVC with CABAC and H.263 AD	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
8	80.3902	1.4333E+02	2.7752E+02	3.4287E+02	4.7691E+02	6.2570E+02	1.9362	2.3921	3.3273	4.3654
11	63.1403	1.2471E+02	2.2973E+02	2.7894E+02	3.8669E+02	5.0230E+02	1.8422	2.2368	3.1008	4.0279
14	47.1895	1.0667E+02	1.8583E+02	2.2234E+02	3.0301E+02	3.9135E+02	1.7420	2.0842	2.8405	3.6686
17	31.3045	8.8954E+01	1.4393E+02	1.6863E+02	2.2431E+02	2.8582E+02	1.6180	1.8957	2.5216	3.2131
20	18.9680	7.3751E+01	1.0875E+02	1.2435E+02	1.5940E+02	1.9820E+02	1.4745	1.6860	2.1613	2.6874
23	10.2831	6.1392E+01	8.1324E+01	9.0039E+01	1.0990E+02	1.3181E+02	1.3247	1.4666	1.7901	2.1470
26	5.0630	5.0862E+01	6.1309E+01	6.5751E+01	7.5689E+01	8.7004E+01	1.2054	1.2927	1.4881	1.7106
29	2.6073	4.4126E+01	4.9748E+01	5.2225E+01	5.7595E+01	6.3751E+01	1.1274	1.1835	1.3052	1.4447
32	1.4660	3.9941E+01	4.3295E+01	4.4835E+01	4.7969E+01	5.1799E+01	1.0840	1.1225	1.2010	1.2969
35	0.9072	3.7501E+01	3.9968E+01	4.1007E+01	4.3001E+01	4.5674E+01	1.0658	1.0935	1.1467	1.2179
38	0.6172	3.6204E+01	3.8203E+01	3.8914E+01	4.0579E+01	4.2486E+01	1.0552	1.0749	1.1208	1.1735
41	0.4838	3.5391E+01	3.7217E+01	3.7773E+01	3.9219E+01	4.0939E+01	1.0516	1.0673	1.1082	1.1568
44	0.4171	3.4173E+01	3.5999E+01	3.6542E+01	3.7907E+01	3.9439E+01	1.0534	1.0693	1.1093	1.1541
Results for CREW test sequence										
8	79.3321	1.4093E+02	2.7459E+02	3.3392E+02	4.6875E+02	6.1923E+02	1.9485	2.3695	3.3263	4.3940
11	61.8917	1.2282E+02	2.2733E+02	2.7375E+02	3.7974E+02	4.9847E+02	1.8510	2.2289	3.0920	4.0587
14	45.9035	1.0511E+02	1.8318E+02	2.1757E+02	2.9619E+02	3.8695E+02	1.7427	2.0698	2.8178	3.6813
17	29.3435	8.6174E+01	1.3741E+02	1.6004E+02	2.1146E+02	2.7220E+02	1.5946	1.8571	2.4538	3.1587
20	17.2745	7.1219E+01	1.0243E+02	1.1588E+02	1.4637E+02	1.8318E+02	1.4383	1.6272	2.0552	2.5720
23	9.5826	5.9609E+01	7.7495E+01	8.5089E+01	1.0215E+02	1.2319E+02	1.3001	1.4275	1.7137	2.0667
26	5.3341	5.1564E+01	6.1980E+01	6.6244E+01	7.5870E+01	8.8034E+01	1.2020	1.2847	1.4714	1.7073
29	3.2550	4.6329E+01	5.3293E+01	5.6075E+01	6.2183E+01	7.0080E+01	1.1503	1.2104	1.3422	1.5127
32	2.1080	4.2641E+01	4.7528E+01	4.9512E+01	5.3684E+01	5.9222E+01	1.1146	1.1611	1.2590	1.3889
35	1.4310	3.9859E+01	4.3637E+01	4.5044E+01	4.8185E+01	5.2221E+01	1.0948	1.1301	1.2089	1.3101
38	0.9793	3.7610E+01	4.0498E+01	4.1576E+01	4.3889E+01	4.6987E+01	1.0768	1.1055	1.1670	1.2493
41	0.7332	3.5999E+01	3.8419E+01	3.9309E+01	4.1138E+01	4.3737E+01	1.0672	1.0919	1.1428	1.2150
44	0.5744	3.4438E+01	3.6607E+01	3.7310E+01	3.8888E+01	4.1112E+01	1.0630	1.0834	1.1292	1.1938



(a)

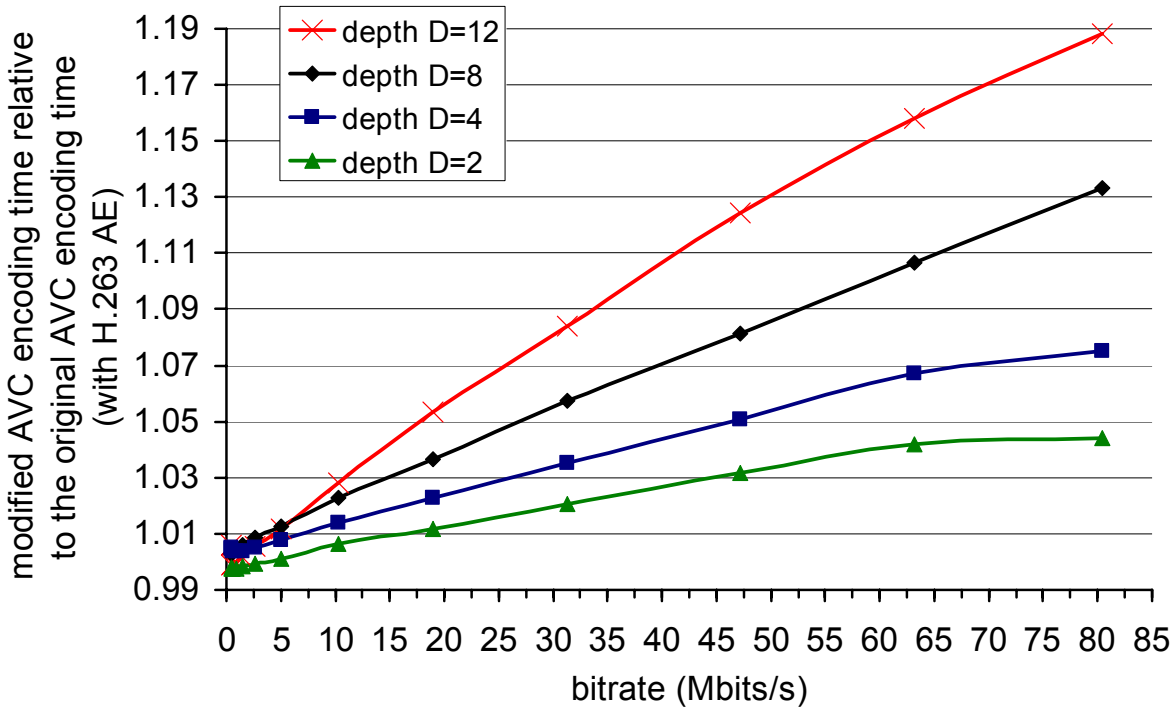


(b)

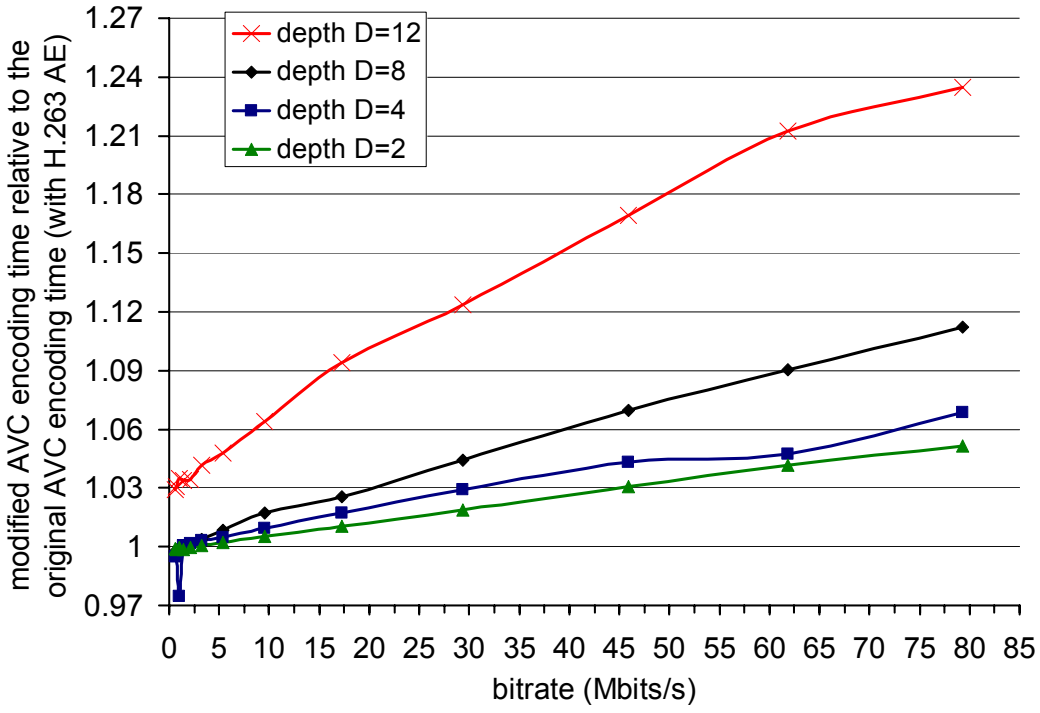
Figure 8.7. Increase of the total decoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic decoder core) relative to the total decoding time of AVC with CABAC (and H.263 arithmetic decoder core) for (a) CITY and (b) CREW test sequences.

Table 8.8. Increase of the total encoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic encoder core) relative to the total encoding time of AVC with CABAC and H.263 arithmetic encoder core.

		AVC encoding times [seconds]					modified AVC with CABAC and CTW encoding time relative to AVC with CABAC and H.263 AE encoding time			
			modified AVC with CABAC and CTW (with H.263 AE core) for different depths D							
QP parameter	bitrate for CABAC [Mbits/s]	AVC with CABAC and H.263 AE	D=2	D=4	D=8	D=12	D=2	D=4	D=8	D=12
Results for CITY test sequence										
8	80.3902	1.7691E+03	1.8474E+03	1.9018E+03	2.0050E+03	2.1019E+03	1.0443	1.0750	1.1334	1.1882
11	63.1403	1.7350E+03	1.8075E+03	1.8517E+03	1.9204E+03	2.0096E+03	1.0417	1.0672	1.1068	1.1582
14	47.1895	1.7171E+03	1.7716E+03	1.8045E+03	1.8568E+03	1.9310E+03	1.0317	1.0509	1.0813	1.1245
17	31.3045	1.7096E+03	1.7445E+03	1.7696E+03	1.8077E+03	1.8532E+03	1.0204	1.0351	1.0574	1.0840
20	18.9680	1.7040E+03	1.7242E+03	1.7425E+03	1.7666E+03	1.7950E+03	1.0119	1.0226	1.0367	1.0534
23	10.2831	1.7028E+03	1.7134E+03	1.7265E+03	1.7415E+03	1.7509E+03	1.0062	1.0139	1.0227	1.0283
26	5.0630	1.7112E+03	1.7129E+03	1.7240E+03	1.7329E+03	1.7315E+03	1.0010	1.0075	1.0127	1.0119
29	2.6073	1.7324E+03	1.7316E+03	1.7409E+03	1.7472E+03	1.7416E+03	0.9995	1.0049	1.0085	1.0053
32	1.4660	1.7681E+03	1.7654E+03	1.7749E+03	1.7790E+03	1.7711E+03	0.9985	1.0038	1.0062	1.0017
35	0.9072	1.8096E+03	1.8054E+03	1.8161E+03	1.8182E+03	1.8089E+03	0.9977	1.0036	1.0048	0.9996
38	0.6172	1.8577E+03	1.8533E+03	1.8643E+03	1.8645E+03	1.8563E+03	0.9976	1.0036	1.0037	0.9993
41	0.4838	1.9006E+03	1.8960E+03	1.9091E+03	1.9065E+03	1.9117E+03	0.9976	1.0045	1.0031	1.0058
44	0.4171	1.9266E+03	1.9230E+03	1.9361E+03	1.9322E+03	1.9246E+03	0.9981	1.0050	1.0030	0.9990
Results for CREW test sequence										
8	79.3321	1.8483E+03	1.9436E+03	1.9752E+03	2.0556E+03	2.2820E+03	1.0515	1.0687	1.1121	1.2347
11	61.8917	1.8145E+03	1.8896E+03	1.9006E+03	1.9787E+03	2.1999E+03	1.0414	1.0474	1.0905	1.2124
14	45.9035	1.7969E+03	1.8525E+03	1.8745E+03	1.9224E+03	2.1013E+03	1.0309	1.0432	1.0699	1.1694
17	29.3435	1.8144E+03	1.8487E+03	1.8676E+03	1.8950E+03	2.0391E+03	1.0189	1.0294	1.0444	1.1238
20	17.2745	1.8204E+03	1.8397E+03	1.8520E+03	1.8671E+03	1.9914E+03	1.0106	1.0173	1.0256	1.0939
23	9.5826	1.8258E+03	1.8351E+03	1.8434E+03	1.8573E+03	1.9426E+03	1.0051	1.0096	1.0172	1.0640
26	5.3341	1.8358E+03	1.8397E+03	1.8448E+03	1.8509E+03	1.9239E+03	1.0021	1.0049	1.0082	1.0480
29	3.2550	1.8565E+03	1.8577E+03	1.8620E+03	1.8629E+03	1.9340E+03	1.0007	1.0030	1.0035	1.0418
32	2.1080	1.8806E+03	1.8800E+03	1.8833E+03	1.8824E+03	1.9456E+03	0.9997	1.0015	1.0010	1.0346
35	1.4310	1.9004E+03	1.8981E+03	1.9017E+03	1.8993E+03	1.9650E+03	0.9988	1.0006	0.9994	1.0340
38	0.9793	1.9140E+03	1.9128E+03	1.8660E+03	1.9126E+03	1.9806E+03	0.9993	0.9749	0.9992	1.0348
41	0.7332	1.9179E+03	1.9155E+03	1.9085E+03	1.9140E+03	1.9772E+03	0.9988	0.9951	0.9980	1.0309
44	0.5744	1.9150E+03	1.9129E+03	1.9057E+03	1.9099E+03	1.9708E+03	0.9989	0.9952	0.9973	1.0291



(a)



(b)

Figure 8.8. Increase of the total encoding time of the modified AVC with CABAC and CTW (with H.263 arithmetic encoder core) relative to the total encoding time of AVC with CABAC (and H.263 arithmetic encoder core) for (a) CITY and (b) CREW test sequences.

Depending on the depth D of context trees and the value of the target bitrate the application of CTW in CABAC slows down the decoding speed of the modified AVC by 1 to 4.4 times and the encoding speed of the modified AVC by 1 to 1.24 times.

8.6.3. Complexity of the modified AVC relative to the original AVC – conclusions

The application of more exact technique of data statistics estimation based on CTW in CABAC increases the complexity of both a video encoder and a video decoder. The increase of the total encoding and the total decoding times for the modified AVC with CABAC and CTW depends on the value of target bitrate and the depth D of context trees. For the depth $D = 8$ and the range of useful bitrates of AVC stream used for Standard-definition Television (less or equal to 10 Mbits/s in *Baseline*, *Extended* and *Main* Profiles of AVC) the total encoding time increases up to 2.5% and total decoding time increases up to 80% after application of CTW in CABAC. Nevertheless, recent increases of performance of digital processors have made even more complicated techniques become attractive for real-time video coding.

8.7. Complexity and coding efficiency of the modified AVC with CTW – final conclusions

The application of data statistics estimation technique based on CTW in CABAC improves compression performance of entropy coder, nevertheless the complexity of the video encoder and the video decoder also increase. Both compression performance improvement and increase of complexity depend on target bitrate. Additionally, the increase of complexity is different for video encoder and video decoder. In Figure 8.9 quotient of percentage increase of complexity and percentage reduction of bitrate has been presented for the modified AVC codec (with CTW and context tree depth $D = 8$) for the target bitrates less than 10 Mbits/s. Both parameters (increase of complexity and reduction of bitrate) have been established with reference to the original AVC with CABAC.

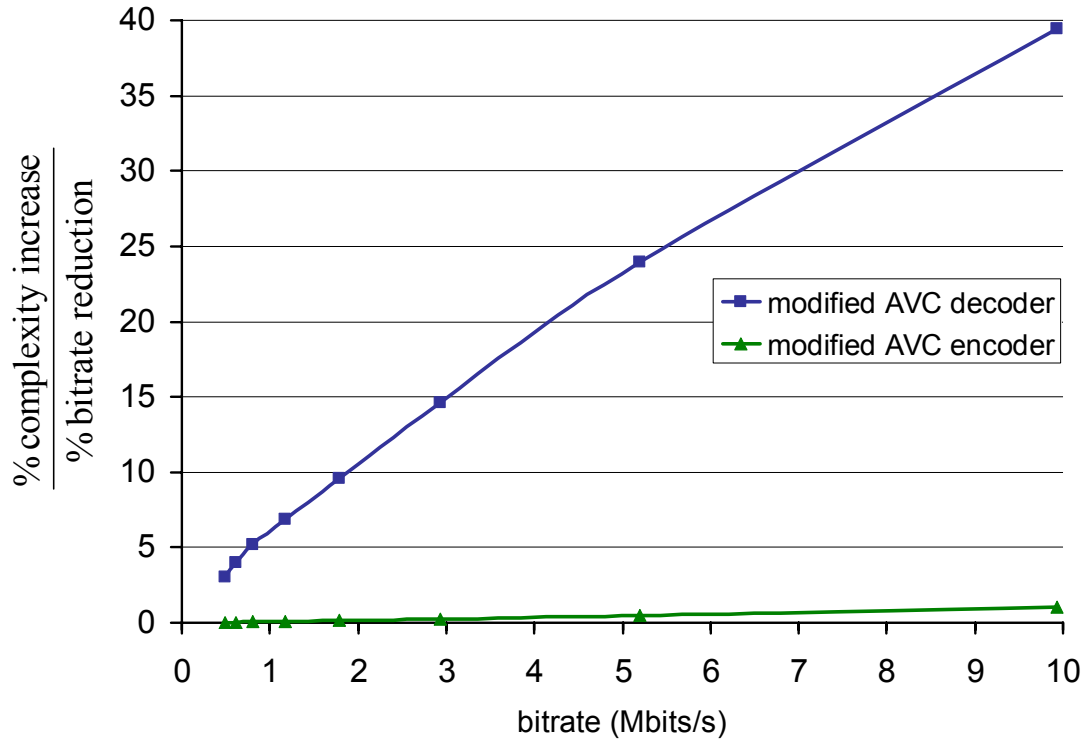


Figure 8.9. The relationship between increase of complexity and reduction of bitrate for the modified AVC codec relative to the original AVC codec (average for CITY and CREW test sequences and I29P GOP structure).

Application of CTW technique in CABAC only insignificantly influences the complexity of the video encoder in presented range of bitrates (less than 10 Mbits/s). However, the complexity of the video decoder has significantly increased. The ratio of increase of complexity to reduction of bitrate ranges from 3 to 40 for considered range of bitrates. Nevertheless, higher coding efficiency of CABAC relative to UVLC is also burdened with significantly higher complexity of the video decoder. The author has investigated complexity and efficiency of optimized AVC decoder with CABAC relative to AVC with UVLC for bitrates less than 7 Mbits/s [Graj05]. The ratio of increase of complexity to reduction of bitrate ranged from 0.45 to 6.5. Thus, better coding efficiency of AVC with CABAC is obtained with smaller increase of complexity of AVC decoder. Nevertheless, improvement of compression for contemporary video coders is obtained by exponential increase of complexity. As an example, the new AVC encoder [AVC] provides about 50% bitrate savings with reference to MPEG-2 encoder [MPEG-2], but AVC decoder is approximately four times more complex than MPEG-2 decoder [Sunna05]. Further improvement of efficiency of state-of-the-art video compression technologies is even more difficult and needs even more computational outlay.

Chapter 9

Implementation of advanced entropy codecs

9.1. Software version of CABAC with CTW

9.1.1. Implementation of CABAC entropy codec

Contemporary adaptive entropy coders significantly improve the compression performance of video coders. High coding efficiency of advanced entropy coders is a result of using efficient arithmetic coding and sophisticated techniques of data statistics estimation. As it has been stated earlier, the state-of-the-art entropy coder used in video compression is CABAC. The extremely high compression performance of CABAC has been obtained at a price of high complexity of encoding and decoding. The implementation of CABAC entropy codec is also far more difficult and far more time-consuming than any other entropy codec commonly used in video compression.

The author was a member of a team that has implemented fast AVC video decoder. In the author's knowledge, it was the first implementation of AVC decoder in Poland and one of the first all over the world. This AVC decoder has been sold by Advanced Digital Broadcast (ADB) [ADB] in a few hundred thousands copies all over the world until now. Besides, the author was a member of a team that has implemented fast AVC encoder that is used by ADB.

Within the confines of these projects the author has fully implemented both CABAC encoder and CABAC decoder in C programming language [Kern88]. The complexity and the outlay of work of implementing CABAC encoder and CABAC decoder are comparable. The author's implementation of the optimized CABAC codec (the encoder and the decoder)

contains approximately 5200 lines of program code in C. The core of the binary arithmetic codec contains only 380 lines of program code. Thus, from the point of view of implementation of CABAC codec the core of arithmetic codec makes only about 7% of the whole implementation of CABAC. It means that implementations of binarization, context model selection and probability estimation and update make about 93% of the whole CABAC implementation. These figures are similar to those obtained for other implementations of CABAC codec. For comparison, in the implementation of CABAC from x264 video codec [x264Soft] the core of arithmetic codec makes approximately 9% of the whole CABAC and in the implementation of CABAC in JM 10.2 reference software [AVCSoft] the core of arithmetic codec makes about 12% of the whole CABAC. It has been presented in Figure 9.1.

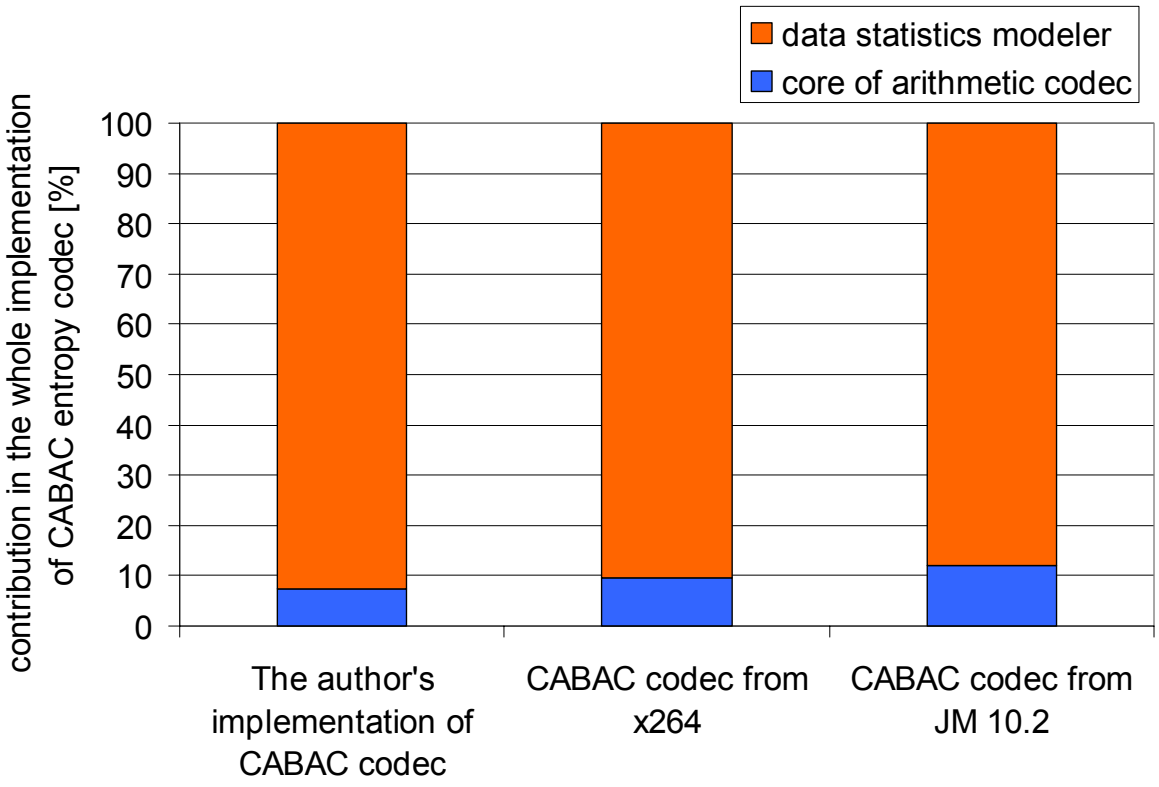


Figure 9.1. Contribution of arithmetic codec core in three implementations of CABAC codec.

Thus, the block of data statistics modeling makes an essential part of contemporary entropy codecs that to a large extent determines the complexity and the compression performance of entropy coding. Sophisticated mechanisms of data statistics modeling together with binary arithmetic coding cause that a considerable amount of computations is required when CABAC encoding or CABAC decoding of a binary symbol. The author took part in the project of putting into practice of fast AVC video decoder dedicated to signal processor platforms. Measurements on complexity of fast AVC decoder with CABAC revealed that

high-performance digital signal processor TMS320DM642 [TI642] (with frequency of 600 MHz) is able to decode only a bitstream of up to four megabits per second in real-time. Decoding of one binary symbol with CABAC absorbs about 75 cycles of TMS320DM642 processor. Different processor power is needed for data statistics modeling and binary arithmetic coding. In author's implementation of CABAC, the core of binary arithmetic decoder needs about 30 processor cycles to decode one binary symbol. It means that data statistics modeling makes about 60% of the total CABAC decoding time of a binary symbol (see Figure 9.2).

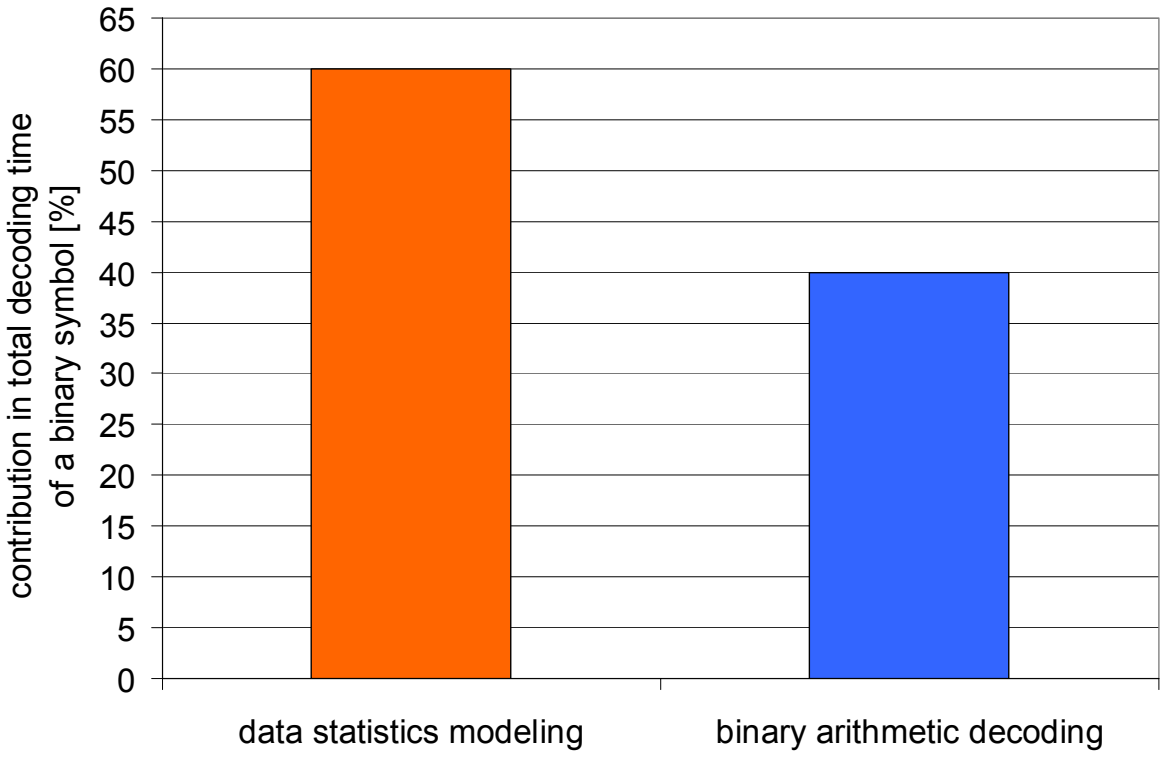


Figure 9.2. Contribution of data statistics modeling and binary arithmetic decoding in the total decoding time of a binary symbol.

Thus, data statistics modeling is considerably more time-consuming than binary arithmetic decoding in CABAC. It causes that CABAC coding is a processor-intensive task that demands high-performance digital processors for real-time coding.

9.1.2. Implementation of CTW technique within CABAC

The author has implemented CTW technique for both CABAC encoder and CABAC decoder within AVC reference software in C programming language. CTW technique has been implemented in a way described in Section 6.3.1. The author's implementation of CTW

technique adds about 500 lines of program code extra to implementation of CABAC codec. This figure does not take into consideration declarations of LUT that are used in logarithmic-domain implementation of CTW.

The application in CABAC of the more exact technique of the conditional probabilities estimation based on CTW additionally increases the complexity of adaptive entropy codec. Experiments on the increase of the complexity of CABAC after application of CTW have been presented in Section 8.3. According to them, the modified CABAC entropy codec (with CTW) is several times slower relative to the original CABAC codec. Thus, the contemporary advanced entropy codecs that exploit more exact techniques of the conditional probabilities estimation are a great challenge even for today's high-performance processors. The real-time entropy coding for transmission bitrates greater than 10 Mbits/s is a very difficult task for digital media processors. However, it is commonly known that Field Programmable Gate Arrays (FPGA) platforms are characterized by considerably higher processing capabilities in comparison to digital signal processors. Therefore, power demanding advanced entropy coding techniques can be efficiently realized on hardware platforms.

9.2. Hardware version of CABAC entropy codec

9.2.1. Implementation of CABAC entropy decoder

The author has designed and implemented a hardware version of CABAC decoder. In this implementation, CABAC decoder has been clearly divided into three main functional blocks: a block of arithmetic decoder core, a block of de-binarization and control of syntax elements decoding and a block of local context management. The task of de-binarization and control of syntax elements decoding is realized with two functional blocks: a *syntax elements decoding* and a *transform coefficients decoding*. The *transform coefficients decoding* block realizes de-binarization and decoding of block of transform coefficients. The *syntax elements decoding* block controls the process of de-binarization and decoding of all remaining syntax elements. For the reason of the application of several different binarization schemes for syntax elements in CABAC, ROM memory which contains the methods of decoding and de-binarization of individual syntax elements has been used. In order to decode a binary symbol, *syntax elements decoding* and *transform coefficients decoding* modules strobe the *arithmetic decoder core* module that realizes arithmetic decoding of symbols with taken into

consideration the proper probability model saved in *context models* block. The number of probability model is calculated by *management of local context* module based on the values of symbols in neighboring blocks. The core of arithmetic decoder decodes encoded bitstream from *input buffer* block that is filled with bitstream of encoded data. The block diagram of author's hardware CABAC decoder has been presented in Figure 9.3.

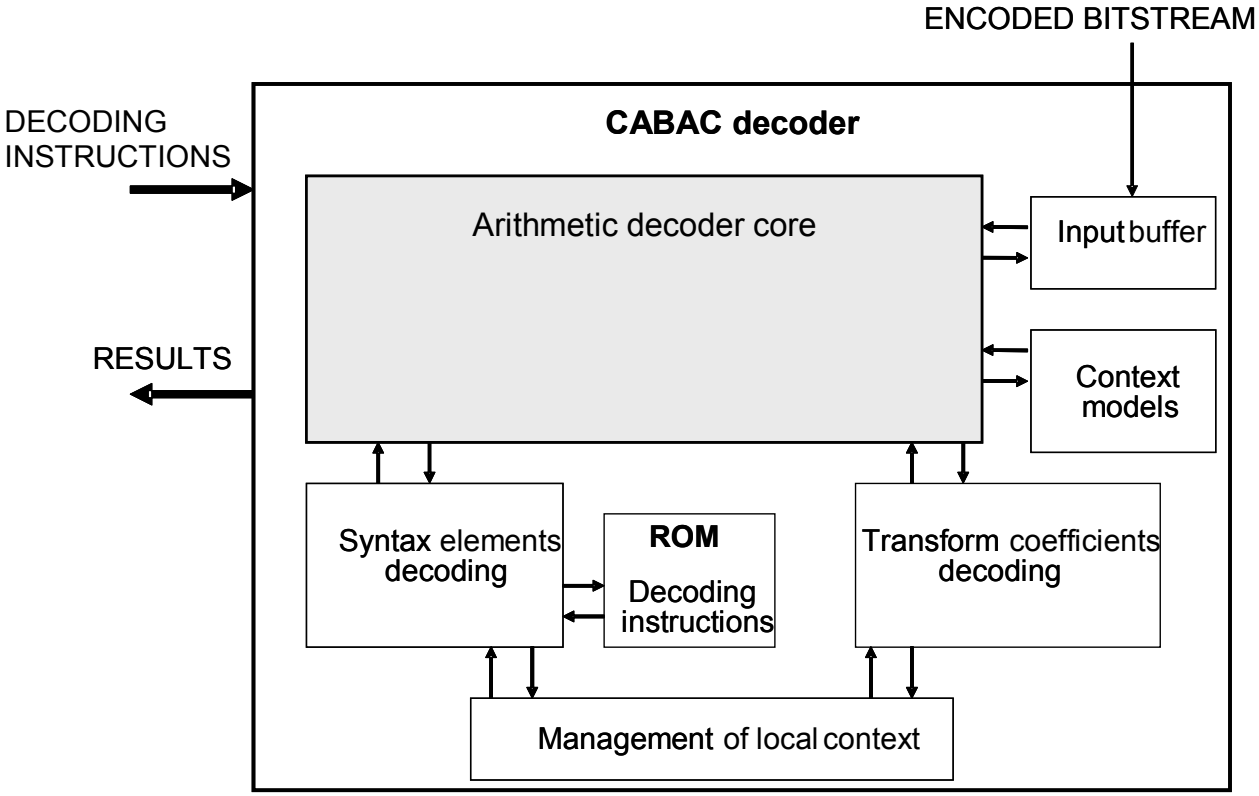


Figure 9.3. General block diagram of author's hardware version of CABAC decoder.

CABAC algorithm in a great deal exploits dependencies between symbols. For that reason, both CABAC encoding and CABAC decoding is a sequential process. It is very difficult to do computations in parallel in CABAC. Nevertheless, there are some possibilities to accelerate CABAC coding in hardware realization. Speaking in the most general terms, CABAC decoding of a binary symbol can be divided into the following tasks:

- Calculating of the number of probability model;
- Arithmetic decoding of a binary symbol with taking into consideration the probability model;
- Renormalization of registers of arithmetic decoder core;
- Updating of the probability model with respect to the value of decoded binary symbol.

In author's implementation of CABAC decoder these computations are performed with exploiting parallelism and tasks pipelining. When arithmetic decoder decodes the current symbol, computations for register renormalization and updating of the probability model can be done in parallel. Calculation of the number of probability model for the successive binary symbol can also be started at the same time. After calculation of the number of probability model for the successive symbol, it has to be checked if the process of renormalization of registers of arithmetic decoder core has been already finished. If it was, the core of arithmetic decoder can start decoding of the successive binary symbol. In this way, the throughput of CABAC decoder has been significantly increased.

9.2.2. Features of author's hardware version of CABAC decoder

The author's implementation of hardware CABAC decoder contains about 5500 lines of program code written in Verilog [Verilog] hardware description language (HDL). The project has been synthesized on Virtex 5 FPGA platform [Virtex-5] with ISE 9.2i software [XilinxISE]. The maximum clock frequency of CABAC decoder is 192.397 MHz and it utilizes about 1600 Virtex 5 slices. It is commonly known that approximately three times higher performance can be achieved when realizing the design as an application-specific integrated circuit (ASIC) [Kuon07]. According to that, the author's CABAC decoder realized as an ASIC can work with maximum frequency of about 600 MHz when using the same process technology as FPGA platform.

There have been done tests on the performance of hardware CABAC decoder with a set of a hundred thousands binary symbols. Experimental results revealed that the author's hardware CABAC decoder decodes a binary symbol in 7.5 clock cycles in average. For comparison, high-performance digital media processor TMS320DM642 needs about 75 clock cycles to decode a binary symbol for author's software implementation of CABAC decoder. Thus, the hardware version of CABAC decoder needs 10 times smaller number of clock cycles in comparison to the software version of CABAC decoder to decode a binary symbol (see Figure 9.4).

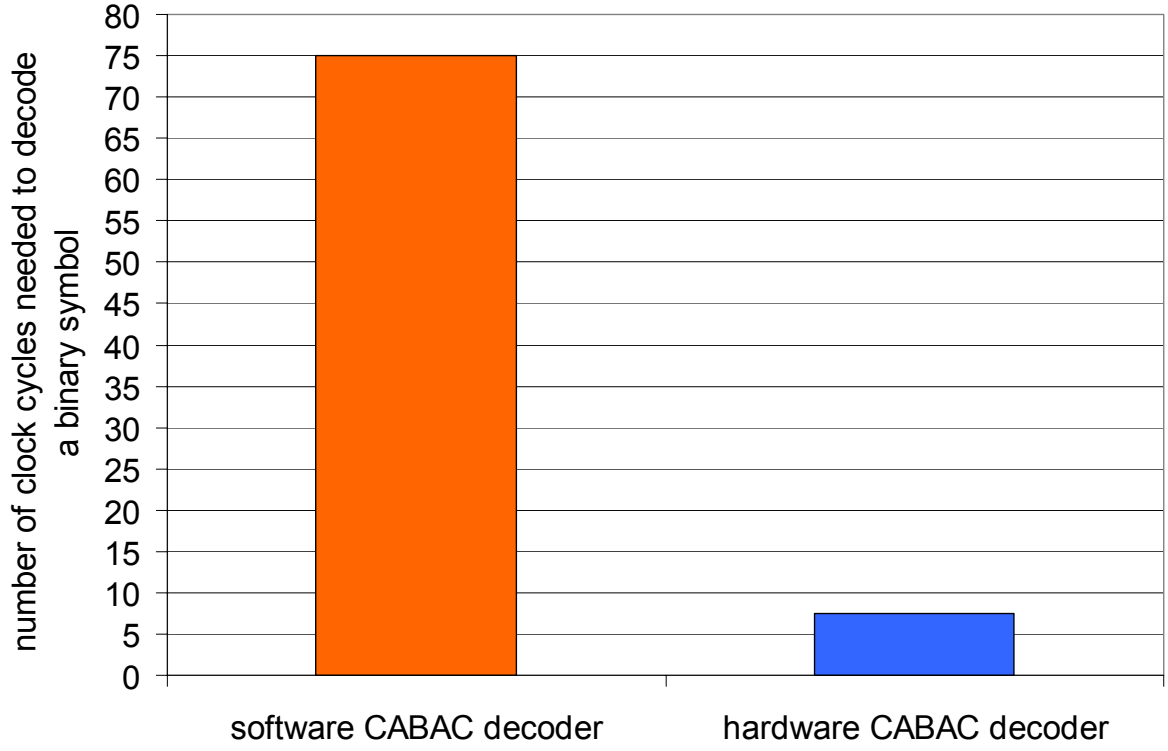


Figure 9.4. Number of clock cycles needed to decode a binary symbol for software and hardware version of CABAC decoder.

The author has not implemented CTW modeling technique for hardware platforms yet. Therefore, the gain of performance of hardware version of the modified CABAC with CTW relative to its software counterpart is currently unknown. Nevertheless, the complementation of the hardware version of CABAC decoder with CTW technique can be a subject of future works.

9.3. Implementation of advanced entropy codecs - conclusions

High compression performance of contemporary advanced entropy coders is mainly a result of application of sophisticated mechanisms of data statistics estimation. Nevertheless, the techniques of data statistics estimation that are used in advanced entropy codecs significantly affect the complexity of the whole entropy codec. In arithmetic entropy coders of newer generation data statistics estimation makes about 60% of time of the whole process of

entropy coding. Therefore, the real-time entropy coding for entropy coders of new generation is a difficult task even for high-power digital processors, especially for higher bitrates.

On the basis of author's implementation of hardware and software version of CABAC decoder it has been proved that the throughput of advanced entropy decoder can be significantly increased when it is realized on the hardware platform. The experiment has been done for CABAC decoder only. Nevertheless, taking into consideration the fact of symmetry in complexity of entropy decoder and entropy encoder based on arithmetic coding the same conclusions are expected for CABAC encoder.

Chapter 10

Recapitulation and conclusions

10.1. Recapitulation

The dissertation has been focused on improvement of compression in adaptive arithmetic encoders by using more exact mechanisms of data statistics estimation. Three more accurate techniques of the conditional probabilities estimation have been proposed to be used in advanced entropy coders in video compression. These are:

- Context-Tree Weighting (CTW) (Section 5.4.1);
- Prediction with Partial Matching (PPM) (Section 5.4.2);
- Original proposal of joint application of Context-Tree Weighting and Prediction with Partial Matching technique (Section 5.4.3).

The proposed techniques of data statistics gathering have been adopted into the state-of-the-art Context-based Adaptive Binary Arithmetic Coding (CABAC) algorithm [Marp03a] that is currently the most efficient entropy coder used in hybrid video compression. In this way, the author has proposed and built three new modified versions of CABAC codec that use sophisticated techniques of data statistics estimation.

The compression performance of each of the three modified CABAC entropy encoders has been thoroughly investigated with the test video sequences and confronted with the coding efficiency of the original CABAC encoder. The coding efficiency of the modified and the original CABAC entropy encoders has been tested in the state-of-the-art Advanced Video Coder AVC [AVC]. In order to do that, about two thousands hours of experiments have been done. Different results have been obtained for the modified CABAC entropy encoders (see Section 6.7). In series of experiments the author has pointed out which technique of data

statistics estimation allows for the greatest improvement of the compression performance for the advanced entropy encoder. The best coding efficiency has been observed for the modified CABAC encoders that took advantage of CTW technique. The author has experimentally proved that considerable bitrate reduction of 2%-4.5% is possible when application of these modified CABAC encoders within AVC framework. Thus, the thesis of the dissertation has been proved (see Chapter 6).

The algorithms of data statistics initialization were different for the modified and the original CABAC entropy encoders. In series of experiments the author has investigated in what extent it influences the compression performance of the tested entropy encoders. Obtained experimental results showed a great importance of the algorithm of data statistics initialization on the compression performance of contemporary adaptive entropy encoders (see Section 6.8).

The modified CABAC encoders are characterized by better coding efficiency in comparison to the original CABAC. Nevertheless, for the reason of restrictions of the core of binary arithmetic codec (M-codec) used in CABAC (see Section 7.1), different core of arithmetic codec had to be used in the modified CABAC codecs. The core of arithmetic codec defined in H.263 video coding standard has been used. In series of experiments the author has thoroughly investigated how the application of arithmetic codec core from H.263 influences the compression performance of the modified CABAC encoders. Obtained experimental results proved a marginal influence of tested cores of arithmetic codec on the coding efficiency of the modified CABAC encoders (see Chapter 7). The author has proved that better coding efficiency of the modified CABAC encoders (relative to the original CABAC) is a result of application of proposed techniques of the probabilities estimation and not different core of arithmetic encoder.

The goal of the dissertation was also to test the influence of application of more sophisticated techniques of data statistics modeling in entropy codec on its complexity. The complexity of the modified CABAC encoder and decoder with CTW has been investigated with the test video sequences. Additionally, it has been tested how the application of the modified CABAC entropy codec influences the complexity of the whole AVC video encoder and video decoder (see Chapter 8). For the context trees of depth $D = 8$ and the useful bitrates less than 5 Mbits/s AVC encoding time increases up to 1.3 % and AVC decoding time increases up to 50% after application of the modified CABAC with CTW. The depth $D = 8$ has been experimentally determined and gives the best compromise between gain of coding efficiency and increase of complexity for the modified AVC coder with CTW.

Advanced entropy coders are an essential element of contemporary video coders that in bigger and bigger extent determine both compression performance and complexity of the whole video codec. The author has presented his experiences in implementation of the optimized advanced entropy codecs dedicated to both processor-based and hardware platforms. The results on complexity of software and hardware versions of author's CABAC decoder have been introduced (see Chapter 9). Percentage contribution of data statistics modeling and binary arithmetic decoding in processing a binary symbol has been investigated.

10.2. Original achievements of the dissertation

The **main achievement** of the dissertation is the proposal of the original extensions to CABAC algorithm that considerably improve the compression performance of entropy encoder. The following well-known and commonly used techniques of data statistics modeling in data compression have been proposed:

a) Context-Tree Weighting method (Section 5.4.1);
b) Prediction with Partial Matching method (Section 5.4.2);
c) The author's method of joint application of Context-Tree Weighting and Prediction with Partial Matching technique (Section 5.4.3).

The proposed extensions of CABAC entropy encoder can be used to improve the compression performance of contemporary video encoders. Proposed extensions can also find the application in video encoders of the next generations [VCEG07].

The **important achievement** of the dissertation is the answer to the question how the application of CTW and/or PPMA technique influences the compression performance of contemporary adaptive entropy encoders. Besides, the dissertation explicitly points out which technique of data statistics gathering allows for achieving the greatest improvement of the coding efficiency of entropy encoder. The dissertation also answers another important question, how the application of the more accurate data modeling techniques influences the complexity of the modified entropy encoder and entropy decoder.

Other original results of the dissertation are:

1. Proposal of new method of joint application of data statistics estimation techniques based on Context-Tree Weighting (CTW) and "A" variant of Prediction with Partial Matching (PPMA) in contemporary adaptive entropy encoders (see Section 5.4.3);

2. Experimental investigations of the compression performance of the modified CABAC entropy encoders (with more sophisticated techniques of the conditional probabilities estimation based on CTW and/or PPMA) relative to the coding efficiency of the original CABAC (Chapter 6);
3. Experimental test of influence of the context length on the compression performance of the modified CABAC entropy encoders. Achieved experimental results proved that depending on the context length even 2% - 4.5% bitstream reduction is possible after application of the modified CABAC (with CTW) entropy encoder relative to the original CABAC encoder (Chapter 6);
4. Experimental investigations of the influence of data statistics initialization method on the compression performance of advanced entropy encoders (Chapter 6);
5. Experimental investigations of the coding efficiency of the M-arithmetic encoder core and the coding efficiency of the traditional arithmetic encoder core from H.263 video coding standard (Section 7.4);
6. Experimental comparison of the total entropy encoding times and the total entropy decoding times of the original CABAC entropy codec and the modified CABAC entropy codec with CTW technique (Section 8.3, Section 8.4.3). In the case of the modified CABAC entropy codec experiments have been done for different context lengths. Obtained experimental results showed that the total entropy decoding time for the modified CABAC is approximately 2.5 – 6.5 times greater in comparison to the total decoding time of the original CABAC entropy decoder. The total entropy encoding time for the modified CABAC entropy encoder is about 2 – 5.5 times greater relative to the total encoding time of the original CABAC entropy encoder. The total entropy decoding times as well as the total entropy encoding times obtained for the modified CABAC entropy codec were different for different context lengths. Additionally, experiments proved that total encoding/decoding times for both the original and the modified CABAC entropy codecs are dependent on the transmission bitrate;
7. Experimental comparison of total encoding and total decoding times for the original AVC with unmodified CABAC and the modified AVC with CABAC and CTW technique (Section 8.6.2). In the case of the modified AVC experiments have been done for different context lengths. For the bitrates less than 10 Mbits/s total decoding time increases up to 80% and total encoding time increases up to 2.5% after application of CTW in CABAC (for depth $D = 8$);

8. Proposal and implementation of the original architecture of the software version of CABAC encoder and CABAC decoder and the hardware version of CABAC decoder. Experimental investigations of throughput of both software and hardware version of CABAC decoder (Section 9.1 and Section 9.2);
9. Experimental investigations of the coding efficiency of CABAC entropy encoder with reference to the compression performance of the Universal Variable-length Coding (UVLC) method (based on Exp-Golomb coding and Context-Adaptive Variable Length Coding) (Section 4.3.1);
10. Experimental test of the complexity of CABAC entropy decoder relative to the complexity of author's proposal of fast UVLC entropy decoding (Section 4.3.2).

10.3. General conclusions

The dissertation answered the important question how much the application of more sophisticated techniques of adaptation of arithmetic coding may improve the compression performance of contemporary adaptive arithmetic coders used in advanced video coding. Experiments have been done with the state-of-the-art Context-based Adaptive Binary Arithmetic Coder (CABAC) that is used in Advanced Video Codec (AVC). Moreover, the dissertation discusses details of practical implementations of adaptive entropy encoders as well as adaptive entropy decoders in application to hybrid compression of video.

The obtained experimental results proved that improvement of adaptation of contemporary arithmetic coders that are used in video compression lead to a reasonable increase of the compression of entropy coding. The modified CABAC entropy coder with CTW data statistics estimation technique outperforms the original CABAC entropy coder by even 2% - 4.5%. In author's opinion it is a very good result. For comparison, in H.263 video coder, the optional, more efficient entropy coder based on arithmetic coding outperforms the simpler, VLC-based technique by approximately 5% [Côté98, Erol98] and it became a part of H.263 international video coding standard. Moreover, the improvement of compression performance of contemporary adaptive entropy coders is more and more difficult.

The gain in compression performance of the modified CABAC entropy coders relative to the original CABAC coder has been achieved even when simplified algorithm of context trees initialization has been used as compared to the standard CABAC coder. The

experimental results proved that the coding efficiency of the modified CABAC coders may be reasonably increased if more sophisticated technique of the context trees initialization is used.

The gain of coding efficiency of the modified CABAC coders relative to the compression performance of the original CABAC coder is heavily dependent on:

- The content of the test sequence that affects the probability distribution of coded data;
- The value of QP parameter that influences on data statistics and the size of data stream within a slice;
- The depth D of context trees that defines the number of previously coded symbols used to estimation of the conditional probability for the successive source symbol.

The increase of compression in adaptive entropy coders is obtained at a cost of higher complexity of both the modified entropy encoder and the modified entropy decoder. Complexity of the modified CABAC with CTW strongly depends on the depth D of context trees. Application in CABAC of CTW with context trees of depth $D = 8$ gives good results and extends AVC encoding time up to 2.5% and AVC decoding time up to 80% when operating on bitrates less than 10 Mbits/s.

The obtained experimental results in the dissertation well correspond to those achieved for other contemporary compression improvements. Comparing two the state-of-the-art entropy coders used in the advanced hybrid video coding (UVLC technique and CABAC technique), higher compression performance of CABAC relative to UVLC (the bitrate reduction between 6% and 20%) has been also achieved by significantly increasing of the complexity of entropy coding. The CABAC decoding time is 30% - 130% higher than UVLC decoding time. As a matter of fact, this increase of complexity leads to higher gain of compression performance. Nevertheless, the improvement of more and more advanced entropy coders is more and more difficult.

The dissertation also reveals that entropy coders used in contemporary video coders require a considerable amount of computations to encode or decode a single symbol. High complexity of advanced entropy coders is mainly the result of application of sophisticated mechanisms of the conditional probabilities estimation. It causes that the real-time entropy encoding and decoding in the case of High Definition Television (HDTV) video sequences (with transmission bitrates greater than 10 Mbits/s) is a great challenge even for today's high performance digital media processors. Therefore, the proposal of optimized architecture of advanced entropy coders that will enable real-time processing of bitstreams with transmission

bitrates of the order of tens mega bits per second is a difficult task today that makes a challenge for software designers.

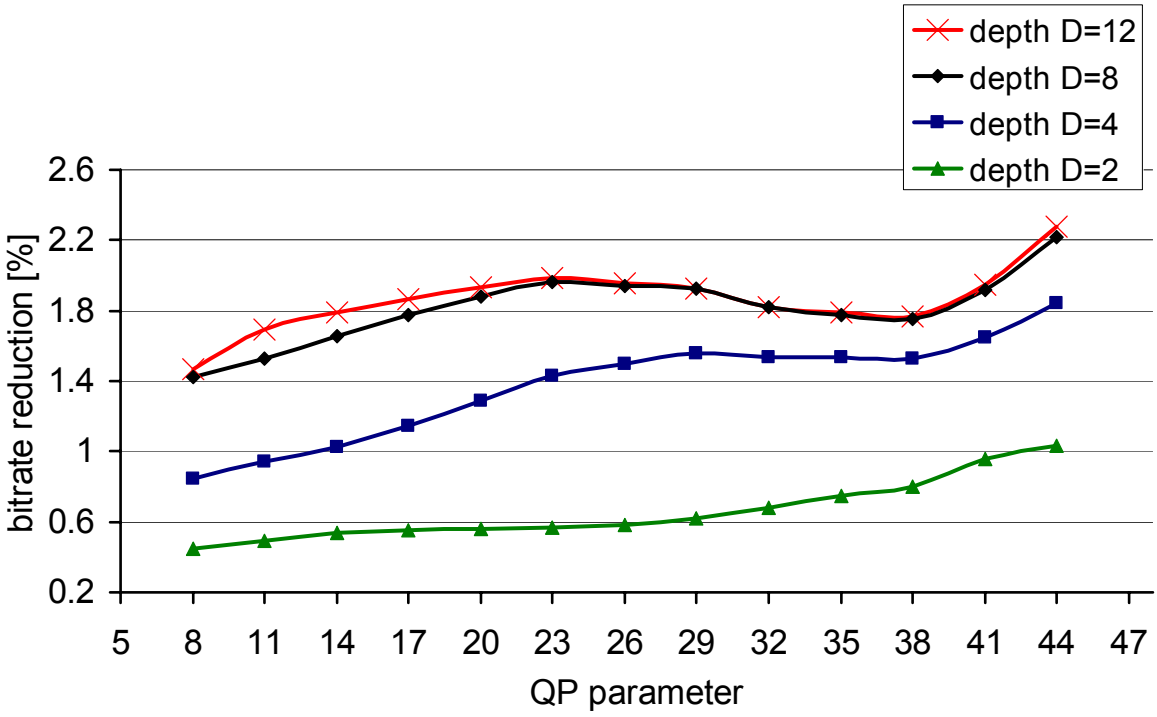
Annex A

Compression performance of the modified AVC with CABAC and CTW relative to the original AVC

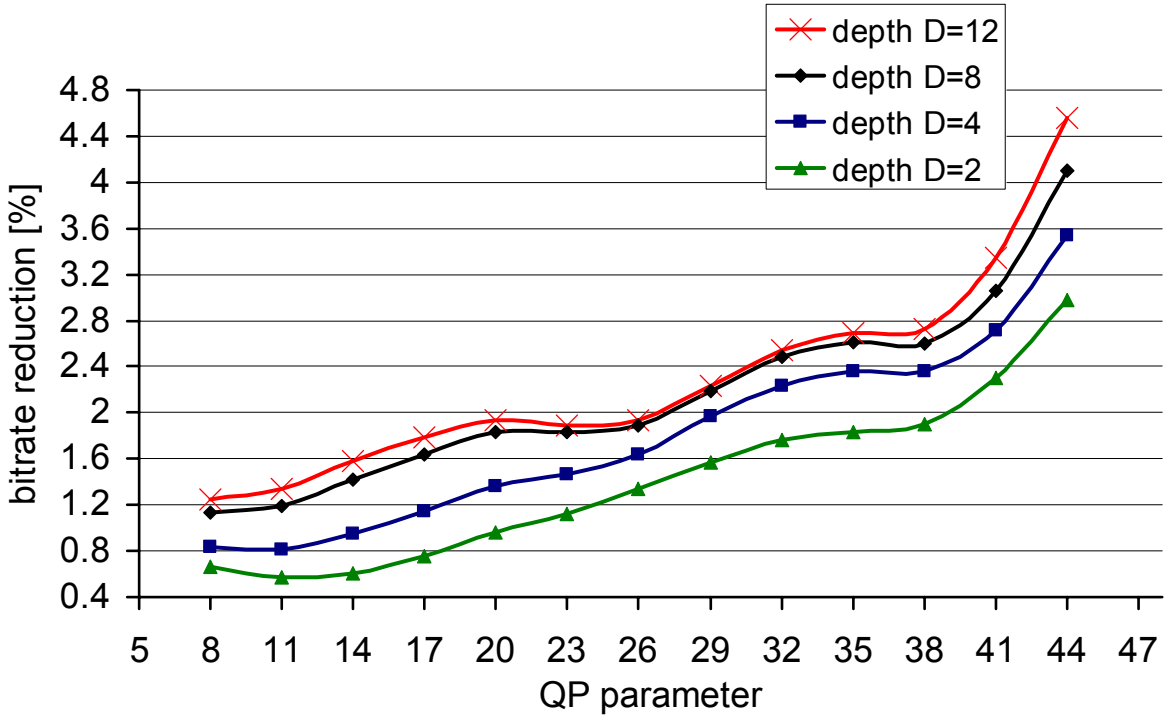
A.1. Experimental results for 4CIF test sequences and I29P structure of GOP

In this section, the detailed experimental results on the coding efficiency of both the original AVC video codec (with standard CABAC entropy codec) and the modified AVC video codec (with modified CABAC that exploits the CTW technique) have been presented. Experiments have been done according to **Scenario 1** (see Section 6.6).

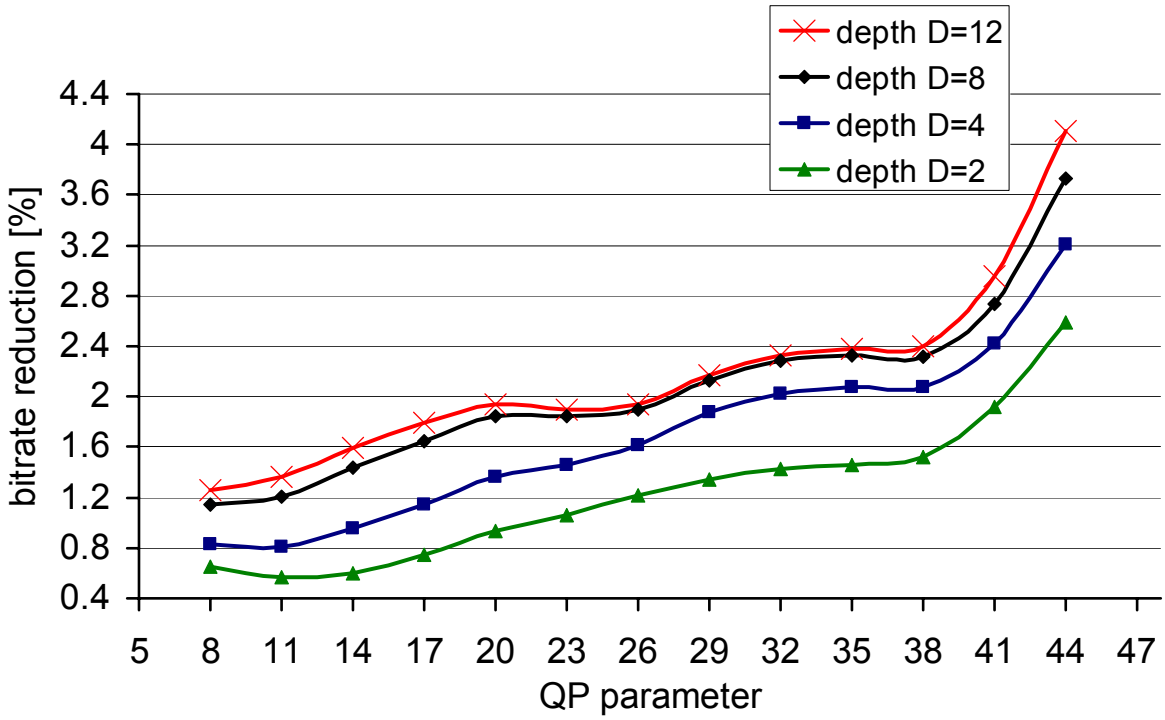
A.1.1. Experimental results for CITY test sequence



(a)



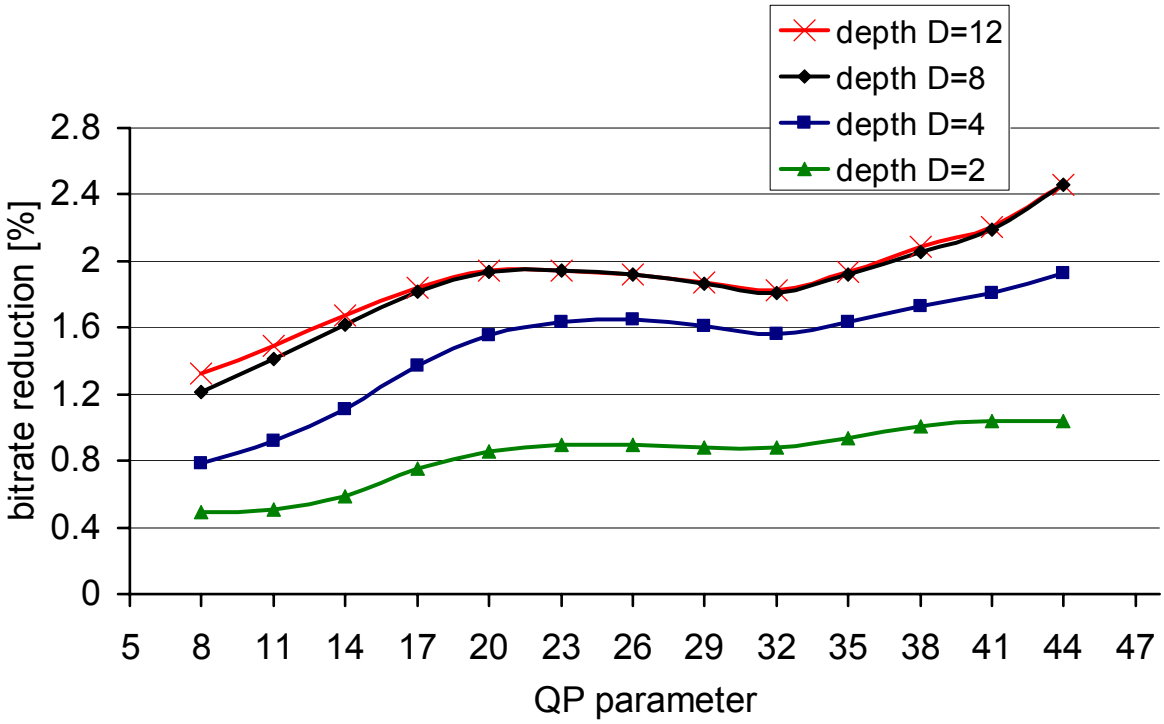
(b)



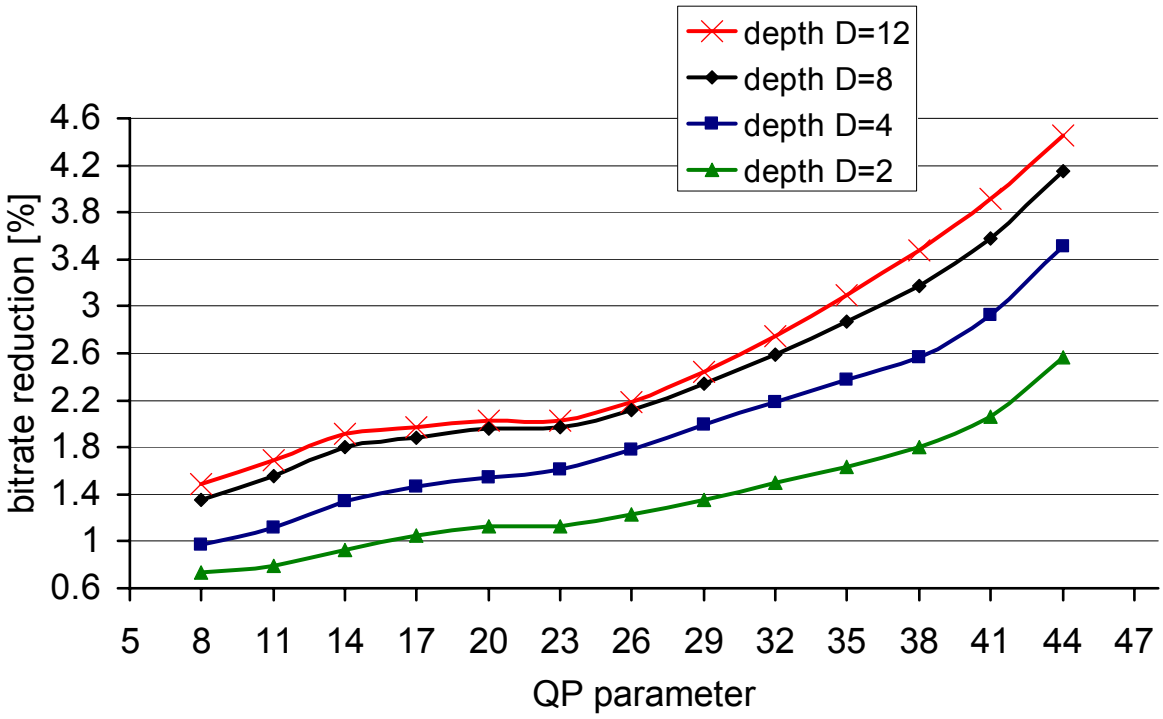
(c)

Figure A.1. Bitrate reduction achieved for the CITY test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and CTW technique in contrast to the original AVC with unmodified CABAC.

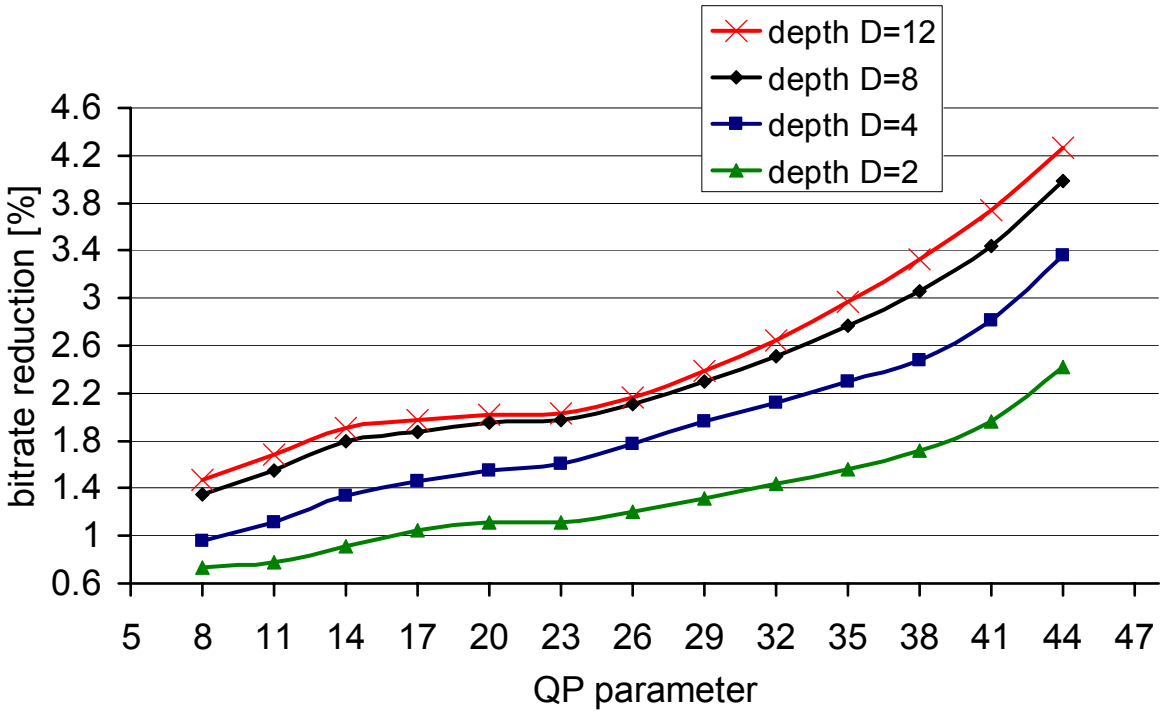
A.1.2. Experimental results for CREW test sequence



(a)



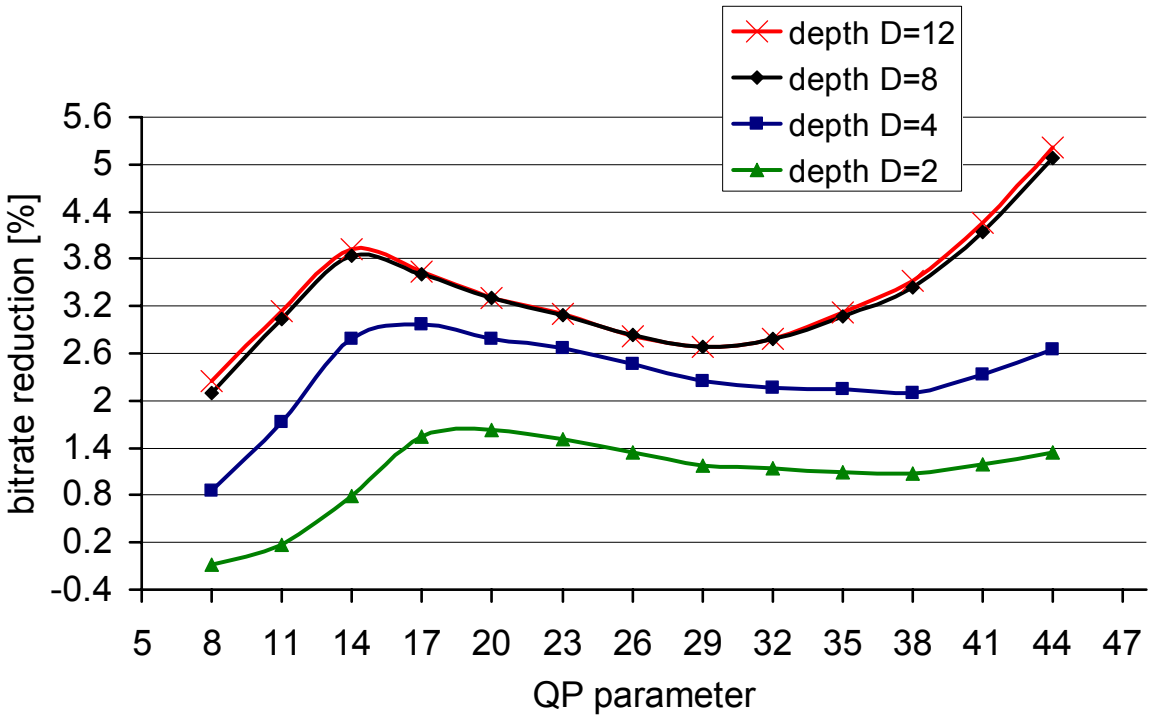
(b)



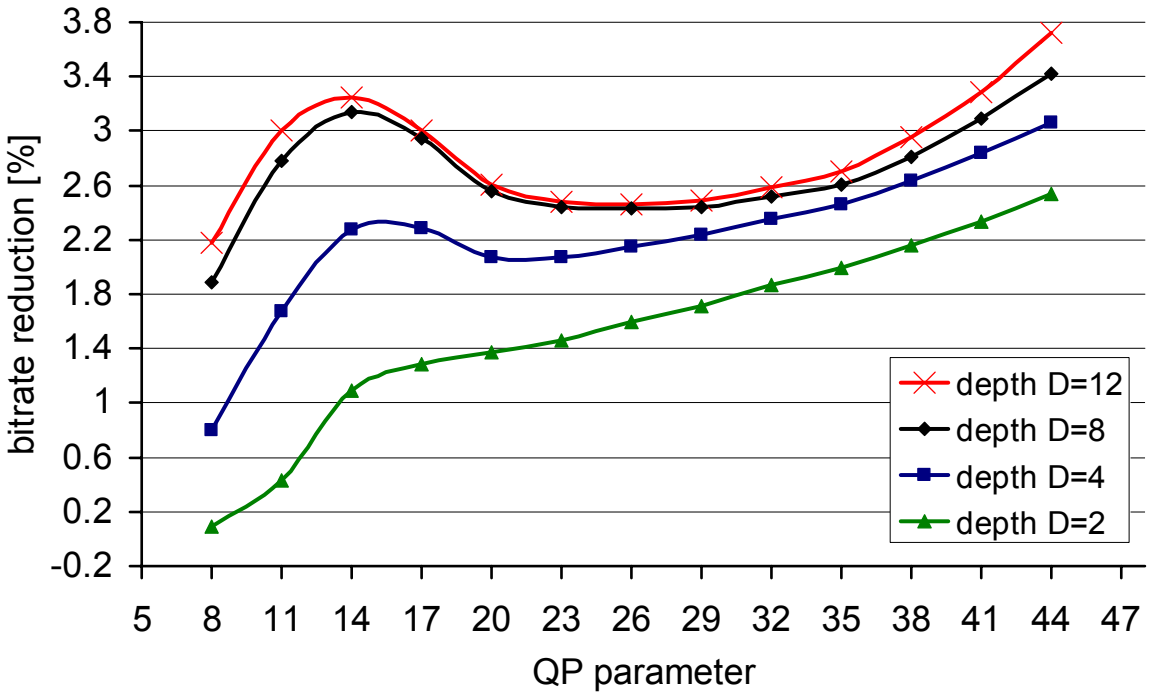
(c)

Figure A.2. Bitrate reduction achieved for CREW test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and CTW technique in contrast to the original AVC with unmodified CABAC.

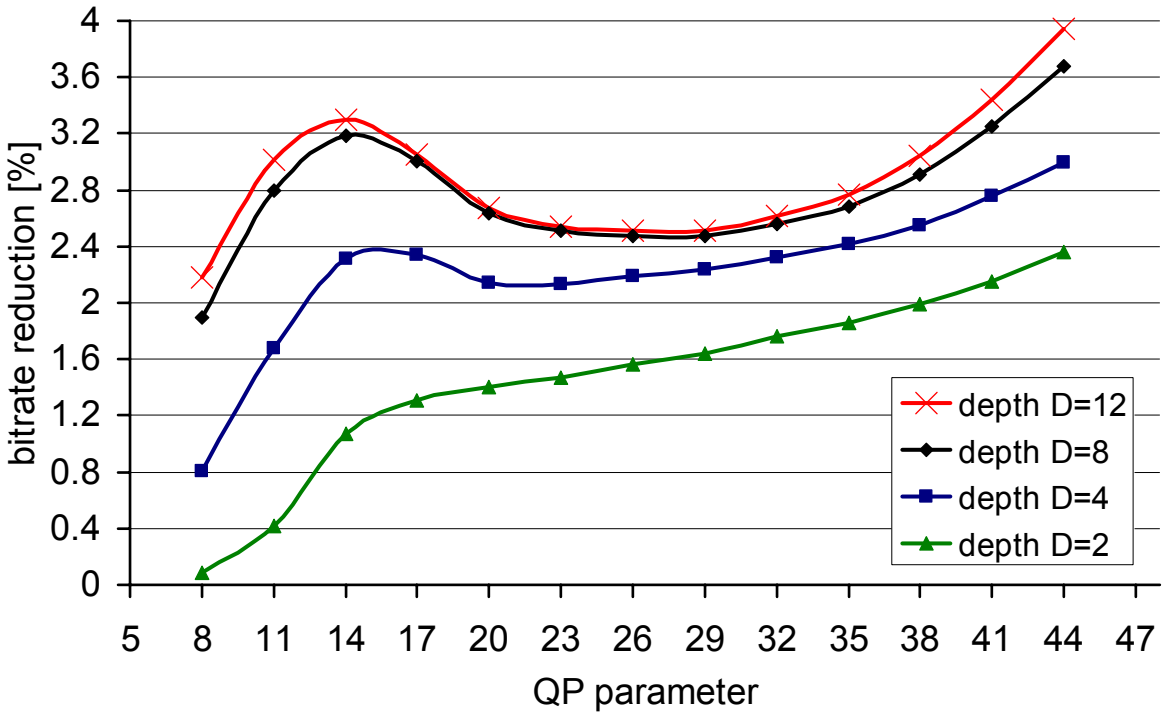
A.1.3. Experimental results for ICE test sequence



(a)



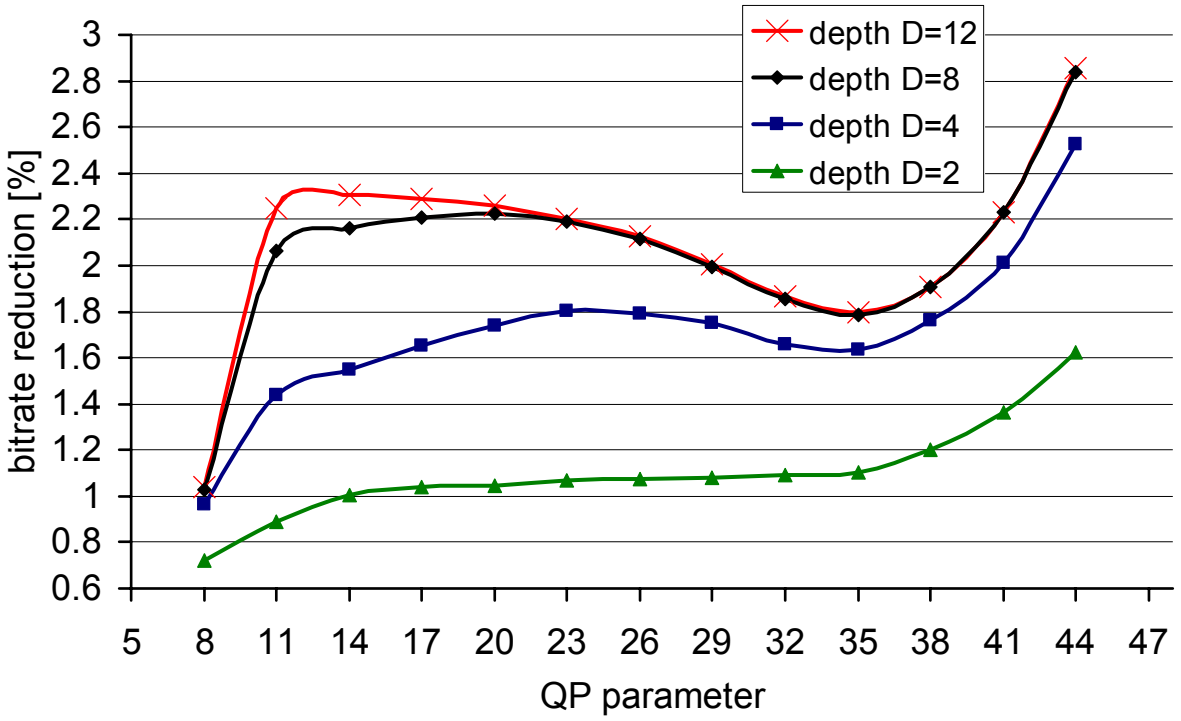
(b)



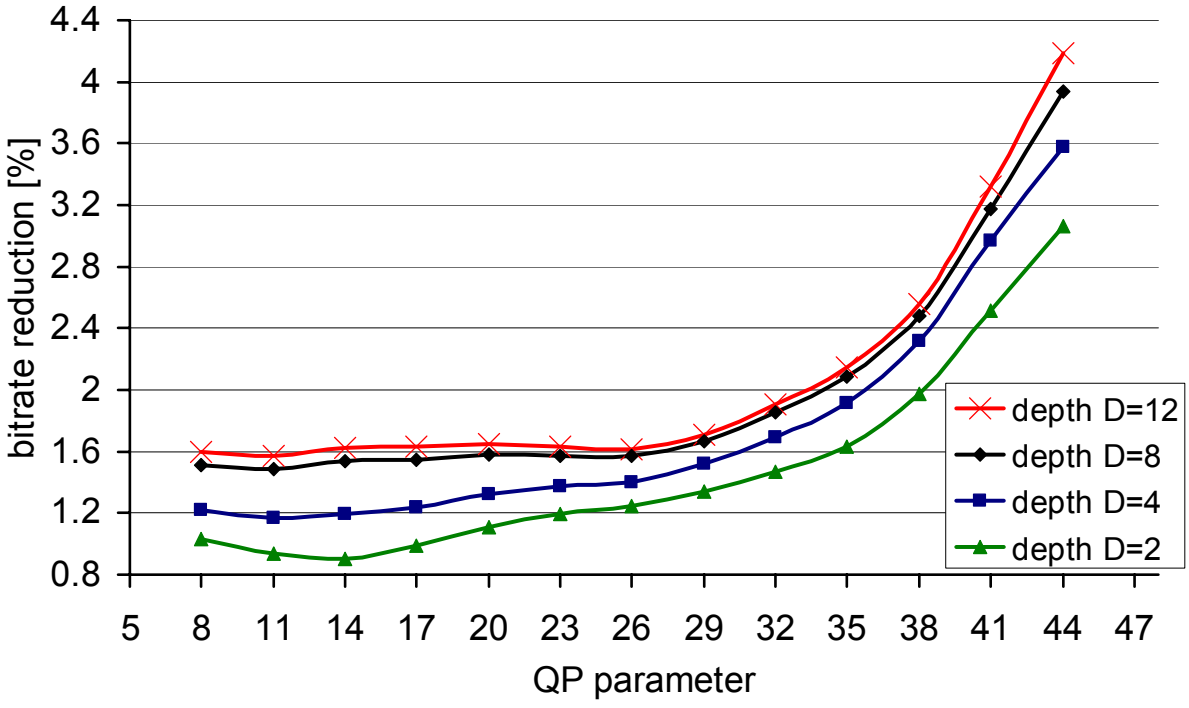
(c)

Figure A.3. Bitrate reduction achieved for ICE test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

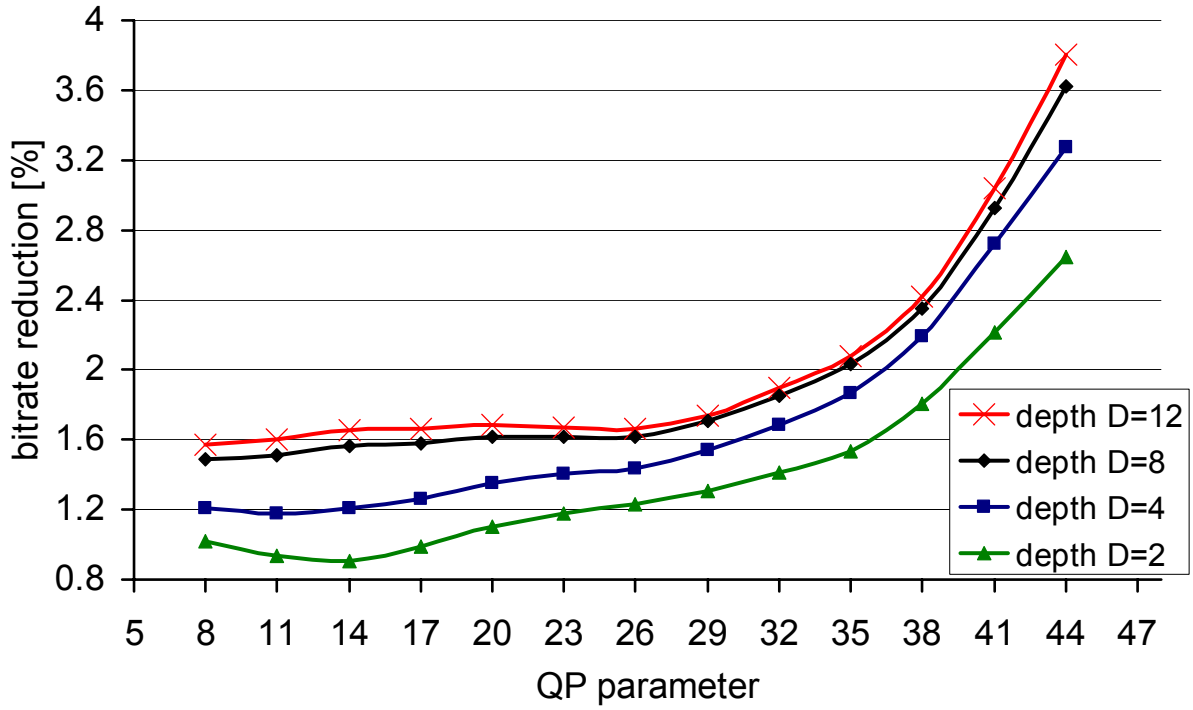
A.1.4. Experimental results for HARBOUR test sequence



(a)



(b)



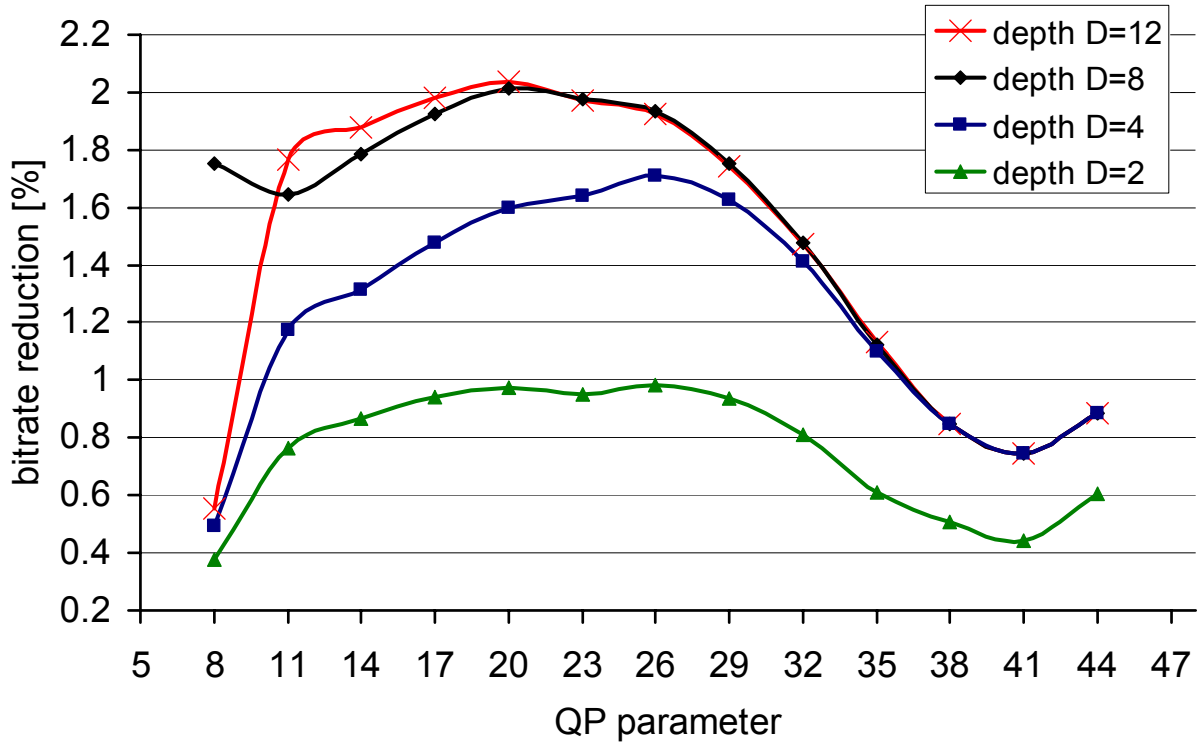
(c)

Figure A.4. Bitrate reduction achieved for HARBOUR test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

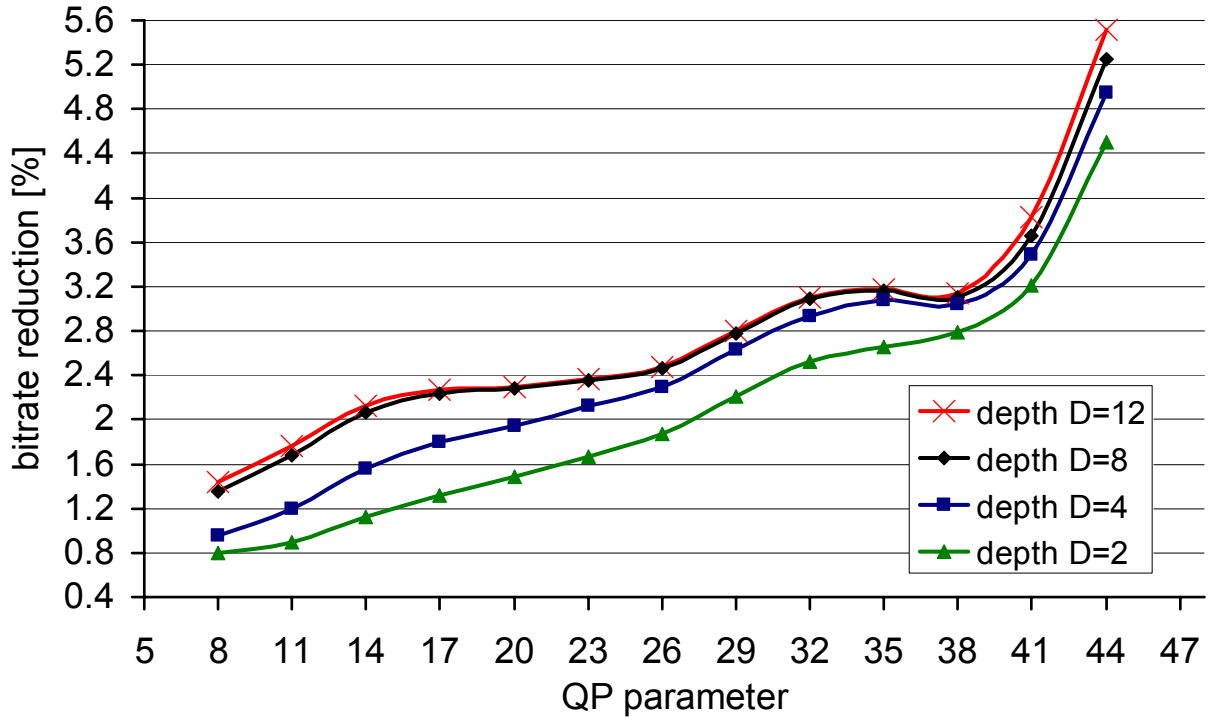
A.2. Experimental results for CIF test sequences and I29P structure of GOP

This section presents detailed experimental results on the compression performance of both the original and the modified AVC coder (with CABAC and CTW). Experiments have been done according to **Scenario 2** (see Section 6.6).

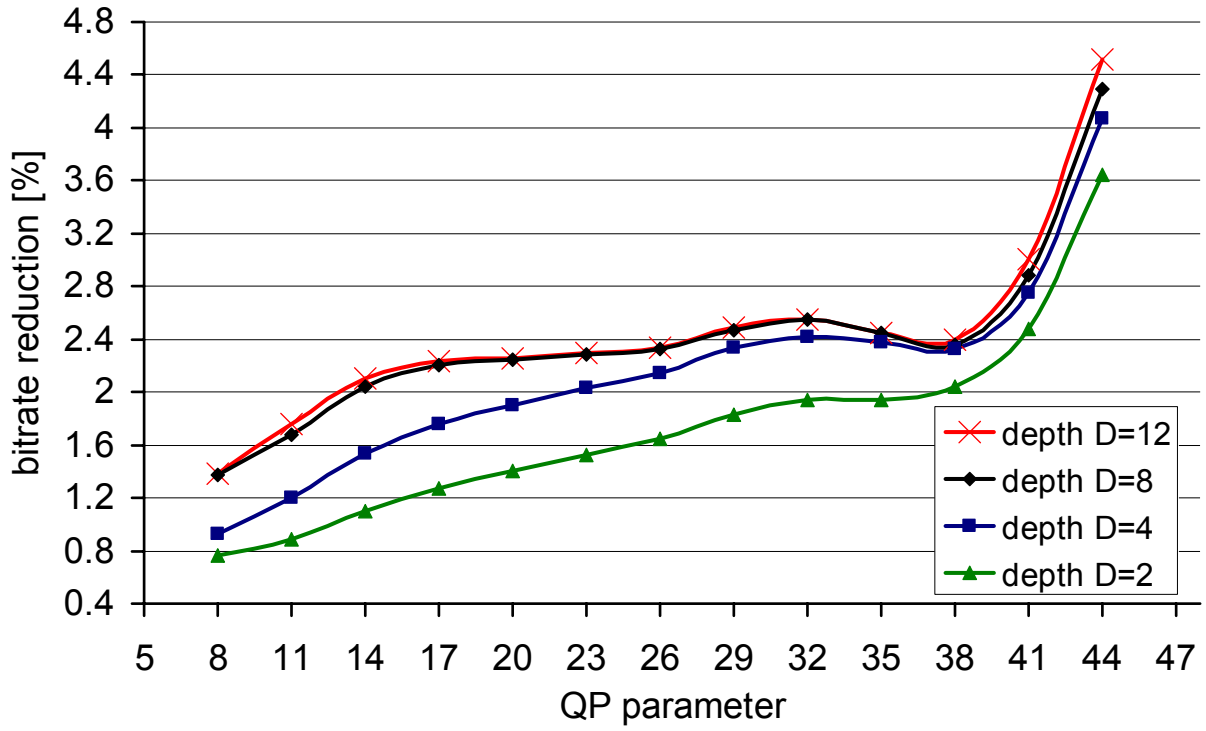
A.2.1. Experimental results for CITY test sequence



(a)



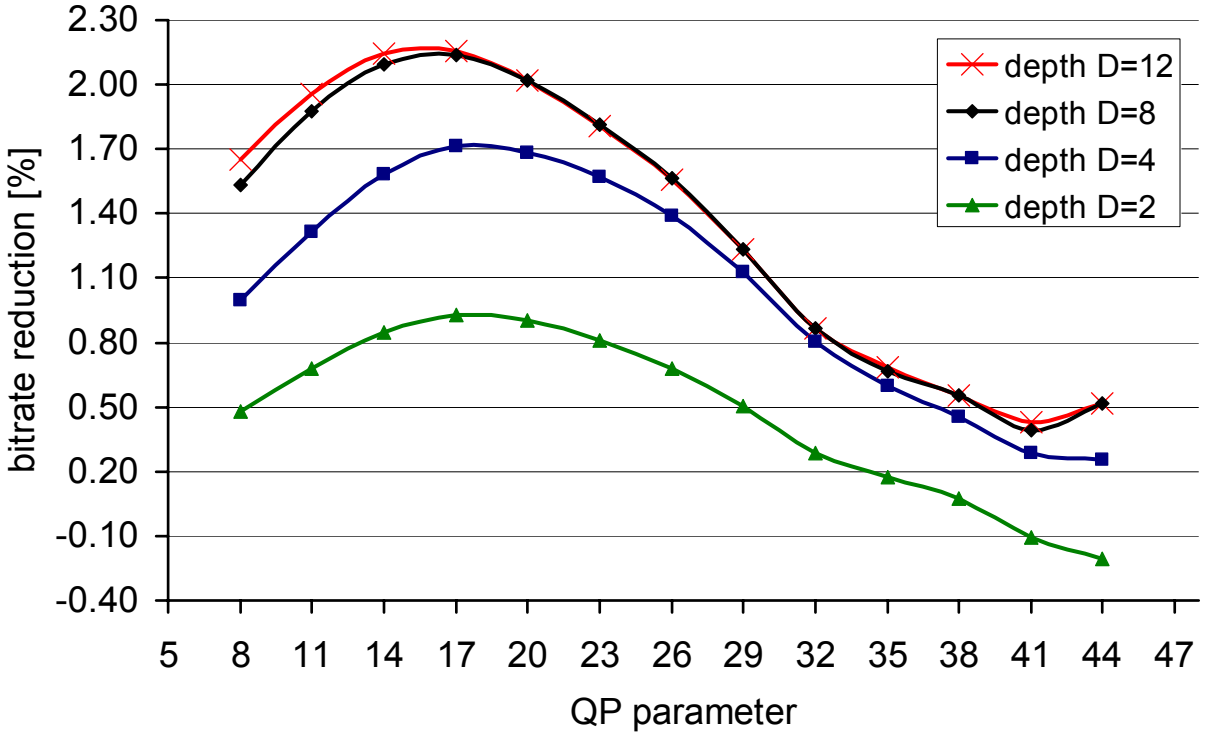
(b)



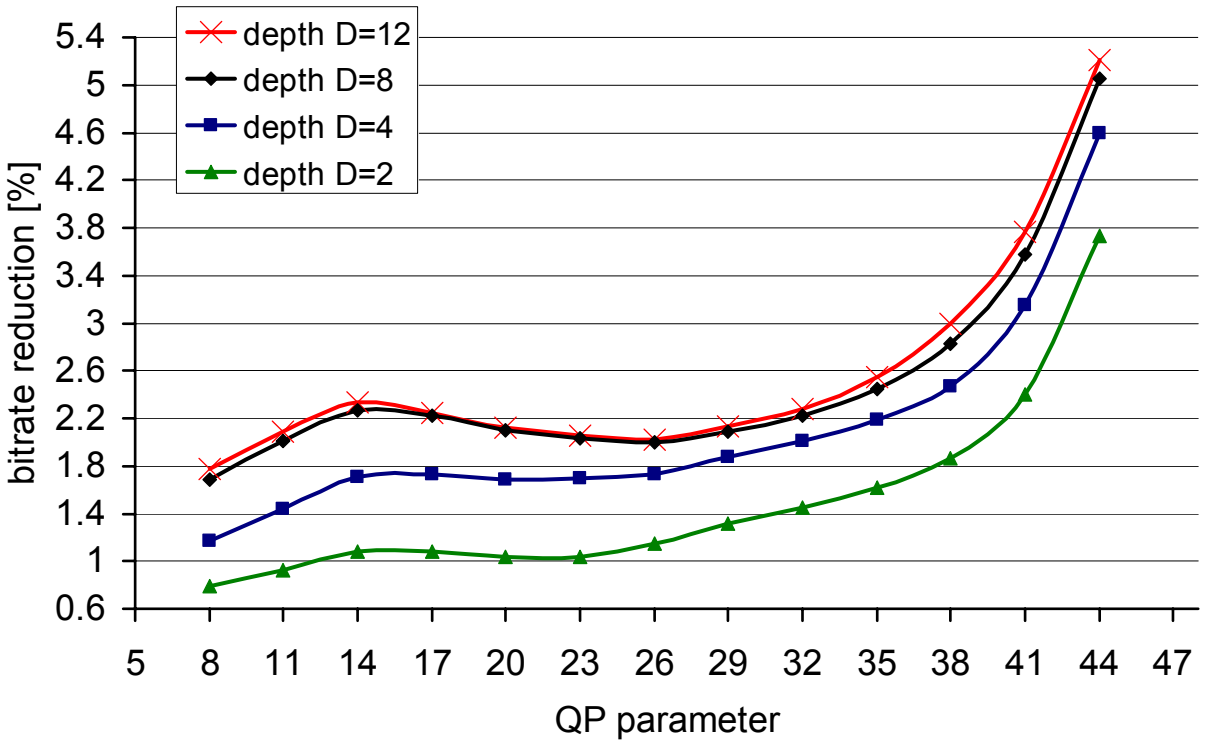
(c)

Figure A.5. Bitrate reduction achieved for CITY test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

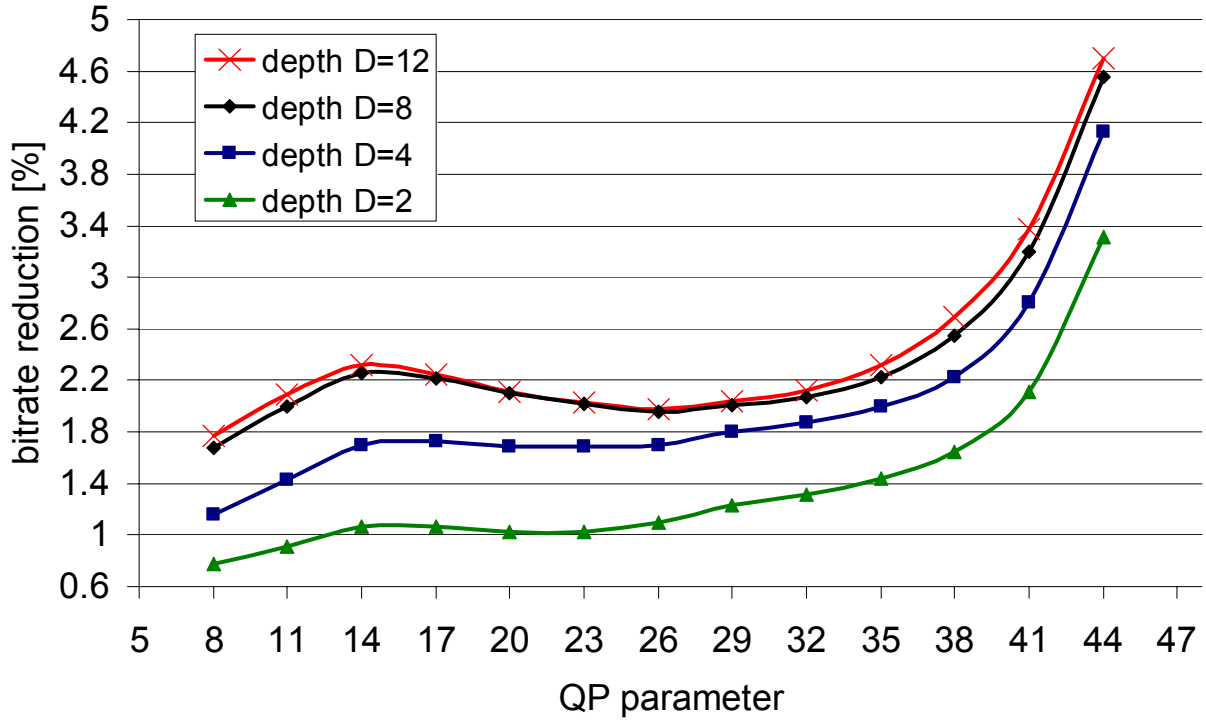
A.2.2. Experimental results for CREW test sequence



(a)



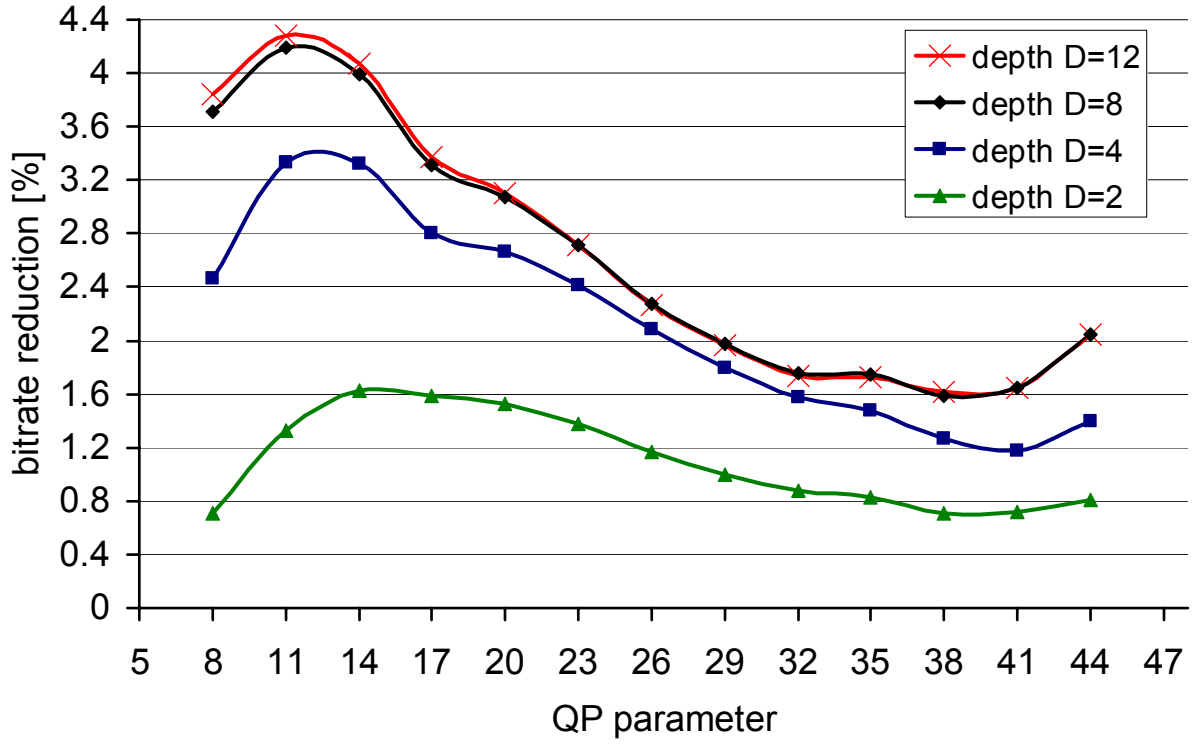
(b)



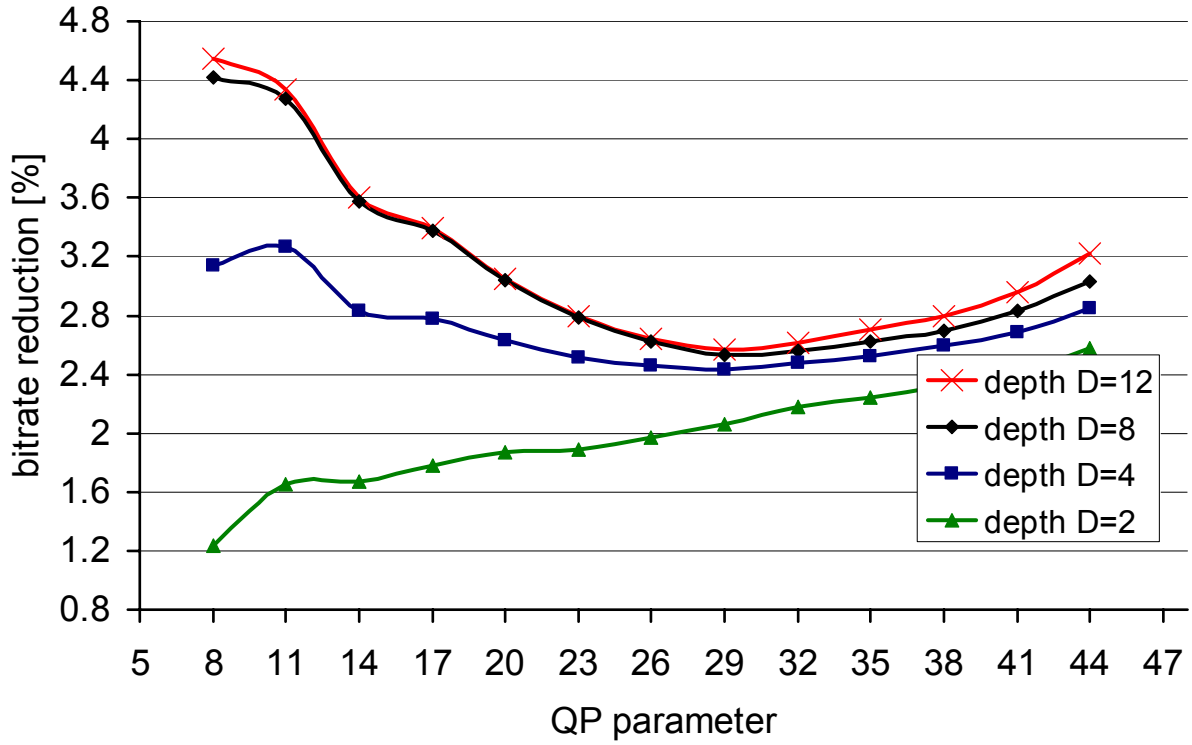
(c)

Figure A.6. Bitrate reduction achieved for CREW test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

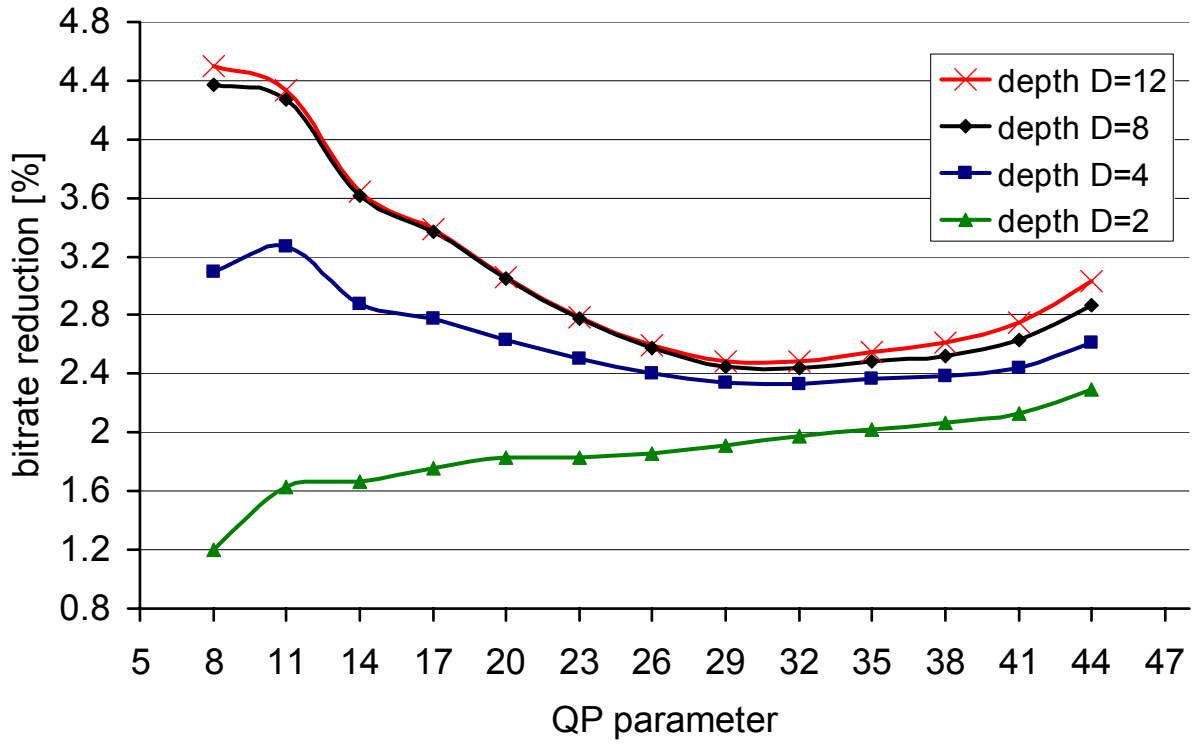
A.2.3. Experimental results for ICE test sequence



(a)



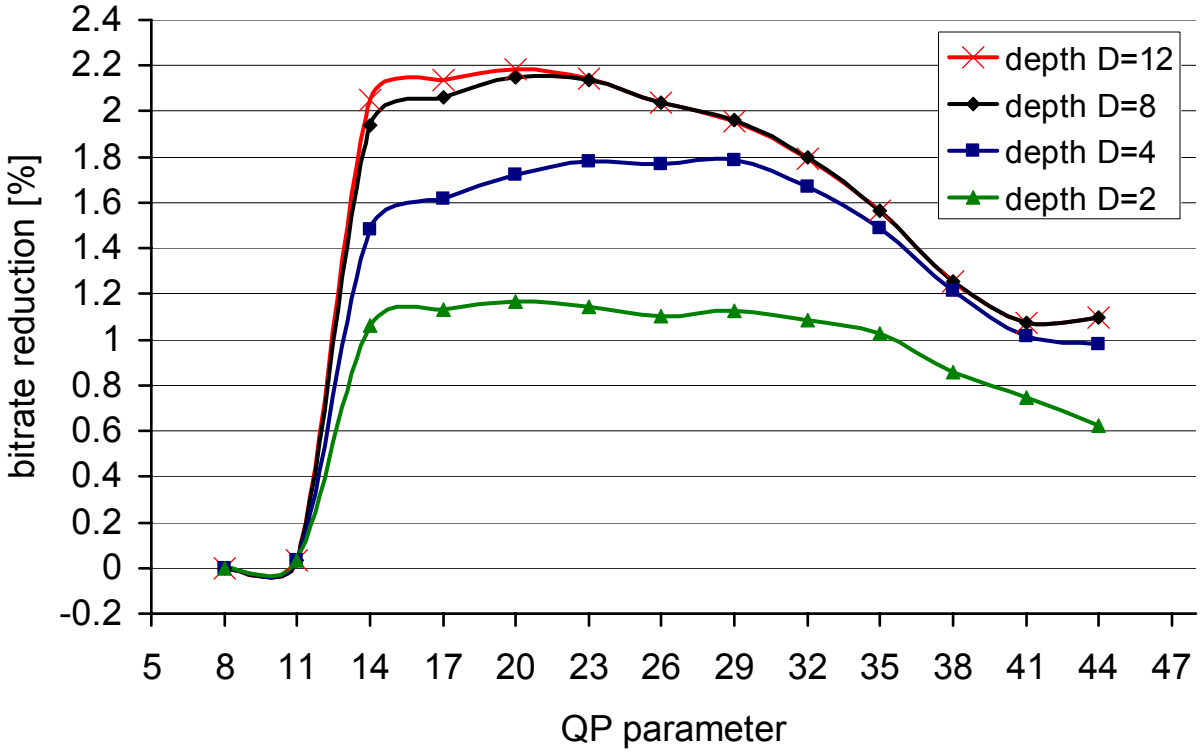
(b)



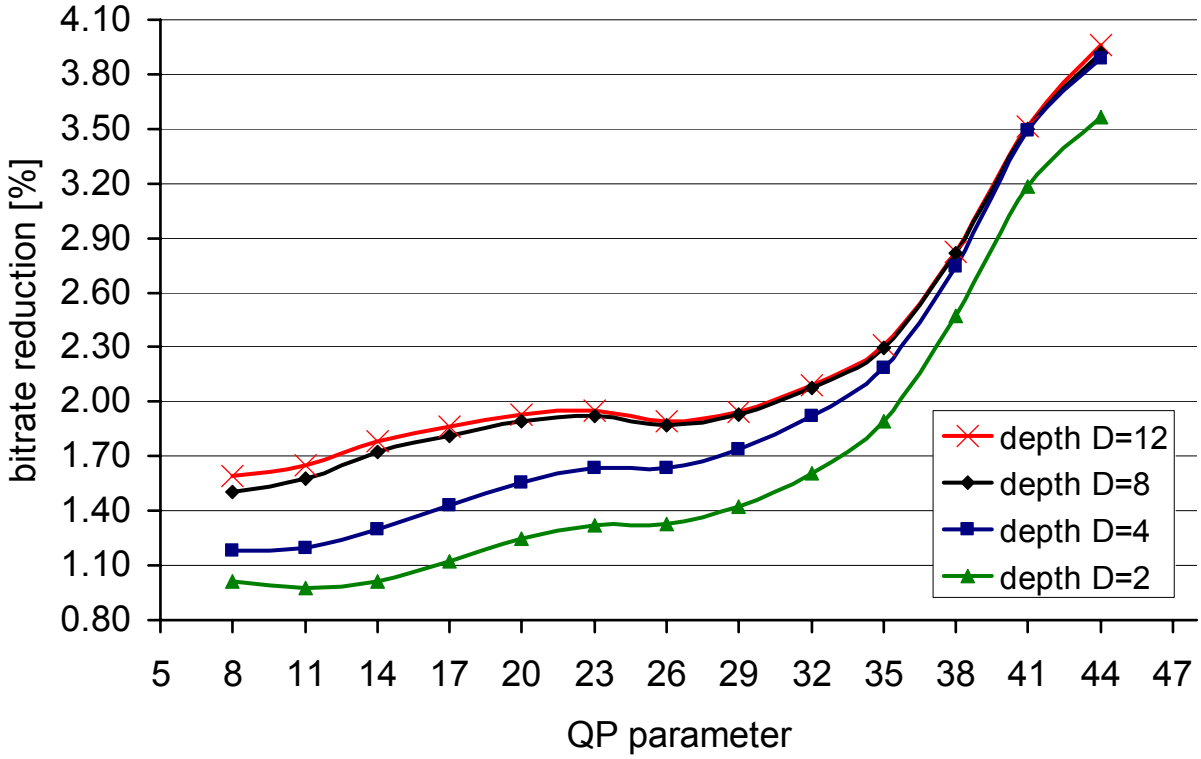
(c)

Figure A.7. Bitrate reduction achieved for ICE test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

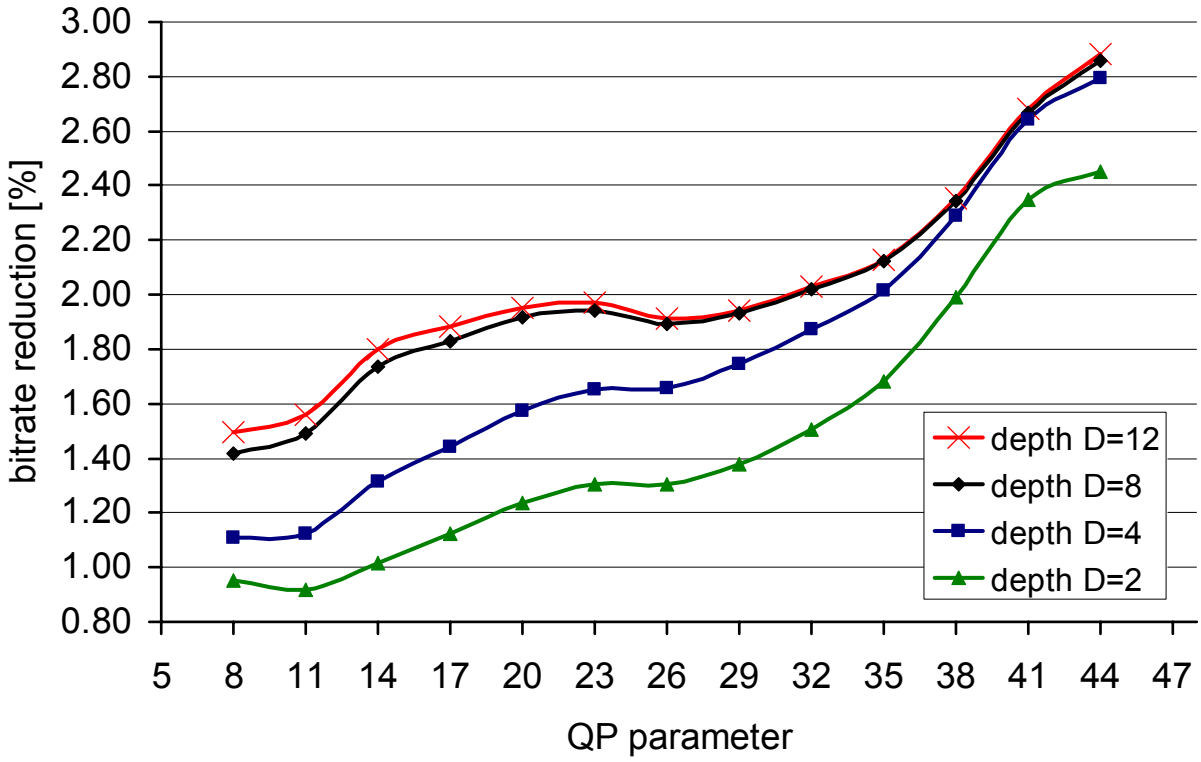
A.2.4. Experimental results for HARBOUR test sequence



(a)



(b)



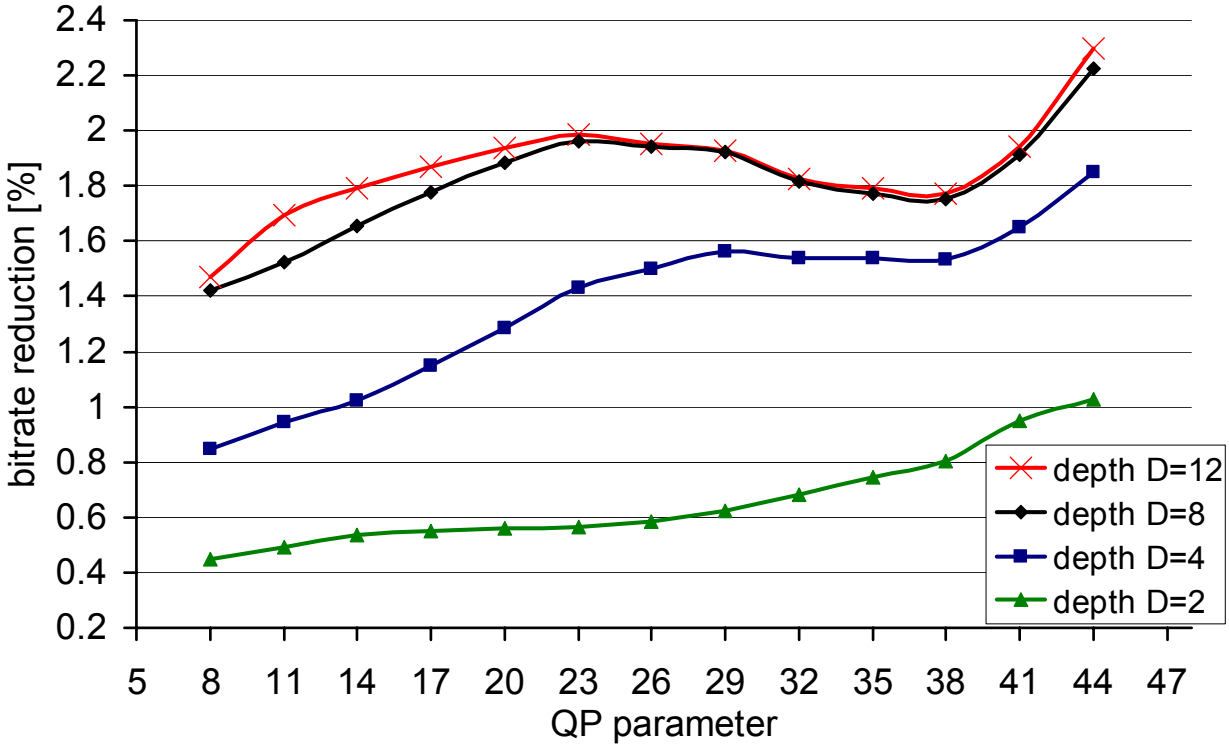
(c)

Figure A.8. Bitrate reduction achieved for HARBOUR test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

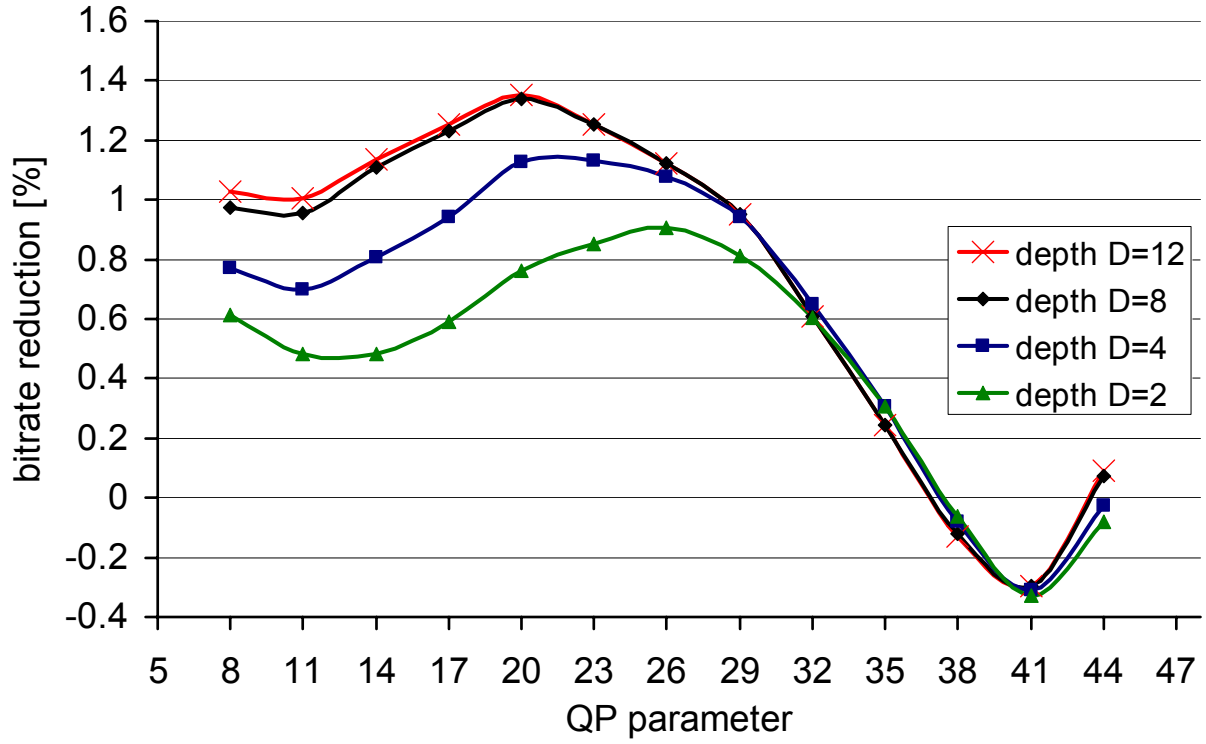
A.3. Experimental results for 4CIF test sequences and IBBPBBP… structure of GOP

This section presents detailed experimental results on the coding efficiency of the original AVC (with unmodified CABAC) and the modified AVC (with CABAC that exploits CTW technique). Experiments have been done according to **Scenario 3** (see Section 6.6).

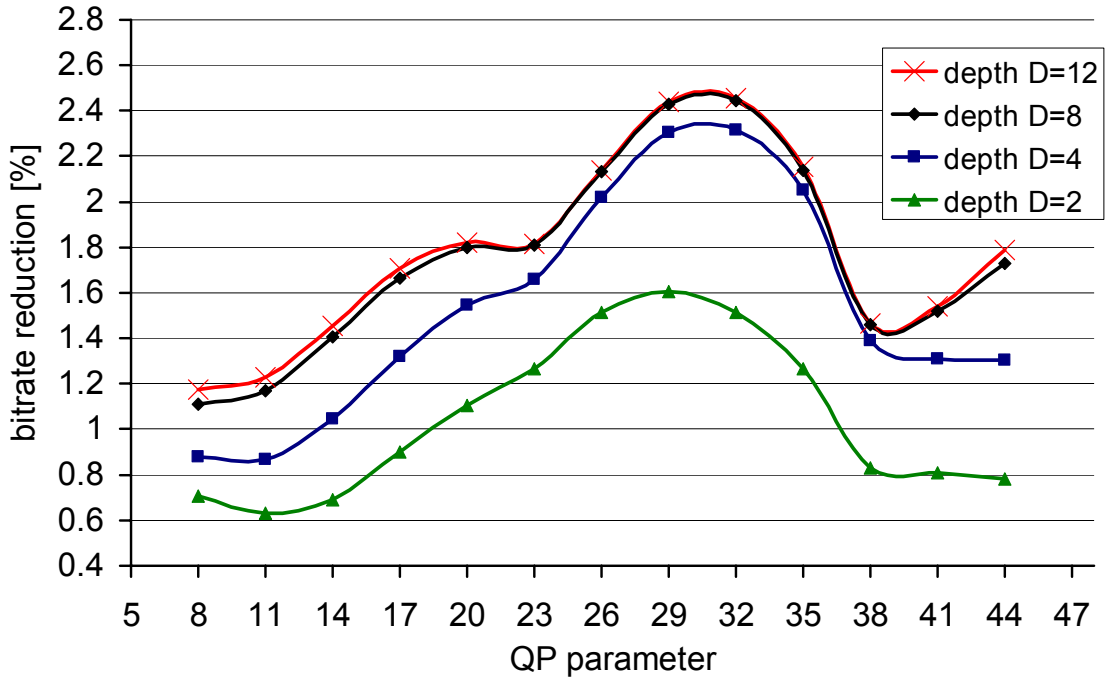
A.3.1. Experimental results for CITY test sequence



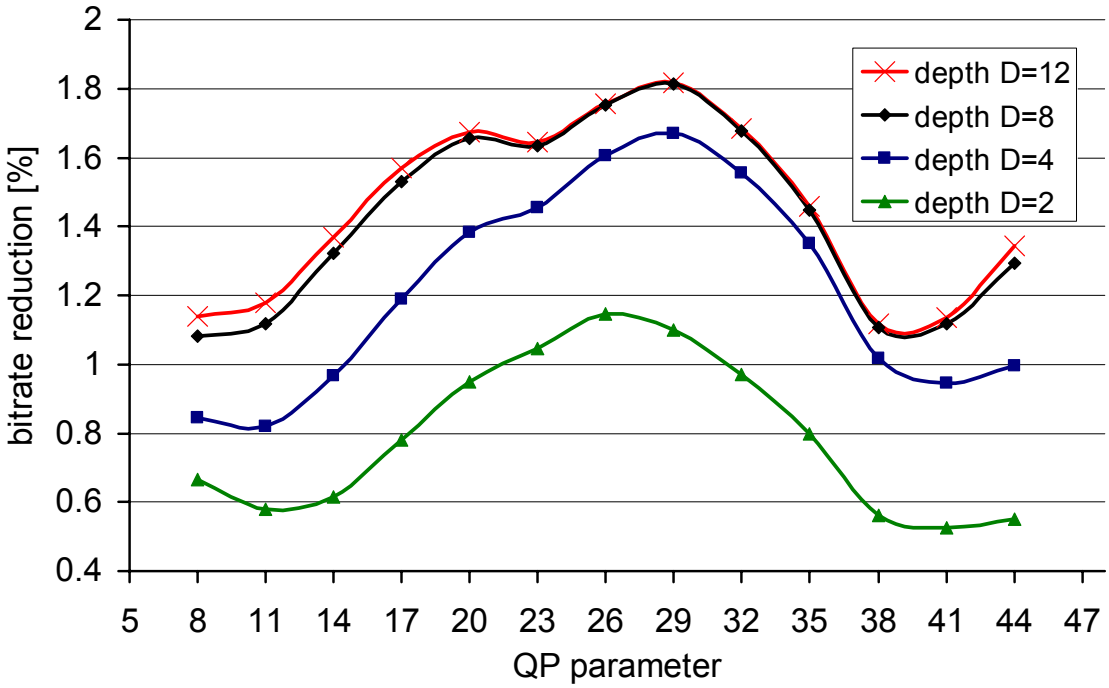
(a)



(b)



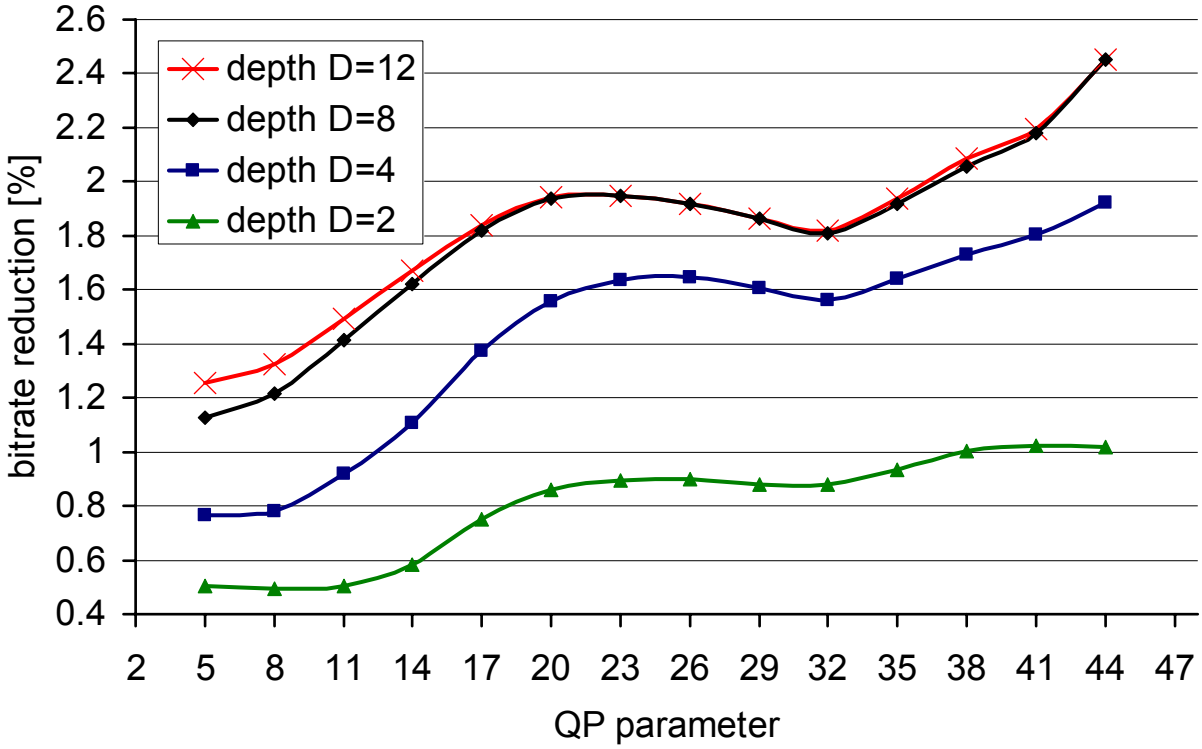
(c)



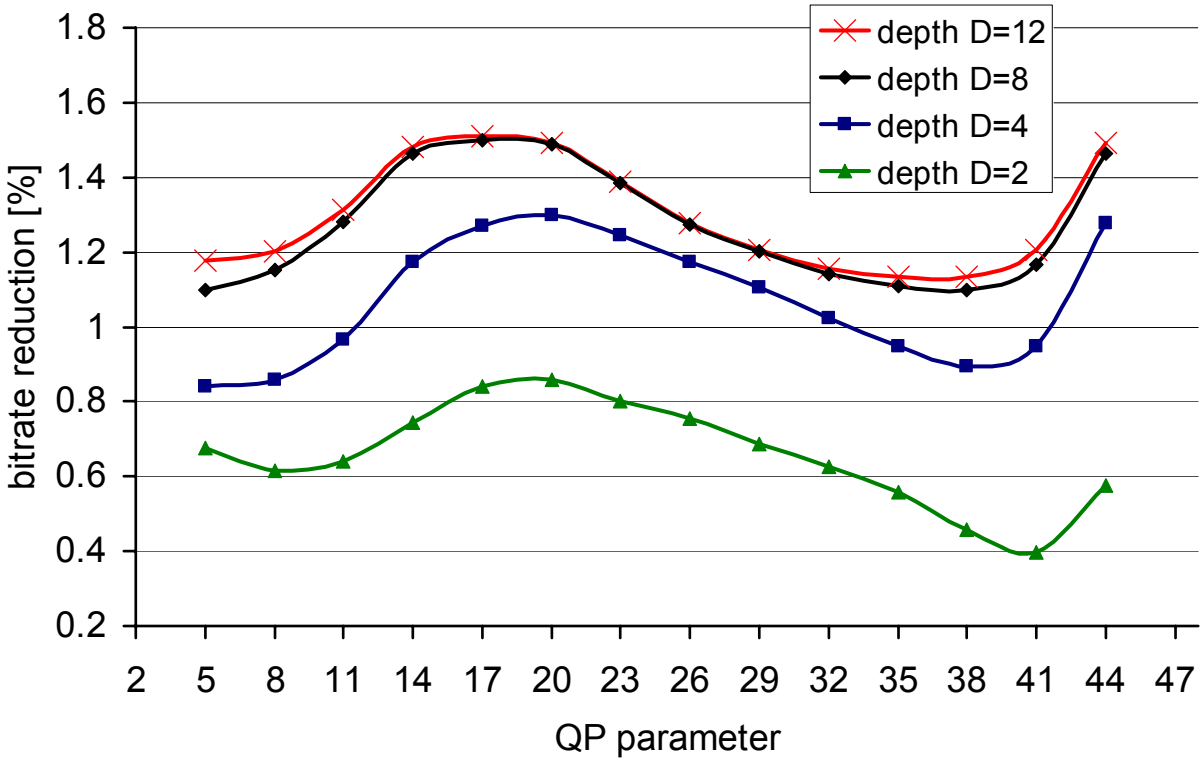
(d)

Figure A.9. Bitrate reduction achieved for CITY test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

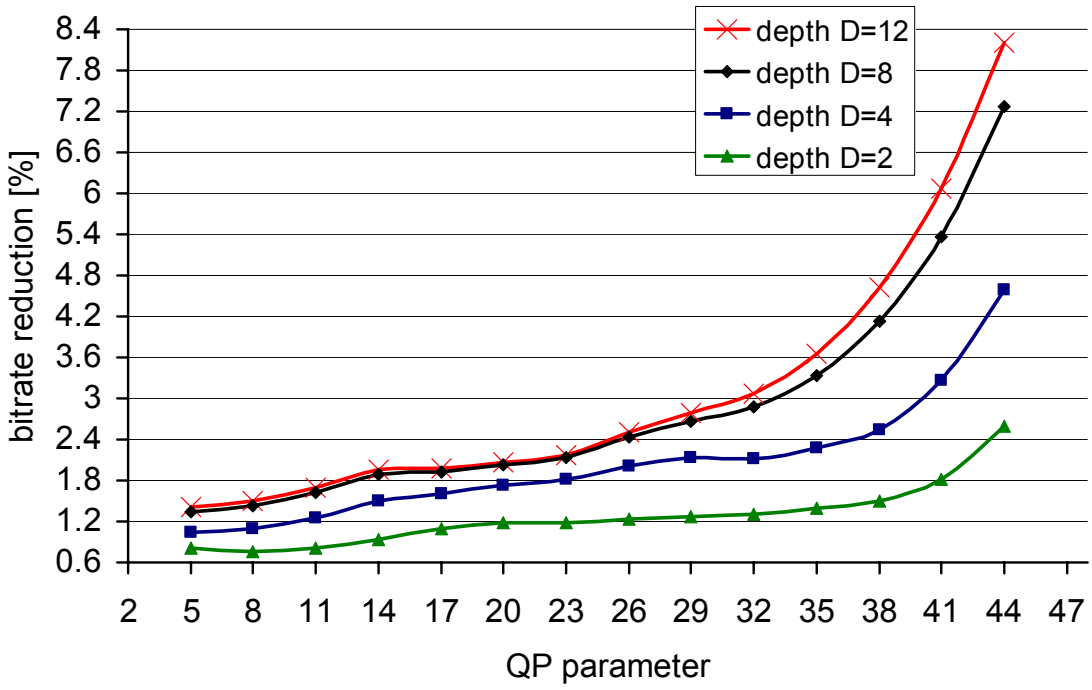
A.3.2. Experimental results for CREW test sequence



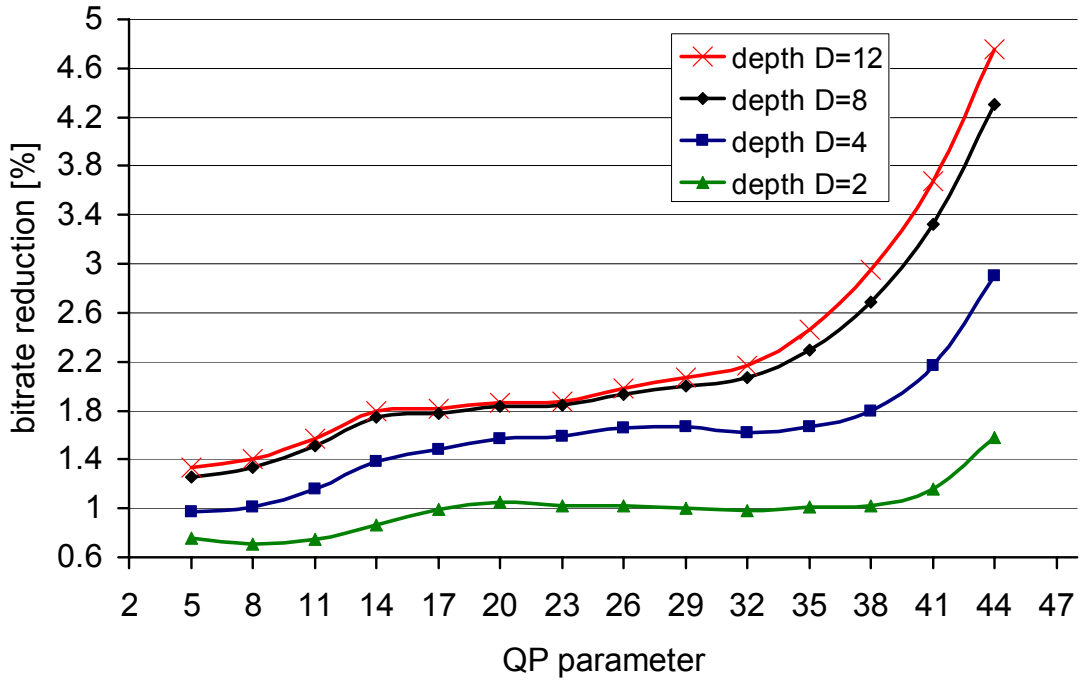
(a)



(b)



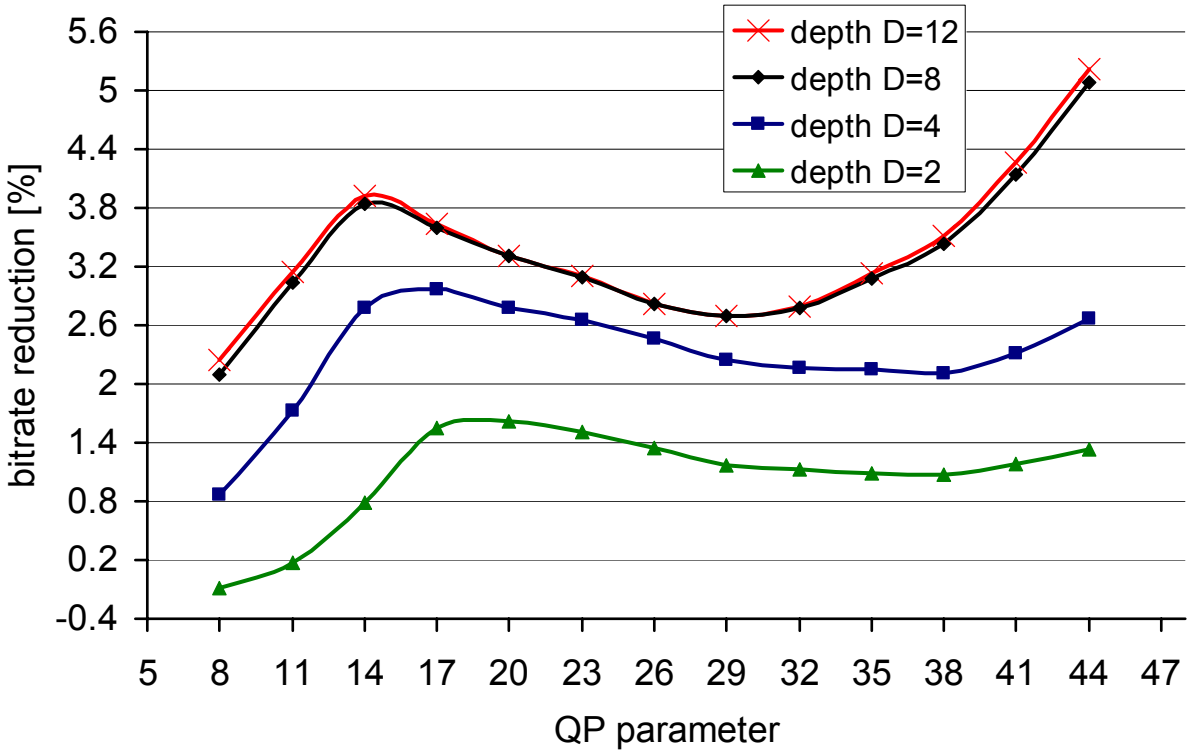
(c)



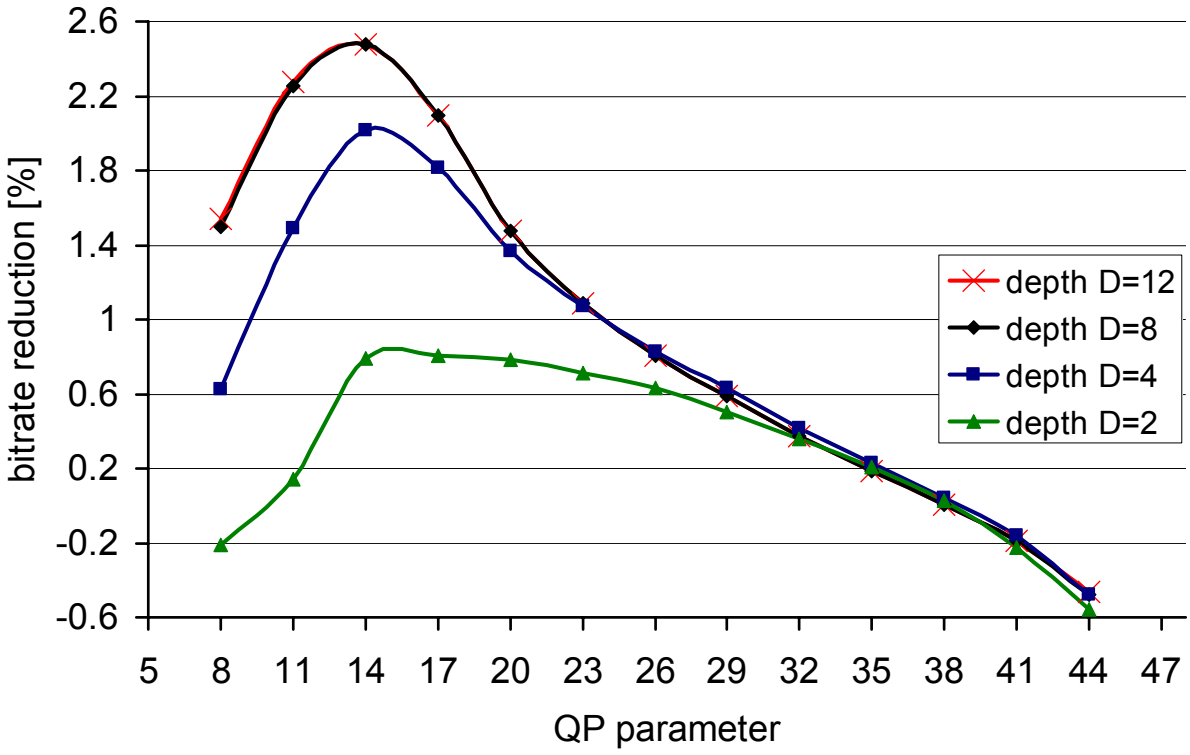
(d)

Figure A.10. Bitrate reduction achieved for CREW test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

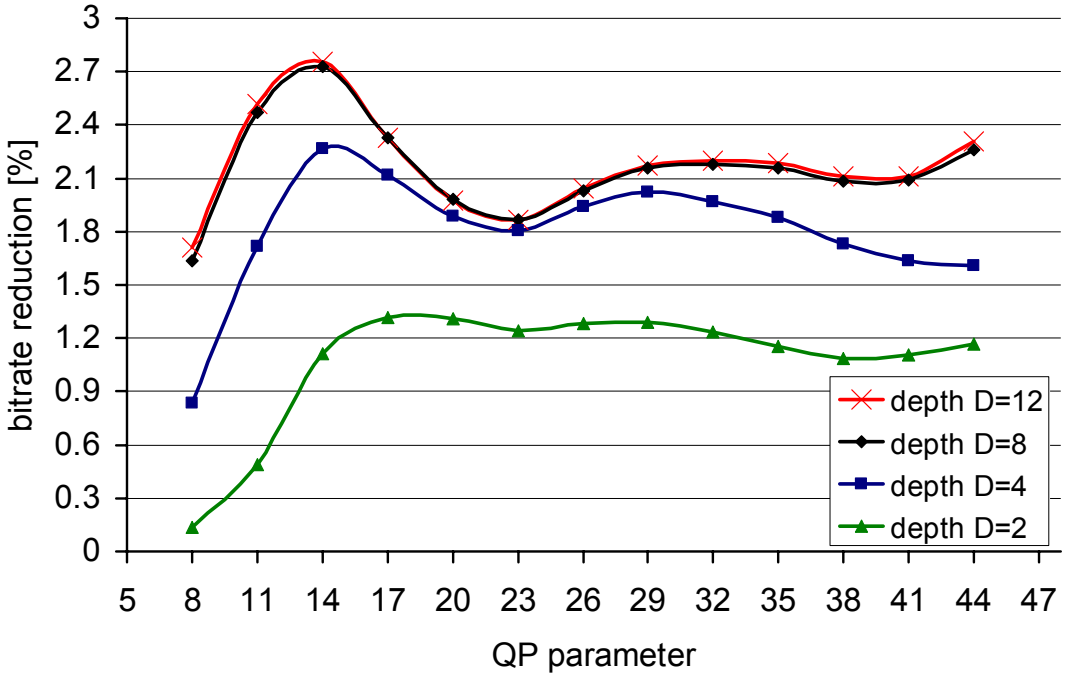
A.3.3. Experimental results for ICE test sequence



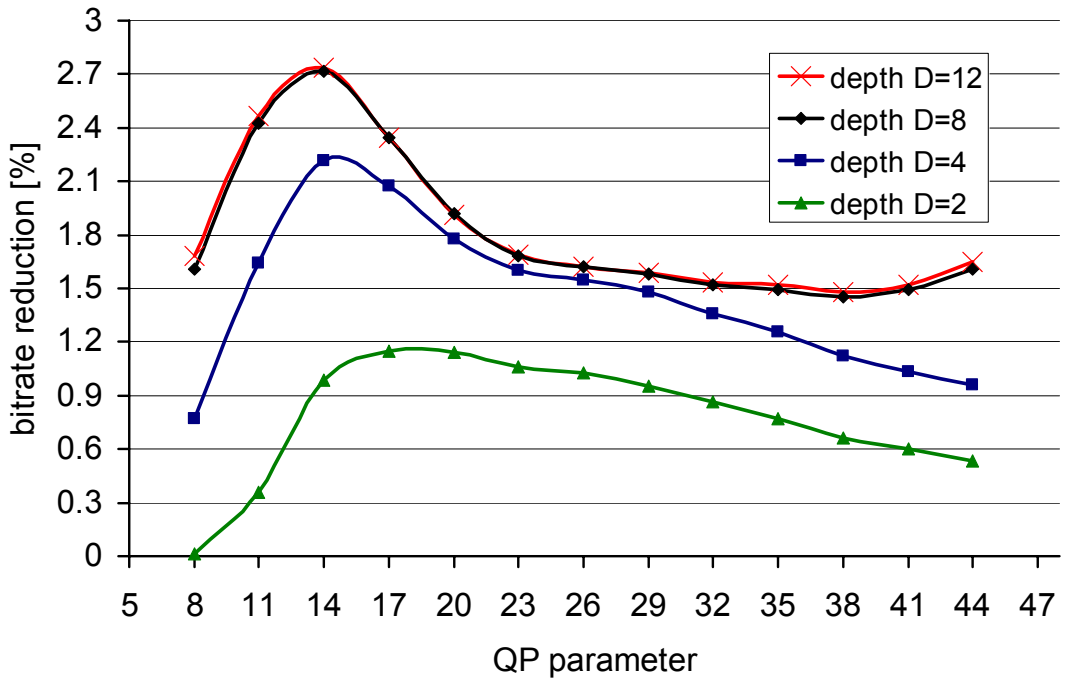
(a)



(b)



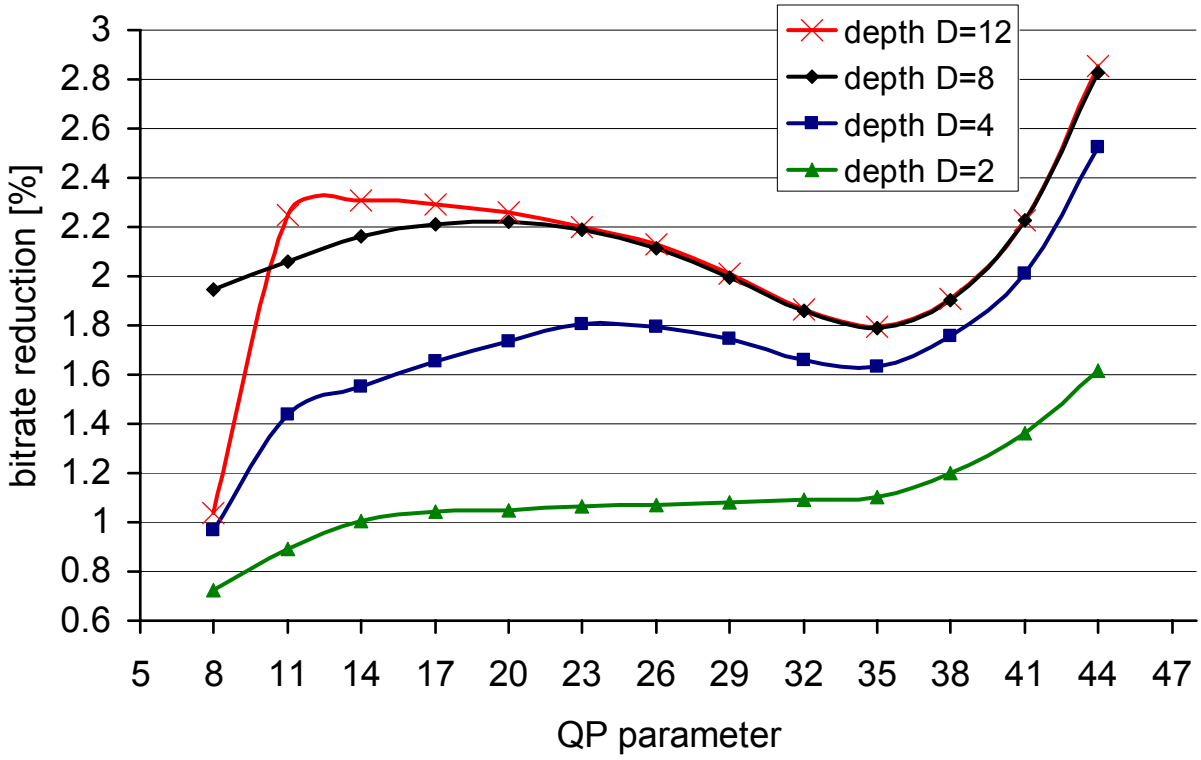
(c)



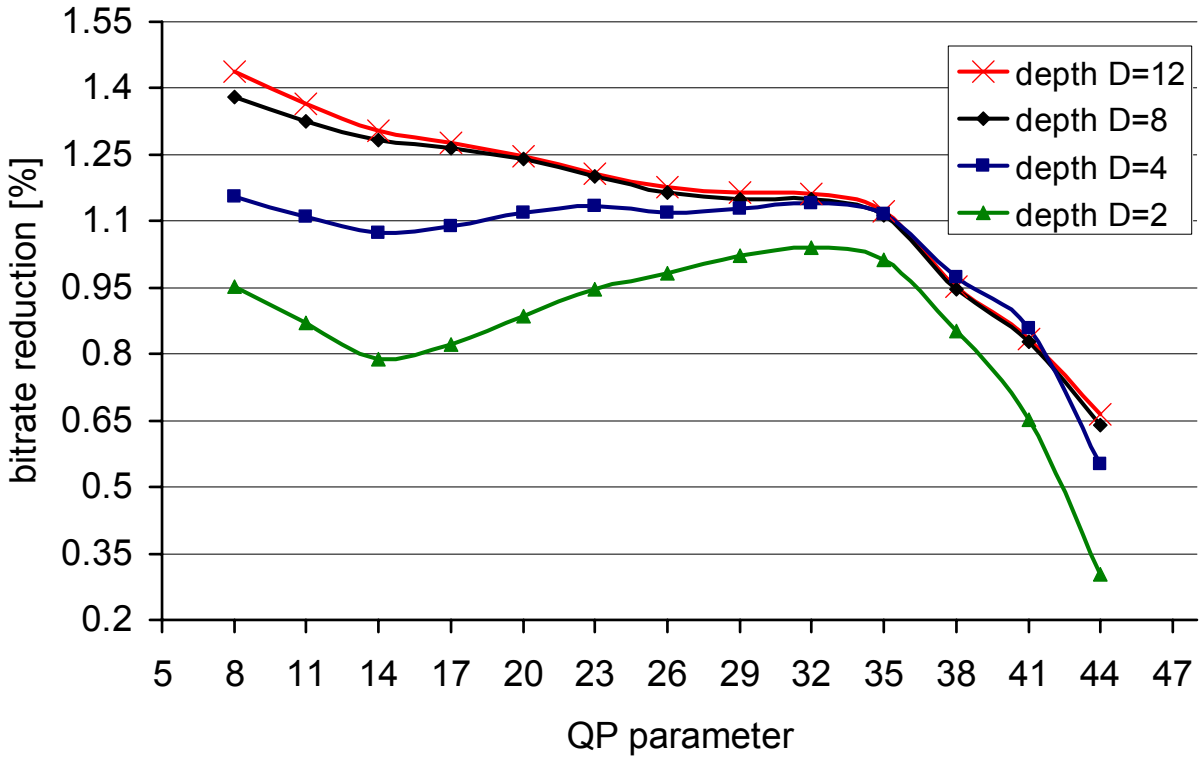
(d)

Figure A.11. Bitrate reduction achieved for ICE test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

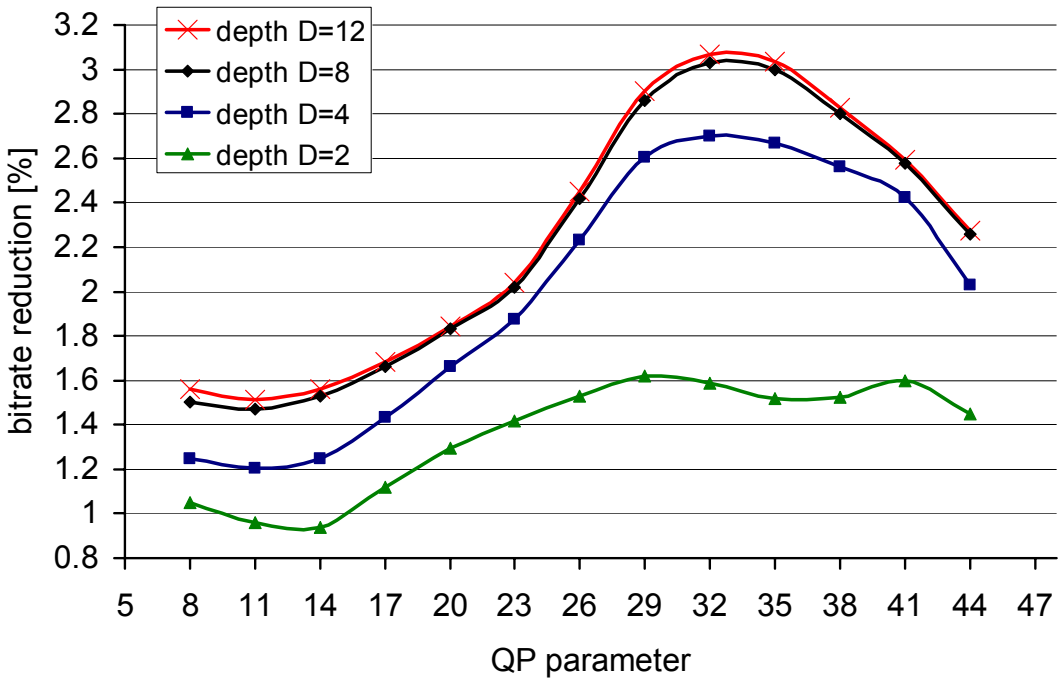
A.3.4. Experimental results for HARBOUR test sequence



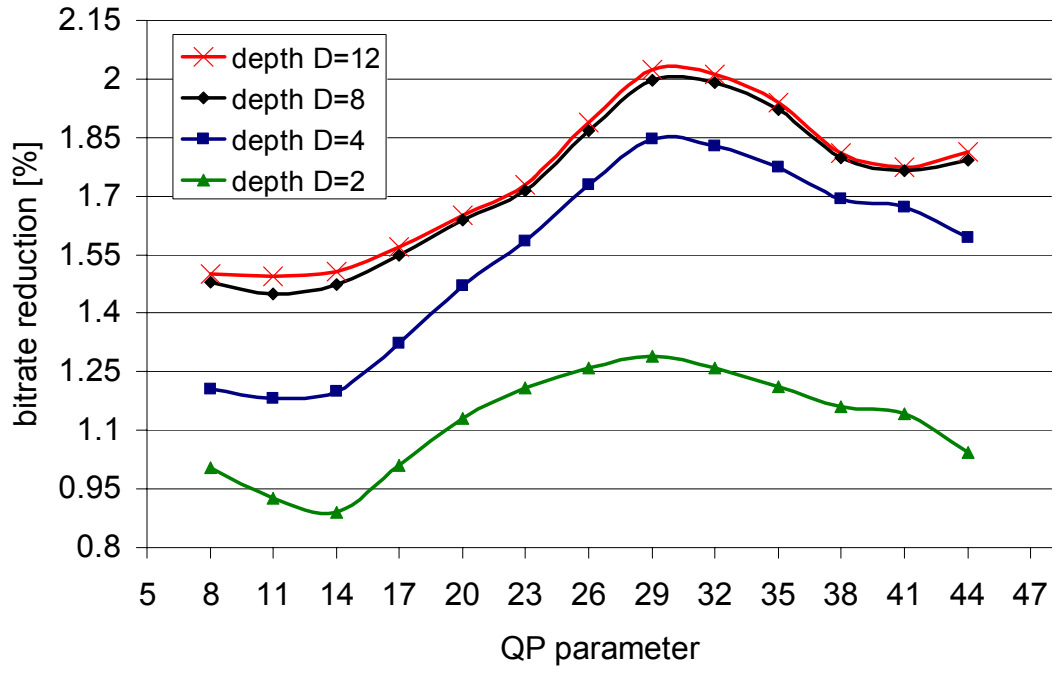
(a)



(b)



(c)



(d)

Figure A.12. Bitrate reduction achieved for HARBOUR test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and the CTW technique in contrast to the original AVC with unmodified CABAC.

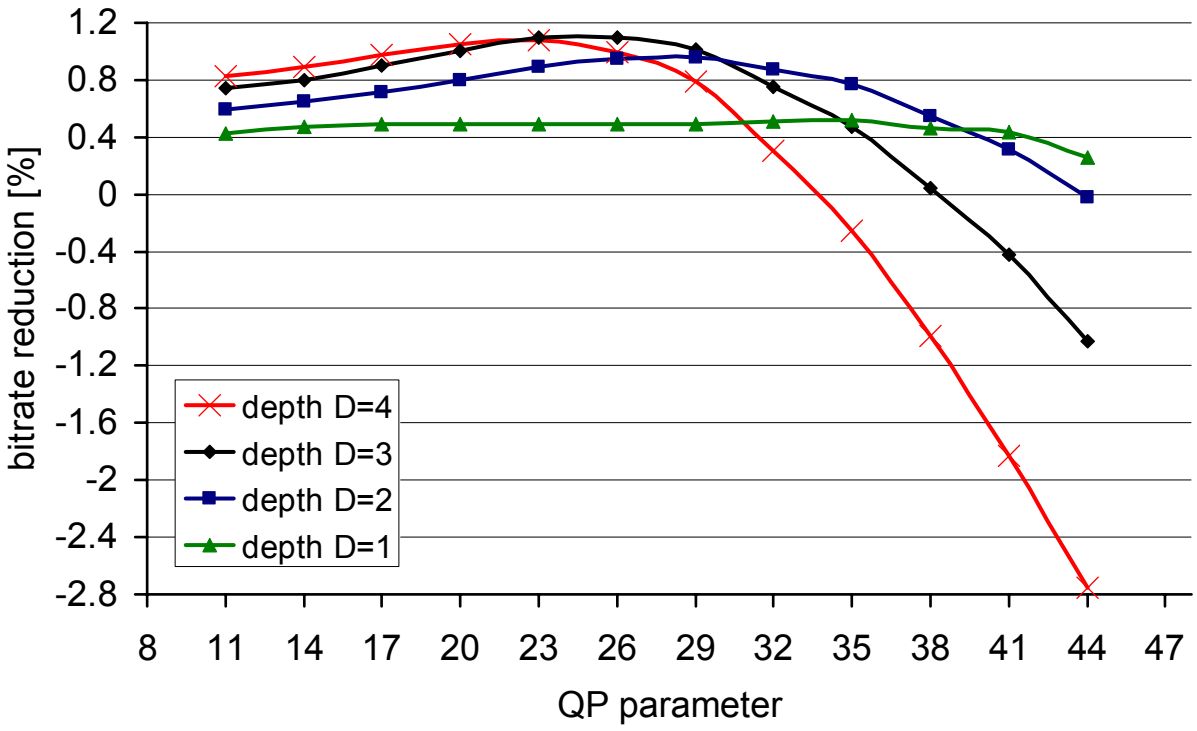
Annex B

Compression performance of the modified AVC with CABAC and PPMA relative to the original AVC

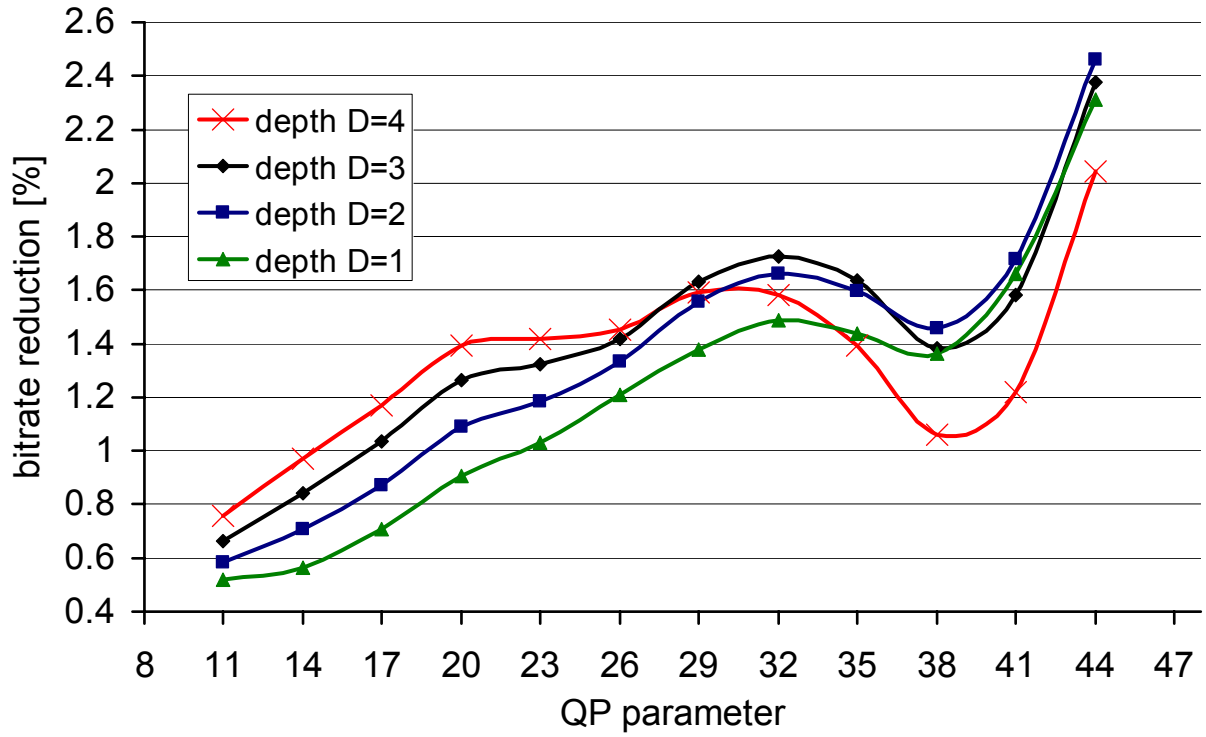
B.1. Experimental results for 4CIF test sequences and I29P structure of GOP

This section presents detailed experimental results on the compression performance of the modified AVC (with CABAC and PPMA technique) relative to coding efficiency of the original AVC. Experiments have been done according to **Scenario 1** (see Section 6.6).

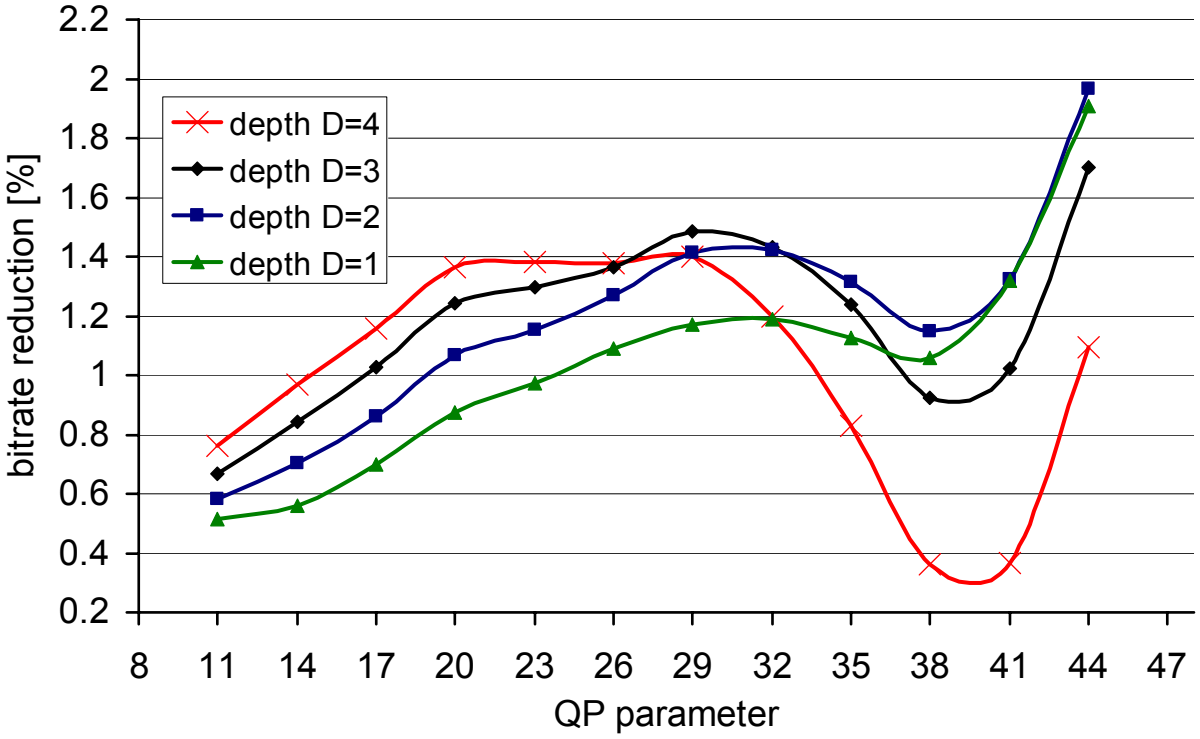
B.1.1. Experimental results for CITY test sequence



(a)



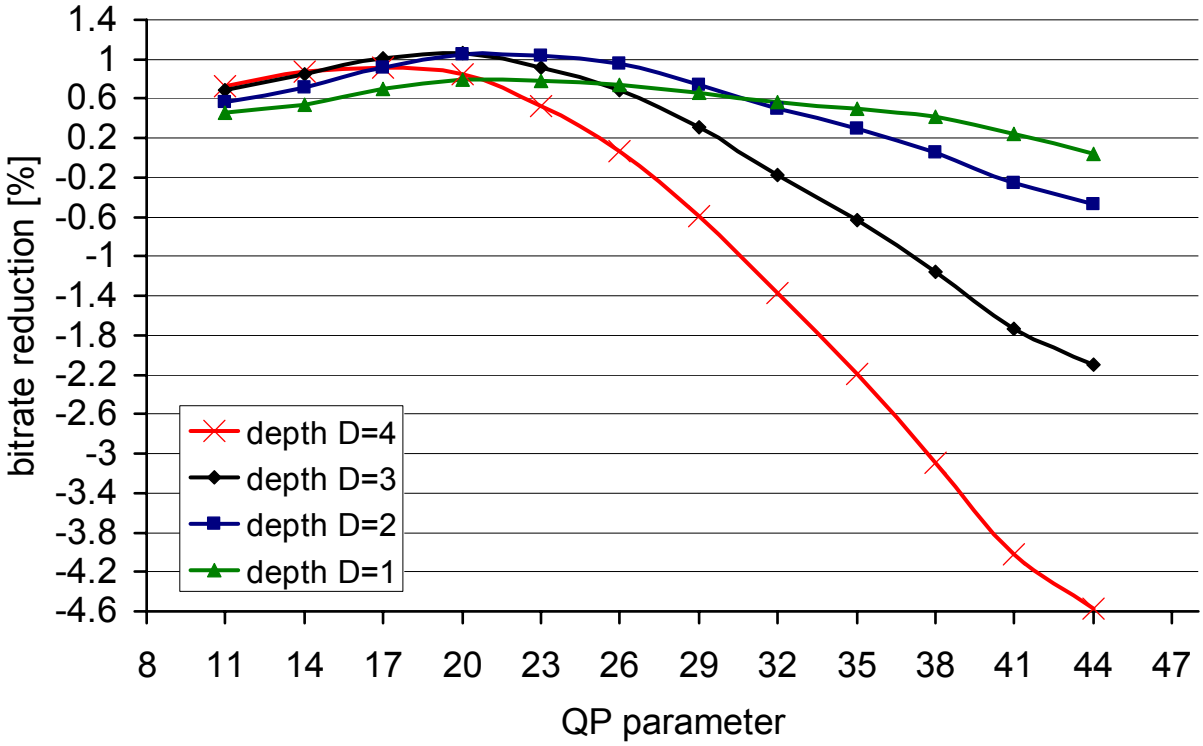
(b)



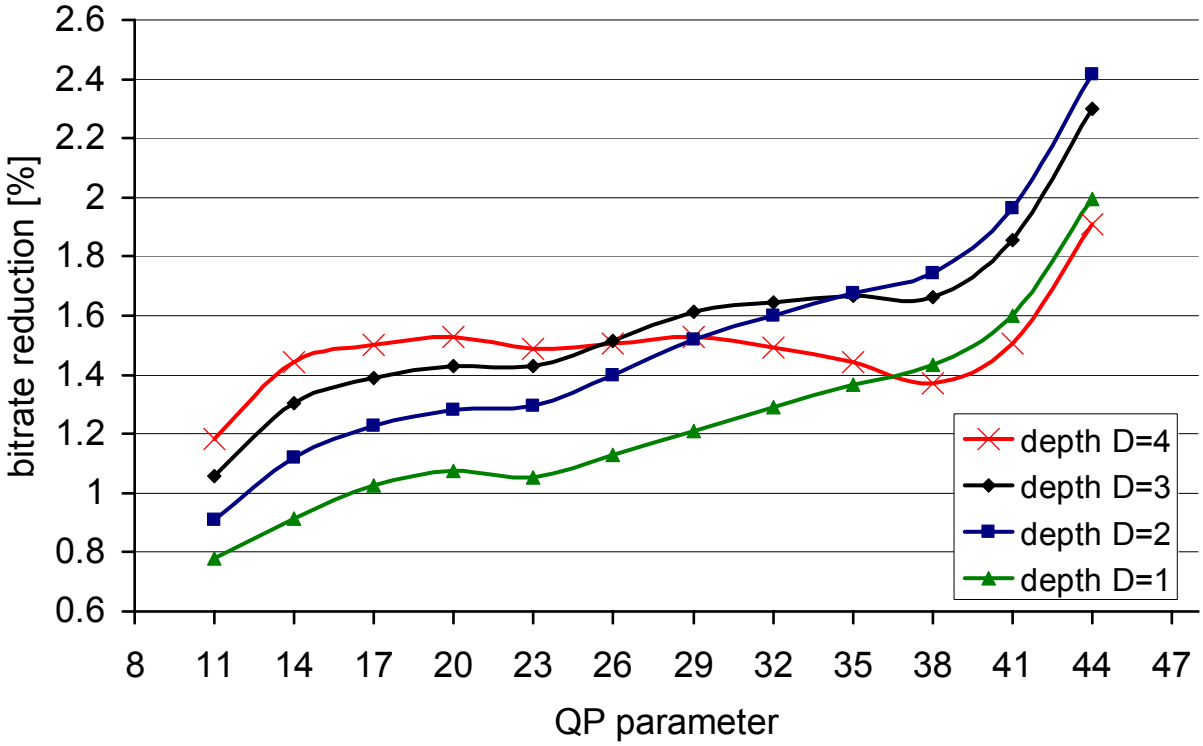
(c)

Figure B.1. Bitrate reduction achieved for CITY test sequence for I-frames (a), P-frames (b), and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the PPMA technique in contrast to the original AVC with unmodified CABAC.

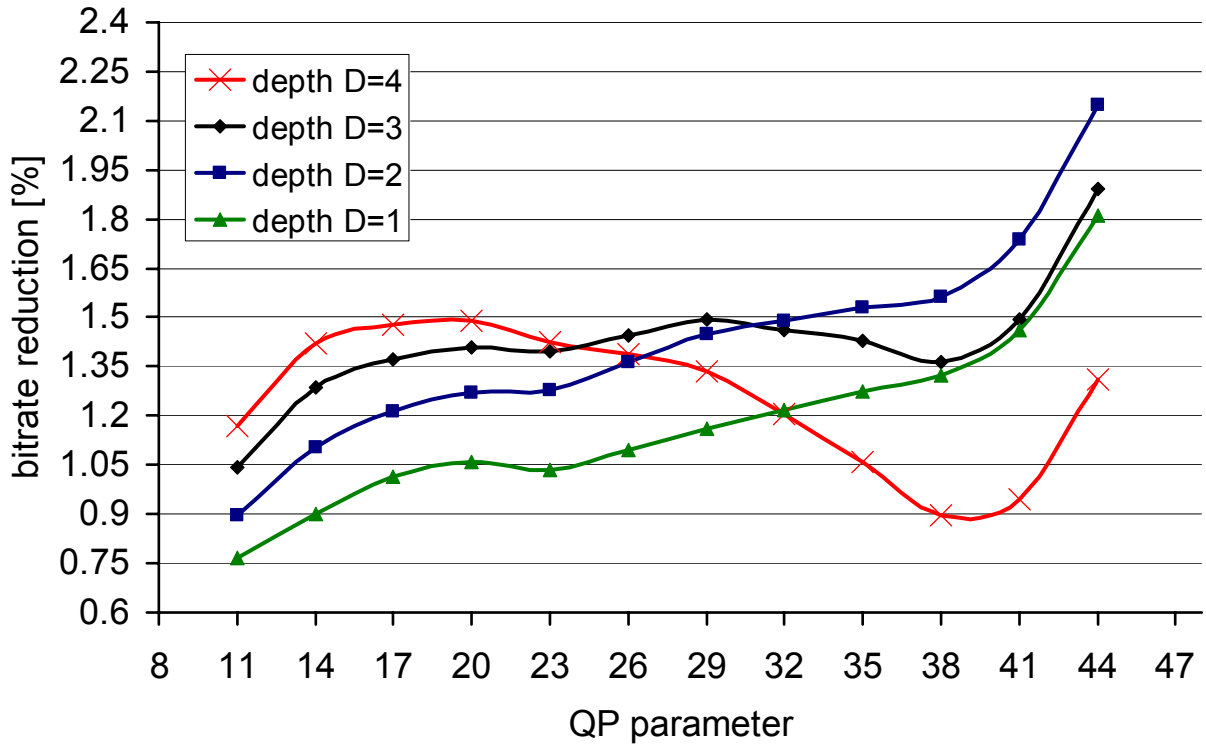
B.1.2. Experimental results for CREW test sequence



(a)



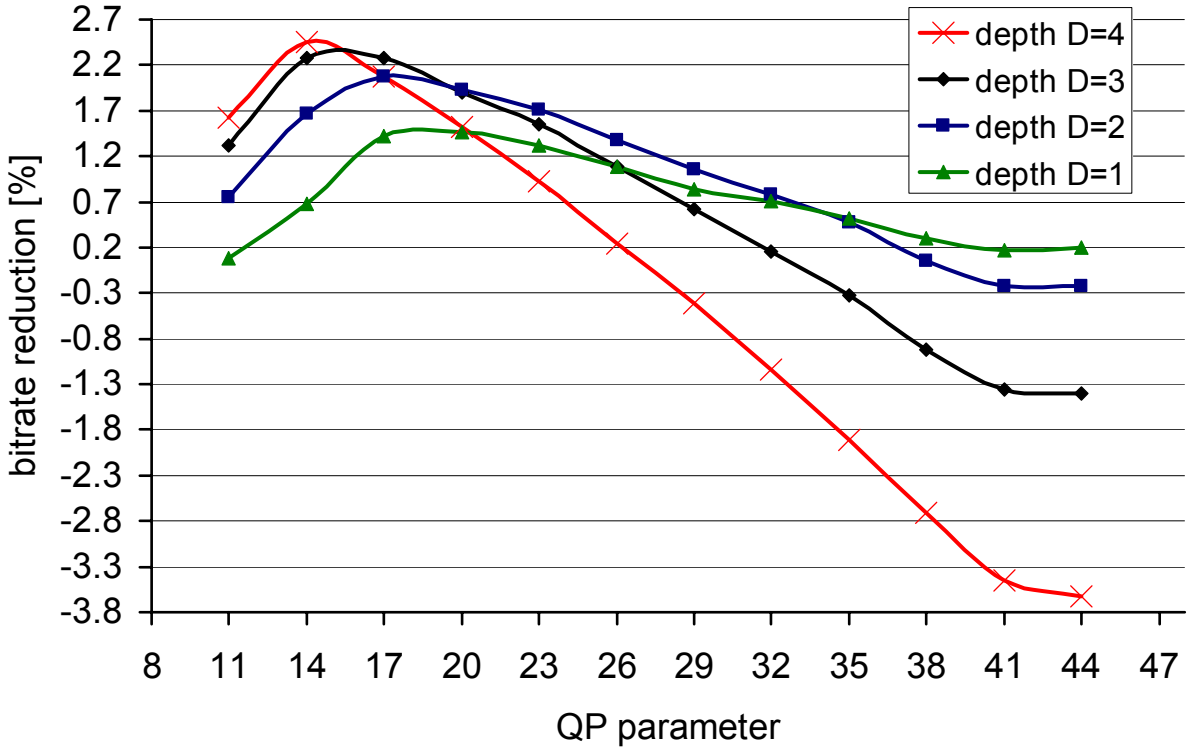
(b)



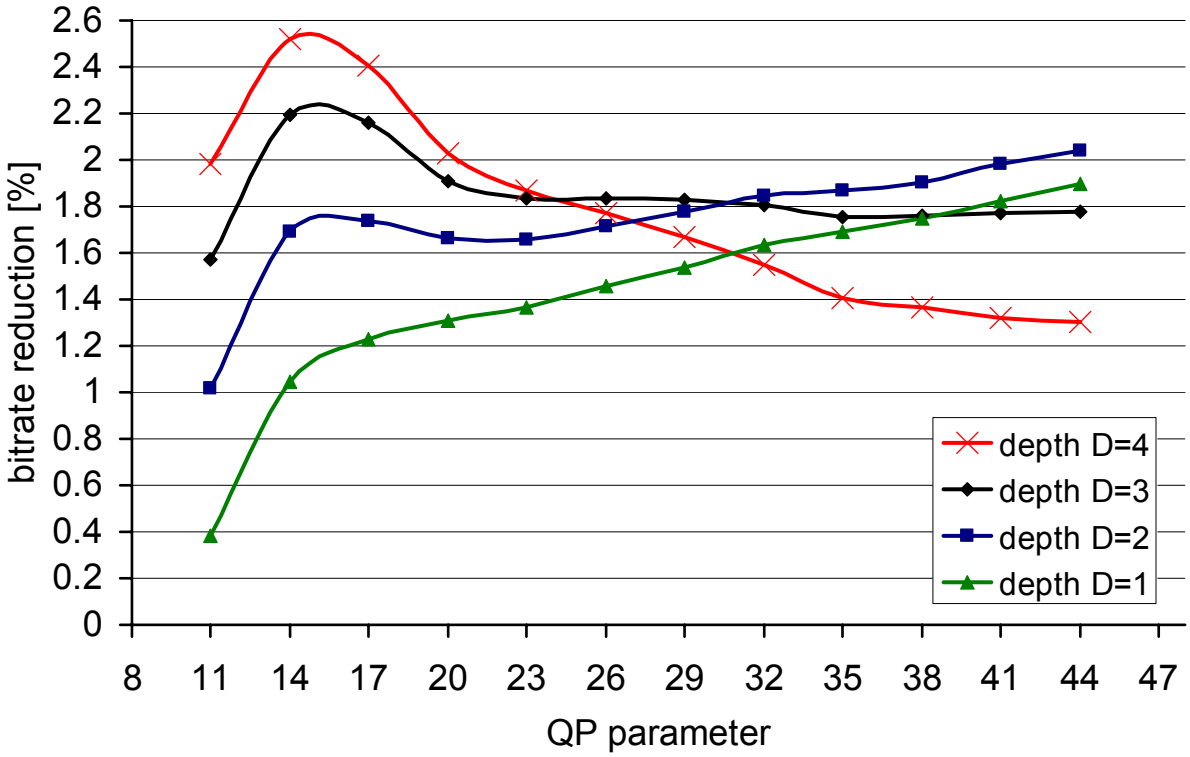
(c)

Figure B.2. Bitrate reduction achieved for CREW test sequence for I-frames (a), P-frames (b), and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the PPMA technique in contrast to the original AVC with unmodified CABAC.

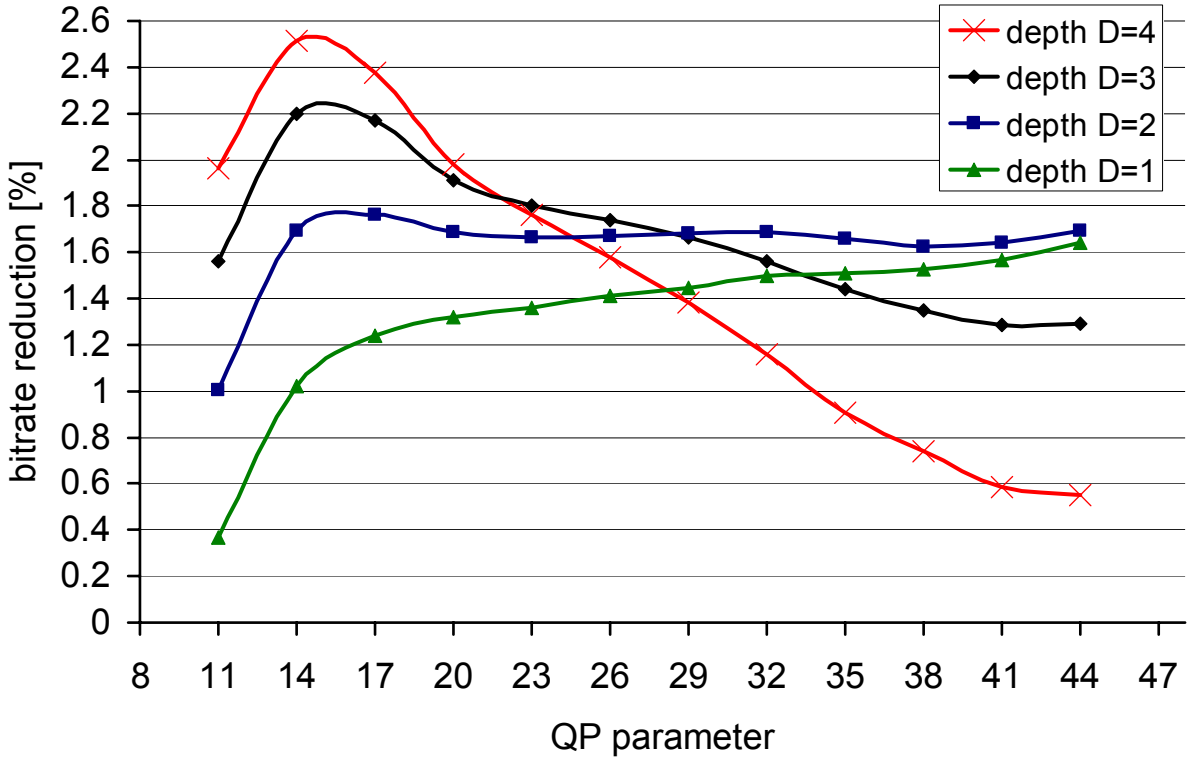
B.1.3. Experimental results for ICE test sequence



(a)



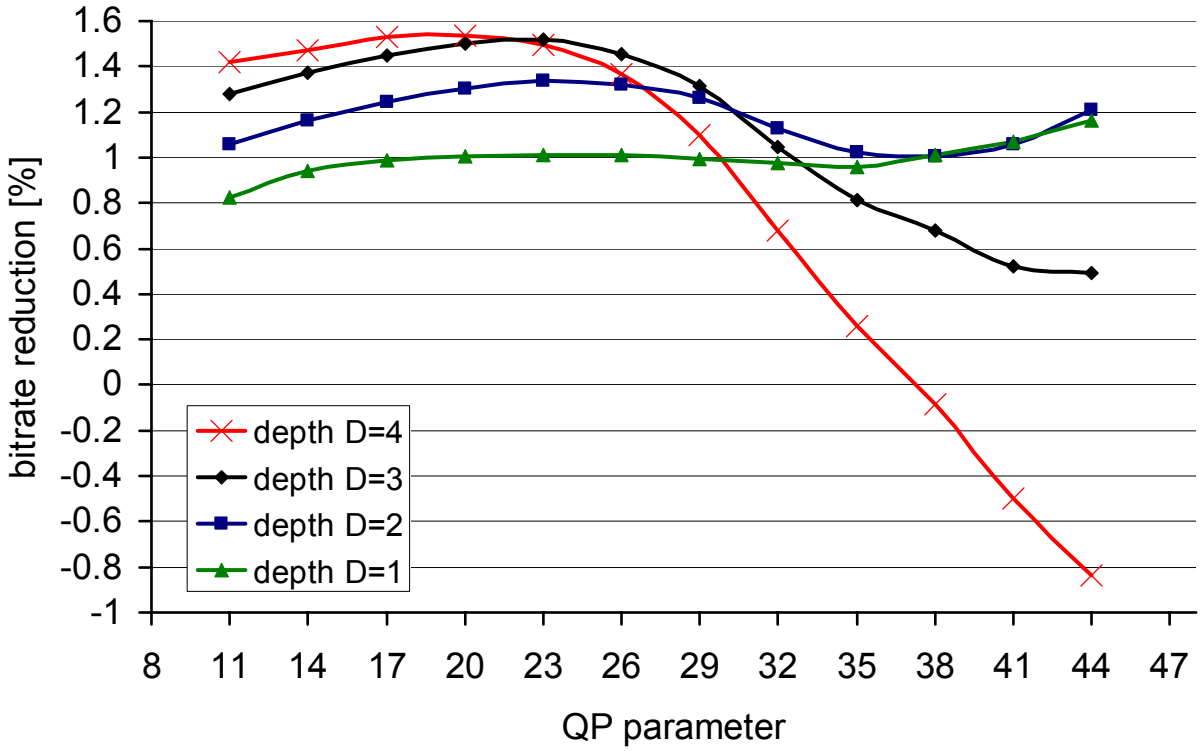
(b)



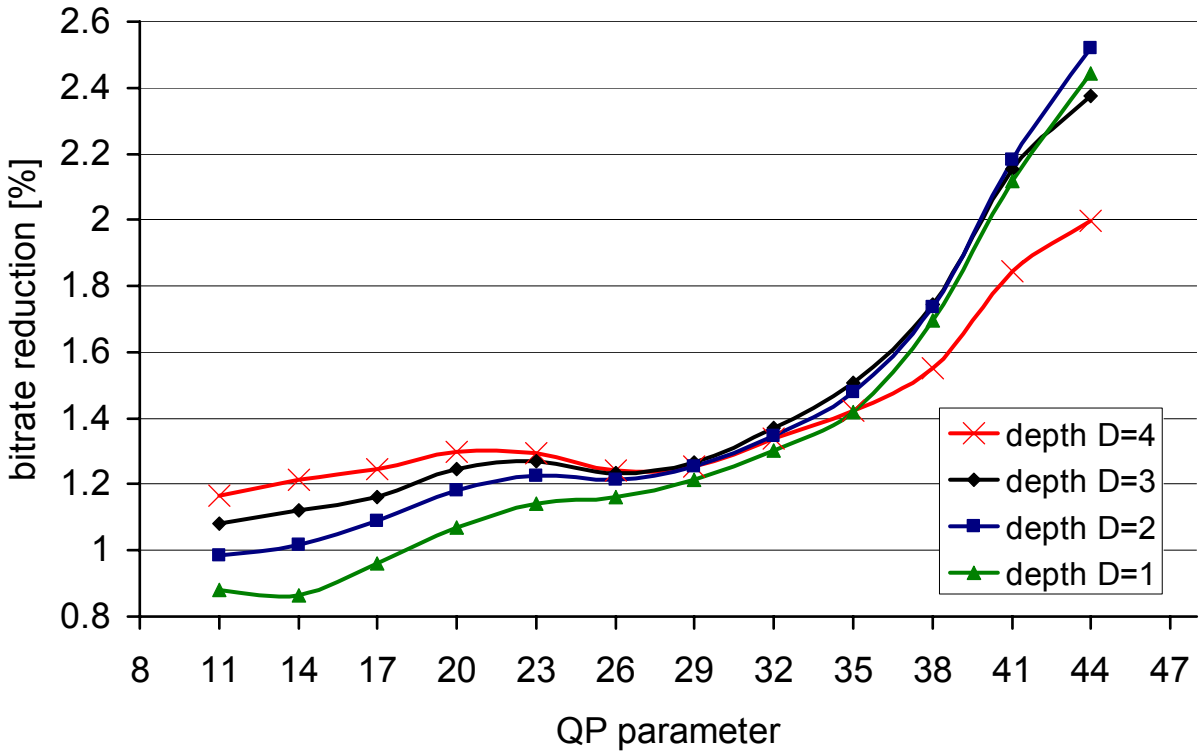
(c)

Figure B.3. Bitrate reduction achieved for ICE test sequence for I-frames (a), P-frames (b), and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the PPMA technique in contrast to the original AVC with unmodified CABAC.

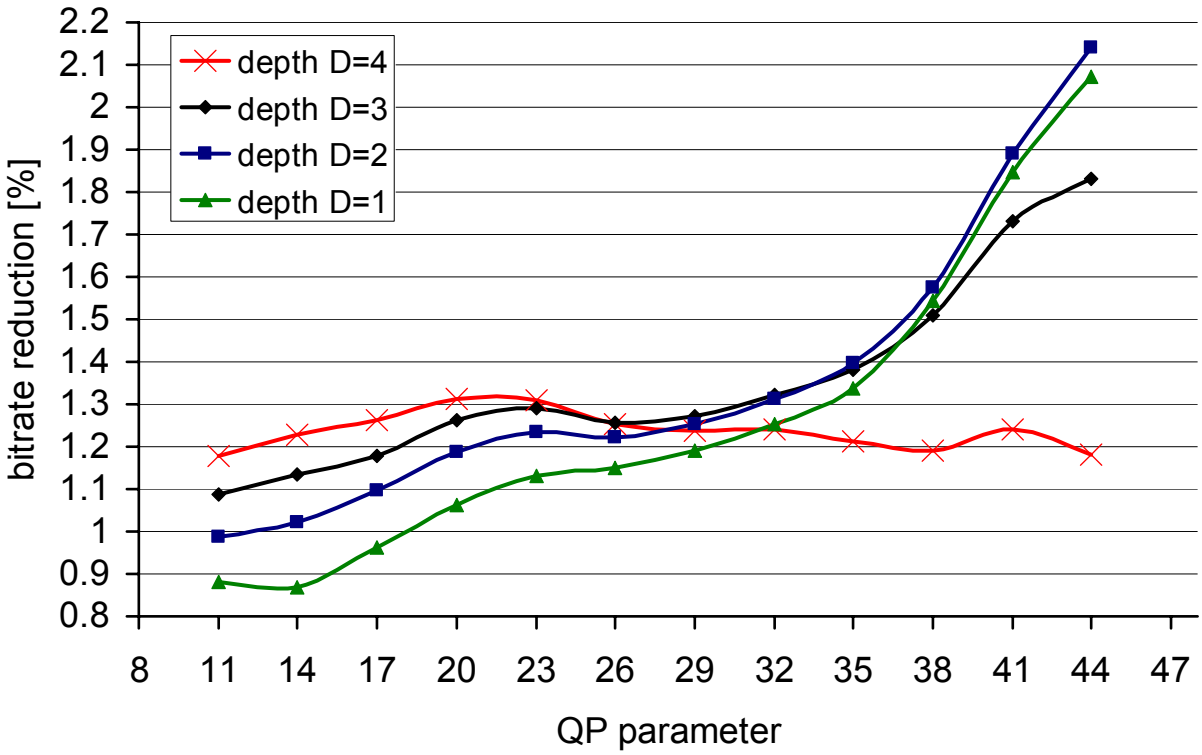
B.1.4. Experimental results for HARBOUR test sequence



(a)



(b)



(c)

Figure B.4. Bitrate reduction achieved for HARBOUR test sequence for I-frames (a), P-frames (b), and the whole test sequence (c). The bitrate reduction is a result of application of the modified AVC with CABAC and the PPMA technique in contrast to the original AVC with unmodified CABAC.

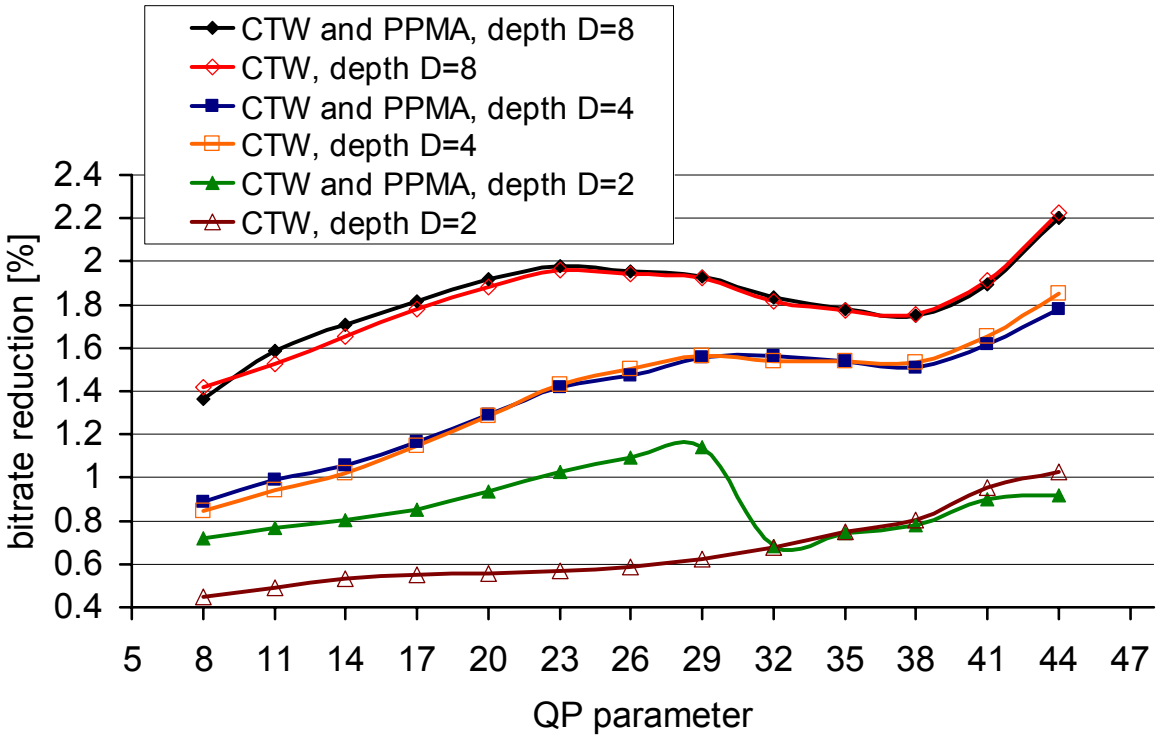
Annex C

Experimental results on coding efficiency of the modified AVC with CABAC and joint application of CTW and PPMA relative to the original AVC

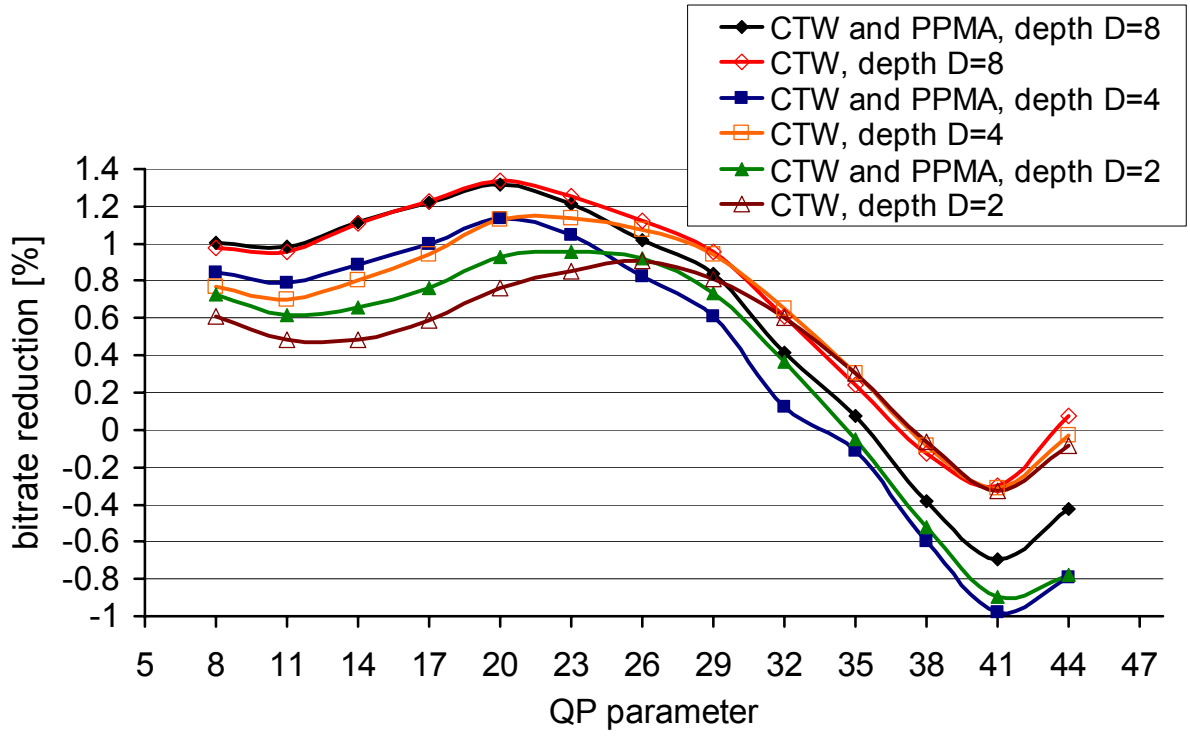
C.1. Experimental results for 4CIF test sequences and IBBPBBP… structure of GOP

In this section, the detailed experimental results on the coding efficiency of the modified AVC (with CABAC and joint application of CTW and PPMA) relative to the compression performance of the original AVC with CABAC have been presented. Experiments have been done in **Scenario 3** (see Section 6.6). The compression performance of the modified AVC encoder with CABAC and CTW and PPMA has been tested for depths *D* of the context trees equal to 2, 4, and 8.

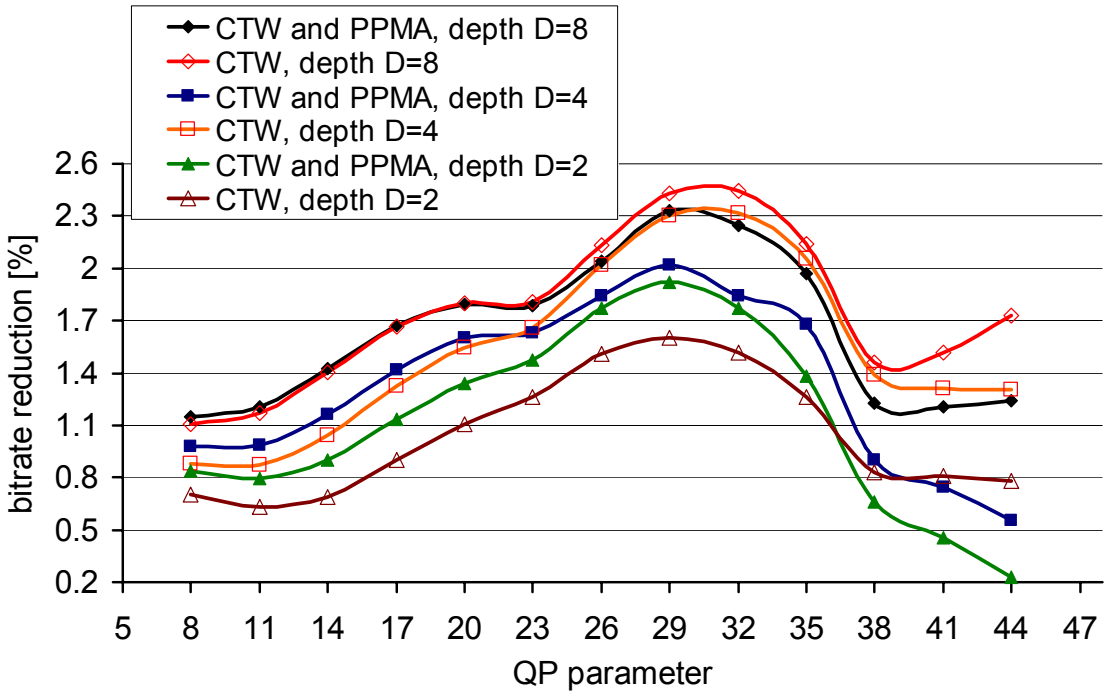
C.1.1. Experimental results for CITY test sequence



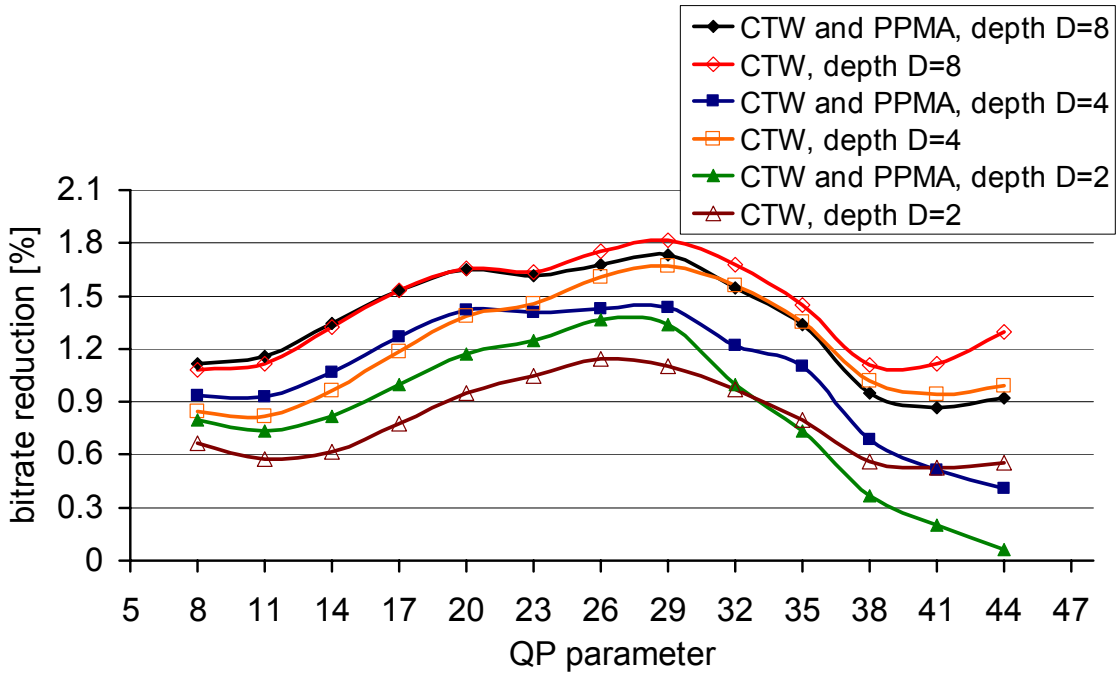
(a)



(b)



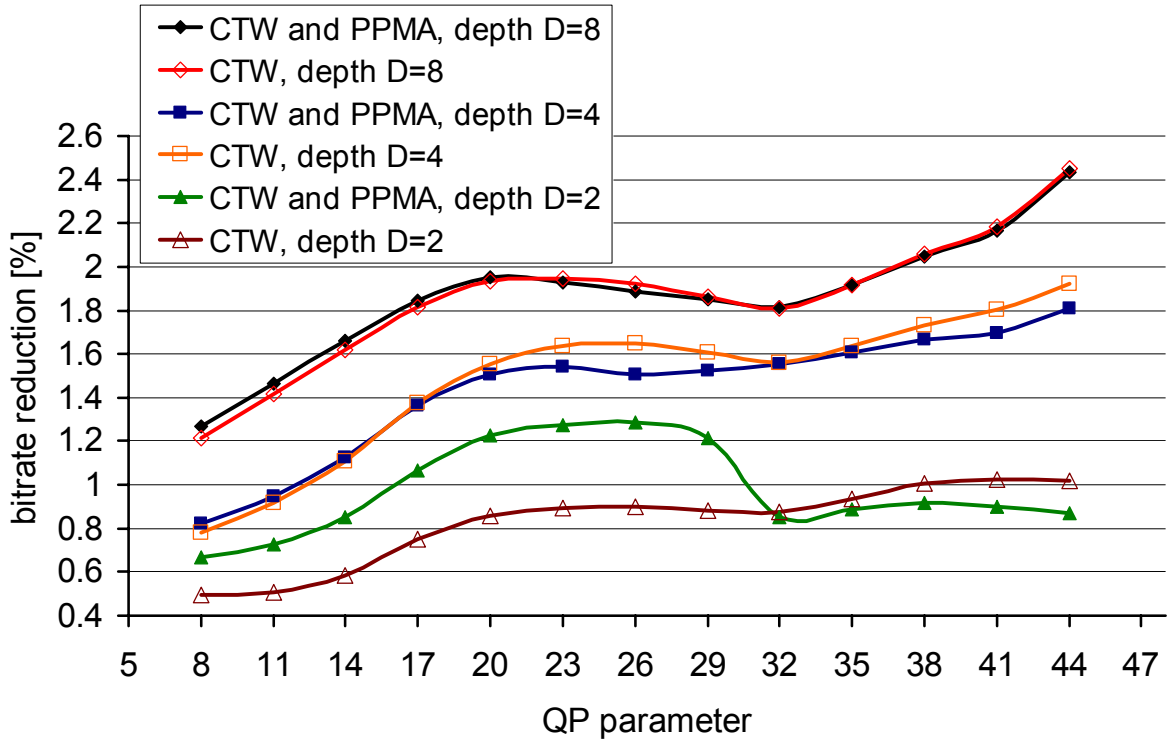
(c)



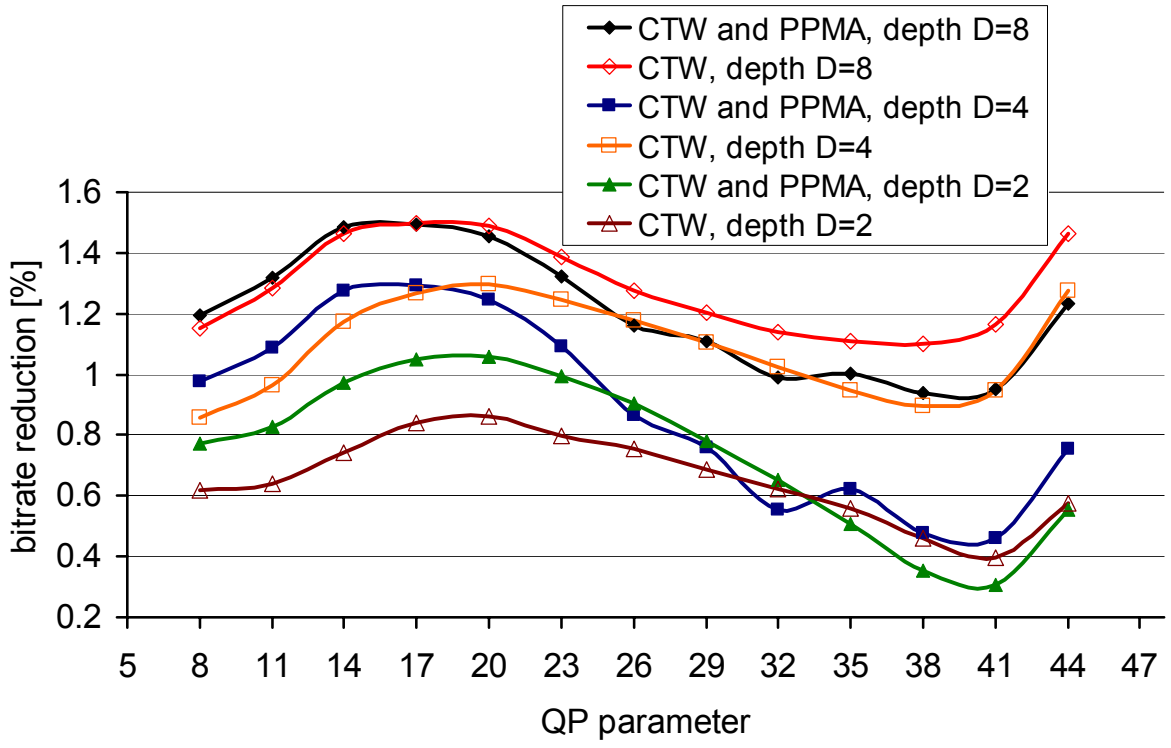
(d)

Figure C.1. Bitrate reduction achieved for CITY test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application the modified AVC with CABAC and joint application of CTW and PPMA technique in contrast to the original AVC with unmodified CABAC.

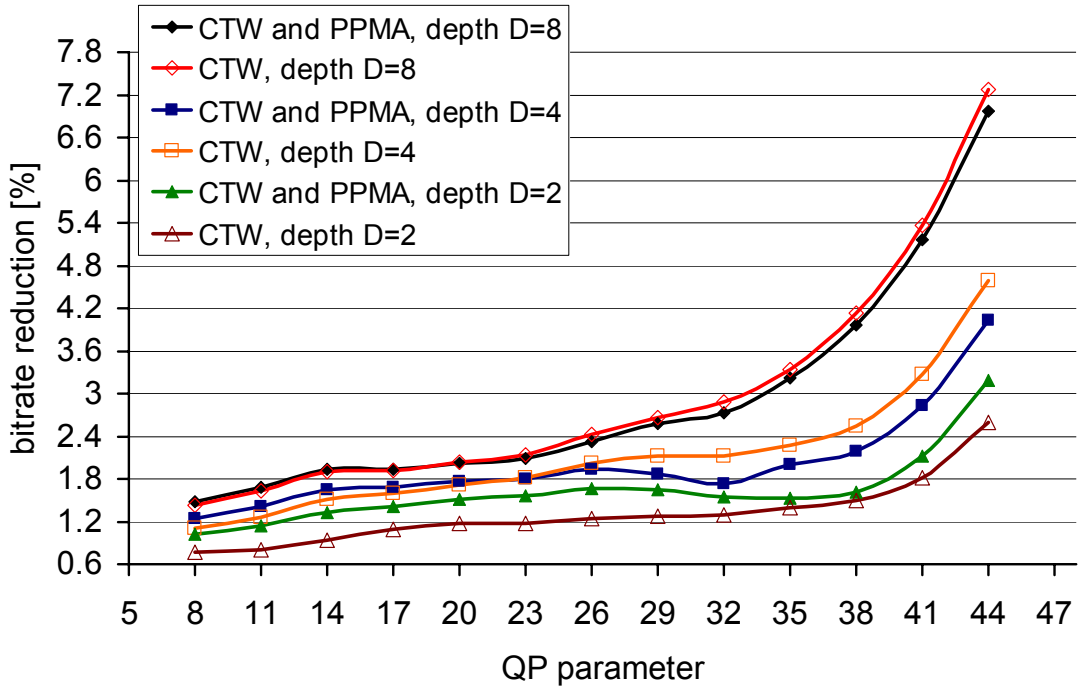
C.1.2. Experimental results for CREW test sequence



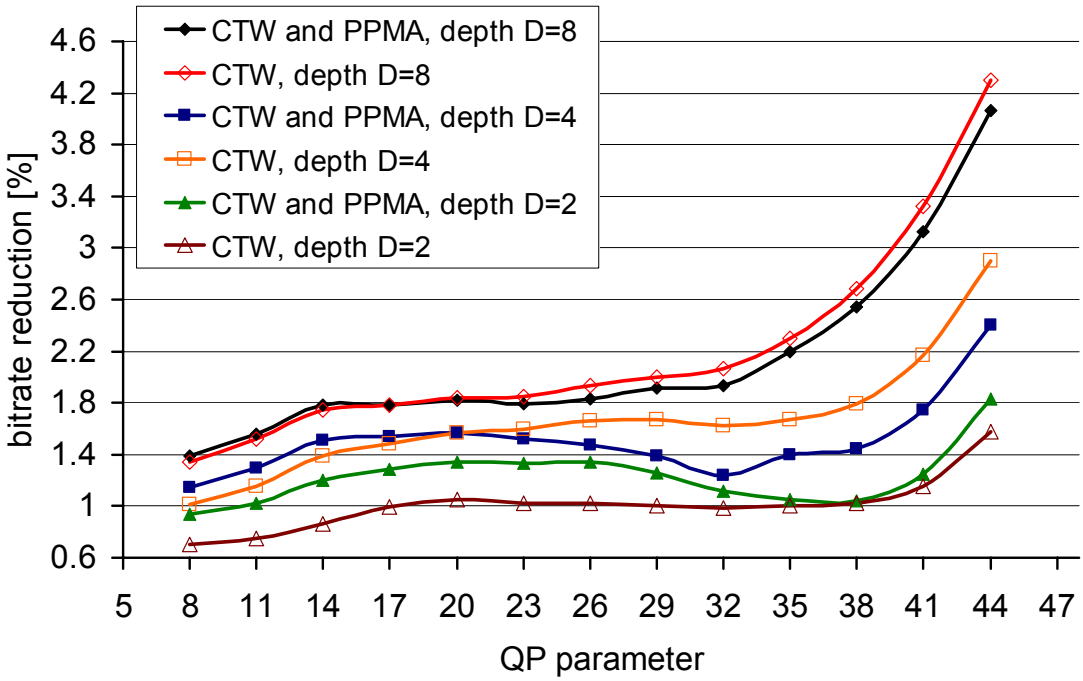
(a)



(b)



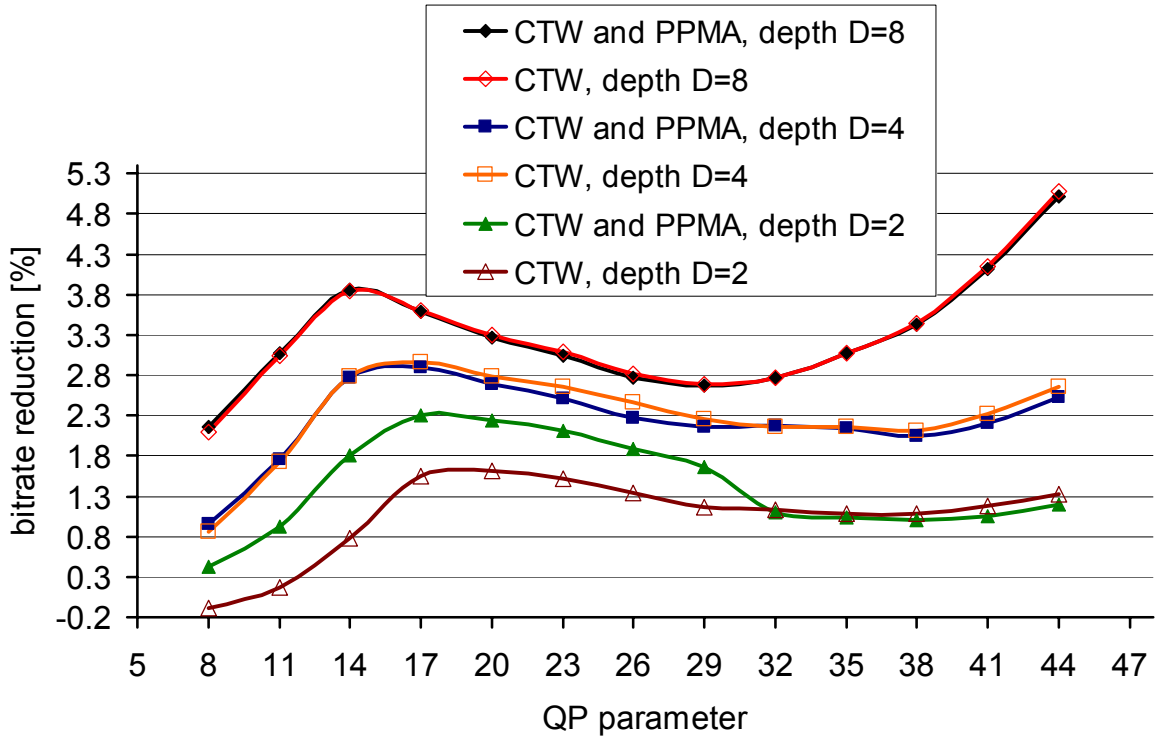
(c)



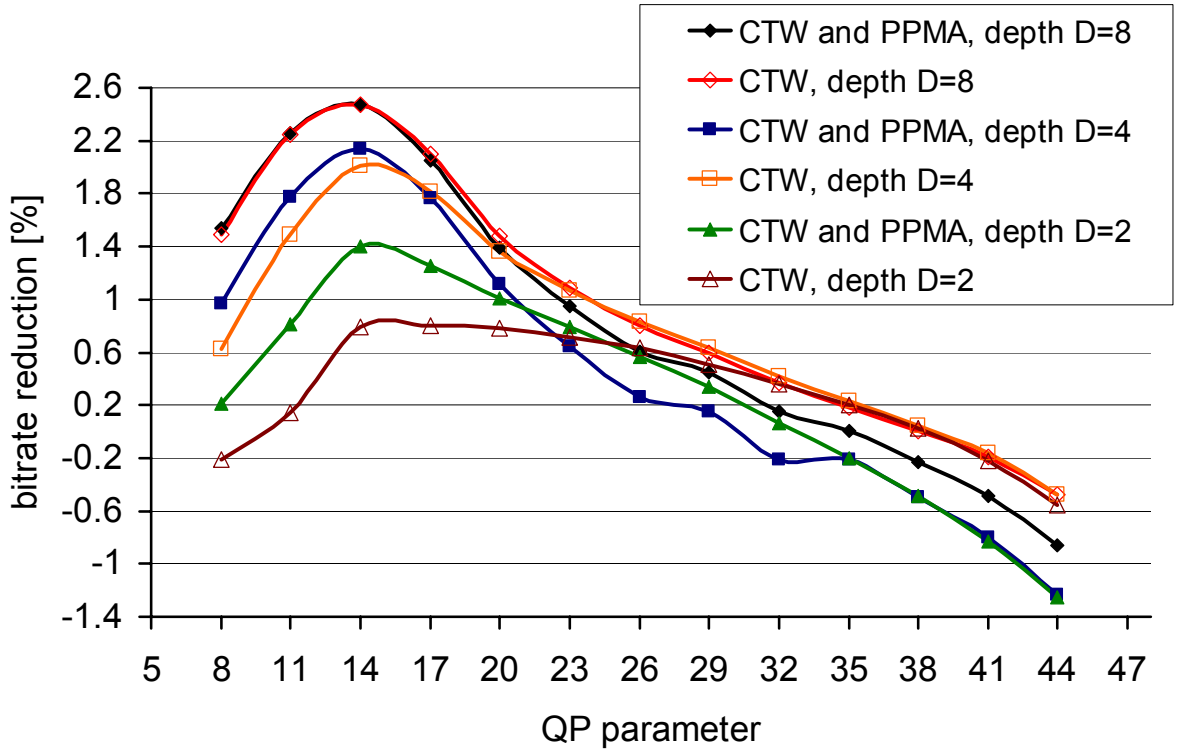
(d)

Figure C.2. Bitrate reduction achieved for CREW test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and joint application of CTW and PPMA technique in contrast to the original AVC with unmodified CABAC.

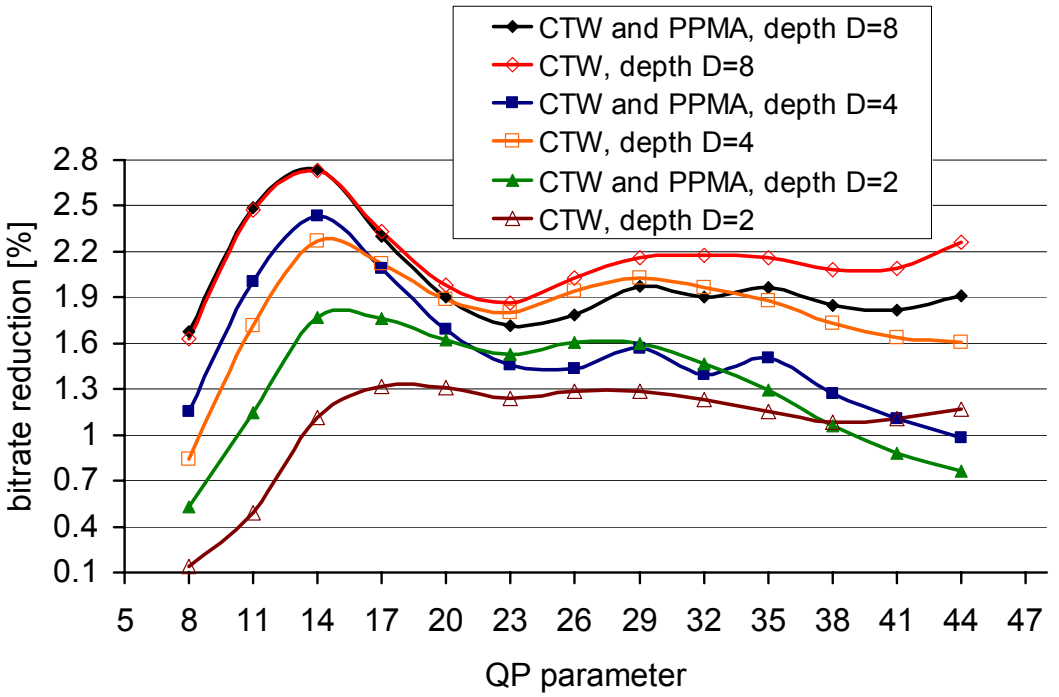
C.1.3. Experimental results for ICE test sequence



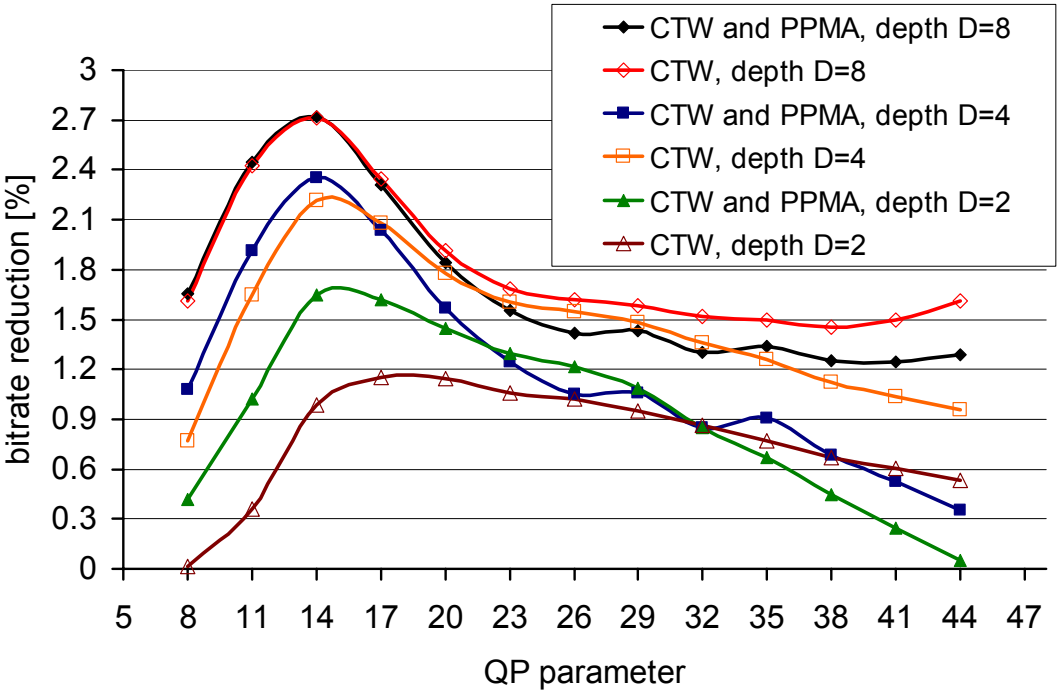
(a)



(b)



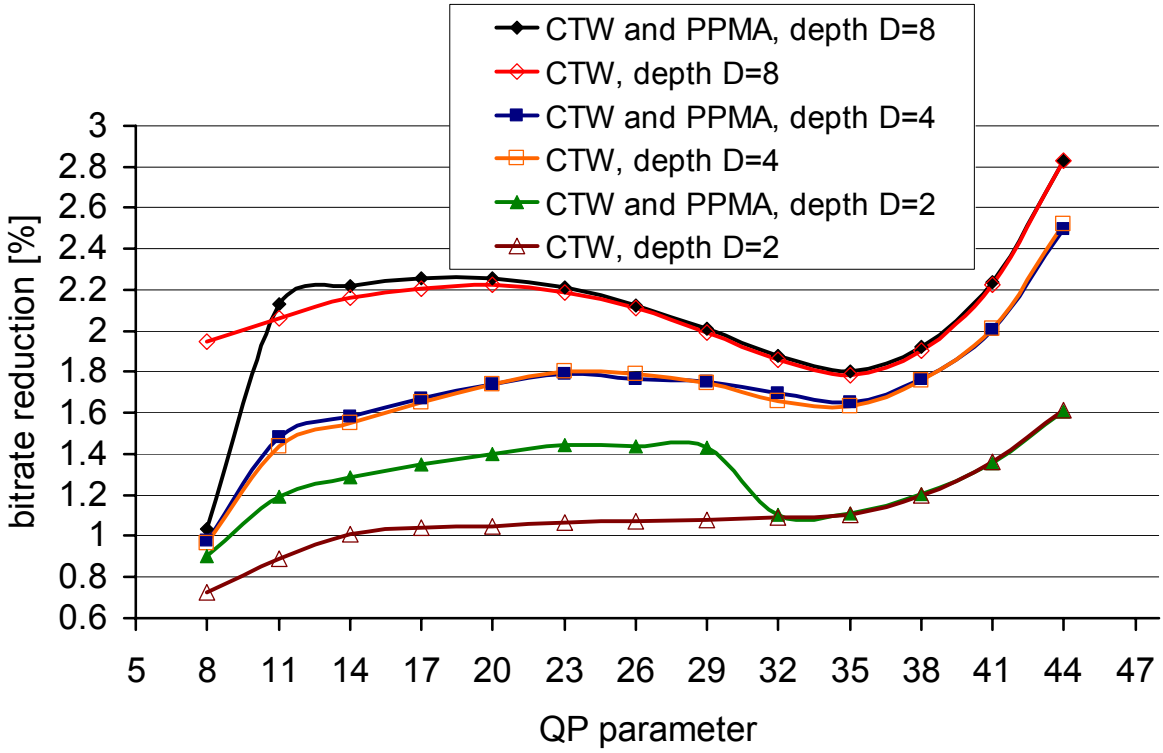
(c)



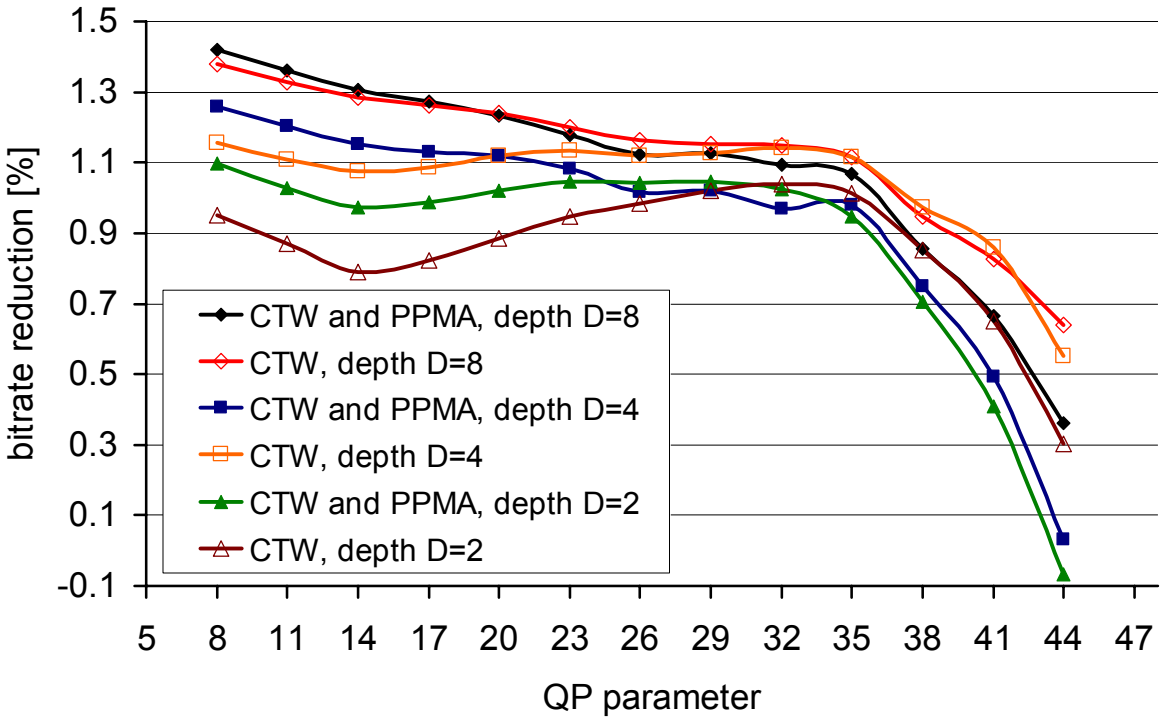
(d)

Figure C.3. Bitrate reduction achieved for ICE test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and joint application of CTW and PPMA technique in contrast to the original AVC with unmodified CABAC.

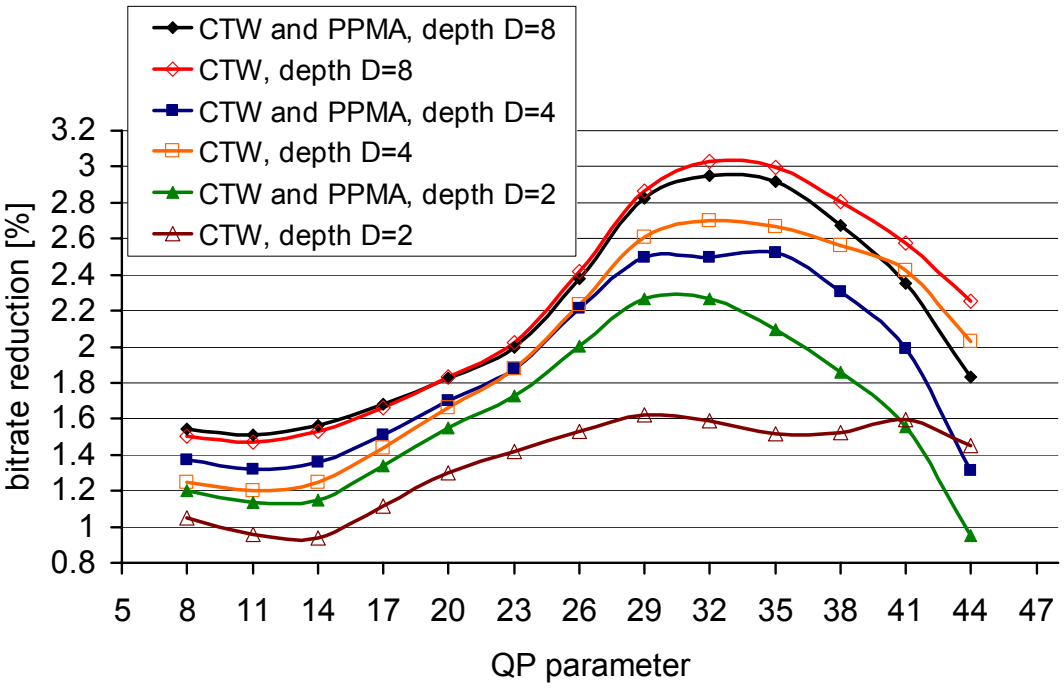
C.1.4. Experimental results for HARBOUR test sequence



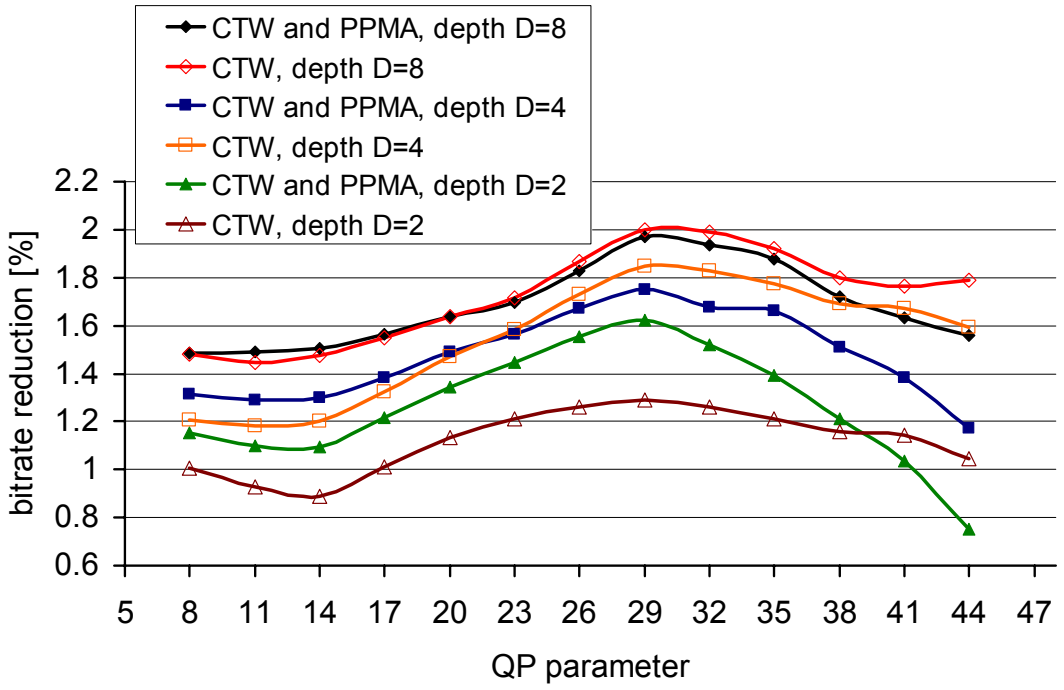
(a)



(b)



(c)



(d)

Figure C.4. Bitrate reduction achieved for HARBOUR test sequence for I-frames (a), P-frames (b), B-frames (c) and the whole test sequence (d). The bitrate reduction is a result of application of the modified AVC with CABAC and joint application of CTW and PPMA technique in contrast to the original AVC with unmodified CABAC.

Annex D

Experimental results on the coding efficiency of arithmetic codec cores

D.1. Experimental results

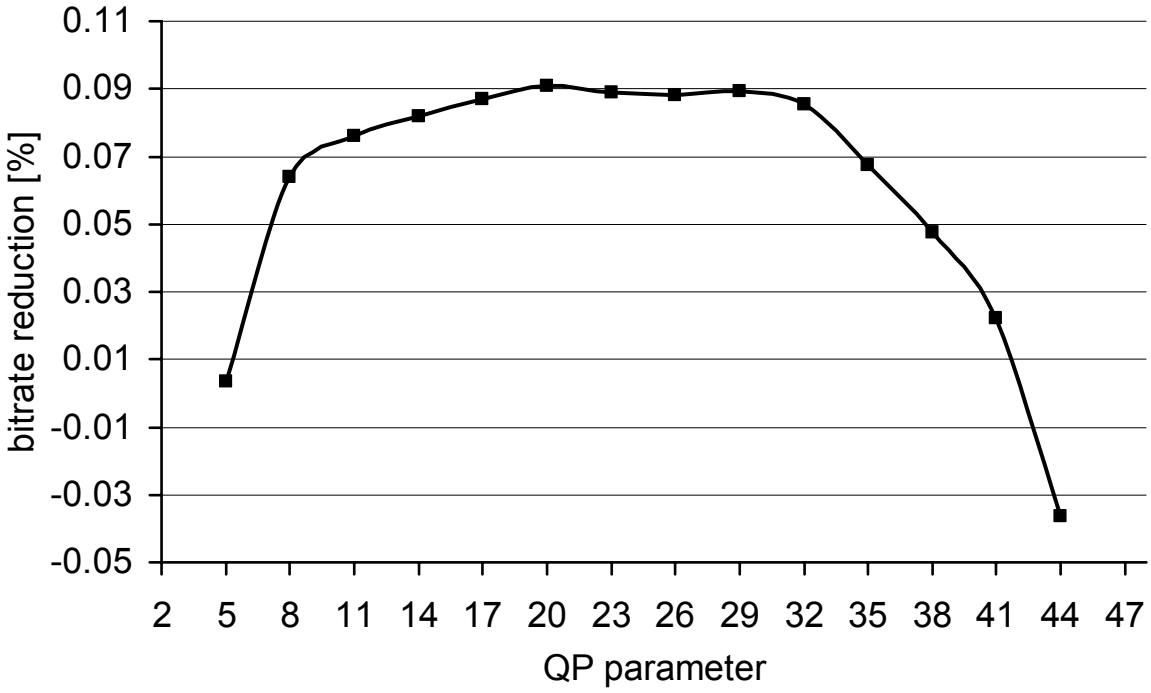
This section presents the detailed experimental results on the coding efficiency of two different cores of arithmetic codec within AVC video encoder. These are:

- M-codec core from CABAC;
- Arithmetic codec core from H.263 video coding standard;

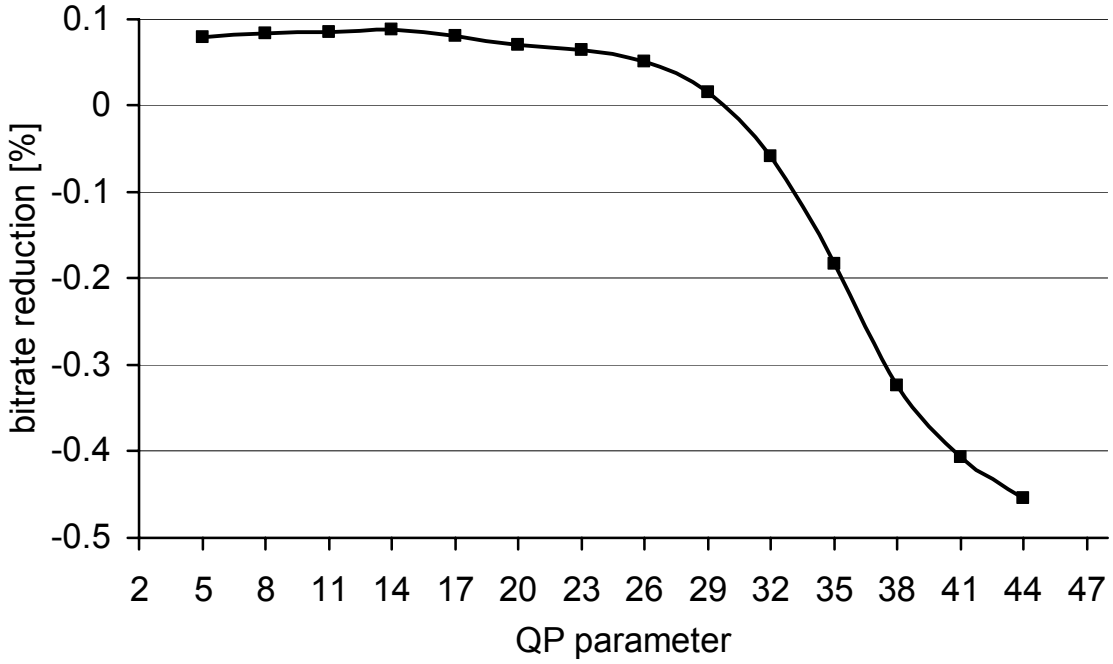
The compression performance of both M-codec core and H.263 arithmetic codec core has been tested in the following conditions:

- The CITY, CREW, ICE and HARBOUR test sequences in 4CIF format have been used;
- The experiments have been done for both intra and inter prediction modes by setting the structure of GOP on I29P;
- Tests have been done for a wide range of the QP parameter that corresponds to video sequences from excellent to bad subjective quality.

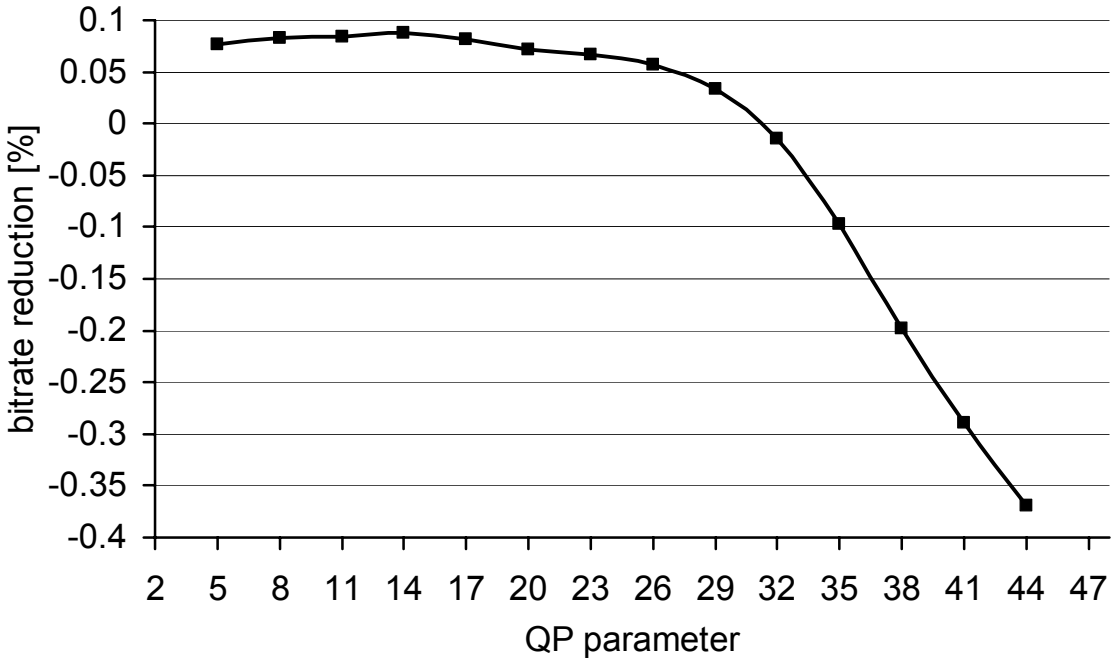
D.1.1. Experimental results for CITY test sequence



(a)



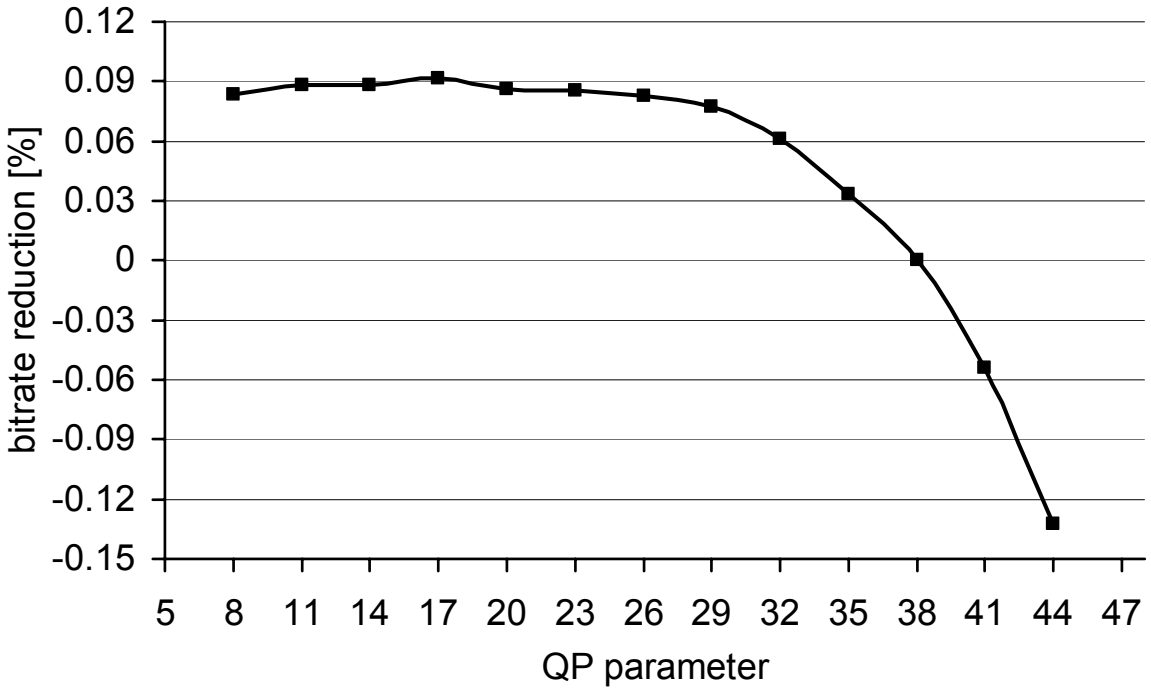
(b)



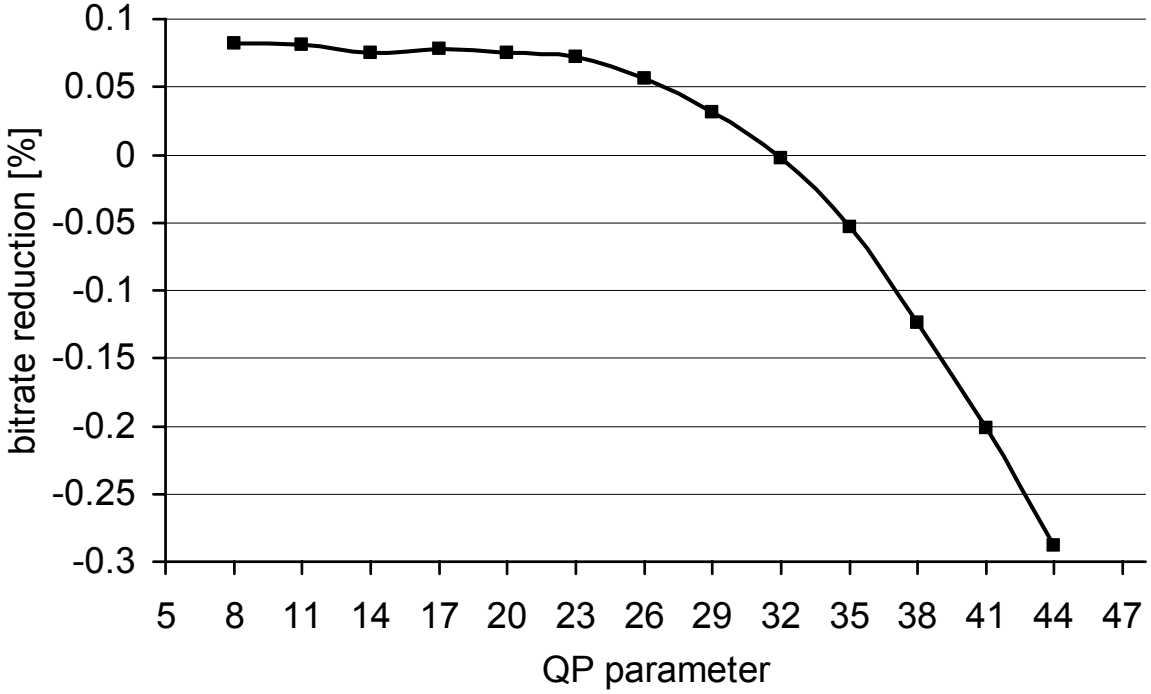
(c)

Figure D.1. Bitrate reduction achieved for CITY test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application in CABAC within the AVC the H.263 arithmetic codec core instead of the M-codec core.

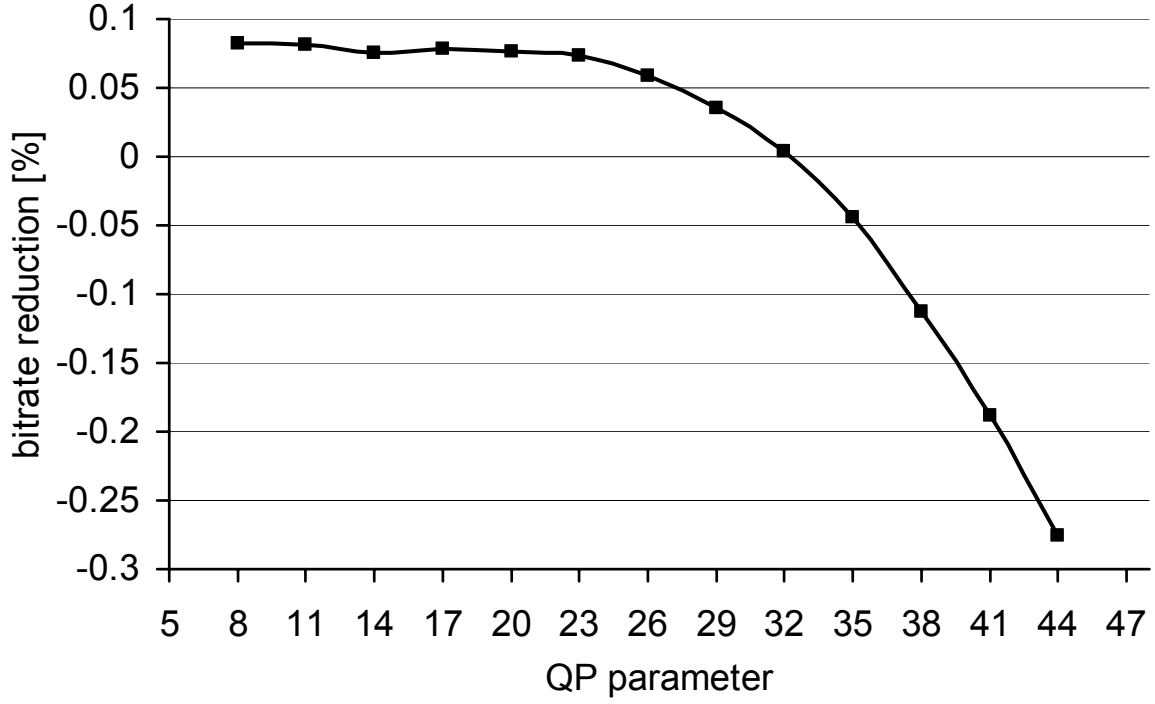
D.1.2. Experimental results for CREW test sequence



(a)



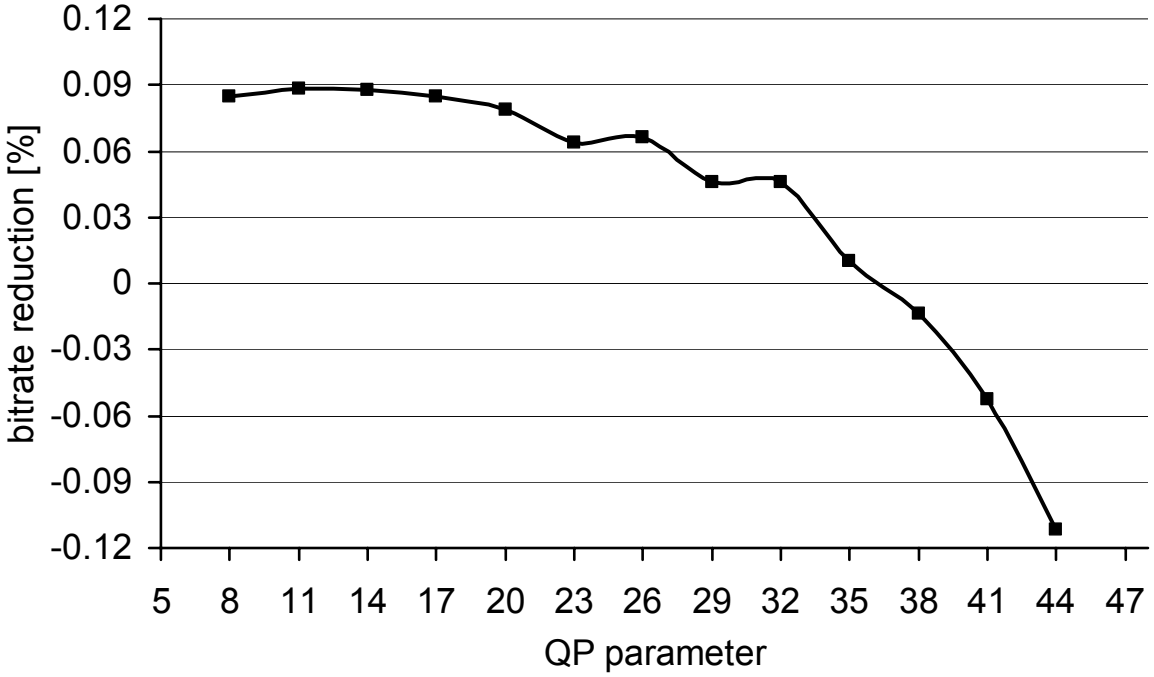
(b)



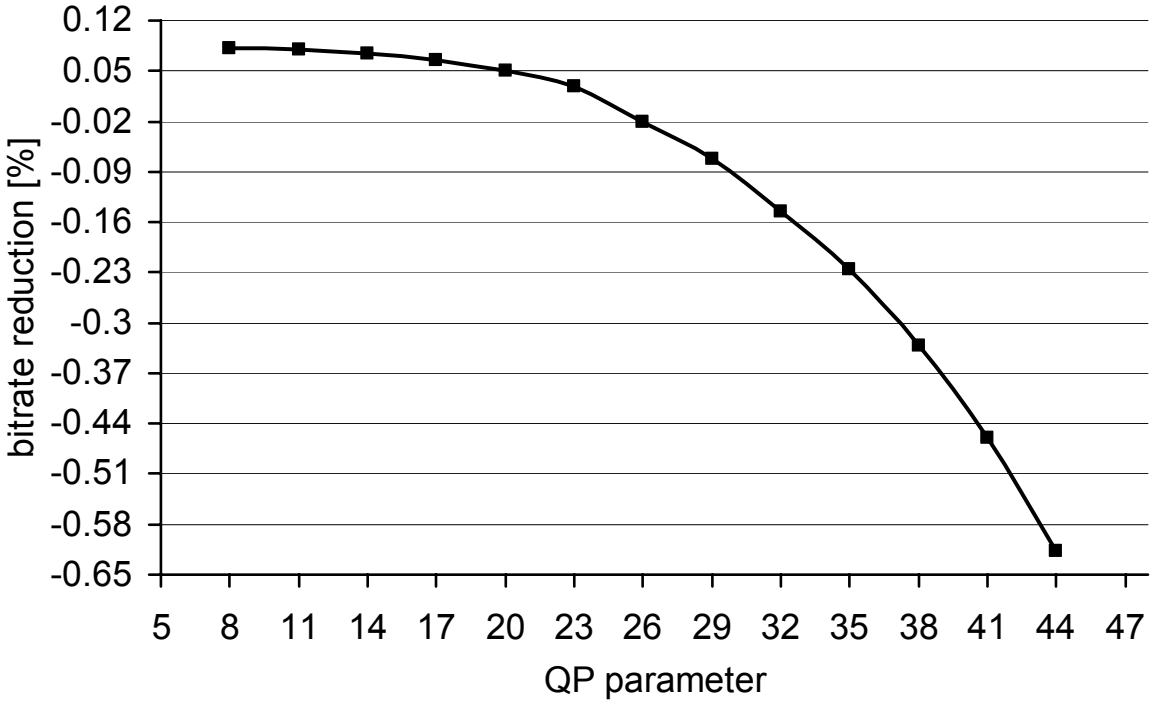
(c)

Figure D.2. Bitrate reduction achieved for CREW test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application in CABAC within the AVC the H.263 arithmetic codec core instead of the M-codec core.

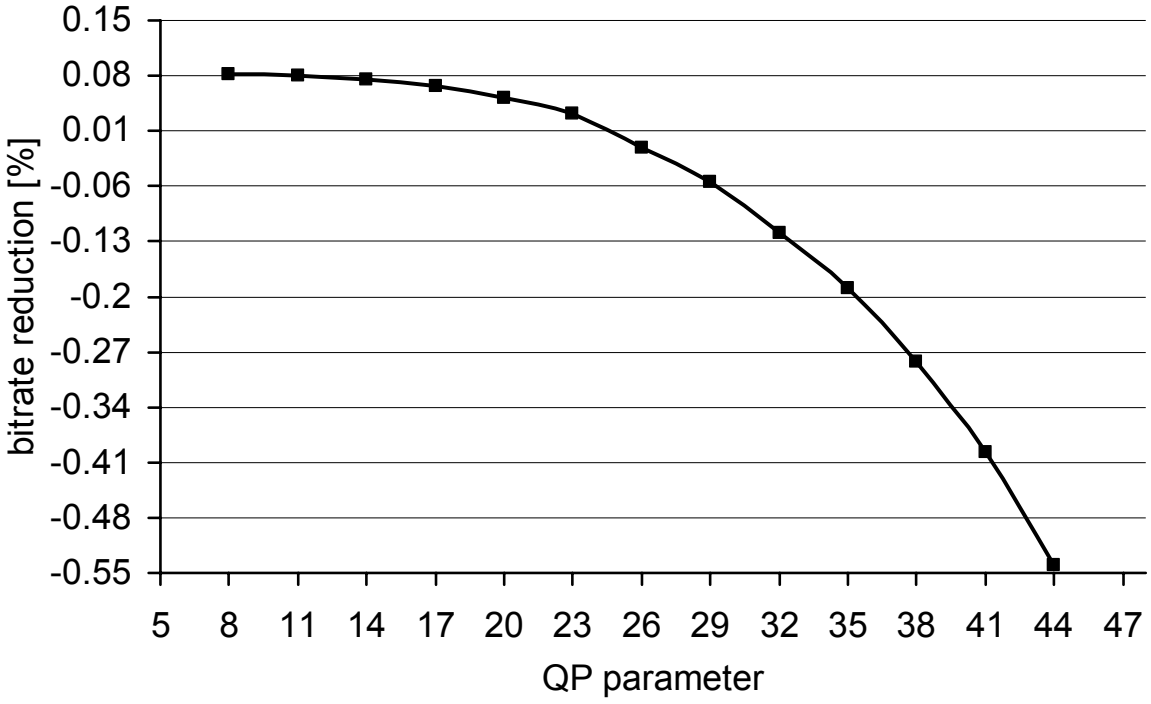
D.1.3. Experimental results for ICE test sequence



(a)



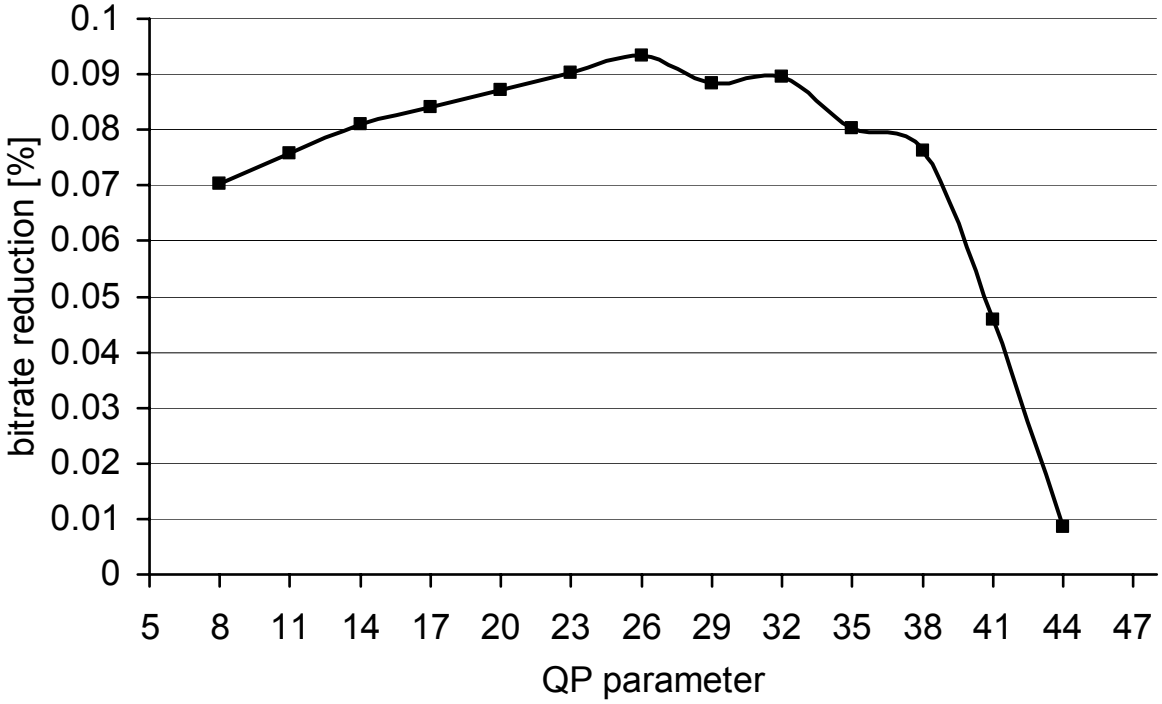
(b)



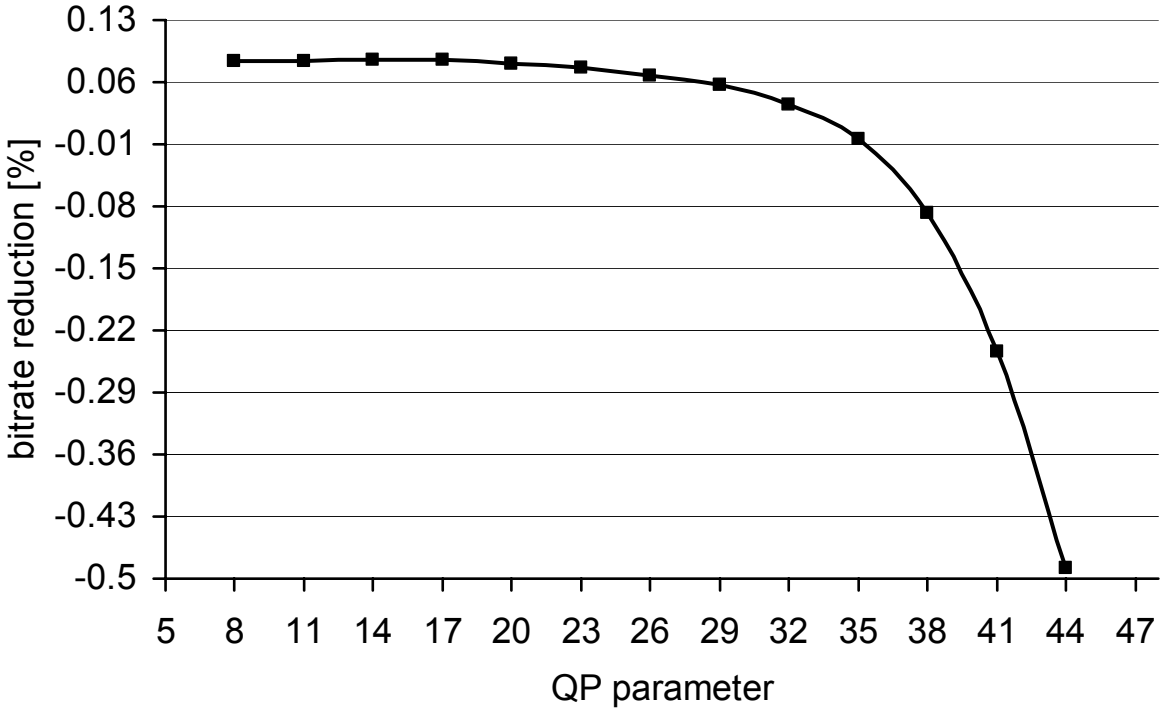
(c)

Figure D.3. Bitrate reduction achieved for ICE test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application in CABAC within the AVC the H.263 arithmetic codec core instead of the M-codec core.

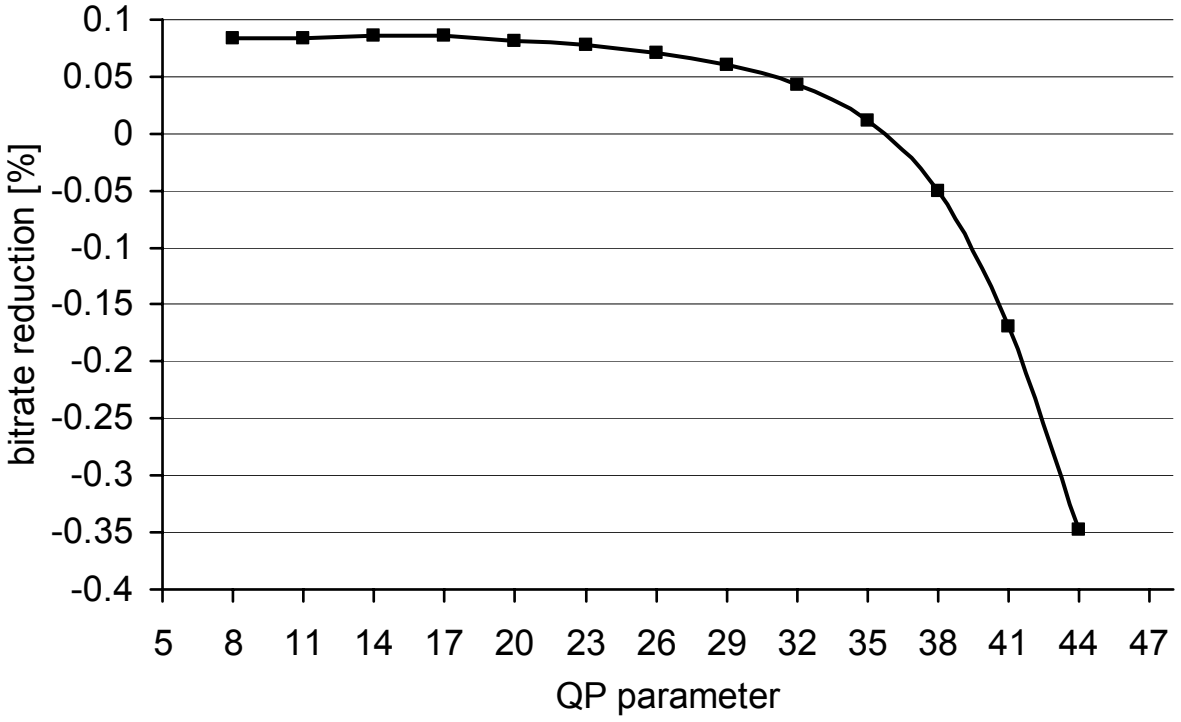
D.1.4. Experimental results for HARBOUR test sequence



(a)



(b)



(c)

Figure D.4. Bitrate reduction achieved for HARBOUR test sequence for I-frames (a), P-frames (b) and the whole test sequence (c). The bitrate reduction is a result of application in CABAC within the AVC the H.263 arithmetic codec core instead of the M-codec core.

Annex E

Experimental comparison of CABAC versus UVLC in AVC codec

E.1. Experimental comparison of coding efficiency of CABAC relative to coding efficiency of UVLC

This section presents the detailed experimental results on the coding efficiency of CABAC relative to UVLC within AVC codec. Experiments have been done in the following scenario:

- The CITY, CREW, ICE and HARBOUR test sequences in 4CIF format have been used;
- The experiments have been done for both intra and inter prediction modes by setting the structure of GOP on I29P;
- Tests have been done for a wide range of the QP parameter (from QP=5 to QP=44 with step equal to 3).

E.1.1. Experimental results for CITY test sequence

Table E.1. Bitrate reduction due to application of CABAC instead of UVLC within AVC for CITY test sequence encoded with I and P slices.

QP parameter	bitrate at the output of CABAC encoder [Mbits/s]	bitrate at the output of UVLC encoder [Mbits/s]	bitrate reduction due to application of CABAC (against UVLC) [%]
5	101.1722	108.9547	7.1428
8	80.3902	86.9222	7.5148
11	63.1403	68.5240	7.8566
14	47.1895	51.3465	8.0961
17	31.3045	34.1807	8.4148
20	18.9680	20.8798	9.1562
23	10.2831	11.3967	9.7715
26	5.0630	5.5775	9.2241
29	2.6073	2.8487	8.4750
32	1.4660	1.6005	8.4033
35	0.9072	1.0032	9.5711
38	0.6172	0.6992	11.7277
41	0.4838	0.5651	14.3879
44	0.4171	0.5014	16.8139

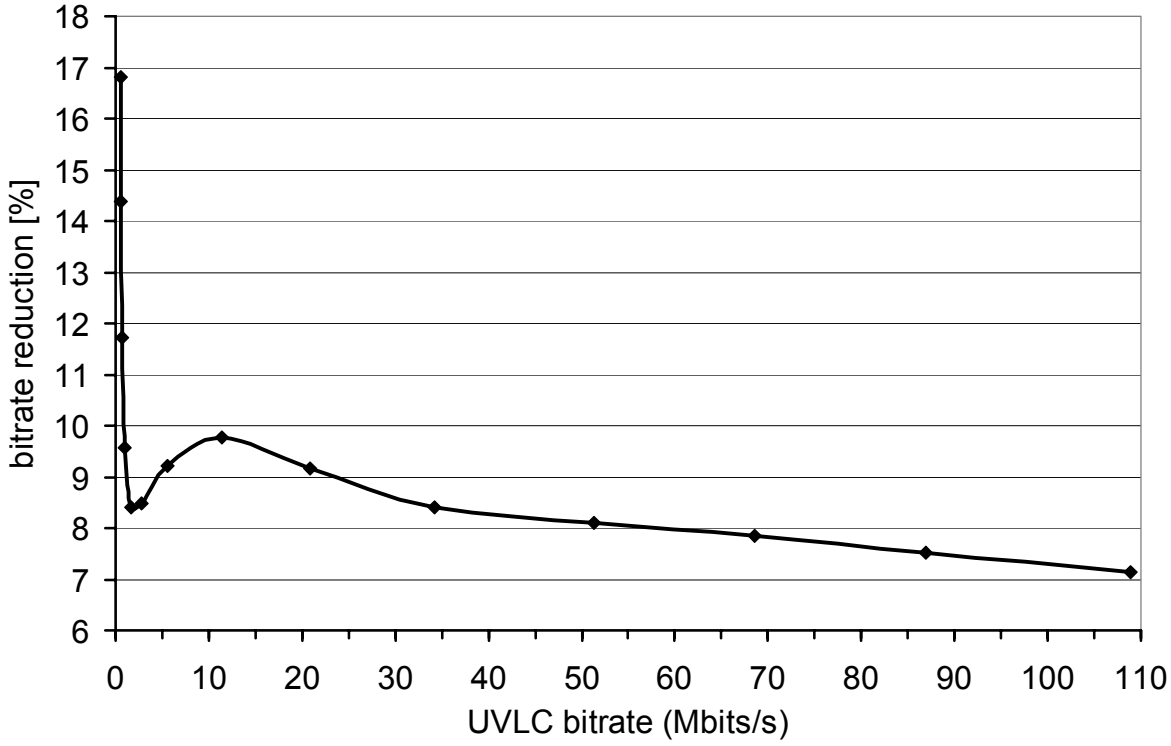


Figure E.1. Bitrate reduction due to application of CABAC instead of UVLC within AVC for CITY test sequence encoded with I and P slices.

E.1.2. Experimental results for CREW test sequence

Table E.2. Bitrate reduction due to application of CABAC instead of UVLC within AVC for CREW test sequence encoded with I and P slices.

QP parameter	bitrate at the output of CABAC encoder [Mbits/s]	bitrate at the output of UVLC encoder [Mbits/s]	bitrate reduction due to application of CABAC (against UVLC) [%]
5	100.5702	109.1457	7.8569
8	79.3321	85.9527	7.7027
11	61.8917	67.2689	7.9935
14	45.9035	50.3233	8.7829
17	29.3435	31.8296	7.8105
20	17.2745	18.7389	7.8147
23	9.5826	10.4331	8.1522
26	5.3341	5.8460	8.7572
29	3.2550	3.6290	10.3067
32	2.1080	2.3900	11.7972
35	1.4310	1.6583	13.7065
38	0.9793	1.1670	16.0830
41	0.7332	0.9065	19.1183
44	0.5744	0.7427	22.6541

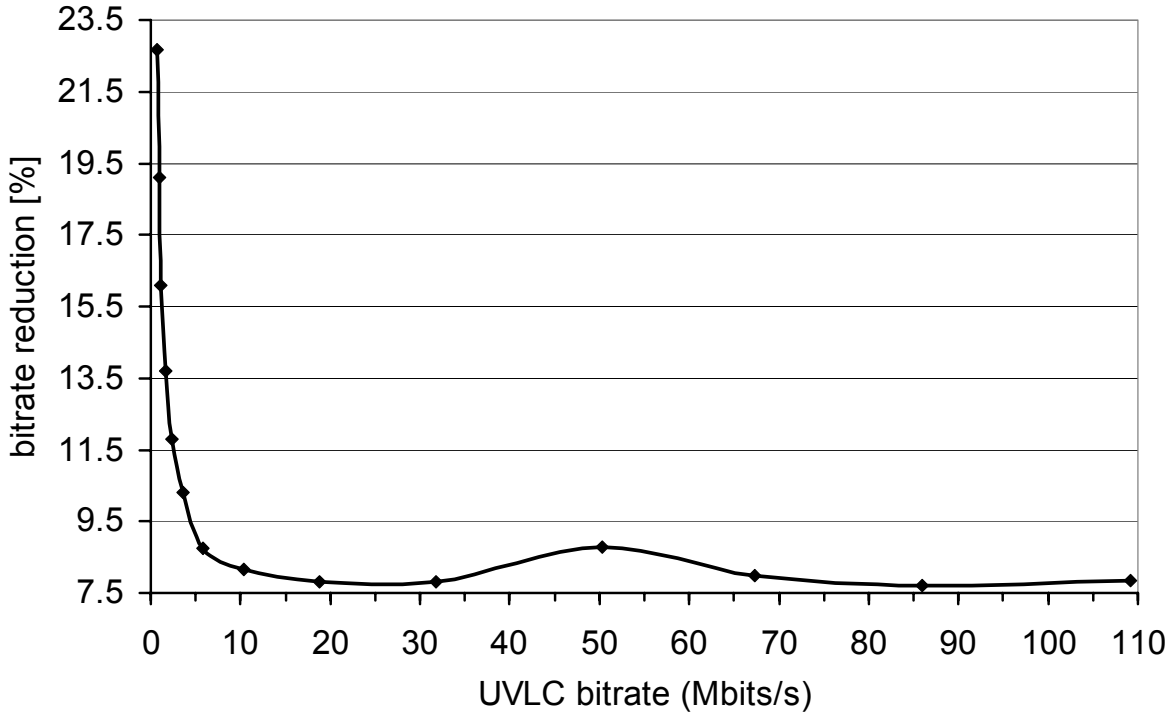


Figure E.2. Bitrate reduction due to application of CABAC instead of UVLC within AVC for CREW test sequence encoded with I and P slices.

E.1.3. Experimental results for HARBOUR test sequence

Table E.3. Bitrate reduction due to application of CABAC instead of UVLC within AVC for HARBOUR test sequence encoded with I and P slices.

QP parameter	bitrate at the output of CABAC encoder [Mbits/s]	bitrate at the output of UVLC encoder [Mbits/s]	bitrate reduction due to application of CABAC (against UVLC) [%]
5	107.3329	114.4082	6.1842
8	86.3388	92.3902	6.5498
11	68.8442	73.9043	6.8469
14	52.6116	56.5264	6.9256
17	36.8574	39.4563	6.5867
20	24.4665	26.4426	7.4732
23	15.5480	17.0500	8.8097
26	9.3372	10.3772	10.0219
29	5.5396	6.2514	11.3864
32	3.2329	3.6889	12.3622
35	1.9208	2.2075	12.9873
38	1.1240	1.3002	13.5548
41	0.6755	0.7959	15.1322
44	0.4117	0.4960	16.9862

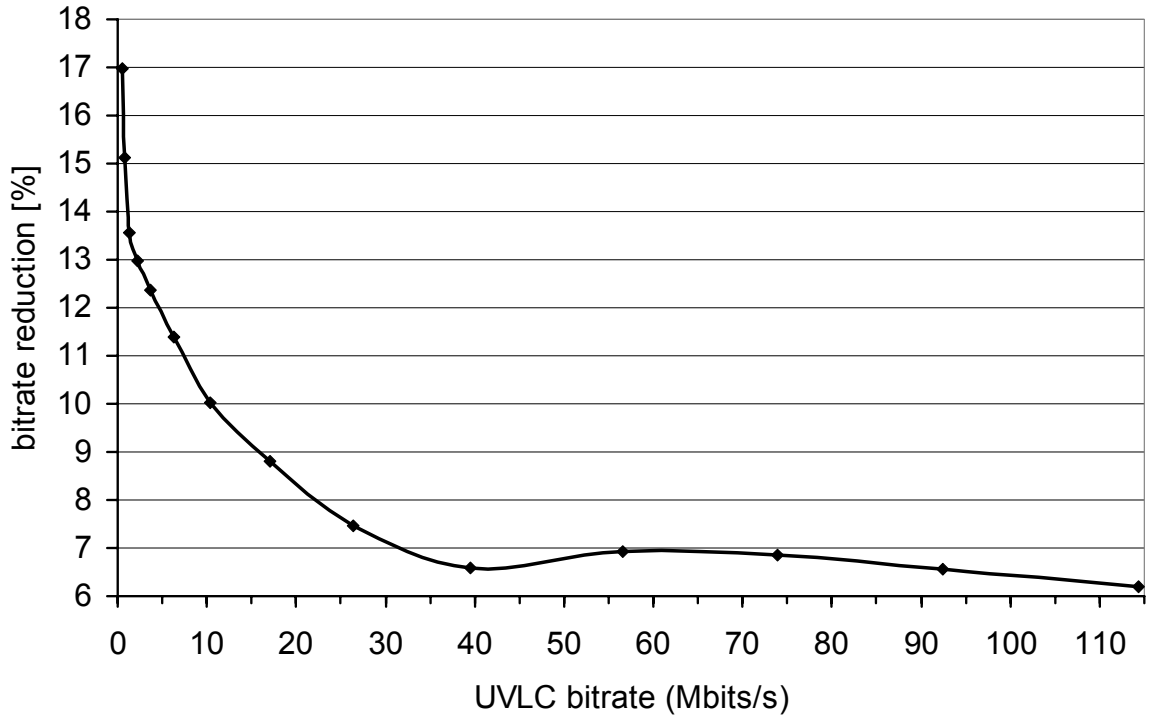


Figure E.3. Bitrate reduction due to application of CABAC instead of UVLC within AVC for HARBOUR test sequence encoded with I and P slices.

E.1.4. Experimental results for ICE test sequence

Table E.4. Bitrate reduction due to application of CABAC instead of UVLC within AVC for ICE test sequence encoded with I and P slices.

QP parameter	bitrate at the output of CABAC encoder [Mbits/s]	bitrate at the output of UVLC encoder [Mbits/s]	bitrate reduction due to application of CABAC (against UVLC) [%]
5	60.1457	65.9202	8.7598
8	42.1920	45.8985	8.0752
11	28.8298	31.2145	7.6398
14	16.3375	17.3981	6.0962
17	8.7447	9.3223	6.1962
20	5.1634	5.5235	6.5207
23	3.1231	3.3408	6.5153
26	1.9275	2.0545	6.1824
29	1.2852	1.3747	6.5071
32	0.8922	0.9636	7.4097
35	0.6484	0.7105	8.7430
38	0.4817	0.5382	10.4974
41	0.3708	0.4246	12.6784
44	0.2947	0.3470	15.0831

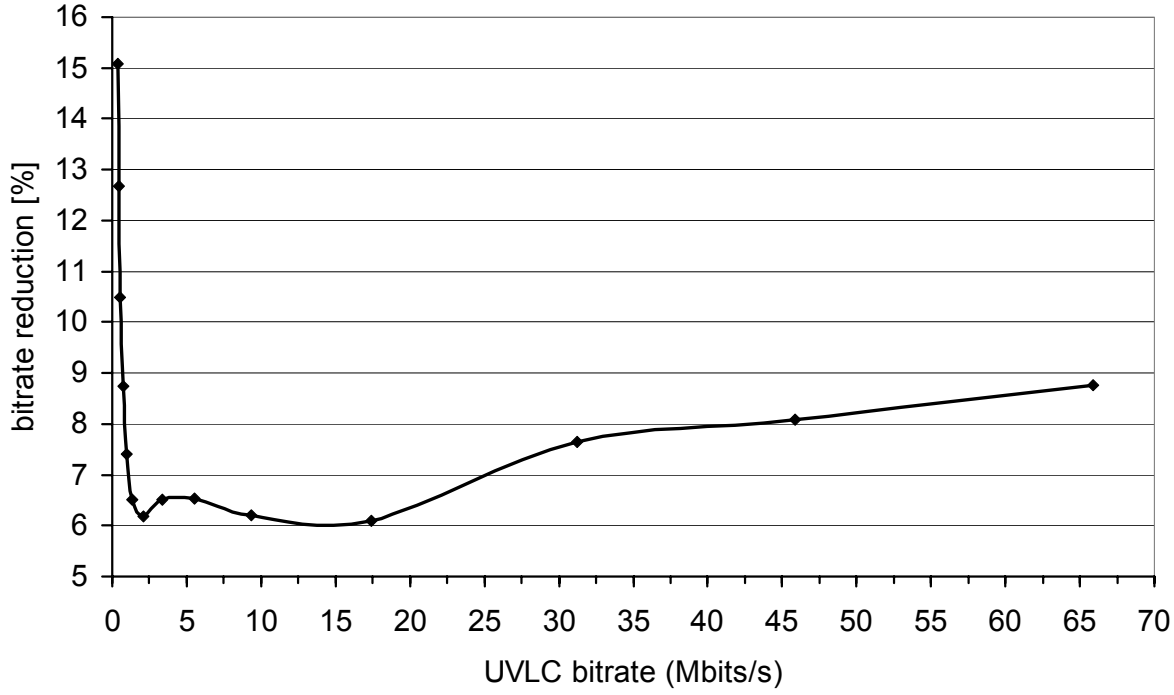


Figure E.4. Bitrate reduction due to application of CABAC instead of UVLC within AVC for ICE test sequence encoded with I and P slices.

E.2. Experimental comparison of complexity of CABAC decoder relative to complexity of UVLC decoder

This section presents the detailed experimental results on the complexity of CABAC decoder relative to the complexity of UVLC decoder within AVC decoder. Experiments have been done under the same conditions as described in Section E.1. Measurements of total decoding times of CABAC and UVLC have been done in a way presented in Section 4.3.2.

E.2.1. Experimental results for CITY test sequence

Table E.5. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for CITY sequence encoded with I and P slices.

QP parameter	bitrate after using CABAC [Mbits/s]	bitrate after using UVLC [Mbits/s]	CABAC decoding time [processor ticks]	UVLC decoding time [processor ticks]	CABAC decoding time relative to UVLC decoding time
5	101.1722	108.9547	1.0897E+11	4.8385E+10	2.2521
8	80.3902	86.9222	9.1391E+10	4.2540E+10	2.1484
11	63.1403	68.5240	7.5966E+10	3.6876E+10	2.0601
14	47.1895	51.3465	5.9757E+10	3.0232E+10	1.9766
17	31.3045	34.1807	4.1352E+10	2.1758E+10	1.9005
20	18.9680	20.8798	2.6296E+10	1.4387E+10	1.8277
23	10.2831	11.3967	1.5323E+10	8.6725E+09	1.7668
26	5.0630	5.5775	8.1347E+09	4.6143E+09	1.7629
29	2.6073	2.8487	4.4584E+09	2.5675E+09	1.7364
32	1.4660	1.6005	2.6125E+09	1.5257E+09	1.7124
35	0.9072	1.0032	1.6709E+09	9.9747E+08	1.6752
38	0.6172	0.6992	1.1752E+09	7.3516E+08	1.5985
41	0.4838	0.5651	9.5141E+08	6.3840E+08	1.4903
44	0.4171	0.5014	8.2589E+08	6.3840E+08	1.2937

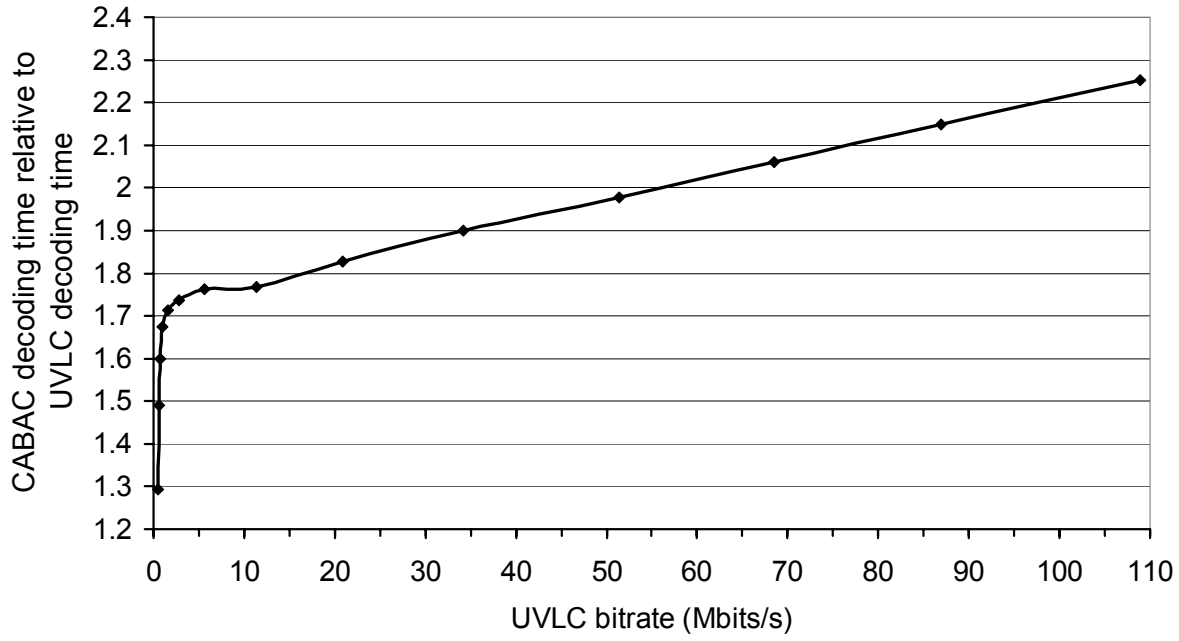


Figure E.5. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for CITY sequence encoded with I and P slices.

E.2.2. Experimental results for CREW test sequence

Table E.6. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for CREW sequence encoded with I and P slices.

QP parameter	bitrate after using CABAC [Mbits/s]	bitrate after using UVLC [Mbits/s]	CABAC decoding time [processor ticks]	UVLC decoding time [processor ticks]	CABAC decoding time relative to UVLC decoding time
5	100.5702	109.1457	1.0941E+11	4.8230E+10	2.2686
8	79.3321	85.9527	9.0711E+10	4.2229E+10	2.1481
11	61.8917	67.2689	7.4898E+10	3.6448E+10	2.0549
14	45.9035	50.3233	5.8844E+10	3.0356E+10	1.9385
17	29.3435	31.8296	4.0817E+10	2.2190E+10	1.8395
20	17.2745	18.7389	2.5784E+10	1.4953E+10	1.7244
23	9.5826	10.4331	1.5162E+10	9.2631E+09	1.6368
26	5.3341	5.8460	8.9253E+09	5.5688E+09	1.6027
29	3.2550	3.6290	5.7442E+09	3.6594E+09	1.5697
32	2.1080	2.3900	3.8902E+09	2.5746E+09	1.5110
35	1.4310	1.6583	2.7283E+09	1.7970E+09	1.5183
38	0.9793	1.1670	1.9093E+09	1.2971E+09	1.4719
41	0.7332	0.9065	1.4757E+09	1.0140E+09	1.4553
44	0.5744	0.7427	1.2078E+09	8.4030E+08	1.4374

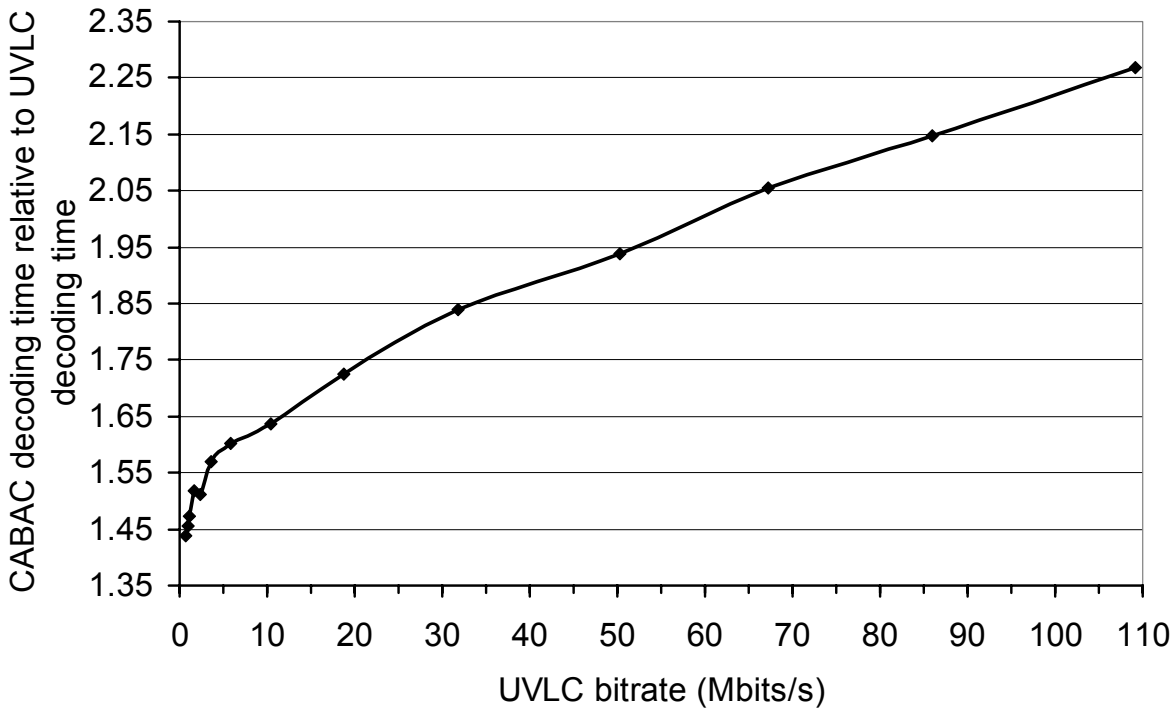


Figure E.6. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for CREW sequence encoded with I and P slices.

E.2.3. Experimental results for HARBOUR test sequence

Table E.7. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for HARBOUR sequence encoded with I and P slices.

QP parameter	bitrate after using CABAC [Mbits/s]	bitrate after using UVLC [Mbits/s]	CABAC decoding time [processor ticks]	UVLC decoding time [processor ticks]	CABAC decoding time relative to UVLC decoding time
5	107.3329	114.4082	1.1591E+11	5.0737E+10	2.2845
8	86.3388	92.3902	9.7428E+10	4.4898E+10	2.1700
11	68.8442	73.9043	8.1951E+10	3.9400E+10	2.0800
14	52.6116	56.5264	6.5862E+10	3.3004E+10	1.9956
17	36.8574	39.4563	4.8080E+10	2.4979E+10	1.9248
20	24.4665	26.4426	3.3290E+10	1.8073E+10	1.8420
23	15.5480	17.0500	2.2659E+10	1.2885E+10	1.7585
26	9.3372	10.3772	1.4719E+10	8.6947E+09	1.6928
29	5.5396	6.2514	9.3622E+09	5.7378E+09	1.6317
32	3.2329	3.6889	5.8244E+09	3.6042E+09	1.6160
35	1.9208	2.2075	3.6139E+09	2.2612E+09	1.5982
38	1.1240	1.3002	2.1783E+09	1.3718E+09	1.5879
41	0.6755	0.7959	1.3499E+09	8.8930E+08	1.5180
44	0.4117	0.4960	8.3098E+08	5.7410E+08	1.4474

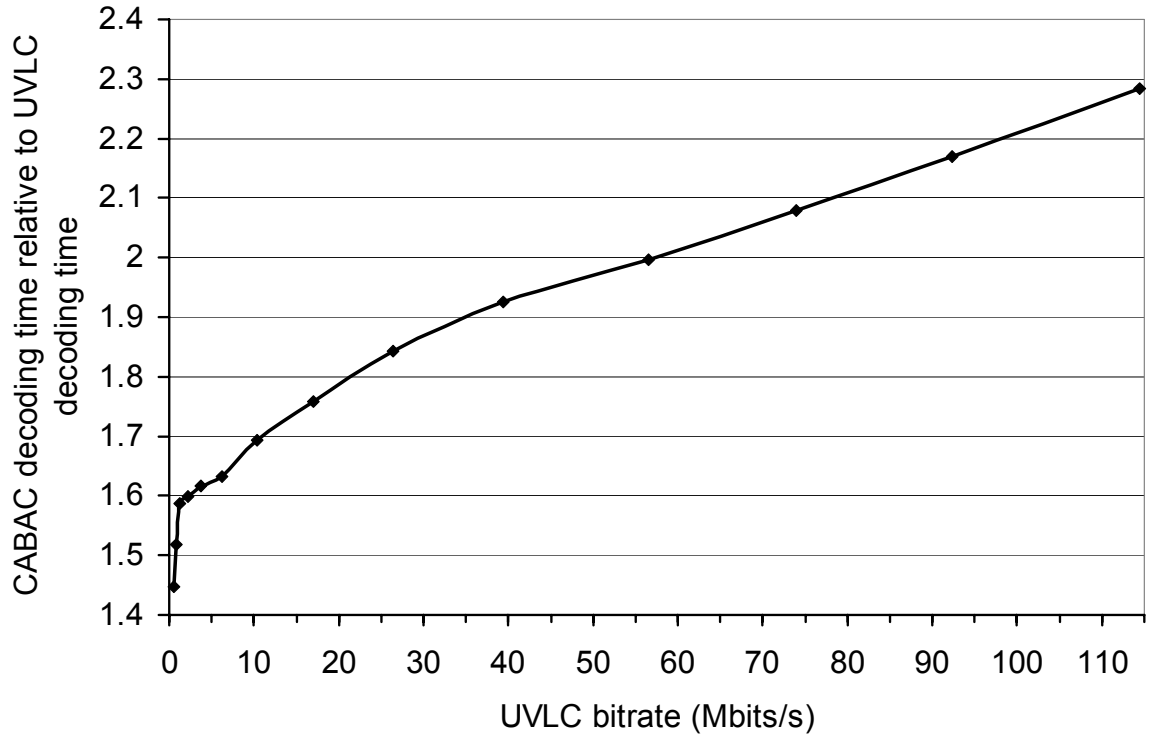


Figure E.7. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for HARBOUR sequence encoded with I and P slices.

E.2.4. Experimental results for ICE test sequence

Table E.8. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for ICE sequence encoded with I and P slices.

QP parameter	bitrate after using CABAC [Mbits/s]	bitrate after using UVLC [Mbits/s]	CABAC decoding time [processor ticks]	UVLC decoding time [processor ticks]	CABAC decoding time relative to UVLC decoding time
5	60.1457	65.9202	5.7369E+10	2.7867E+10	2.0587
8	42.1920	45.8985	4.1873E+10	2.1422E+10	1.9547
11	28.8298	31.2145	3.0575E+10	1.6114E+10	1.8974
14	16.3375	17.3981	1.7991E+10	9.7086E+09	1.8531
17	8.7447	9.3223	9.9273E+09	5.6224E+09	1.7657
20	5.1634	5.5235	6.0758E+09	3.5675E+09	1.7031
23	3.1231	3.3408	3.8261E+09	2.3361E+09	1.6378
26	1.9275	2.0545	2.4391E+09	1.4667E+09	1.6630
29	1.2852	1.3747	1.6800E+09	9.8662E+08	1.7028
32	0.8922	0.9636	1.2094E+09	7.1314E+08	1.6960
35	0.6484	0.7105	9.0878E+08	5.3615E+08	1.6950
38	0.4817	0.5382	6.9189E+08	4.1141E+08	1.6818
41	0.3708	0.4246	5.4746E+08	3.3624E+08	1.6282
44	0.2947	0.3470	4.4673E+08	2.7892E+08	1.6016

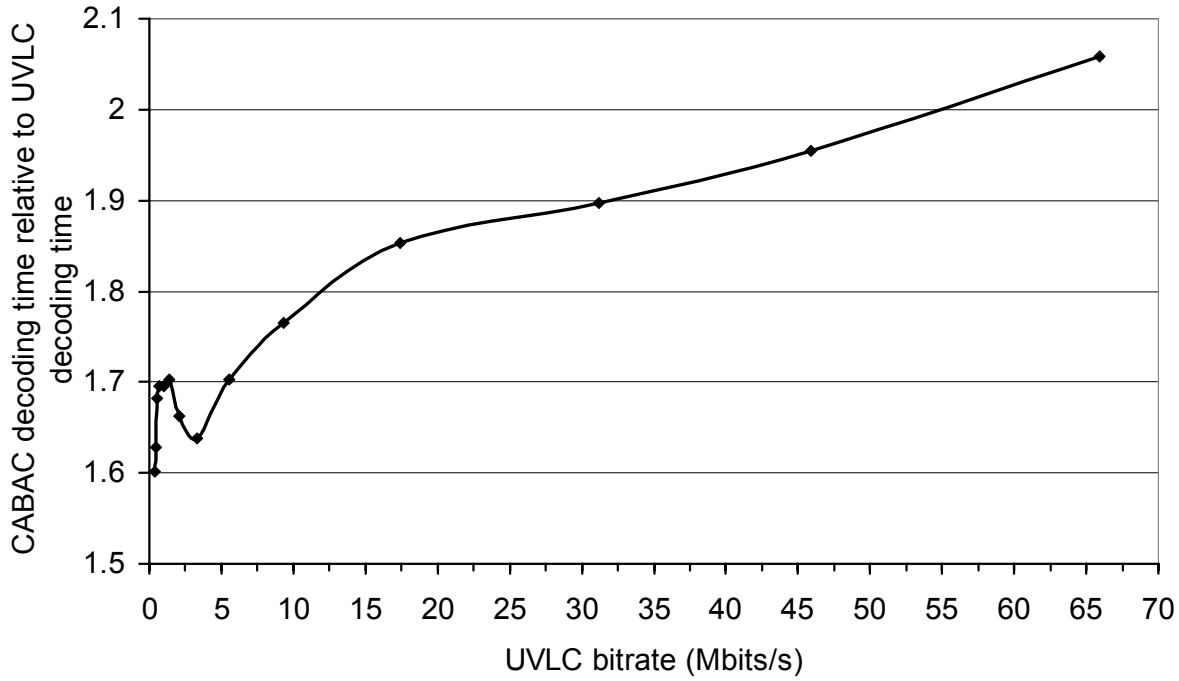


Figure E.8. Increase of total decoding time of CABAC decoder relative to total decoding time of UVLC decoder within AVC for ICE sequence encoded with I and P slices.

Annex F

Test video sequences that have been used to explore the compression performance of the modified AVC relative to the original AVC

The compression performance of the original AVC encoder as well as the modified AVC encoders with sophisticated techniques of data statistics estimation based on CTW and/or PPMA have been analyzed with the set of four test video sequences. These are the CITY, CREW, ICE and HARBOUR test video sequences that have been used in call for proposal by Scalable Video Coding [Schw07] standardization in MPEG.

These are relatively new standard test sequences that are appropriate to evaluate coding artifacts in higher-quality video. The sequences have been adopted for standardization activities by expert groups MPEG (ISO/IEC JTC1/SC29/WG11) and JVT (joint group of MPEG and ITU-T VCEG). The test sequences have 704x576 spatial resolution and 60 frames per second. The 0-th frame of each of used test video sequences has been presented in this annex.

Figure F.1. The 0-th frame of CITY test sequence.

Figure F.2. The 0-th frame of CREW test sequence.

Figure F.3. The 0-th frame of HARBOUR test sequence.

Figure F.4. The 0-th frame of ICE test sequence.

References

[Åberg97] J. Åberg, and Y. M. Shtarkov, Text Compression by Context Tree Weighting. IEEE Data Compression Conference, pp. 377-386, March 1997.

[Achar05a] T. Acharya, and A. K. Ray, Image Processing. Principles and Applications. New Jersey, John Wiley & Sons, 2005.

[Achar05b] T. Acharya, and P. S. Tsai, JPEG 2000 Standard for Image Compression. Concepts, Algorithms and VLSI Architectures. New Jersey, John Wiley & Sons, 2005.

[ADB] Advanced Digital Broadcast, www.adbglobal.com.

[Apos85] A. Apostolico, and A. S. Fraenkel, Robust transmissions of unbounded strings using Fibonaci representations. Technical Report CS85-14, Dpt of Applied Mathematics, The Weizmann Institute of Science, Rehovot Israel, 1985.

[AVC] ISO/IEC 14496-10, Generic Coding of Audio-Visual Objects, Part 10: Advanced Video Coding, March 2006.

[AVCSoft] H.264/AVC software coordination site – http://bs.hhi.de/~suehring/tml.

[AVS] Audio Video Coding Standard Workgroup of China (AVS), The Standards of People's Republic of China GB/T 20090.2-2006, Information Technology – Advanced Coding of Audio and Video – Part 2:Video, 2006.

[Bar04] V. Baronchini, and T. Oelbaum, Subjective test results for the CfP on scalable video coding technology. Doc. ISO/IEC JTC1/SC29/WG11 M10737, Munich, March 2004.

[Begl04] R. Begleiter, R. El-Yaniv, and G. Yona, On Prediction Using Variable Order Markov Models. Journal of Artificial Intelligence Research, Vol. 22 (2004), pp. 385-421, December 2004.

[Bely06] E. Belyaev, M. Gilmutdinov, and A. Turlikov, Binary Arithmetic Coding System with Adaptive Probability Estimation by "Virtual Sliding Window". IEEE International Symposium on Consumer Electronics, pp. 1-5, 2006.

[Bloom98] C. Bloom, Solving the Problems of Context Modeling, http://www.cbloom.com/papers/ppmz.zip, 1998.

[Bobi02] P. Bobiński i W. Skarbek, Kodowanie Binarne w Rodzinie Standardów H.26X. Multimedialne i Sieciowe Systemy Informacyjne, MiSSI 2002, Kliczków, str. 249-258, Wrzesień 2002.

[Bonc06] C. Boncelet, Lossless Image Compression with BCTW. IEEE International Conference on Image Processing 2006, pp. 2281-2284, 8-11 October, 2006.

[Bovik00] A. Bovik (Editor), Handbook of Image and Video Processing. Canada, Academic Press, A Harcourt Science and Technology Company, 2000.

[Boy04] J. M. Boyce, Weighted Prediction in the H.264/MPEG AVC Video Coding Standard. International Symposium on Circuit and Systems (ISCAS) 2004, pp. 789-792, Canada, 23-26 May 2004.

[bzip2] J. Seward, Bzip2 and libbzip2, Version 1.0.3. A program and library for data compression. http://www.bzip.org.

[Clear84] J. G. Cleary, and I. H. Witten, Data Compression Using Adaptive Coding and Partial String Matching. IEEE Transactions on Communication, Vol. 32, No. 4, pp. 396-402, April 1984.

[Clear93] J. G. Cleary, and W. J. Teahan, Unbounded Length Contexts for PPM. Computer Journal, Vol. 36, No. 5, pp. 1-9, 1993.

[Côté98] G. Côté, B. Erol, M. Gallant, and F. Kossentini, H.263+: Video Coding at Low Bit Rates. IEEE Transactions on Circuit and Systems for Video Technology, Vol. 8, No. 7, pp. 849-866, November 1998.

[Doma98] M. Domański, Zaawansowane techniki kompresji obrazów i sekwencji wizyjnych. Poznań, 1998.

[Ekstr96] N. Ekstrand, Lossless Compression of Grayscale Images via Context Tree Weighting. Data Compression Conference 1996, pp. 132-139, 31 March – 3 April 1996.

[Elias75] P. Elias, Universal Codeword Sets and Representation of the Integers. IEEE Transactions on Information Theory, Vol. 21, No. 2, pp. 194-203, 1975.

[Erol98] B. Erol, M. Gallant, G. Côté, and F. Kossentini, The H.263+ Video Coding Standard: Complexity and Performance. Data Compression Conference, pp. 259-268, 30 March-1 April 1998.

[Fan04] L. Fan, S. Ma, and F. Wu, Overview of AVS Video Standard. IEEE International Conference on Multimedia and Expo 2004 (ICME), pp. 423-426, Taipei, Taiwan, 27-30 June 2004.

[Fere03] C. Feregrino, High Performance PPMC Compression Algorithm. Fourth Mexican International Conference on Computer Science 2003, pp. 135-142, 8-12 September, 2003.

[Firo06] M. H. Firooz, and M. S. Sadri, Improving H.264/AVC Entropy Coding Engine Using CTW method. Picture Coding Symposium, April 2006.

[Flier04] M. Flierl, and B. Girod, Video Coding with Superimposed Motion-Compensated Signals. Applications to H.264 and Beyond, United States of America, Kluwer Academic Publisher, 2004.

[Gall75] R. G. Gallager, and D. Van Voorhis, Optimal Source Codes for Geometrically Distributed Integer Alphabets. IEEE Transactions on Information Theory, Vol. 21, pp. 228-230, March IT-1975.

[Gall78] R. G. Gallager, Variations on a Theme by Huffman. IEEE Transactions on Information Theory, Vol. IT-24, No. 6, pp. 668-674, 1978.

[Gard98] T. R. Gardos, H.263+, The New ITU-T Recommendation for Video Coding at Low Bit Rates. IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 6, pp. 3793-3796, 12-15 May 1998.

[Ghan04] M. Ghandi, M. M. Ghandi, and M. B. Shamsollahi, A Novel Context Modeling Scheme for Motion Vectors Context-Based Arithmetic Coding. IEEE Canadian Conference on Electrical & Computer Engineering (CCECE04), May 2-5, 2004, Ontario, Canada.

[Golo66] S. W. Golomb, Run-Length Encoding. IEEE Transactions on Information Theory, Vol. IT-12, pp. 399-401, 1966.

[Graj05] T. Grajek i D. Karwowski, Złożoność Obliczeniowa i Efektywność Kodowania Entropijnego w Standardzie H.264/AVC. Krajowa Konferencja Radiokomunikacji, Radiofonii i Telewizji, str. 377-380, Kraków, 15-17 Czerwca 2005.

[gzip] GZIP Homepage, http://www.gzip.org.

[H263] ITU-T Rec. H.263, Video Coding for Low Bit Rate Communication, August 2005.

[Hong04] D. Hong, M. van der Schaar, and B. Pesquet – Popescu, Arithmetic Coding with Adaptive Context-Tree Weighting for the H.264 Video Coders. Visual Communications and Image Processing 2004, Vol. 5308, pp. 1226-1235, January 2004.

[Howa92] P. G. Howard, and J. S. Vitter, Practical Implementations of Arithmetic Coding. Image and Text Compression, Storer, Ed., pp. 85-112, Kluwer Academic, 1992.

[Howa93] P. G. Howard, The Design and Analysis of Efficient Lossless Data Compression Systems, PhD Dissertation, Department of Computer Sciences, Brown University, 1993.

[Huff52] D. A. Huffman, A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the I. R. E, pp. 1098-1101, September, 1952.

[IntelComp] Intel C++ Compiler for Windows, http://www.intel.com.

[Jack05] K. Jack, Video Demystified. A Handbook for the Digital Engineer, fourth edition. Newnes, 2005.

[Jain81] J. R. Jain, and A. K. Jain, Displacement Measurement and Its Application in Interframe Image Coding. IEEE Trans. Commun. Vol. COM-29, pp. 1799-1808, 1981.

[JBIG] ISO/IEC 11544 and ITU-T Rec. T.82, Information Technology – Coded Representation of Pictures and Audio Information – Progressive Bi-Level Image Compression, March 1993.

[JBIG2] ISO/IEC 14492 and ITU-T Rec. T.88, JBIG2 Bi-Level Image Compression standard, 2000.

[JPEG] ISO/IEC 10918-1 and ITU-T Rec. T.81, Information Technology – Coded Representation of Picture and Audio Information – Digital Compression and Coding of Continuous-Tone Still Images (JPEG standard), 1993.

[JPEGLS] ISO/IEC 14495-1 and ITU-T Rec. T.87, Information Technology – Lossless and Near Lossless Compression of Continuous – Tone still Images, 1999.

[JPEG2000] ISO/IEC 15444-1 and ITU-T Rec. T.800, Information Technology – JPEG 2000 image coding system, 2000.

[JSVM07] JSVM Software Manual, Joint Video Team/ISO/IEC, June 2007.

[Kalv07] H. Kalva, and J. B. Lee, The VC-1 Video Coding Standard. IEEE Multimedia, October-December 2007, Vol. 14, No. 4, pp. 88-91.

[Kam03] N. Kamaci, and Y. Altunbasak, Performance Comparison of the Emerging H.264 Video Coding Standard with the Existing Standards. IEEE International Conference on Multimedia and Expo (ICME), Vol. 1 (6-9), pp. 345-348, 2003.

[Karw04a] D. Karwowski, Kodowanie Entropijne w Standardzie H.264/AVC. Krajowa Konferencja Radiokomunikacji, Radiofonii i Telewizji, str. 471-474, Warszawa, 16-18 czerwca 2004.

[Karw04b] D. Karwowski, An Efficient Architecture of H.264/AVC CAVLC Decoder. 11th International Workshop on Systems, Signals and Image Processing (IWSSIP), pp. 151-154, Poznań, Poland, September 13-15, 2004.

[Karw06] D. Karwowski, Ulepszone Adaptacyjne Kodowanie Arytmetyczne w Zaawansowanym Koderze Wizyjnym H.264/AVC Wykorzystujące Ważenie Drzew Kontekstów (CTW). V Sympozjum Naukowe "Techniki Przetwarzania Obrazu", str. 469-475, Serock, Listopad 2006.

[Karw07a] D. Karwowski, Improved Arithmetic Coding in H.264/AVC Using Context-Tree Weighting and Prediction by Partial Matching. European Signal Processing Conf. EUSIPCO 2007, pp. 1270-1274, September 2007, Poznań, Poland.

[Karw07b] D. Karwowski, and M. Domański, Improved Arithmetic Coding In H.264/AVC Using Context-Tree Weighting Method. Picture Coding Symposium PCS 2007, November 2007, Lisboa, Portugal.

[Kern88] W. Kernighan, and M. Ritchie, C Programming Language (2nd Edition). Prentice Hall Software, 1988.

[Kuon07] I. Kuon, and J. Rose, Measuring the Gap Between FPGAs and ASICs. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., Vol. 26, No. 2, p. 203, February 2007.

[Krich81] R. E. Krichevsky, and V. K. Trofimov, The Performance of Universal Encoding, IEEE Transactions on Information Theory, Vol. IT-27, pp. 199-207, March 1981.

[Lange06] R. Lange, Multiresolution Representation of Motion Vectors in Video Compression, PhD Dissertation, Poznań University of Technology, 2006.

[Lam06] P. Lambert, W. Neve, P. Neve, I. Moerman, P. Demeester, and R. Walle, Rate-Distortion Performance of H.264/AVC Compared to State-of-the-art Video Codecs. IEEE Transactions on Circuit and Systems for Video Technology, Vol. 16, issue 1, pp. 134-140, January 2006.

[Luo03] Y. Luo, and R. K. Ward, Removing the Blocking Artifacts of Block-Based DCT Compressed Images. IEEE Transactions on Image Processing, Vol. 12, No. 7, pp. 838-842, July 2003.

[Mah05] M. Mahoney, Adaptive Weighting of Context Models for Lossless Data Compression. Florida Tech. Technical Report CS-2005-16, 2005.

[Marp01] D. Marpe, G. Blättermann, G. Heising, and T. Wiegand, Video Compression Using Context-Based Adaptive Arithmetic Coding. IEEE International Conference on Image Processing (ICIP'01), Thessaloniki, Greece, September 2001.

[Marp02a] D. Marpe, G. Blättermann, G. Heising, and T. Wiegand, Efficient Entropy Coding for Video Compression by Using Context-Based Adaptive Binary Arithmetic Coding. 4th International ITG Conference on Source and Channel Coding (ICSCC'02), Berlin, Germany, January 2002.

[Marp02b] D. Marpe, H. Schwarz, G. Blättermann, G. Heising, and T. Wiegand, Context-Based Adaptive Binary Arithmetic Coding in JVT/H.26L. IEEE International Conference on Image Processing (ICIP'02), Rochester, NY, USA, September 2002.

[Marp03a] D. Marpe, H. Schwarz, and T. Wiegand, Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. IEEE

Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620-636, July 2003.

[Marp03b] D. Marpe, and T. Wiegand, A highly Efficient Multiplication-Free Binary Arithmetic Coder and Its Application in Video Coding. IEEE International Conference on Image Processing (ICIP'03), Barcelona, Spain, September 2003.

[Marp04] D. Marpe, Adaptive Context-Based and Tree-Based Algorithms for Image Coding and Denoising. Doctoral Dissertation, Universität Rostock, Rostock 2004.

[Marp05a] D. Marpe, S. Gordon, and T. Wiegand, H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance, and Application Areas. IEEE International Conference on Image Processing (ICIP'05), Genova, Italy, September 2005.

[Marp05b] D. Marpe, The H.264 / MPEG4-AVC Standard, Core Coding technology and Recent Extensions. Proc. IWSSIP 2005, Chalkida, Greece, September 22-24, 2005.

[Marp06a] D. Marpe, G. Marten, and H. L. Cycon, A Fast Renormalization Technique for H.264/MPEG4-AVC Arithmetic Coding. 51st Internationales Wissenschaftliches Kolloquium Technische Universitat Ilmenau, September 2006.

[Marp06b] D. Marpe, T. Wiegand, and G. J. Sullivan, The H.264/MPEG4 Advanced Video Coding Standard and its Applications. IEEE Image Communications Magazine, Vol. 44, Is. 8, pp. 134-143, August 2006.

[Marp06c] D. Marpe, H. Kirchhoffer, and G. Marten, Fast Renormalization for H.264/MPEG4-AVC Arithmetic Coding. Proc. 14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006.

[Mila06] S. Milani, and G. A. Mian, An Improved Context Adaptive Binary Arithmetic Coder for the H.264/AVC Standard. Proc. of European Signal Processing Conference (EUSIPCO 2006), September 4-8, 2006, Florence, Italy.

[Moff90] A. Moffat, Implementing the PPM Data Compression Scheme. IEEE Transactions on Communications, Vol. 38, No. 11, pp. 1917-1921, November 1990.

[MPEG-1] ISO/IEC 11172-2, Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s, Part 2: Video. (MPEG-1), November 1993.

[MPEG-2] ISO/IEC 13818-2 and ITU-T Rec. H.262, Generic Coding of Moving Pictures and Associated Audio Information – Part 2: Video. (MPEG-2), November 1994.

[MP2AAC] ISO/IEC 13818-7, Information Technology – Generic Coding of Moving Pictures and Associated Audio, Part 7: Advanced Audio Coding, 1997.

[MP4AAC] ISO/IEC 14496-3, Information Technology – Coding of Audio Visual Objects – Audio, 2001.

[Mrak03a] M. Mrak, D. Marpe, and S. Grgic, Comparison of Context-Based Adaptive Binary Arithmetic Coders in Video Compression. Proc. EC-VIP-MC'03, July 2003.

[Mrak03b] M. Mrak, D. Marpe, and T. Wiegand, Application of Binary Context Trees in Video Compression. Picture Coding Symposium (PCS'03), St. Malo, France, April 2003.

[Mrak03c] M. Mrak, D. Marpe, and T. Wiegand, A Context Modeling Algorithm and its Application in Video Coding. IEEE International Conference on Image Processing (ICIP'03), Barcelona, Spain, September 2003.

[Mual02] M. E. Al-Mualla, Video Coding for Mobile Communications. Efficiency, Complexity, and Resilience. United States of America, Academic Press, An Elsevier Science Imprint, 2002.

[Ohm04] J. –R. Ohm, Multimedia Communication Technology. Representation, Transmission and Identification of Multimedia Signals, Springer. 2004.

[Oster04] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, Video Coding with H.264/AVC: Tools, Performance, and Complexity. IEEE Circuits and Systems Magazine, Vol. 4, No. 1, pp. 7-28, April 2004.

[Pas76] R. Pasco, Source Coding Algorithms for Fast Data Compression. Doctoral Dissertation, Stanford University, 1976.

[Penn88] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. B. Arps, An Overview of the Basic Principles of the Q-coder Adaptive Binary Arithmetic Coder. IBM Journal of research and development, Vol. 32, No. 6, pp. 771-726, November 1998.

[PNG] ISO/IEC 15948, Information technology – Computer Graphics and Image Processing – Portable Network Graphics (PNG): Functional specification, 2003.

[Przel05] A. Przelaskowski, Kompresja danych. Wydawnictwo BTC, Warszawa, Polska, 2005.

[Raja04] G. Raja, and M. J. Mirza, Performance Comparison of Advanced Video Coding H.264 Standard with Baseline H.263 and H.263+ Standards. International Symposium on Communications and Information Technologies, pp. 743-746, Sapporo, Japan, October 26-29, 2004.

[Ran95] X. Ran, and C. Choo, Syntax-Based Arithmetic Video Coding for Very Low Bitrate Visual Telephony, International Conference on Image Processing, Vol. 2, pp. 410-413, October 1995.

[Rice79] R. F. Rice, Some Practical Universal Noiseless Coding Techniques. Jet Propulsion Laboratory, Pasadena, California, JPL Publication 79-22, March 1979.

[Richa02] I. E. G. Richardson, Video Codec Design. Developing Image and Video Compression Systems. John Wiley & Sons, 2002.

[Richa03] I. E. G. Richardson, H.264 and MPEG-4 Video Compression. John Wiley & Sons, 2003.

[Rijk96] K. Rijkse, H.263: Video Coding for Low-Bit-Rate Communication. IEEE Communication Magazine, Vol. 34, pp. 42-45, December 1996.

[Riss76] J. J. Rissanen, Generalized Kraft Inequality and Arithmetic Coding. IBM Journal of Research and Development, Vol. 20, pp. 198-203, May 1976.

[Riss79] J. J. Rissanen, and G. G. Langdon, Arithmetic Coding. IBM Journal of Research and Development, Vol. 23, No. 2, pp. 149-162, March 1979.

[Ryab96] B. Y. Ryabko, Imaginary Sliding Window as a Tool for Data Compression. Problems of information transmission, pp. 156-163, January 1996.

[Said04] A. Said, Introduction to Arithmetic Coding – Theory and Practice. Imaging systems Laboratory, HP Laboratories Palo Alto, April 21, 2004.

[Salom06] D. Salomon, Data Compression. The Complete Reference – 4th Edition. Springer-Verlag, 2006.

[Salom07] D. Salomon, Variable-Length Codes for Data Compression. Springer-Verlag, 2007.

[Sayo00] K. Sayood, Introduction to Data Compression, 2nd ed. Morgan Kaufmann, 2000.

[Schäf03] R. Schäfer, T. Wiegand, and H. Schwarz, The Emerging H.264/AVC Standard. EBU Technical Review, Special Issue on “Best of 2003", January 2003.

[Schw02a] H. Schwarz, and T. Wiegand, The Emerging JVT/H.26L Video Coding Standard. International Broadcast Convention (IBC'02), Amsterdam, Netherlands, July 2002.

[Schw02b] H. Schwarz, D. Marpe, and T. Wiegand, CABAC and slices, Joint Video Team of ISO/IEC JTC1/SC29/WG11 & ITU-T SG16/Q.6 Doc. JVT-D020, Klagenfurt, Austria, July 2002.

[Schw07] H. Schwarz, D. Marpe, and T. Wiegand, Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuit and Systems for Video Technology, Vol. 17, No. 9, pp. 1103-1120, September 2007.

[Shan48] C. E. Shannon, A Mathematical Theory of Communications. Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, 1948.

[Shkar02] D. Shkarin, PPM: One Step to Practicality. Data Compression Conference 2002, pp. 202-211, 2-4 April 2002.

[Skarb93] W. Skarbek, Metody Reprezentacji Obrazów Cyfrowych. Warszawa, 1993.

[Skarb98] W. Skarbek, Multimedia. Algorytmy i Standardy Kompresji. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1998.

[Stock03] T. Stockhammer, and T. Wiegand, H.264/AVC for Wireless Applications. IEEE International Workshop on Mobile Multimedia Communications (MoMuC), Munich, Germany, October 2003.

[Sull04] G. Sullivan, P. Topiwala, and A. Luthra, The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions, SPIE Conference on Applications of Digital Image Processing XXVII, August, 2004.

[Sull05] G. Sullivan, and T. Wiegand, Video Compression - From Concepts to the H.264/AVC Standard. Proceedings of the IEEE, Special Issue on Advances in Video Coding and Delivery, Vol. 93, No. 1, pp. 18-31, January 2005.

[Sunna05] P. Sunna, AVC/H.264 – An Advanced Video Coding System for SD and HD Broadcasting, EBU Technical Review No. 302, April 2005.

[Taub02] D. S. Taubman, and M. W. Marcellin, JPEG2000 Image Compression Fundamentals. Standards and Practice, Kluwer Academic Publishers, Boston, 2002.

[Teuh78] J. Teuhola, A Compression Method for Clustered Bit-Vectors. Information Processing Letters, Vol. 7, pp. 308-311, October 1978.

[TIFF] Adobe Systems Incorporated, TIFF, revision 6.0. Final – June 3, 1992.

[TI642] Texas Instrument, TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor. www.ti.com, July, 2002.

[Tri02] G. A. Triantafyllidis, D. Tzovaras, and M. G. Strintzis, Blocking Artifact Detection and Reduction in Compressed Data. IEEE Transactions on Circuit and Systems for Video Technology, Vol. 12, No. 10, pp. 877-890, October 2002.

[Tzir94] G. Tziritas, C. Labit, Motion Analysis for Image Sequence Coding. Amsterdam, Elsevier 1994.

[VCEG07] ITU-T VCEG SG 16 Q.6, Doc. VCEG-AG01, 33rd VCEG Meeting Report, Shenzhen, China, 20 October 2007.

[VC-1] Society of Motion Picture and Television Engineers, VC-1 Compressed Video Bitstream Format and Decoding Process, SMPTE 421M-2006, 2006.

[Verilog] IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language, IEEE Std 1364-1995, IEEE, 1995.

[Virtex-5] Virtex-5 FPGA Family Datasheet. Xilinx, Inc., San Jose, CA, 2007, http://www.xilinx.com/.

[Volf98] P. A. J. Volf, and F. M. J. Willems, Switching Between Two Universal Source Coding Algorithms. Data Compression Conference, pp. 491–500, Snowbird, Utah, March 30 – April 1 1998.

[Volf99] P. A. J. Volf, F. M. J. Willems, and Tj. J. Tjalkens, Complexity Reducing Techniques for the CTW Algorithm. In Symp. on Inform. Theory in the Benelux, Vol. 20, pp. 25–32, Haasrode, Belgium, May 27-28 1999.

[Volf02] P. A. J. Volf, Weighting Techniques in Data Compression: Theory and Algorithms. Doctoral Dissertation, Technische Universiteit Eindhoven, Eindhoven 2002.

[Wan04] Q. Wang, D. Zhao, S. Ma, Y. Lu, Q. Huang, and W. Gao, Context-Based 2D-VLC for Video Coding. IEEE International Conference on Multimedia and Expo 2004 (ICME), pp. 89-92, Taipei, Taiwan, 27-30 June 2004.

[Welch84] T. A. Welch, A Technique for High-Performance Data Compression. IEEE Computer, pp. 8-19, June 1984.

[Wieg03a] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, Overview of the H.264/AVC Video Coding Standard. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 560-576, July 2003.

[Wieg03b] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, Rate-Constrained Coder Control and Comparison of Video Coding Standards. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 688-703, July 2003.

[Wieg07] T. Wiegand, and G. J. Sullivan, The H.264/AVC Video Coding Standard {Standards in a nutshell}, IEEE Signal Processing Magazine, Vol. 24, No. 2, March 2007.

[Will06] F. M. J. Willems, T. J. Tjalkens, and T. Ignatenko, Context-Tree Weighting and Maximizing: Processing Betas. Inaugural Workshop ITA (Information Theory and its Applications), UCSD Campus, La Jolla, Febr. 6 - 10, 2006.

[Will95] F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens, The Context-Tree Weighting Method: Basic Properties. IEEE Transactions on Information Theory, Vol. 41, No. 3, pp. 653-664, May 1995.

[Will97a] F. M. J. Willems, and Tj. J. Tjalkens, Complexity Reduction of the Context-Tree Weighting Algorithm: A study for KPN research. Technical Report EIDMA-RS.97.01, Euler Institute for Discrete Mathematics and its Applications, Eindhoven University of Technology, 1997.

[Will97b] F. M. J. Willems, and Tj. J. Tjalkens, Complexity Reduction of the Context-Tree Weighting Method. In Symp. on Inform. Theory in the Benelux, Vol. 18, pp. 123–130, Veldhoven, The Netherlands, May 15-16 1997.

[Will98a] F. M. J. Willems, The Context-Tree Weighting Method: Extensions. IEEE Transactions on Information Theory, Vol. 44, No. 2, pp. 792-797, March 1998.

[Will98b] F. M. J. Willems, and Tj. J. Tjalkens, Reducing Complexity of the Context-Tree Weighting Method. Proc. IEEE International Symposium on Information Theory, p. 347, Cambridge, Mass., August 16-21, 1998.

[Witt87] I. H. Witten, J. G. Cleary, and R. Neal, Arithmetic Coding for Data Compression. Commun ACM., no. 6, pp. 520-540, June 1987.

[Woot05] C. Wootton, A Practical Guide to Video and Audio Compression. From Sprockets and Rasters to Macro Blocks, Focal Press, 2005.

[Wysz82] G. Wyszecki, and W. S. Styles, Color Science: Concepts and Methods, Quantitative Data and Formulae. Second Edition, Wiley 1982.

[Xiao06] S. Xiao, and C. G. Boncelet, On the Use of Context-Weighting in Lossless Bilevel Image Compression. IEEE Transactions on Image Processing, Vol. 15, No. 11, November 2006.

[XilinxISE] Xilinx ISE 9.2i software manuals, http://www.xilinx.com/.

[x264Soft] x264 Project. http://www.videolan.org/developers/x264.html.

[Ziv77] J. Ziv, and A. Lempel, A Universal Algorithm for Data Compression. IEEE Transactions on Information Theory, Vol. IT-23, No. 3, pp. 337-343, May 1977.

[Ziv78] J. Ziv, and A. Lempel, A Universal Algorithm for Data Compression. IEEE Transactions on Information Theory, Vol. IT-24, No. 5, pp. 530-536, September 1978.