Multi-Frame Optimized Quantization for High Efficiency Video Coding

vorgelegt von
Dipl.-Ing.
Martin Winken
geb. in Berlin

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuß:

Vorsitzender:	Prof. Dr.-Ing. Thomas Sikora
Gutachter:	Prof. Dr.-Ing. Thomas Wiegand
Gutachter:	Prof. Dr.-Ing. Jens-Rainer Ohm (RWTH Aachen)
Gutachter:	Dr.-Ing. Markus Flierl (KTH Stockholm, Schweden)

Tag der wissenschaftlichen Aussprache: 28. Mai 2015

Berlin 2015

Abstract

In video coding, there are inter-frame dependencies due to motion-compensated prediction. The achievable rate distortion performance of an inter-coded frame depends on the coding decisions made during the encoding of its reference frames. Typically, in the encoding of a reference frame, these dependencies are either not considered at all or only via some rough heuristic.

In this thesis, a multi-frame transform coefficient optimization method for H.265/ HEVC is developed and studied. The inter-frame dependencies are described using a linear signal model. Based on this model, the optimization problem is cast in the form of an ℓ_1-regularized least squares problem. For solving this problem, an optimization algorithm is developed, which is applicable to H.265/HEVC without imposing excessive demands in terms of computational complexity and memory requirements. A simple functional relationship between the regularization parameter and the quantization paramter is empirically found. The accuracy of the linear signal model is studied, the bit rate savings due to the proposed method are evaluated, and its complexity is assessed. Finally, an extension of the method for spatially scalable video coding using SVC, the scalable extension of H.264/AVC, is presented.

Zusammenfassung

Bei der Codierung von Videosignalen ergeben sich aufgrund der bewegungskompensierten Prädiktion Abhängigkeiten zwischen den einzelnen Frames. Die erzielbare Rate-Distortion-Performance eines inter-codierten Frames hängt dadurch von den Codierentscheidungen ab, die bei der Codierung seiner Referenzbilder getroffen wurden. Typischerweise werden diese Abhängigkeiten bei der Codierung eines Referenzbildes entweder gar nicht beachtet oder aber nur mittels einer groben Heuristik.

In dieser Arbeit wird ein Verfahren zur Optimierung der Transformationskoeffizienten unter Berücksichtigung der Abhängigkeiten zwischen den einzelnen Frames für H.265/HEVC entwickelt und untersucht. Die Abhängigkeiten werden durch ein lineares Signalmodell beschrieben. Mit Hilfe dieses Modells wird das Optimierungsproblem in der Form eines ℓ_1-regularisierten Least-Squares-Problem formuliert. Zum Lösen dieses Problems wird ein Optimierungsalgorithmus entwickelt, der sich ohne übermäßige Anforderungen hinsichtlich Komplexität und Speicherbedarf auf H.265/ HEVC anwenden läßt. Ein einfacher funktionaler Zusammenhang zwischen dem Regularisierungsparameter und dem Quantisierungsparameter wird empirisch hergeleitet. Die Genauigkeit des linearen Signalmodells wird untersucht, die sich ergebenden Bitraten-Einsparungen werden ausgewertet, und die Komplexität des Verfahrens wird bewertet. Ferner wird eine Erweiterung des Verfahrens für örtlich skalierbare Videocodierung mit SVC, der skalierbaren Erweiterung von H.264/AVC, vorgestellt.

Acknowledgments

First of all, I wish to express my sincere thanks to my supervisor Professor Thomas Wiegand for giving me the opportunity to start my career at the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute (HHI). The fruitful amalgamation of practice-oriented contribution to H.265/HEVC standardization and profound scientific research which was shaped by Prof. Wiegand has made this thesis possible. I am also deeply grateful to Professor Jens-Rainer Ohm for his interest in my work and for co-reviewing my thesis. I would like to thank Professor Thomas Sikora for engaging my interest in the field of this work, starting with the undergraduate course "Signals and Systems." I owe special thanks to Dr. Detlev Marpe and Dr. Heiko Schwarz for their continuous support and motivation since my initial days at Fraunhofer HHI and for the numerous inspiring discussions. I am very thankful to all my colleagues at HHI who made working there so enjoyable, in particular Heiner Kirchhoffer and Haricharan Lakshman for their interest in my work and their discerning remarks which often brought new ideas to my mind.

Finally, I would like to thank my family, my parents Inge and Manfred, for being there for me and for their encouragement and support during all stages of my education. Last but not least, I would like to thank someone very special who gave me the strength to pursue this thesis — not mentioned by name does not mean forgotten!

Contents

List of Figures

List of Tables

1 Introduction

At the time of the writing of this thesis, a revolution is taking place in the area of video technology. According to a recent study by Cisco [Cis14], video services will amount to 79 % of all consumer Internet traffic in 2018, up from 66 % in 2013 (peer-to-peer sharing of video files not even counted in), and the amount of video on demand (VoD) traffic will double by 2018, corresponding to the equivalent of 6 billion DVDs per month. So-called 4K or Ultra High Definition (UHD) television sets, offering four times the resolution of High Definition Television (HDTV), are becoming available for end-consumers. Video streaming services like YouTube, Netflix etc. are competing with traditional linear TV broadcasting. Key enabler for all of these developments is availability of efficient methods for video compression. In particular, the international video coding standard H.264/AVC has been a major driver of HDTV deployment as well as video streaming to mobile devices. Meanwhile, in early 2013, the first version of its successor H.265/HEVC has been finalized and formally ratified as an international standard. Recent studies have shown that H.265/HEVC is able to provide the same subjective quality of the video signal at half the bit rate as H.264/AVC on average, with even higher bit rate savings of approximately 64 % for sequences at UHD resolution [OSS$^+$12, TMBR14]. The video coding standards, however, only specify the bitstream format and the decoding process, leaving a lot of freedom to the designer of a video encoder. This thesis is concerned with rate distortion optimization of the video encoder under consideration of inter-frame dependencies, within the given constraints of the video coding standard. Results are shown for H.265/HEVC as well as H.264/AVC-based Scalable Video Coding (SVC).

1.1 Problem statement

In video coding, there are inter-frame dependencies due to motion-compensated prediction. The achievable rate distortion performance of an inter-coded frame depends on the coding decisions made during the encoding of its reference frames. Typically, in the encoding of the reference frames, the impact on their referring frames is either not considered at all or only via some rough heuristic (e.g., by using some fixed QP cascading rule in hierarchical prediction structures). It is the aim of this thesis to improve the overall coding performance by applying modern numerical optimization methods in order to exploit part of these dependencies.

1.2 Main contributions

The subject of this thesis goes back to initial work done by Schumitsch [SSW04, SSW05]. Schumitsch's objective is to optimize the transform coefficient level[1] selection by considering inter-picture dependencies. For this purpose, he proposes to use matrix formulation in order to obtain an approximation of the video reconstruction process (i.e., inverse transform and motion-compensated prediction) based on the transform coefficients. A key part of his work is to assume that, firstly, the reconstructed samples of a transform block can be obtained as a linear combination of the corresponding transform coefficient levels, and secondly, the reconstructed samples of an inter-coded block can be represented as a linear combination of previously decoded samples. He then uses a Quadratic Program[2] formulation in order to solve for the optimal transform coefficient levels in consideration of the inter-frame dependencies within a set of video frames. In [SSW05], he presents results for two sequences at QCIF resolution (176×144 luma samples), encoded using H.264/AVC Main Profile (4×4 transform block size). As will be explained in more detail later, direct application of his method to high

[1]Within this thesis, the terms *transform coefficient* and *transform coefficient level* are used as defined in H.264/AVC and H.265/HEVC, i.e. the transform coefficient level refers to the value that is actually transmitted in the bitstream and whose interpretation depends on the chosen quantization step size, whereas the transform coefficient refers to the intermediate value in the reconstruction process after inverse scaling of the corresponding transform coefficient level. Since, given the quantization step size, the one can be derived from the other, a differentiation is made only when necessary.

[2]A Quadratic Program is a numerical optimization problem with quadratic objective function and linear constraints.

resolution video sequences, to complicated prediction structures, and to video coding schemes using large transform block sizes is not feasible due to practical constraints. For example, when only a transform block size of 4×4 is used as in H.264/AVC Main Profile, each residual sample depends only on 16 transform coefficients. When allowing 32×32 transform blocks, however, as in the H.265/HEVC video coding standard, one residual sample may depend on 1024 transform coefficients, which significantly increases both memory and computational requirements.

The main contributions of this thesis are as follows:

- The resulting rate distortion performance of applying the optimization method to the encoding of a first-order Gauss-Markov source using differential pulse code modulation (DPCM) is studied. It is shown that, for lower to medium entropy rates, significant improvements over scalar quantization with an optimal adaptation of the rounding control parameter are achieved. Furthermore, the impact of the optimization on the power spectral density of the residual signal is studied and it is found that for lower bit rates, the energy of the residual signal decreases and its spectrum becomes more and more low-pass.

- A significantly more efficient approach to the joint optimization problem that does not rely on the Quadratic Program formulation, but instead uses a variant of the *iterative shrinkage/thresholding algorithm (ISTA)* [DDDM04] is presented. A comparison with different state of the art solution algorithms to this mathematical problem class is given and the decision for using ISTA in the context of multi-frame optimization is justified. This method allows usage of larger sized transform blocks with negligible impact on memory requirements and moderate impact on computational complexity.

- In the optimization problem, the trade-off between distortion and approximated bit rate is controlled by a regularization parameter. A rule for selecting this regularization parameter based on the Quantization Parameter (QP) is empirically derived. The impact of using this fixed rule compared to determining the optimal parameter in rate distortion sense for each sequence and QP individually is shown.

- In state of the art video coding standards, like H.264/AVC or H.265/HEVC, there are highly efficient coding modes for signaling all-zero transform blocks.

This aspect is not captured in the optimization problem where the impact of each transform coefficient to the overall bit rate is treated individually. Therefore, a method that determines for each block the impact of the all-zero coding mode on the overall distortion of the subsequent frames and considers the result in the rate distortion optimization of the encoding process is proposed.

- A spatial sliding window process is proposed, that allows application of Schumitsch's method to higher resolution video sequences and/or prediction structures requiring joint consideration of a larger number of frames. By this method, the original optimization problem, which is too big in order to be solved directly, is split into a series of smaller sized sub-problems which are solved successively.

- The resulting rate distortion behaviour for a set of different non-convex regularization functions is studied under the ISTA framework and experimental results are presented, providing a comparison with the well-known ℓ_1-norm regularizer.

- An extension for inter-layer dependencies in spatial scalable video coding using SVC is developed. In SVC, the base layer residual and texture signals may be used in order to facilitate encoding of the enhancement layer. It is shown, that by considering these inter-dependencies during encoding of the base layer, significant coding gains for the enhancement layer can be achieved with no impact on the coding performance of the base layer.

2 State of the art

In this chapter, the state of the art of video coding standards and prior approaches to rate distortion optimization in video coding are reviewed. First, an overview over the hybrid video coding paradigma, which is the basis for all relevant video coding standards, is given. Then, the unique features of the new H.265/HEVC video coding standard are described. Further, the common Lagrangian approach to the rate distortion optimization problem in the operational control of a video encoder is elucidated. Finally, prior approaches to the bit allocation problem, in particular such approaches which take the inter-frame dependencies into account, are discussed.

2.1 Hybrid video coding

All relevant video coding standards (i.e., H.261, MPEG-1, MPEG-2/H.262, H.263, H.264/AVC, H.265/HEVC) are based on the so-called hybrid approach. The term “hybrid” stems from the fact that along different dimensions of the input video signal different decorrelating coding techniques are applied [Eri85]. In particular, along the temporal dimension, a motion-compensated prediction is used, in order to exploit temporal dependencies. The spatial dependencies within one video frame, which remain after the temporal prediction, are reduced by applying a separable block transform to the prediction residual. Typically, the discrete cosine transform (DCT) [ANR74] or an approximation thereof is used for this purpose.

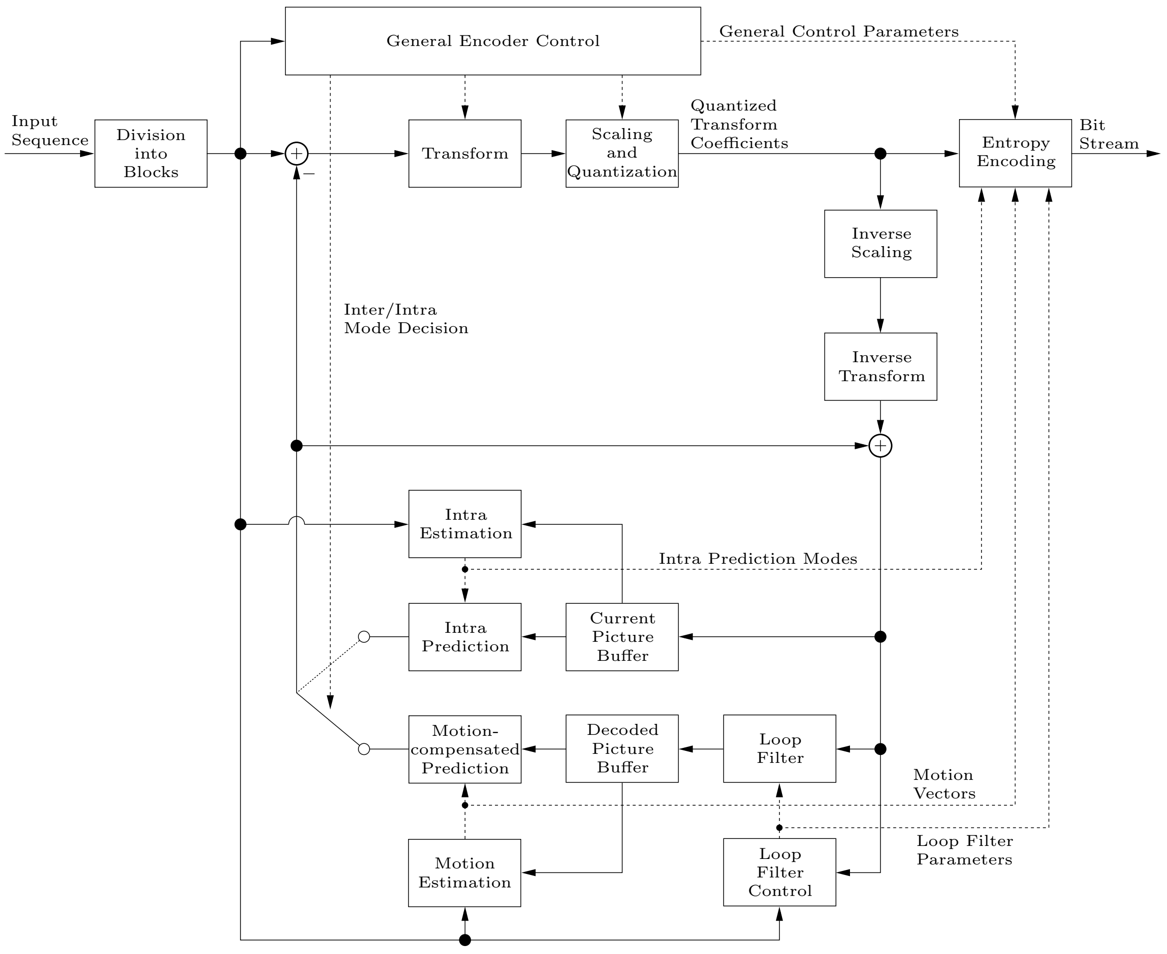


Figure 2.1: Basic architecture of a hybrid video encoder.

2.1.1 General structure

The basic architecture of a hybrid video encoder is depicted in Fig. 2.1. The solid lines represent the signal flow of the signal samples, whereas the so-called side information or control data (e.g., prediction modes, motion vectors, block sizes) are shown using dashed lines. The input signal is first split into individual coding blocks. For each block, a prediction signal is generated, which may be obtained either by intra prediction from already encoded blocks of the current frame (or by using some fixed value for the first block) or by motion-compensated prediction using past encoded frames. This prediction signal is subtracted from the original input signal and the resulting residual signal is then transformed and quantized, leading to quantized transform coefficient levels which are fed to the entropy coding stage. The entropy encoder typically uses either variable-length coding (VLC) (e.g, using Huffman codes [Huf52]) or arithmetic coding [RL79, WNC87] (e.g, context-based adaptive binary arithmetic coding (CABAC) [MSW03]). After reconstruction (i.e., inverse scaling/transform and adding of the prediction signal), a forward-adaptive loop filter is applied in the exemplary encoder of Fig. 2.1. The loop filtered reconstructed frames are stored in the decoded picture buffer, where they are available as reference frames for motion-compensated prediction. The motion-compensated prediction signal is determined by the motion vector and the corresponding reference frame. Typically, a motion vector accuracy finer than one sample (or *pel*, for picture element) is used. The state of the art video coding standards H.264/AVC and H.265/HEVC use a motion vector accuracy of one quarter-pel for the luminance component. Therefore, a sub-pel interpolation filter has to be applied to the reconstructed frame, in order to obtain the sample values at sub-pel positions.

2.1.2 The video coding standard H.265/HEVC

H.265/HEVC is the latest video coding standard which has been jointly developed by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), the two most relevant standardization organizations in the area of video compression. The video coding standard which precedes H.265/HEVC is H.264/AVC, which is in wide use today in different application areas such as digital television, BluRay, internet video streaming, and video conferencing. Therefore, in the following the major enhancements of H.265/HEVC relative to H.264/AVC are sum-

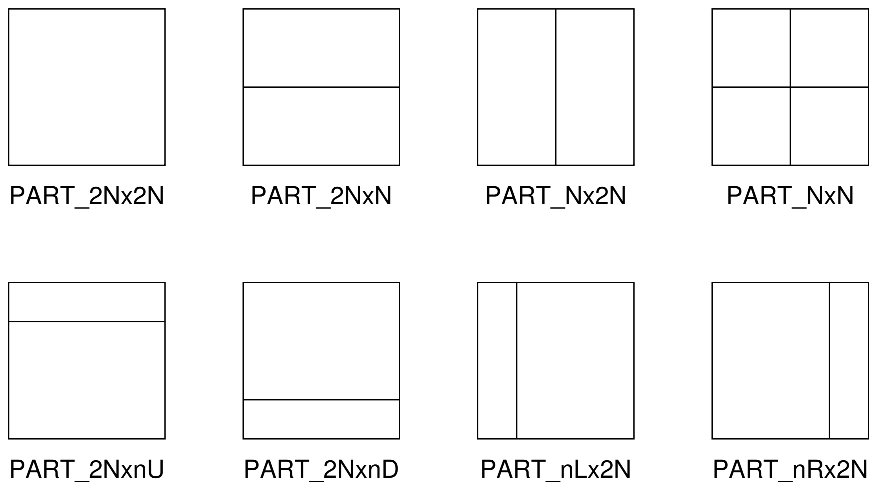


Figure 2.2: Possible partitionings of a Coding Unit (CU) into Prediction Units (PUs).

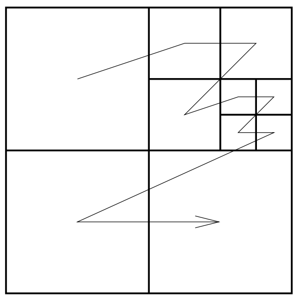


Figure 2.3: Illustration of the Z scan (depth first) traversal of a quadtree.

marized. A more detailed description of H.265/HEVC can be found in [SOHW12].

Prediction block partitioning

Whereas in H.264/AVC each frame is split into macroblocks of 16×16 luma samples, the basic structure in H.265/HEVC is the Coding Tree Unit (CTU). The block size of a CTU is variable, but fixed within a sequence. The following CTU sizes are possible: 16×16, 32×32, or 64×64 luma samples. Typically, a larger CTU size leads to better coding efficiency, especially at higher resolutions [SOHW12]. Each CTU can be further split by the use of a quadtree decomposition into Coding Units (CUs). For each CU, it is specified which type of prediction (intra or inter) is used. A CU can be further split into two or four Prediction Units (PUs). The prediction parameters (i.e., motion vectors, reference frames, intra prediction directions etc.) are transmitted for each PU. H.265/HEVC allows PU sizes ranging from 4×4 to 64×64 samples[1]. The possible partitionings are shown in Fig. 2.2. The rectangular partitionings are only supported for inter-predicted CUs. The CTUs within a frame are processed in raster scan order (row by row, left to right), whereas the CUs within a CTU are traversed in so called Z scan order as illustrated in Fig. 2.3. The Z scan order is used because thus for each CU, the top and left neighboring CUs will processed afore, and therefore the coding decision made for these will be available and can be used for context-modeling in the entropy coding stage. More information about the block partitioning in H.265/ HEVC can be found in [KML+12].

Motion-compensated prediction

Just like H.264/AVC, H.265/HEVC uses a motion vector accuracy of one quarter-pel. The interpolation filter that is used in order to obtain the sample values at sub-pel positions of the luma component is an 8-tap FIR filter. For the chroma interpolation process, a 4-tap FIR filter is used. Like H.264/AVC, H.265/HEVC allows bi-prediction, i.e. a superposition of two prediction signals, and weighted prediction, i.e. a scaling of the prediction signal. In addition to prior video coding standards, a so-called merge mode is supported, where the motion parameters (i.e., motion vectors and reference frames) can be inherited from a set of candidates, which include neighboring blocks as well as a temporally collocated merge candidate. This allows efficient representation of contiguous regions, where the motion parameters are shared across different CUs. Furthermore, a so-called CU SKIP mode is supported, where it is very efficiently signaled, that the merge mode is used without transmitting a residual signal.

Transform domain representation of the residual signal

Each CU is split into one or more Transform Units (TUs) by the usage of secondary quadtree structure, called Residual Quadtree (RQT). H.265/HEVC supports transform sizes of 4×4, 8×8, 16×16, and 32×32 samples. The basis functions are integer approximations of the corresponding DCT [ANR74] basis functions. For intra-predicted luma blocks of 4×4 samples, an integer approximation of the discrete sine

[1]4×4 PUs are only supported for intra-predicted CUs.

transform (DST) [WH85] is used instead. Furthermore, for 4×4 transform blocks, a *transform skip* mode is supported, where the transform of the residual signal from spatial to frequency domain is omitted, which is especially beneficial to screen and computer graphics content [LXSW12, PLXS12].

In the bitstream, for each TU, so-called *transform coefficient levels* are transmitted. From each transform coefficient level, the actual transform coefficient is derived by multiplication by a scaling factor, which depends on the Quantization Parameter (QP) value. A smaller QP value corresponds to a smaller scaling factor, which results in a finer accuracy of the transform coefficients. For 8 bit video sequences, there are 52 different scaling factors supported, corresponding to a QP range of 0–51. The same uniform-reconstruction quantizer (URQ) [Sul96] as in H.264/AVC is used for H.265/HEVC.

In inter-predicted CUs, a transform block may overlap several prediction blocks. E.g., if the PART_Nx2N partitioning according to Fig. 2.2 is used for a 32×32 CU, there will be two 16×32 PUs. In this case, it is possible to encode the residual signal using one 32×32 TU, which overlaps the two PUs. For intra-predicted CUs (with PART_NxN partitioning), this is not possible, since otherwise the prediction signal of the second PU in coding order, which may depend on the reconstructed signal of the first PU in coding order, would depend on the residual signal of the whole CU, causing a causality problem in the encoder, since in order to determine the residual signal, the prediction signal has to be known, which is not the case, if the prediction signal itself (partly) depends on the residual signal. The TUs within a CU are processed in Z scan order (see Fig. 2.3). The transform coding in H.265/HEVC is described in detail in [SJN+12, NHW+13].

Residual coding

Whereas H.264/AVC supports two methods of entropy coding, namely context-based adaptive variable length coding (CAVLC) [BL02] and context-based adaptive binary arithmetic coding (CABAC) [MSW03], in H.265/HEVC only the latter (CABAC) is supported. The encoding of the transform coefficient levels using CABAC follows the same three steps as in H.264/AVC:

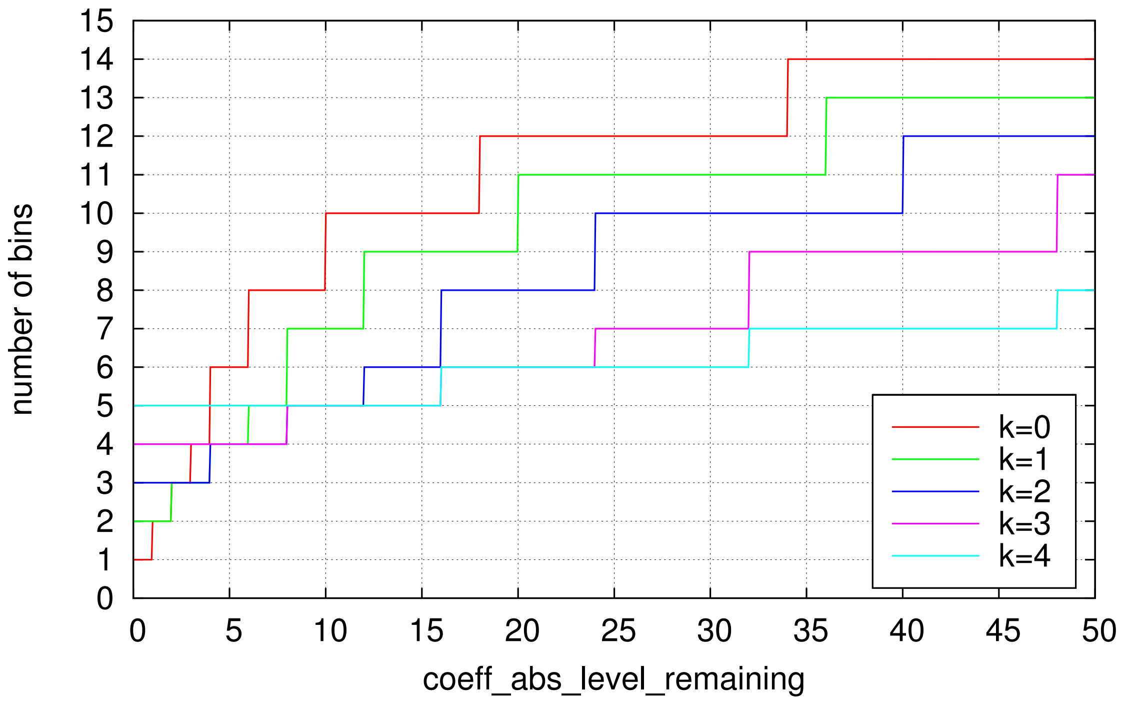


Figure 2.4: Number of bins resulting from the five different binarizations of the syntax element **coeff_abs_level_remaining**.

- binarization
- context modeling
- binary arithmetic coding (BAC)

The last step, where the binary symbols are encoded using the BAC engine also known as *M coder*, is directly taken without any modification from H.264/AVC. The binarization and context modeling are modified in order to improve both throughput and coding efficiency. Each transform block (TB) is split into so-called sub-blocks (SB), where a sub-block is a 4×4 array of 16 transform coefficient levels. First, the locations of the non-zero transform coefficients are transmitted. Using one single syntax element, a flag called **rqt_root_cbf**, it can very efficiently be signaled that all the transform coefficients of the whole CU are zero. If this is not the case, then for each TB of the CU, a coded block flag (cbf) for the luma component is transmitted (**cbf_luma**) which indicates whether all the luma transform coefficients within this TB are zero[1]. If again this is not the case, the location of the last non-zero coefficient in

[1]The coded block flags **cbf_cb** and **cbf_cr** for the chroma components are interleaved with the signaling of the RQT structure.

scanning order within a TB is transmitted. The positions of the non-zero coefficients within a SB are indicated using the so-called *significance map.* For that purpose, for all but the last coefficient in scanning order (which must be non-zero per definitionem), a significance flag (**sig_coeff_flag**) is transmitted. Furthermore, for each following SB, by the usage of the **coded_sub_block_flag**, it can again very efficiently signaled, that all its coefficients are zero.

After the locations of the non-zero coefficients have been signaled, the actual values have to be transmitted. At this point, it is already clear that the value must be at least one, because otherwise it would not be a non-zero coefficient. The absolute value minus one of the non-zero transform coefficient values is encoded using three syntax elements, namely **coeff_abs_level_greater1_flag**, **coeff_abs_level_greater2_flag**, and **coeff_abs_level_remaining**. The first two of the three are encoded using context-modeling and binary arithmetic coding, whereas the last one is encoded in so-called bypass mode, i.e. without context modeling and each binary symbol (bin) results in one additional output bit. The binarization of **coeff_abs_level_remaining**, i.e. the decomposition of the corresponding value into a sequence of binary symbols, is based on Golomb-Rice codes [Gol66] and Exp-Golomb codes [Teu78]. Five different binarizations are supported, where the actual binarization is chosen backward-adaptively. The resulting numbers of bins are shown in Fig. 2.4. Generally, it can be seen, that the number of bins grows slowly (i.e., logarithmically) with the value of **coeff_abs_level_remaining**.

For the encoding of the sign of the non-zero transform coefficients, H.265/HEVC supports a new coding tool called *sign data hiding (SDH).* With SDH enabled, for each sub-block which fulfills the condition that the distance between the first and last non-zero coefficient in scan order is larger than 3, the sign information of the last coefficient in coding order is not explicitly signaled, but instead it is derived from the parity of the sum of the transform coefficient levels. Obviously, from an encoder perspective, this imposes a constraint such that only those combinations of transform coefficient levels can be transmitted which fulfill the parity condition. Therefore, if the "real" transform coefficient levels result in a violation of the parity condition, an adaptation of one level by $+1$ or -1 has to be made. In the H.265/HEVC reference encoder HM, this adaptation can be made such that the impact on either the rate distortion cost[1] or solely on the induced distortion is minimized, where the former is used in conjunction with so-called *rate distortion optimized quantization (RDOQ)*

[1]a weighted sum $D + \lambda R$ of the resulting distortion D and the required bit rate R, see Sec. 2.2

[KYC08, KCYJ09], and the latter in conjunction with ordinary scalar quantization.

More details about the residual coding in H.265/HEVC can be found in [SJN+12, NHW+13].

In-loop filtering

In addition to the deblocking filter, which is similar to the one of H.264/AVC, in H.265/HEVC a secondary in-loop filter, which is called sample adaptive offset (SAO), is supported. The SAO targets at a better reconstruction of the original sample amplitude. Each sample is classified into one out of several predefined categories. A look-up table is transmitted in the bitstream which contains an offset value for each category. Consequently, the SAO is a non-linear forward-adaptive filter. A detailed description of the SAO can be found in [FAA+12].

2.2 Rate Distortion Optimization

By a video coding standard, only the bitstream format and the decoding process are prescribed, the question how to do the actual video compression, i.e. how to obtain the coded bitstream from the sample values of the original input video sequence, is intentionally left open. This leaves much freedom to the designer of a video encoder. The performance of an actual video encoder can be judged based on a variety of different criteria, e.g.

- latency,
- computational complexity (i.e., in the simplest form, run time),
- memory requirements, or
- reconstruction quality of the decoded video signal at a given bit rate.

Typically, in order to improve the performance for one of the criteria, one has to accept losses for the others. In this thesis, the focus is on the last criterion, i.e. improving the rate distortion performance, whithout imposing excessive demands in terms of the other criteria, such that practical implementability is maintained. Generally, the rate distortion optimization problem can be stated as minimizing the distortion D under a given constraint on the bit rate R, i.e.

$$\min D \quad \text{subject to } R \leq R_{max}. \tag{2.1}$$

As has been shown in [Eve63], problems of this kind can be recast into an unconstrained optimization problem by the usage of a discrete version of the Lagrangian multiplier method. This results in the following formulation

$$\min J \quad \text{with } J = D + \lambda\, R, \tag{2.2}$$

where the solution of Eq. 2.2 for a given value of the Lagrangian multiplier λ is identical to a solution of Eq. 2.1 with a corresponding bit rate constraint R_{max}. Both D and R depend on all the choices made during the encoding of the video sequence, i.e. prediction modes, block sizes, motion vectors, quantization step size etc. If the Lagrangian multiplier λ is fixed, the resulting rate distortion cost J can be computed for a set of coding options and the one resulting in the lowest J is chosen. In the reference encoder implementations for H.264/AVC and H.265/HEVC, this optimization is done block-by-block. In [WG01], for the video coding standard H.263+, it is proposed to select λ depending on the quantization step size according to

$$\lambda = 0.85 \cdot Q^2, \tag{2.3}$$

where Q is the QUANT parameter of H.263+, which is half the distance of two neighboring non-zero quantizer reconstruction values[1]. In [WSJ$^+$03], the following rule has been empirically derived for H.264/AVC

$$\lambda = 0.85 \cdot 2^{(QP-12)/3}, \tag{2.4}$$

[1]Non-zero is important here, because in H.263 (as in its predecessors H.261 and H.262) the distance between the zero reconstruction value and the first non-zero reconstruction value is 1.5 times the distance between the non-zero reconstruction values, which is referred to as *central dead-zone*.

where QP is the quantization parameter. Since the quantization step size Δ doubles, if QP is incremented by six, the following relations hold:

$$\Delta \propto 2^{QP/6} \tag{2.5}$$

$$\Delta^2 \propto 2^{QP/3} \tag{2.6}$$

$$\lambda \propto \Delta^2 \tag{2.7}$$

This shows that in both Eq. 2.3 and Eq. 2.4 the Lagrangian multiplier is proportional to the square of the quantization step size. A similar rule for chosing λ is also used in the H.265/HEVC reference encoder software.

2.3 Previous approaches

Subject of this thesis is rate distortion optimization of transform coefficients under consideration of inter-frame dependencies. This can broadly be viewed as a specific case of a bit allocation problem: Given a total budget R_{budget}, how to distribute the available bit rate among the individual transform coefficients without exceeding R_{budget}? In this section, previous work on the bit allocation problem is reviewed and their differences to the work within this thesis are elaborated.

Huang and Schultheiss were the first to address the bit allocation problem within the context of source coding in their seminal work [HS63]. They considered block coding of Gaussian variables using a decorrelating linear transform, whose output is fed to a set of Lloyd-Max [Max60, Llo82] scalar quantizers. For fixed-length coding of the quantizer outputs, they give a formular which approximates the number of bits to be assigned to each scalar quantizer in order to minimize the distortion under a constraint on the total number of bits available. In their solution, fractional or even negative bit assignments may occur. Furthermore, they only considered coding of Gaussian sources. In [Seg76], a solution to the bit allocation problem under consideration of the non-negativity constraint and application of entropy coding to the quantizer outputs is presented.

In [SG88], Shoham and Gersho propose an algorithm for bit allocation to an arbitrary set of quantizers which relies on a discrete version of the Lagrangian multiplier method [Eve63]. The algorithm in this paper also does not rely on model assumptions about the rate distortion performance of a quantizer, but instead can be applied on-line to the actual costs of any coding scheme (similar to the rate distortion optimization (RDO) in typical current video encoders). It does not, however, cover the case of inter-dependencies between the individual quantizers.

Operational dependent bit allocation

The bit allocation problem in a scenario where there are inter-dependencies in the sense, that the input to one quantizer depends on the output of another quantizer, is first treated in the work by Ramchandran et al. [ROV93, ROV94], which also does not rely on model-based rate distortion functions. In this work, the inter-dependencies are modeled using a trellis, and the optimal solution to the bit allocation problem is then found by searching for the minimal-cost path through the trellis. Note that only in very simple constellations the Viterbi algorithm (VA) [For73] can be used for this purpose, because the VA relies on the Markov property. In the context of multi-frame optimization in video coding, the Markov property would mean that the rate distortion cost of the next frame (in coding order) only depends on the coding decisions made for the current frame. This is true for a I-B-I coding scenario, since the independent I frames decouple the B frame from one another [ROV93, Sec. 2.1]. But in a more general setup, the coding decisions made for all previous frames also impact the next frame. In order to ease the burden of having to explore the whole exponentially growing dependency tree, a monotonicity property is assumed in [ROV93, ROV94]. This monotonicity property basically means that a better predictor will lead to more efficient coding of the residue. Furthermore, a suboptimal heuristic is proposed, such that in each stage of the trellis, except for the first one which corresponds to the initial I frame, only the lowest cost branch is retained. Still, applicability of the joint-optimization algorithm is limited to setups with manageable search space (e.g., deciding between three different frame-wise quantizers for a group of five frames, as shown in [ROV93, ROV94]), because the search space grows exponentially with the number of frames under consideration.

Model-based dependent bit allocation

In [USC93], the theoretical optimal bit allocation in the presence of quantizer feedback under the model assumption of an exponential distortion-rate function is derived, i.e. for the first frame, which does not depend on other frames, the relation between bit rate R_1 and quantization error variance E_1 is modeled as

$$E_1 = e^{-\alpha R_1} X_1, \tag{2.8}$$

where α and X_1 are free parameters which have to be estimated for each specific encoder and sequence by encoding at different bit rates and regression of the empirical rate distortion curve. Under the assumption that frame m is temporally predicted from frame $m-1$, the inter-frame dependencies are modeled as follows in [USC93]

$$E_m = e^{-\alpha R_m}(X_m + \rho_m E_{m-1}), \tag{2.9}$$

where $0 \leq \rho_m \leq 1$ is the coefficient of quantizer feedback. Even though the actual bit allocation, which is derived from these model assumptions, is rather academical, the authors of [USC93] conclude that, by considering inter-frame dependencies in the rate distortion optimization, "in the case of quantizer feedback, frames that are either easily predicted, or good predictors are encoded to higher quality since propagating quantization errors contribute to the total error in predicted frames."

A very similar rate distortion model for the inter-frame dependencies is assumed in [CLK97], where the distortion D_i of the frame i is obtained from its bit rate R_i and the distortion of its reference frame D_{i-1} as well as the coding efficiency parameter β_i and the frame dependency parameter α_i as:

$$D_i = 2^{-\beta_i R_i}(\sigma_i^2 + \alpha_i D_{i-1}) \tag{2.10}$$

The main difference between Eq. 2.9 and Eq. 2.10 is, that in Eq. 2.10 the exponential decay parameter β_i is frame-specific, whereas α in Eq. 2.9 is fixed for the whole sequence. Based on the model of Eq. 2.10, the optimal bit-allocation among the individual frames for a wavelet video coder is derived in [CLK97].

Inter-frame optimization by quantization step size variation

Note that all previously described approaches on inter-frame bit allocation only consider the problem of how to temporally distribute the available bit-budget *among* the individual frames of the sequence, but not how to distribute it spatially *within* the frames. This is addressed in [KK98], which aims at determining the optimal quantization step size for each macroblock. It is assessed there that "There are too many possible branches in the trellis construction to apply dynamic programming[1] because of the inter-dependency among macroblocks caused by the motion compensation." Therefore, a two step approach is proposed, where in the first step, using the inter-frame dependency model of [CLK97], the optimal bit rate distribution among the individual frames is determined. Then, in a second step, given a fixed bit rate budget for each frame, the quantization step sizes for each macroblock are determined by the method of [ROV94] (i.e., constructing the trellis for all possible quantization step sizes and determining the path with the lowest Lagrangian rate distortion cost). Note, however, that the second step is a local optimization where the inter-frame dependencies and consequently the impact on referring frames is not considered. Consequently, regions of a reference frame which are referred more often by motion-compensated prediction (e.g., uncovered background) will not be favored in terms of bit allocation over regions which are referred less often (e.g., background which is about to be covered), even though it might be advisable with respect to the overall rate distortion performance.

In [RCL00], a heuristic for the bit allocation considering inter-frame dependencies is proposed, where the bit rate spent for the encoding of reference frames is increased at the cost of the bit rate of the non-reference frames. This is a backward-adaptive method which does not require pre-analysis of the actual inter-frame dependencies.

In [BWO02], an inter-frame optimization method is proposed, where those blocks of a reference frame which are simply copied (i.e., used as reference for motion-compensated prediction without coded residual signal) in subsequent referring frames are encoded using a finer quantization step size. The reduced quantization step size is the smaller the more subsequent frames are copying from a particular block of a reference frame.

In [RO06], based on a linear signal model of the decoding process, an algorithm is developed which determines the optimal quantization step sizes for each macroblock in SNR scalable video coding using hierarchical B frames, such that the error accumulation within the B frame hierarchy is taken into account.

[1]e.g. [SG88] or [ROV94]

Transform coefficient thresholding and soft decision quantization

All previously described methods rely on a modulation of the quantization step size. The question of which values to actually encode for a certain block of transform coefficients, given a fixed quantization step size, is also an important aspect in terms of the resulting rate distortion performance. In [RV94], a rate distortion optimal thresholding algorithm is described. For each transform coefficient, it is decided on a rate distortion criterion, whether to keep or to drop (i.e., set to zero) this particular coefficient. This is again a local optimization approach, which does not consider inter-frame dependencies.

In [WLV00], this idea is extended such that several possible values for each transform coefficient are checked. The quantization of a transform block is modeled using a trellis, where each stage of the trellis corresponds to one transform coefficient. The states within each stage represent the possible values to which this coefficient can be quantized. Each path through the trellis therefore corresponds to one particular vector of quantized transform coefficients. The idea is to search the path which has the lowest rate distortion cost. The design of the trellis is specifically matched to the entropy coding of the H.263+ video coding standard.

In [YY07], this concept is adapted to the context-based adaptive variable length (CAVLC) entropy coding method of the H.264/AVC standard. Furthermore, the terms hard decision quantization (HDQ) and soft decision quantization (SDQ) are coined. HDQ refers to the conventional method of quantization, where the quantized transform coefficient level c is obtained from the unquantized transform coefficient x as

$$c = \operatorname{sgn}(x) \left\lfloor \frac{|x|}{\Delta} + f \right\rfloor , \tag{2.11}$$

where Δ is the quantization step size, f is a rounding control parameter which is typically chosen to be $1/3$ for I slices and $1/6$ for P slices, and $\lfloor x \rfloor$ denotes rounding to the nearest integer that is less than or equal to x. In SDQ, the transform coefficients $\mathbf{c}_{SDQ}$ are chosen such that the resulting rate distortion cost is minimized:

$$\mathbf{c}_{SDQ} = \arg\min_{\mathbf{c}} D(\mathbf{c}) + \lambda \cdot R(\mathbf{c}) \tag{2.12}$$

Here, $D(\mathbf{c})$ and $R(\mathbf{c})$ represent the distortion and bit rate when using the transform coefficient vector $\mathbf{c}$, and λ is the Lagrangian multiplier which controls the trade-off between reconstruction quality and required bit rate. Note that $\mathbf{c}_{SDQ}$ represents the vector of all the transform coefficients for a given transform block, since the resulting bit rate cannot be derived for a single transform coefficient on its own, but instead the other coefficients have to be considered as well. This is caused by the entropy coding which takes dependencies among coefficients of the same block into account. The SDQ is done block-by-block, neglecting any inter-block and inter-frame dependencies, whereas the inter-coefficient dependencies within one block are captured by a trellis (or, more broadly, graph). The rate distortion optimal coefficient vector $\mathbf{c}_{SDQ}$ is, again, found as the shortest (i.e. lowest rate distortion cost) path through the graph. In [YY09], the concept is further developed in order to adapt it to the context-based adaptive binary arithmetic coding (CABAC) [MSW03] entropy coding method.

In [SW07], a trellis-based algorithm for determining rate distortion optimal transform coefficients for scalable video coding (SVC) is presented, which is applicable to both spatial and fidelity scalability. In scalable video coding, the so-called *base layer* can be decoded independently, whereas the decoding of the *enhancement layer* may depend on coding decisions which are transmitted for the base layer. Using this method, these inter-layer dependencies are taken into account, where the trade-off between base and enhancement layer coding efficiency can be controlled by a weighting factor.

Rate Distortion Optimized Quantization (RDOQ)

In [KYC08, KCYJ09], a simplification of SDQ is presented, which aims at reducing the computational complexity. Instead of checking all (or very large number of) possible values for each transform coefficient level, at most three candidates are tested. With the following definitions

$$c_{float} = \frac{|x|}{\Delta}, \tag{2.13}$$

$$c_{floor} = \lfloor c_{float} \rfloor, \tag{2.14}$$

$$c_{ceil} = \lceil c_{float} \rceil = c_{floor} + 1, \tag{2.15}$$

only 0, c_{floor}, and c_{ceil} are possible outcomes (neglecting the sign), given the unquantized transform coefficient is equal to x. If c_{float} is closer to c_{floor} than to c_{ceil}, only 0 and c_{floor} are considered. Consequently, if $c_{float} \leq 0.5$, the coefficient is set to zero without any further testing. By limiting the number of candidates, the computational complexity is largely reduced relative to [YY07, YY09]. Under the name rate distortion optimized quantization (RDOQ), this method has become part of both the H.264/AVC and the H.265/HEVC reference encoder, and therefore can be considered as state of the art.

An acceleration of RDOQ has recently been proposed in [HSK$^+$11, HKC13]. In the computation of the rate distortion cost for each transform coefficient, the actual bit rate is replaced by a rate model. In [HSK$^+$11], a linear rate model is used, i.e. the bit rate is assumed to be proportional to the ℓ_1-norm of the transform coefficient vector (i.e., for a single coefficient, proportional to its absolute value). Then, the quantized transform coefficient level c is obtained from the unquantized transform coefficient x as follows:

$$c = \begin{cases} 0 & \text{if } \frac{|x|}{\Delta} \leq T \\ \operatorname{sgn}(x) \left\lfloor \frac{|x|}{\Delta} - T + \frac{1}{2} \right\rfloor & \text{otherwise} \end{cases} \tag{2.16}$$

Here, Δ is again the quantization step size and T is a threshold value which arises from the linear bit rate model. It can be seen that each coefficient, whose quantized level would be (in absolute value) smaller than T, is clipped to zero, whereas the remaining coefficients are shrinked by an amount of T towards zero. This operation has also become known under the name *soft thresholding* (as introduced in [DJ94]) and will be considered in more detail in Sec. 3.4.1. In [HKC13], the rate model is extended to

$$R(c) = \alpha|c| + \beta\|c\|_0, \tag{2.17}$$

i.e. a linear combination of the absolute value (ℓ_1-norm) and the ℓ_0-pseudo-norm, which is zero for $c = 0$ and one otherwise.

Low-pass prefiltering for low bit rate DPCM encoding

In [GO01], DPCM encoding of Gaussian autoregressive (AR) sequences at low bit rates is studied. It is shown that using DPCM, the process innovation of the AR sequence plus a feedback quantization error term is encoded. Furthermore, it is shown that, due to the transfer function of the DPCM decoder[1], distortion of the transmitted quantized residual signal at lower frequencies contributes stronger to the overall reconstruction

[1]If the predicted and the reconstructed sample values at time n are denoted as $\hat{x}_n$ and $\tilde{x}_n$, respectively, and a first order predictor with predictor coefficient a is assumed, it holds that $\hat{x}_n = a\tilde{x}_{n-1}$. Since the reconstructed sample $\tilde{x}_n$ is obtained as the sum of the prediction signal $\hat{x}_n$ and the transmitted residual signal u_n, is follows that $\tilde{x}_n = \hat{x}_n + u_n = a\tilde{x}_{n-1} + u_n$. Accordingly, the DPCM decoder is an infinite impulse response (IIR) filter with transfer function $H(z) = 1/(1 - az^{-1})$, whose gain is larger for lower frequencies if $a > 0$.

distortion. Consequently, in order to improve the rate distortion performance at lower bit rates, a low-pass prefiltering of the innovation signal at the encoder side is proposed.

In [KR07], this idea is further extended by inclusion of a downsampling stage after the low-pass filtering, which results in a reduced number of samples being transmitted. Using this set-up, the rate distortion performance can be even further improved. Other than the previously discussed method described in [GO01], this inevitably comes along with modifications at the decoder side, because the downsampling has again to be inversed by upsampling and low-pass filtering.

Linear model based multi-frame optimization

In [SSW04, SSW05], Schumitsch proposes a method for optimization of transform coefficients under consideration of inter-frame dependencies. He also uses a linear rate model based on the ℓ_1-norm of the vector of transform coefficients. Then, an optimization method based on quadratic programming is employed in order to obtain the optimized transform coefficients. Due to memory and computational requirements, this approach is problematic for either high resolution video sequences or for complicated prediction structures, where a large number of frames has to be considered jointly, e.g. hierarchical B frames. In this thesis, based on Schumitsch's approach, a method is proposed that is also applicable in these cases.

3 Multi-frame transform coefficient optimization

In this chapter, the multi-frame transform coefficient optimization problem is formally stated in the form of a regularized least squares problem. For that purpose, first the linear system model of multi-frame video decoding as proposed by Schumitsch in [SSW04, SSW05] is introduced. Then, his optimization method which is based on a Quadratic Program formulation is revisited. After that, several alternative approaches to this problem class are discussed. Finally, the decision to pursue an approach based on the *Iterative Shrinkage/Thresholding Algorithm (ISTA)* is justified.

3.1 Linear system modeling of multi-frame video decoding

A linear signal model of the reconstruction process (including scaling/inverse transform and motion-compensated prediction) for a series of consecutive frames is proposed by Schumitsch in [SSW04, SSW05]. Since this work uses Schumitsch's model, in the following sections the individual parts of the model are described in detail. Note that although to simplify matters only the luma samples are considered, the concepts could easily be extended to also include the chroma samples.

Under the assumption of a group of $N > 1$ frames under consideration, each having a width of W and a height of H luma samples, there are $K = N \cdot W \cdot H$ luma samples total. The following $K \times 1$ column vectors are introduced:

- $\mathbf{s}$, the vector of reconstructed samples,

- $\hat{\mathbf{s}}$, the vector of prediction signal samples, and
- $\mathbf{r}$, the vector of residual signal samples.

Note that the samples from the individual frames are stacked into column vectors according to some mapping rule, e.g. Z-scan or raster scan order. Even though the used mapping, as long as one-to-one and onto, is in principle arbitrary, in the following frame-by-frame Z-scan mapping in coding order is assumed, because this particular mapping has the following nice properties:

- All samples belonging to the same transform block receive consecutive indices.
- A sample with index i can, due to prediction, only depend on other samples with an index smaller than i.

Using the above notation, and neglecting any filtering operation on the reconstructed signal samples, the reconstructed signal can be written as follows:

$$\mathbf{s} = \hat{\mathbf{s}} + \mathbf{r} \tag{3.1}$$

3.1.1 Matrix notation of inverse transform

The residual signal is obtained from the transform coefficient levels by a scaling operation, which depends on the quantization step size, followed by inverse transform. In matrix notation, this can be written by usage of a $K \times K$ scaling/inverse transform matrix $\mathbf{T} = [\mathbf{t}_1 \; \mathbf{t}_2 \; \ldots \; \mathbf{t}_K]$ and the $K \times 1$ column vector $\mathbf{c}$ of the transform coefficient levels as follows:

$$\mathbf{r} = \mathbf{T}\,\mathbf{c} \tag{3.2}$$

Each column $\mathbf{t}_i$ of $\mathbf{T}$ contains the scaled basis function (basis image) corresponding to transform coefficient c_i. Note that if the used transform is non-overlapping, as in all current video coding standards, $\mathbf{T}$ has block-diagonal form, since the influence of each transform coefficient is limited to within its block. Also note that if the maximum transform block size is limited to $k = l_{trafo,max} \cdot l_{trafo,max}$ luma samples, then by definition each column of $\mathbf{T}$ can have at most k non-zero entries, since each transform coefficient has only impact on at most k residual samples. Furthermore, each row of $\mathbf{T}$ can also have at most k non-zero entries, since each residual sample is obtained as a linear combination of the corresponding samples of at most k basis images. Consequently, the fraction of the non-zero entries of $\mathbf{T}$ is limited to be not greater than $\frac{k}{K}$. Since typically $k \ll K$, $\mathbf{T}$ is a sparse matrix. But still, since the number of non-zero entries of $\mathbf{T}$ can be up to $k \cdot K$, and as for the current state of the art video coding standard H.265/HEVC $k = 32 \cdot 32 = 1024$, the memory requirements for storing $\mathbf{T}$, even when making use of the sparsity, can be significant.

3.1.2 Matrix notation of motion-compensated prediction

The prediction signal can be split into two parts,

- the variable part that depends on reconstructed samples of the N frames under consideration, and
- the fixed part which either depends on reconstructed samples of other ("previous" in coding order) frames, that are outside the set of N frames under consideration, or which is generated by some static prediction method (e.g., DC intra prediction mode).

If the fixed part is denoted as $\mathbf{p}$, the prediction signal can be written in matrix notation by usage of a $K \times K$ prediction matrix $\mathbf{M}$ as follows:

$$\hat{\mathbf{s}} = \mathbf{p} + \mathbf{M}\,\mathbf{s} \tag{3.3}$$

Unless stated otherwise, it is assumed in the following that the matrix $\mathbf{M}$ represents only the motion-compensated prediction signal, whereas the intra prediction signal is included in the fixed prediction signal $\mathbf{p}$. Each entry $m_{i,j}$ of the matrix $\mathbf{M}$ gives the value by how much the reconstructed signal sample s_j contributes to the prediction signal sample $\hat{s}_i$. Note that the matrix $\mathbf{M}$ is strictly lower triangular, since each prediction sample can only depend on previous reconstructed samples, and therefore $\forall_i \forall_j (i \leq j \Rightarrow m_{i,j} = 0)$.

In order to illustrate the prediction matrix $\mathbf{M}$, assume for a moment that only motion-compensated prediction with full-pel motion vector accuracy and one single hypothesis is used (i.e., no intra prediction, no sub-pel interpolation, no biprediction, no weighted prediction). In this case, there would be only two possible values for each $m_{i,j}$, namely 0 and 1. Furthermore, for each motion-compensated prediction sample $\hat{s}_i$, there would be exactly one non-zero entry $m_{i,j}$ in row i of the matrix $\mathbf{M}$, corresponding to its reference sample s_j, such that $\hat{s}_i = s_j$. In each column j of the matrix $\mathbf{M}$, however, there could be more than one non-zero entry, since several motion-compensated prediction blocks could possibly refer to the same area of the same reference frame, such that one particular reconstructed sample gets referenced by more than one prediction sample.

If single hypothesis motion-compensated prediction with sub-pel accuracy using a separable h tap FIR interpolation filter is used, there could be up to $h \cdot h$ non-zero entries in each row. In case of biprediction, the maximum number of non-zero entries per row is correspondingly $2 \cdot h \cdot h$. Thus, the fraction of the non-zero entries of $\mathbf{M}$ is limited to be not greater than $\frac{2 \cdot h \cdot h}{K}$. Since typically $2 \cdot h \cdot h \ll K$, $\mathbf{M}$ like $\mathbf{T}$ is a sparse matrix. As in H.265/HEVC $h = 8$, the total number of non-zero entries of $\mathbf{M}$ cannot be greater than $128 \cdot K$, which is still big, but almost an order of magnitude smaller than the maximum number of non-zero entries of $\mathbf{T}$.

3.1.3 Matrix notation of the whole reconstruction process

By using equations 3.2 and 3.3, the reconstructed signal $\mathbf{s}$ can be rewritten as follows, which is a key part of Schumitsch's approach:

$$\mathbf{s} = \hat{\mathbf{s}} + \mathbf{r} \tag{3.1 revisited}$$

$$= \mathbf{p} + \mathbf{M}\,\mathbf{s} + \mathbf{T}\,\mathbf{c} \tag{3.4}$$

Note that $\mathbf{s}$ appears on both sides of the equation sign, which can easily be resolved:

$$0 = \mathbf{p} + (\mathbf{M} - \mathbf{I})\,\mathbf{s} + \mathbf{T}\,\mathbf{c} \tag{3.5}$$

In Schumitsch's method, Eq. 3.5 is a linear equality constraint of the Quadratic Program formulation.

For the approach pursued in this thesis, an explicit expression for $\mathbf{s}$ is required, which is obtained as:

$$\mathbf{s} = (\mathbf{I} - \mathbf{M})^{-1}\,(\mathbf{p} + \mathbf{T}\,\mathbf{c}) \tag{3.6}$$

Note that since $\mathbf{M}$ is a very large matrix, direct computation of the inverse of $(\mathbf{I}-\mathbf{M})$ is not practical. However since

$$(\mathbf{I}-\mathbf{M})\sum_{\nu=0}^{\infty}\mathbf{M}^{\nu}=\mathbf{I}+\mathbf{M}+\mathbf{M}^2+\ldots-\mathbf{M}-\mathbf{M}^2-\mathbf{M}^3-\ldots=\mathbf{I} \tag{3.7}$$

it follows that

$$(\mathbf{I}-\mathbf{M})^{-1}=\sum_{\nu=0}^{\infty}\mathbf{M}^{\nu}. \tag{3.8}$$

The series on the right hand side of the equation sign converges, if and only if for each eigenvalue λ_i of the matrix $\mathbf{M}$, $|\lambda_i|<1$. Since $\mathbf{M}$ is a strictly lower triangular matrix, all its eigenvalues λ_i are equal to zero. Furthermore, any matrix that has zero as its only eigenvalue is nilpotent, such that $\mathbf{M}^{\nu}=\mathbf{0}$ for every ν larger than some ν_{max}. It is assumed that ν_{max} is the smallest such value, i.e. $\mathbf{M}^{\nu_{max}}\neq\mathbf{0}$ if $\mathbf{M}\neq\mathbf{0}$. The value of ν_{max} can be interpreted as the length of the longest prediction chain represented by the matrix $\mathbf{M}$. More formally, from the matrix $\mathbf{M}$ one can obtain a directed graph with K vertices, where there is a directed edge from vertex j to vertex i iff $m_{i,j}\neq 0$. The value of ν_{max} is equal to the length of the longest path within this graph. Since $\mathbf{M}$ is strictly lower triangular, the graph will be acyclic. (It is also intuitively obvious that the graph will be acyclic, since a reconstructed sample cannot depend on itself.) For directed acyclic graphs, the longest path can be determined in linear time. Note that if the matrix $\mathbf{M}$ represents only motion-compensated prediction, $\nu_{max}<N$, since for a group of N frames, the longest possible prediction chain has a length of $N-1$ and occurs, for example, if each frame references its direct preceding frame.

Thus, the reconstructed signal $\mathbf{s}$ can be written as:

$$\mathbf{s}=\sum_{\nu=0}^{\nu_{max}}\mathbf{M}^{\nu}\left(\mathbf{p}+\mathbf{T}\,\mathbf{c}\right) \tag{3.9}$$

3.2 Problem statement

The operational optimization of a video encoder is typically based on a Lagrangian approach, meaning that a weighted sum of the distortion of the reconstructed video samples and the corresponding bit rate is minimized [SW98]. In the reference encoder implementations (JM for H.264/AVC, HM for H.265/HEVC), this rate distortion optimization (RDO) is done block-by-block. This chapter is based on the idea originating from Schumitsch [SSW04, SSW05], that, assuming known and fixed prediction parameters (i.e., prediction modes, motion vectors, and reference indices), the transform coefficients of a reference frame can be chosen in such a way that the impact on the referring frames is taken into account and consequently the overall rate distortion performance is improved. For that purpose, a group of N consecutive frames in coding order is jointly optimized. There is an interdependency between prediction parameters and transform coefficients which is resolved in the following way. In a first step, the prediction parameters and transform coefficients for the individual frames are determined using the ordinary encoding method as in the reference encoder implementation. Then, in a second step, the transform coefficients for this group of frames are redetermined, utilizing the now known inter-frame dependencies. Since these new transform coefficients would presumably again lead to different prediction parameters, an iterative method that iterates between these two steps could be applied. In the following, the second step, in which the transform coefficients are optimized under consideration of inter-frame dependencies, is described in more detail.

Formally, a numerical optimization problem is stated where the optimization variables are the transform coefficients $\mathbf{c}$ of the N frames. As in the usual Lagrangian approach, a weighted sum of the distortion term $D(\mathbf{c})$ and an approximation of the bit rate $R(\mathbf{c})$ is minimized, where the trade-off between the two is controlled by a regularization parameter μ:

$$\mathbf{c}_{opt} = \arg\min_{\mathbf{c}} D(\mathbf{c}) + \mu\, R(\mathbf{c}) = \arg\min_{\mathbf{c}} J(\mathbf{c}) \tag{3.10}$$

The function $J(\mathbf{c})$ is the (approximated) rate distortion cost of the coefficient vector $\mathbf{c}$.

3.2.1 Definition of the distortion function $D(\mathbf{c})$

In video coding, the squared error between original and reconstruction (or a derived quantity thereof) is typically used as the objective distortion measure. Therefore, the sum of squared differences between original and reconstructed sample values (squared ℓ_2-norm of the difference signal) is used here as the distortion metric.

In addition to the nomenclature of the previous chapter, the $K \times 1$ column vector $\mathbf{y}$ of the original samples is introduced. The distortion function $D(\mathbf{c})$ can then be defined as follows:

$$\begin{aligned}
D(\mathbf{c}) &= \|\mathbf{y} - \mathbf{s}\|_2^2 && (3.11)\\
&= \left\| \mathbf{y} - \sum_{\nu=0}^{\nu_{max}} \mathbf{M}^{\nu} \left(\mathbf{p} + \mathbf{T}\,\mathbf{c}\right) \right\|_2^2 && (3.12)\\
&= \left\| \underbrace{\mathbf{y} - \sum_{\nu=0}^{\nu_{max}} \mathbf{M}^{\nu}\,\mathbf{p}}_{\tilde{\mathbf{y}}} - \underbrace{\sum_{\nu=0}^{\nu_{max}} \mathbf{M}^{\nu}\,\mathbf{T}}_{\mathbf{A}}\ \mathbf{c} \right\|_2^2 && (3.13)\\
&= \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 && (3.14)
\end{aligned}$$

The new vector $\tilde{\mathbf{y}}$ is equal to the original sample values, subtracted by the motion-compensated fixed prediction signal. The matrix $\mathbf{A}$ is the reconstruction operator, i.e. inverse transform followed by motion-compensated prediction.

3.2.2 Definition of the rate function $R(\mathbf{c})$

The actual bit rate that results from encoding the transform coefficient vector $\mathbf{c}$ is a very intricate function due to sophisticated entropy coding methods that are employed in state of the art video coding standards. In particular, the bit rate for encoding a transform coefficient c_i depends on the previously encoded transform coefficients. Therefore the actual bit rate is not additive in the sense, that the bit rate for each c_i can be determined independently, and the total bit rate corresponds to the sum of the individual bit rates. Since the actual bit rate cannot be stated in analytical form, a simple surrogate is used. As a first simplification, an additive rate function $R(\mathbf{c})$ is assumed, such that

$$R(\mathbf{c}) = \sum_{i=0}^{K-1} R_i(c_i) \qquad (3.15)$$

Furthermore, it is assumed, that the individual $R_i(\cdot)$ are all the same, such that

$$R(\mathbf{c}) = \sum_{i=0}^{K-1} R_0(c_i). \qquad (3.16)$$

The function $R_0(c_i)$ should be defined in such a way that a smaller transform coefficient (in absolute value) results in a smaller value of $R_0(c_i)$, since a smaller transform coefficient (in absolute value) results in smaller number of encoded binary symbols (bins) and, thus, typically in a smaller bit rate for state of the art video coding standards such as H.264/AVC (see [MSW03]) and H.265/HEVC (see [SJN$^+$12]). As the simplest function with this property, $R_0(c_i)$ is defined as the absolute value of c_i, i.e. $R_0(c_i) = |c_i|$.

The resulting surrogate rate function $R(\mathbf{c})$ is consequently obtained as the ℓ_1-norm of the transform coefficient vector $\mathbf{c}$:

$$R(\mathbf{c}) = \sum_{i=0}^{K-1} |c_i| = \|\mathbf{c}\|_1 \tag{3.17}$$

3.2.3 Regularized least squares problem

With the above definitions of $D(\mathbf{c})$ and $R(\mathbf{c})$, the multi-frame transform coefficient optimization problem (Eq. 3.10) is obtained in the following standard form:

$$\mathbf{c}_{opt} = \arg\min_{\mathbf{c}} \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 + \mu\,\|\mathbf{c}\|_1 \tag{3.18}$$

Numerical optimization problems of this type are called ℓ_1-regularized least squares problems. They have attracted much interest in recent years. An introductory overview with focus on signal processing can be found in [ZE10]. An appealing property of this kind of optimization problem is that it leads to a *sparse* solution vector, i.e. a vector which has many components equal to zero. The sparsity, i.e. the fraction of zero components, is controlled by the regularization parameter μ. A larger value of μ corresponds to a larger sparsity, since the impact of the regularization term is increased. The question how to choose the value of μ in the context of multi-frame transform coefficient optimization is addressed in a later chapter.

3.2.4 Obtaining an integer solution

Since transform coefficient levels can only assume integer values, in multi-frame transform coefficient optimization, one is interested in finding the integer-valued solution of 3.18:

$$\mathbf{c}_{opt,int} = \arg \min_{\mathbf{c} \in \mathbb{Z}^K} \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 + \mu \,\|\mathbf{c}\|_1 \tag{3.19}$$

As has been shown in [Eve63], the solution of the regularized problem 3.19 is equal to the solution of the following constrained problem, with an appropriate value of τ:

$$\mathbf{c}_{opt,int} = \arg \min_{\mathbf{c} \in \mathbb{Z}^K} \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 \qquad \text{subject to } \|\mathbf{c}\|_1 \leq \tau \tag{3.20}$$

This can be rewritten in the following form:

$$\begin{gathered} \mathbf{c}_{opt,int} = \arg \min_{\mathbf{c} \in D \subset \mathbb{Z}^K} \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 \\ \text{with } D := \{\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^K \wedge \|\mathbf{x}\|_1 \leq \tau\} \end{gathered} \tag{3.21}$$

Integer least squares optimization is studied in [HV05, VH05]. Equation 3.21 is problem (2) of [HV05] with a particular definition of D. It is reported there that, for a general $\mathbf{A}$, this problem is NP hard, both in a worst-case sense as well as in an average sense. First, three heuristic methods are proposed in [HV05]:

1. Solve the unconstrained (i.e., real-valued) problem 3.18 and round component-wise to the nearest integer. The integer solution obtained in this way is also called the *Babai estimate* [Bab86].

2. *Nulling and cancelling.* Here, the Babai estimate is kept fixed for one component (e.g., c_0) and its effect is cancelled out from the original $\tilde{\mathbf{y}}$. Then, the process is repeated for the remaining $K-1$ unknowns etc. until the values of all the components are fixed.

3. *Nulling and cancelling with optimal ordering.* This is similar to the previous method, with the difference that an ordering from the "strongest" to the "weakest" component is used.

Then, an exact method using the sphere decoding algorithm of [Poh81, FP85] is presented. In the second part of the paper [VH05, Fig. 1], the expected complexity of the exact algorithm is plotted as a function of the dimension of the problem for various settings. It has to be noted that the maximum dimension considered there is 40, which is several orders of magnitude smaller than for typical scenarios of the multi-frame transform coefficient optimization problem, where for a group of $N = 3$ frames having CIF resolution of 352×288, the resulting dimension will already be

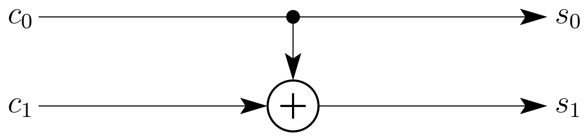


Figure 3.1: Illustration of the reconstruction process for the example with two coefficients.

in the order of $K \approx 10^5$. Therefore, application of the sphere decoding algorithm is computationally intractable even for a small number of video frames at low resolution. From the three suggested heuristics, methods 2 and 3 are also problematic, because they require solving a series of K sub-problems, where the problem size is reduced by one after each step. This is also impractical, since, as stated above, the number of such sub-problems will typically be larger than 10^5, which would require solving way too many sub-problems. Therefore, unless stated otherwise, in the following usage of method 1 is assumed, i.e. the real-valued optimization problem 3.18 is solved and the individual components of the solution vector are rounded to their closest integer value.

In [SSW05, p. 330, Sec. 3.3], Schumitsch proposes an iterative rounding method, which is similar in spirit to the heuristic methods 2 and 3 of [HV05]. The difference is, that using his method in each step several components of the solution vector are fixed, namely all components which fall (in absolute value) within a given interval. The size of this interval is increased in each step. This method is further studied in Sec. 4.1.9.

3.3 Motivation

3.3.1 Illustrating example with two coefficients

In this section, a very simple toy example is used in order to illustrate the optimization problem of Eq. 3.18. In particular, it is shown how the solution vector $\mathbf{c}_{opt}$ changes with varying μ. The mapping $\mu \mapsto \mathbf{c}_{opt}$ is called the *regularization path* of the optimization problem. In the example used throughout this section, multi-frame optimization of two frames having one sample each is assumed (i.e., $W = H = 1$, $N = 2$, and thus $K = 2$). Consequently, $\mathbf{c}_{opt}$ has two components. The signal flow of the reconstruction process in this example is shown in Fig. 3.1. The transform coefficient c_0 of the first

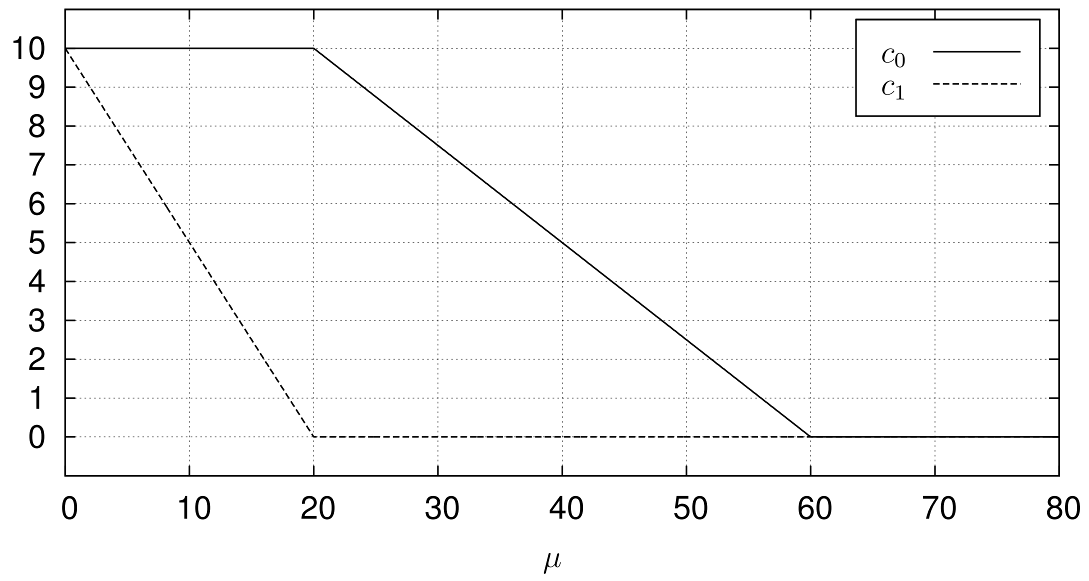


Figure 3.2: Regularization path for the example with two coefficients and $\tilde{\mathbf{y}} = [10\ \ 20]^T$.

frame directly maps to the reconstructed sample s_0. The reconstructed sample s_1 is obtained as the sum of c_0, which here plays the role of the prediction signal, and the residual signal c_1. This corresponds to the following matrix definitions:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.22}$$

The regularization path for $\tilde{\mathbf{y}} = [10\ \ 20]^T$ is shown in Fig. 3.2. Obviously, for $\mu = 0$, there is no regularization term, and thus $\mathbf{c}_{opt} = \mathbf{A}^{-1}\tilde{\mathbf{y}} = [10\ \ 10]^T$. For $\mu \geq 60$, the impact of the regularization term is so strong against the distortion term, that $\mathbf{c}_{opt} = \mathbf{0}$. For $0 < \mu < 60$, $\mathbf{c}_{opt}$ makes a transition from the least squares solution to the all-zero solution. In the first part of this region, for $0 < \mu \leq 20$, $c_{opt,1}$ linearly decreases to zero, whereas $c_{opt,0}$ stays at its initial value of 10. For $20 < \mu \leq 60$, $c_{opt,0}$ linearly shrinks to zero (with half the slope as $c_{opt,1}$ before) and $c_{opt,1}$ stays at its final value of zero.

A visualization that may be helpful to get an intuitive understanding of this behaviour is shown in Fig. 3.3. First note that, according to the well-known method of Lagrange multipliers [Eve63], the optimization problem 3.18 can also be stated in the following form:

$$\mathbf{c}_{opt} = \arg\min_{\mathbf{c}} \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 \qquad \text{subject to } \|\mathbf{c}\|_1 \leq \tau \tag{3.23}$$

This is to be understood in the sense that for each optimization problem in the form of Eq. 3.18 with a given value of μ, there is a corresponding optimization problem in the form of Eq. 3.23 with an appropriate value of τ, such that both have the same solution $\mathbf{c}_{opt}$. There is, however, no direct mapping from μ to τ. The visualization in Fig. 3.3 is based on the ℓ_1-constrained formulation of Eq. 3.23. The ellipse-shaped contour lines show the distortion term $\|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2$. All points along each contour line share the same distortion value. The least squares solution, which is obtained at $\mathbf{c} = [\,10 \;\; 10\,]^T$, corresponds to a distortion value of zero and is shown by a filled circle. Each of the four diagrams in Fig. 3.3 corresponds to a different value of τ (and therefore, in the formulation of Eq. 3.18, to a different regularization parameter μ). The feasible region, i.e. the area for which $\|\mathbf{c}\|_1 \leq \tau$, is shown shaded in gray. The solution $\mathbf{c}_{opt}$ of the ℓ_1-constrained problem 3.23 is that point of the feasible region for which the distortion term has the smallest value. It is marked by a hollow circle. The top left diagram, for example, shows the constellation for $\tau = 15$. The smallest distortion with $\|\mathbf{c}\|_1 \leq 15$ is achieved at $\mathbf{c} = [\,10 \;\; 5\,]^T$. As can be seen from Fig. 3.2, this corresponds to $\mu = 10$. In the top right diagram, the case for $\tau = 10$ is shown, leading to $\mathbf{c} = [\,10 \;\; 0\,]^T$, which corresponds to $\mu = 20$. The regularization path, i.e. the path from the least squares solution to the all-zero solution, is shown by a black arrow line. Each point along this path corresponds to a $\mathbf{c}_{opt}$ that is obtained for a different value of τ (or μ, respectively). It can be seen that c_1 is decreased until $\mathbf{c}_{opt}$ hits the right corner of the shrinking feasible region. Then, $\mathbf{c}_{opt}$ stays at the right corner of the feasible region and is "towed away" towards the origin. This explains why regularization using the ℓ_1-norm favors solutions having components exactly equal to zero. The contour lines are more likely to hit one of the corners of the feasible region, which are aligned with the coordinate axes, where the corresponding component of the solution vector is equal to zero [Tib94].

In the context of multi-frame transform coefficient optimization under a rate constraint, the regularization path of Fig. 3.2 can be interpreted as follows. The case that the available bit rate budget is sufficient to encode both coefficients at their correct value corresponds to $\mu = 0$, leading to the least squares solution. The case that neither of the coefficients can be encoded, because the available bit rate budget is effectively zero, corresponds to $\mu \geq 60$, such that $\mathbf{c}_{opt} = \mathbf{0}$. For $0 < \mu < 60$, there is some bit rate budget available which is not sufficient to encode both coefficients at their correct value, and which therefore has to be distributed among c_0 and c_1. The only parameters here, which control the resulting bit rate, are the values that are encoded for c_0 and c_1, where a smaller absolute value leads to a smaller bit rate. Obviously, c_0 has a larger impact on the overall distortion of the reconstruction, since spending less

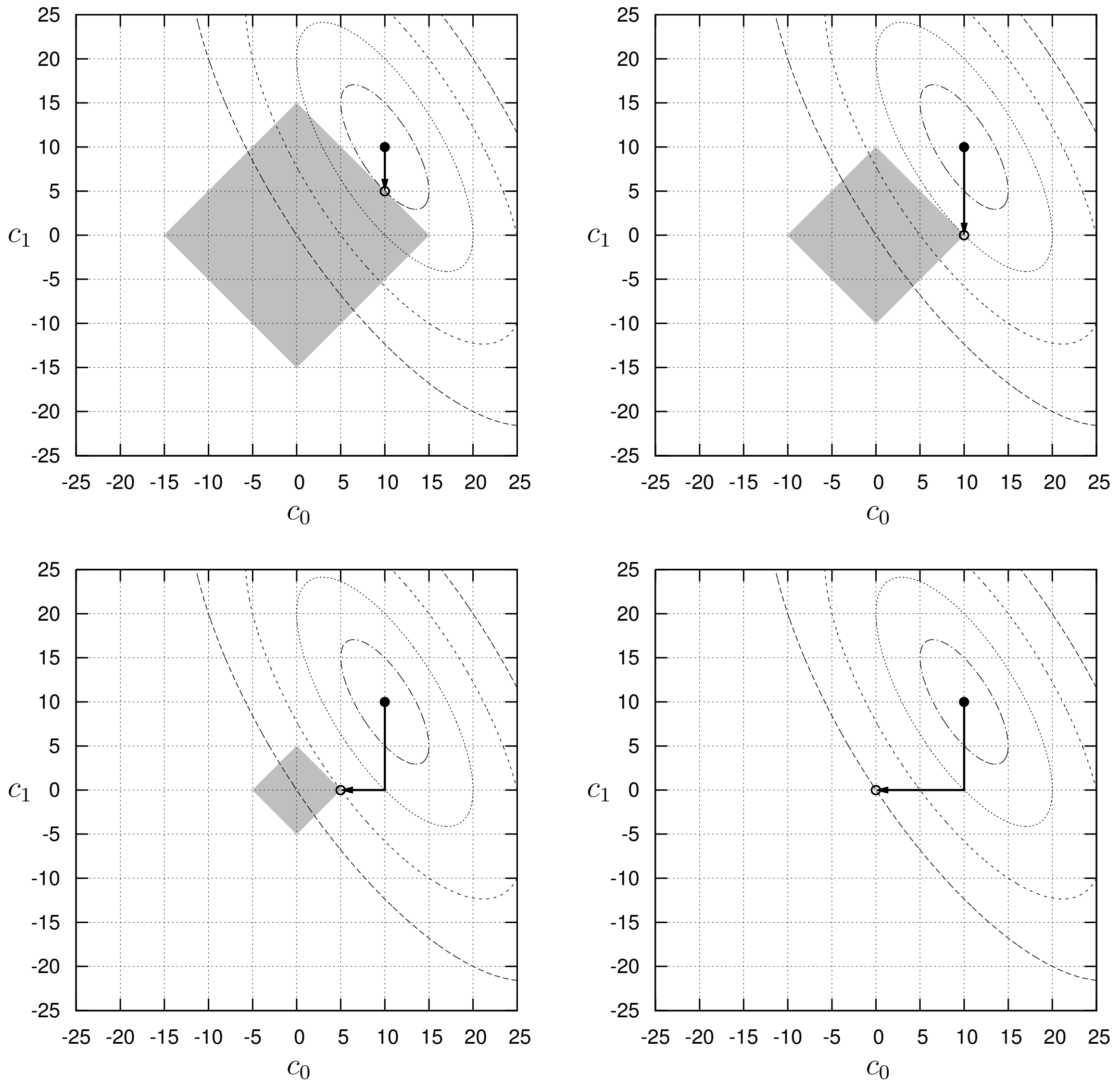


Figure 3.3: Visualization of the distortion term (ellipse-shaped contour lines), the least squares solution (filled circle), the ℓ_1-regularized solution $\mathbf{c}_{opt}$ (hollow circle), the regularization path (arrow line), and the feasible region (gray-shaded area) for $\mu = 10$ (top left), $\mu = 20$ (top right), $\mu = 40$ (bottom left), and $\mu = 60$ (bottom right).

bit rate for encoding ${c_0}^1$ would harm both s_0 and s_1, whereas doing so for c_1 would only impact s_1. Consequently, c_0 is not impaired at first (for $\mu \leq 20$), meaning that it still receives the same share of the original bit rate budget as for $\mu = 0$, whereas the amount spent for the encoding of c_1 is successively decreased. As the bit rate spent for c_1 hits zero, the bit rate for c_0 is also successively decreased until no information is encoded at all. Hence, it can be stated that the optimization problem 3.18, with an appropriate choice of the regularization parameter μ, leads to a solution vector $\mathbf{c}_{opt}$ where the available bit rate is distributed among the individual coefficients in such a way that the impact on the overall reconstruction distortion is taken into account, i.e. coefficients with larger impact on the overall reconstruction distortion are encoded with a smaller error, whereas for coefficients having little impact, a larger coding error can be tolerated.

Another exemplary regularization path is shown in Fig. 3.4. As before, the prediction structure as shown in Fig. 3.1 is used, but applied to a different original signal, namely $\tilde{\mathbf{y}} = [\,6 \;\; 0\,]^T$. It can be seen that in this case, for $0 < \mu < 4$, both c_0 and c_1 are shrinked towards zero. This can be attributed to the fact that in this scenario, by shrinking c_0, the prediction for s_1 is actually improved. Therefore, from a coding perspective, by shrinking c_0 right from the beginning, bit rate is saved in two ways, firstly the bit rate spent for encoding c_0 is reduced, secondly the required bit rate for encoding c_1 is also reduced, since its prediction is improved.

In Fig. 3.5 the regularization path for $\tilde{\mathbf{y}} = [\,10 \;\; -10\,]^T$ is shown, again using the same prediction structure as before. In this case, not only both c_0 and c_1 are shrinked towards zero right from the beginning, like in the previous example, but c_0 even hits zero before c_1. This might be unexpected, since one might assume that as c_0 affects both s_0 and s_1, whereas c_1 only affects s_1, c_0 should generally be considered more important than c_1 and therefore, in terms of rate allocation, always be favored over c_1. However, it has to be noted, that Fig. 3.5 shows a pathological example, since in this case s_0 is an unsuitable predictor for s_1, because — in terms of the ℓ_1-norm as a simplified rate function — coding s_1 directly would require less bits than coding the prediction residual $s_1 - s_0$.

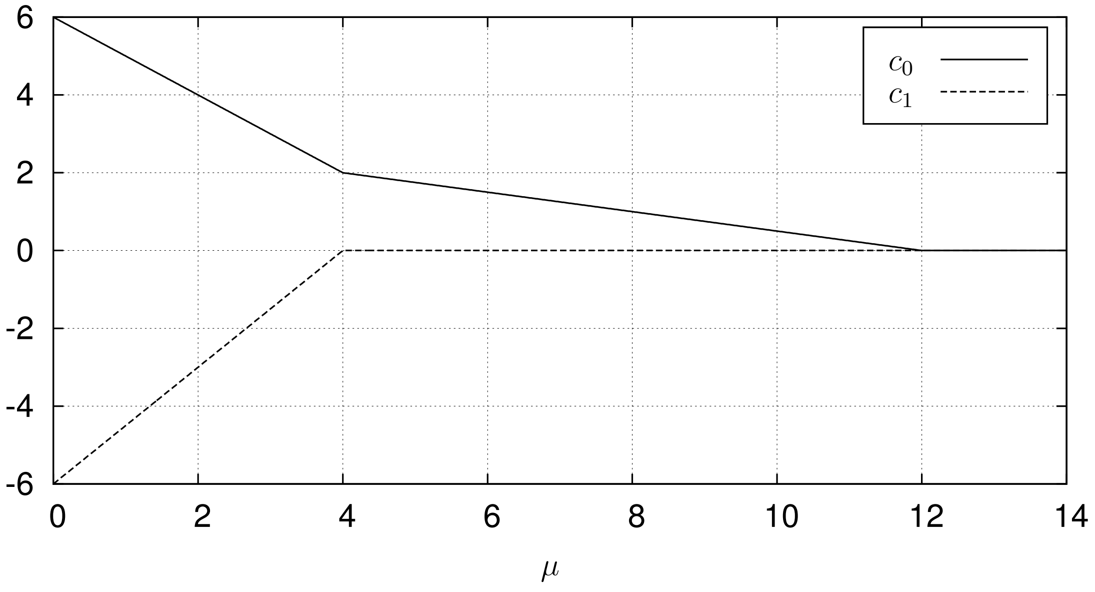


Figure 3.4: Regularization path for the example with two coefficients and $\tilde{\mathbf{y}} = [\,6\;\;0\,]^T$.

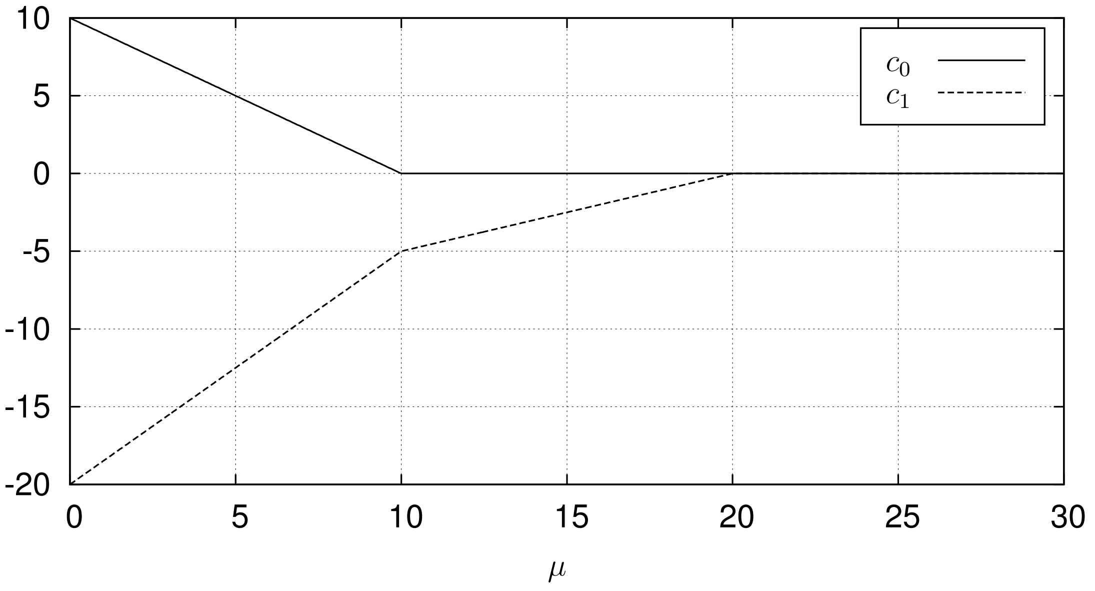


Figure 3.5: Regularization path for the example with two coefficients and $\tilde{\mathbf{y}} = [\,10\;\;-10\,]^T$.

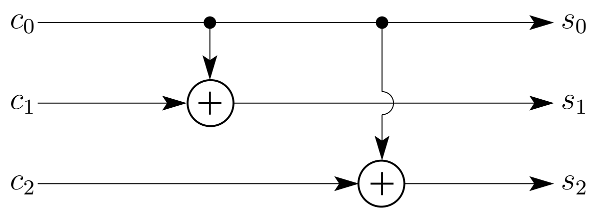


Figure 3.6: Illustration of the reconstruction process for the example with three coefficients.

3.3.2 Illustrating example with three coefficients

In this section, the toy example of the previous section is extended to three frames, as shown in Fig. 3.6. The relationship between c_0, c_1, s_0, and s_1 is the same as before. The additional reconstructed sample s_2 is, just like s_1, predicted from s_0. This corresponds to the following matrix definitions:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{3.24}$$

In Fig. 3.7, the regularization path for $\tilde{\mathbf{y}} = [\,10 \quad 20 \quad 20\,]^T$ is shown. Note that the scenario of Fig. 3.2 can be viewed as a subset of this setting, where c_2 and s_2 are not considered. Even though the settings of Fig. 3.2 and Fig. 3.7 are similar, the behaviour for c_0 is very different. Here, when μ is increased starting from zero, c_0 is not retained at its original value, but instead it is raised from 10 to 15. This is because in this setting, there are two reconstruction samples, s_1 and s_2, that benefit from an increased prediction signal c_0. So, even though more bit rate is spent for coding c_0 and the reconstruction error of s_0 is increased, there is an overall gain, since the prediction error for s_1 and s_2 is reduced, and therefore less bit rate has to be spent for c_1 and c_2.

This example illustrates:

- The optimization problem 3.18 with $\mu > 0$ does not in every case lead to coefficients with smaller or the same absolute value as the least squares solution.

- Solving the optimization problem for only a subset of the frames may lead to

[1]Which effectively means encoding a smaller value for c_0.

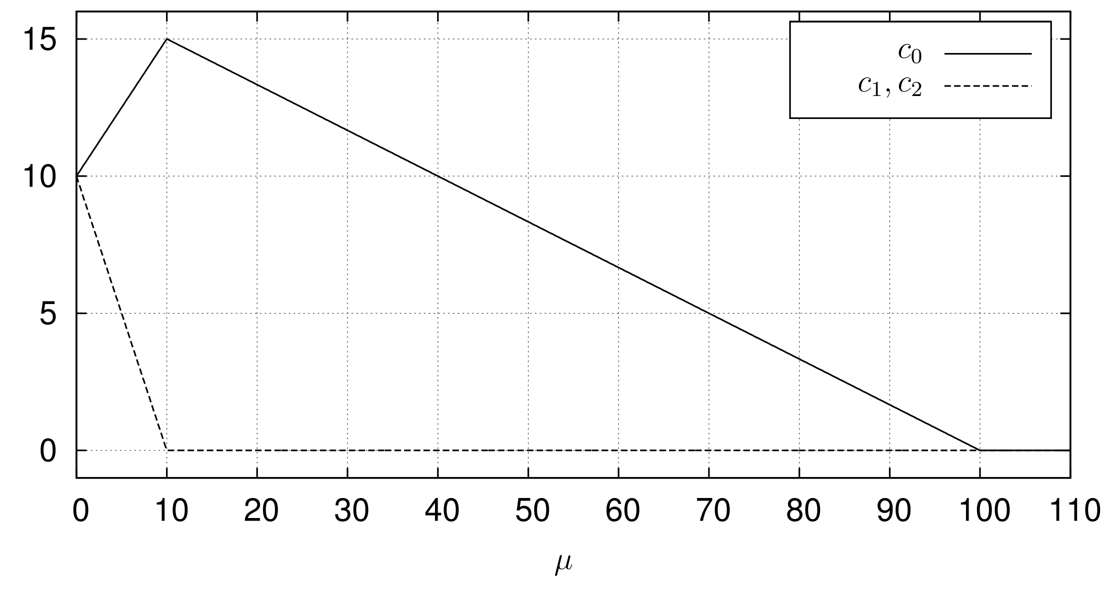


Figure 3.7: Regularization path for the example with three coefficients and $\tilde{\mathbf{y}} = [10 \;\; 20 \;\; 20]^T$.

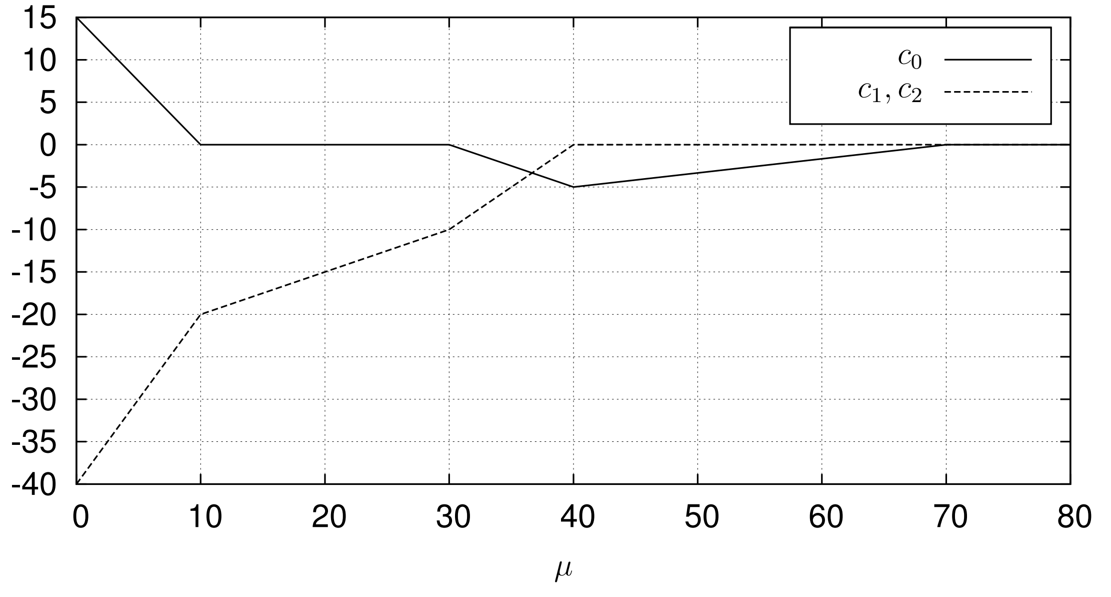


Figure 3.8: Regularization path for the example with three coefficients and $\tilde{\mathbf{y}} = [15 \;\; -25 \;\; -25]^T$.

completely different results (compare c_0 in Fig. 3.2 and Fig. 3.7).

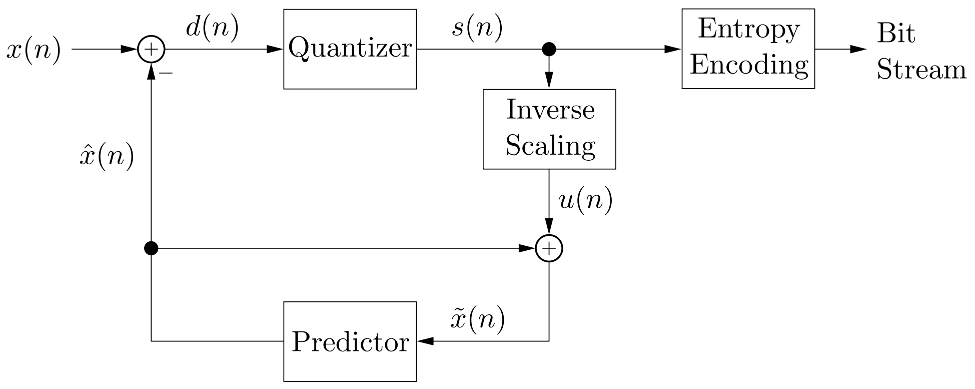
Fig. 3.8 shows the resulting regularization path, if the prediction structure of Fig. 3.6 is used and $\tilde{\mathbf{y}} = [\,15 \;\; -25 \;\; -25\,]^T$. This again is a rather pathological example, since s_0 is a bad predictor for s_1 and s_2. Two distinctive features can be observed for c_0:

- After c_0 has reached zero for $\mu = 10$, it becomes again non-zero for $30 < \mu < 70$.
- For $30 < \mu < 70$, c_0 has the opposite sign as the least squares solution.

From the discussed examples it can be reasoned, that generally it is difficult to predict how the solution of 3.18 will look like. As has been shown, the individual components of $\mathbf{c}_{opt}$ can be larger or smaller in absolute value than the least squares solution, and even the sign may change. In any case, if the ℓ_1-norm is accepted as a simple surrogate for the actual bit rate, 3.18 will lead to a transform coefficient vector whose components are chosen such that the overall reconstruction distortion is minimized under the given bit rate constraint.

3.3.3 Optimization of DPCM for Gauss-Markov sources

In this section, the rate distortion behaviour of the multi-frame optimization method is studied in a simple setting. For that purpose, it is applied to the encoding of first-order Gauss-Markov sources using differential pulse code modulation (DPCM). The general architecture of DPCM is depicted in Fig. 3.9. Following the notation of [JN84], the input signal is denoted as $x(n)$, the prediction signal as $\hat{x}(n)$, the unquantized prediction error as $d(n)$, the quantized prediction error as $u(n)$, the reconstructed signal as $\tilde{x}(n)$, and the decoder output signal as $y(n)$. Additionally, the sequence of quantization indices (or *levels*, following the terminology of video coding standards) is denoted here as $s(n)$. Note that all the $s(n)$ are integer-valued, i.e. $s(n) \in \mathbb{Z}$. In the following, the simple predictor $\hat{x}(n) = \tilde{x}(n-1)$ is assumed, i.e. the previous reconstructed sample is used as the predictor for the current sample. Furthermore, a *uniform reconstruction with unity ratio quantizer* (URURQ) [Sul96] is assumed,



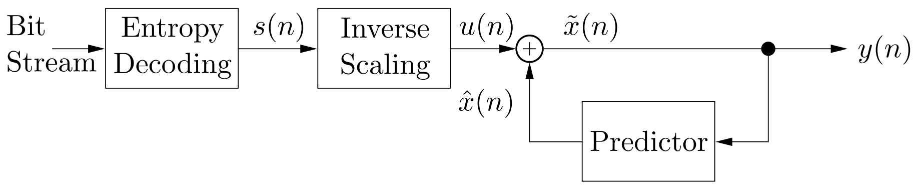


Figure 3.9: Basic architecture of a DPCM encoder (top) and decoder (bottom).

leading to the following relations:

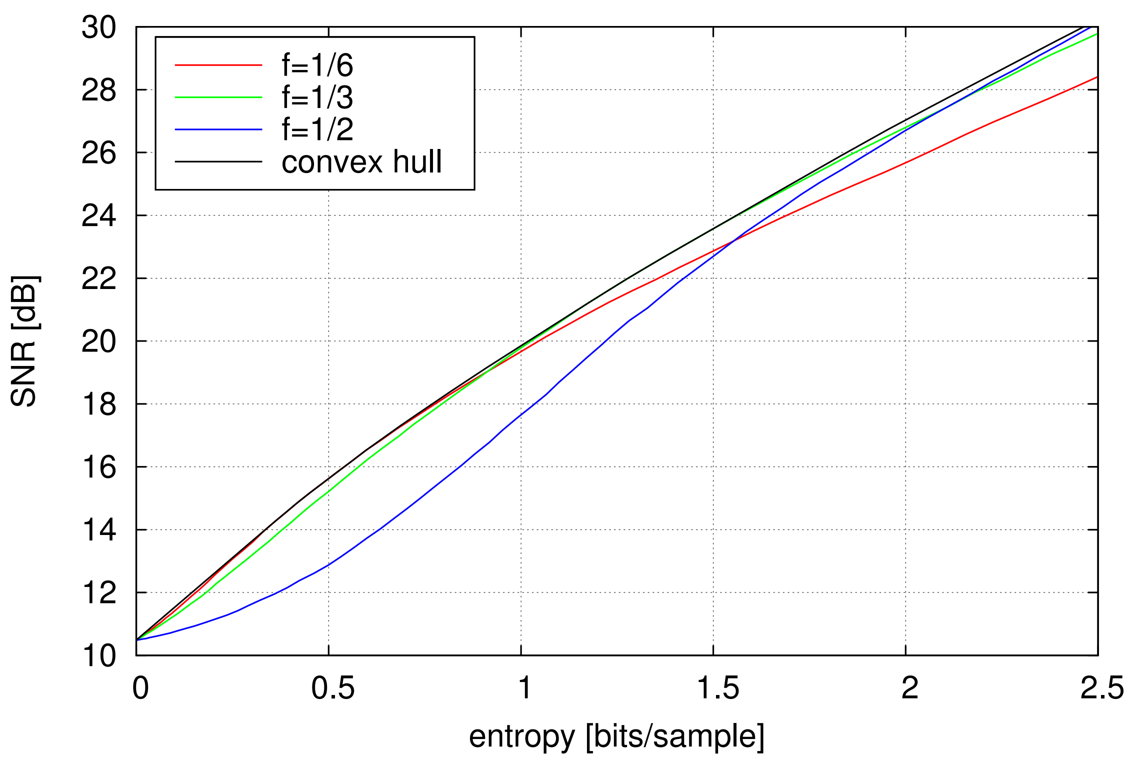
$$\begin{aligned} s(n) &= \operatorname{sgn}(\, d(n)\,) \left\lfloor \frac{|d(n)|}{\Delta} + f \right\rfloor & (3.25) \\ u(n) &= \Delta \cdot s(n) & (3.26) \end{aligned}$$

The parameter Δ is the quantization step size, f is a rounding control parameter, and $\lfloor x \rfloor$ denotes rounding to the nearest integer that is less than or equal to x (see Eq. 2.11).

A first-order Gauss-Markov process $X(n)$ (for $n = 0, 1, \ldots$) with correlation coefficient ρ can be generated as follows from a memoryless standard (i.e., zero mean and unit variance) Gaussian distributed *innovation sequence* $Z(n) \sim \mathcal{N}(0, 1)$:

$$X(n) = \begin{cases} Z(n) & \text{if } n = 0 \\ \rho X(n-1) + \sqrt{1-\rho^2}Z(n) & \text{if } n > 0 \end{cases} \tag{3.27}$$

Note that the innovation signal is scaled by a factor of $\sqrt{1-\rho^2}$ in the recursive part of the definition, in order to make the $X(n)$ for $n > 0$ also standard Gaussian distributed (i.e., $X(n) \sim \mathcal{N}(0, 1)$).



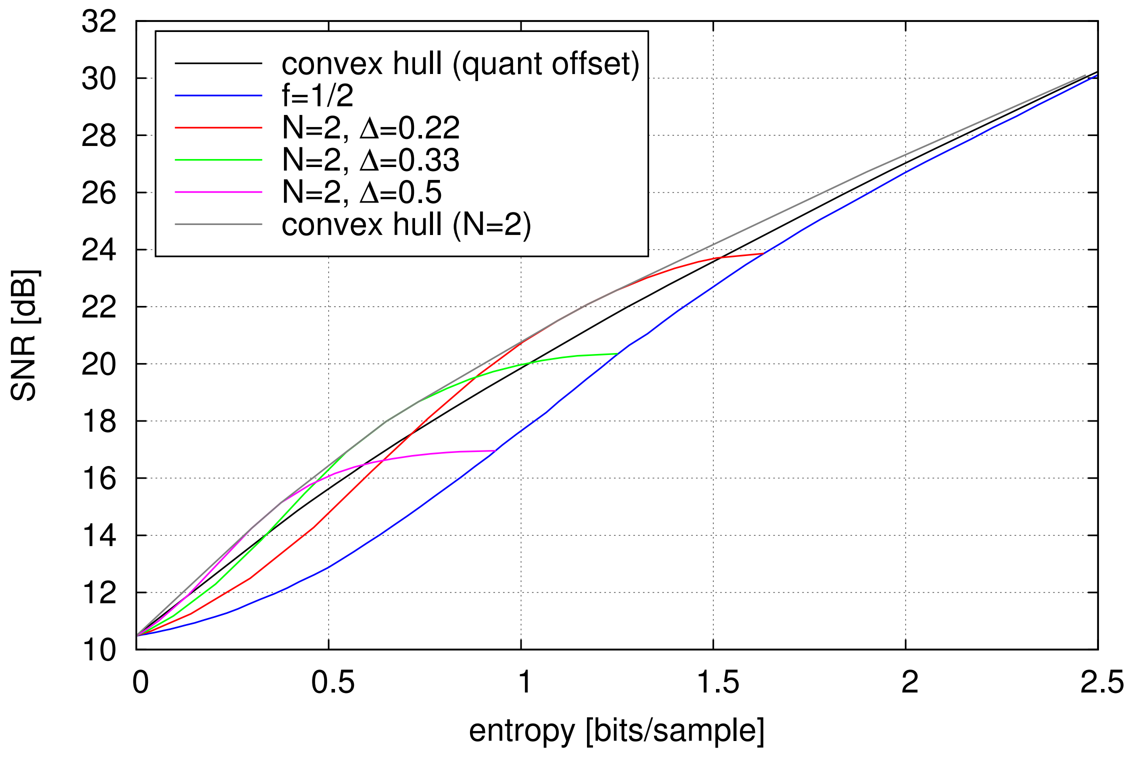


Figure 3.10: Empirical rate distortion curves for a first-order Gauss-Markov source with $\rho = 0.99$ (top: variation of the rounding offset f; bottom: variation of the regularization parameter μ).

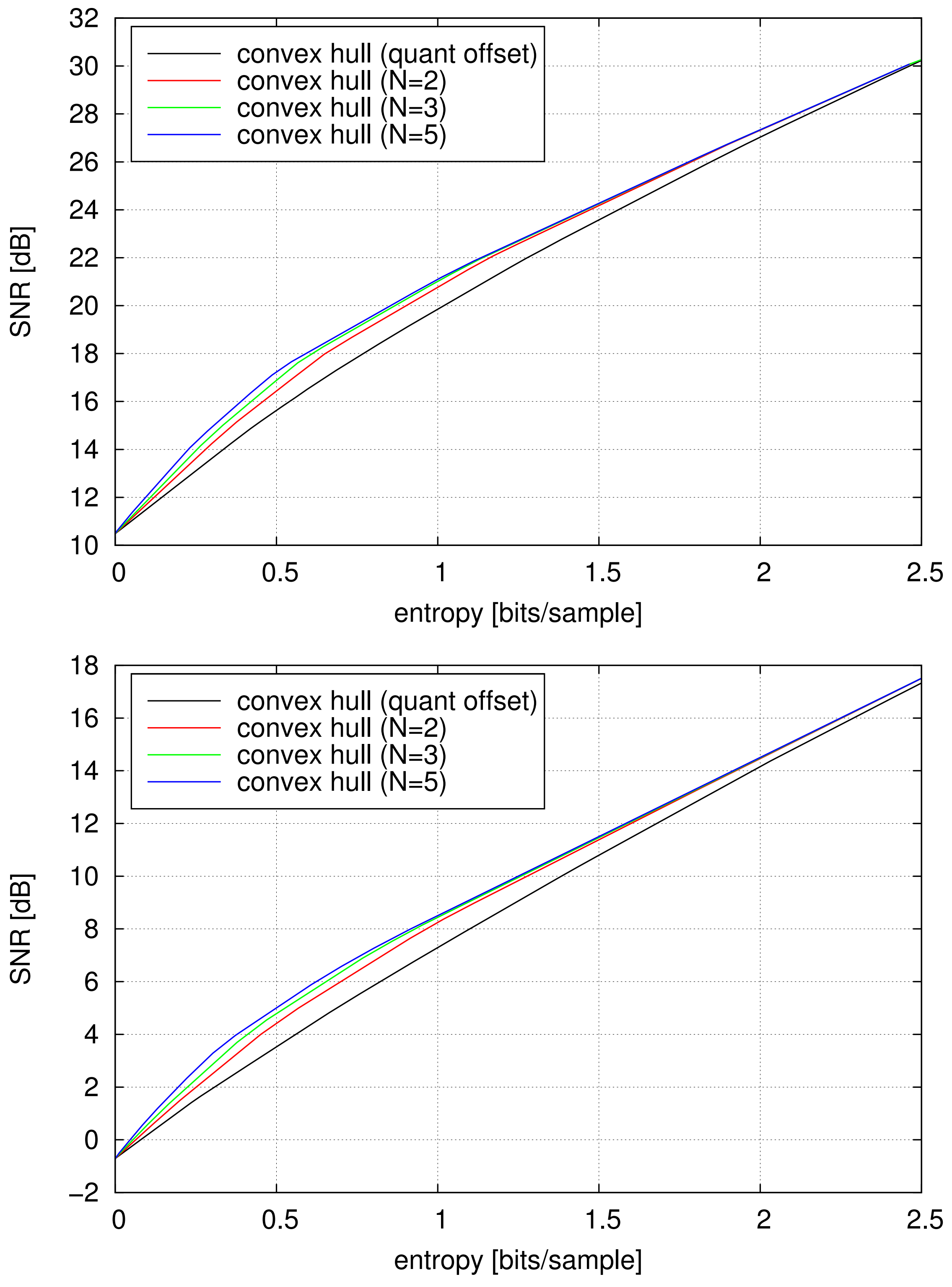


Figure 3.11: Empirical rate distortion curves for a first-order Gauss-Markov source with $\rho = 0.99$ (top) and $\rho = 0.8$ (bottom).

This can easily be verified, since per definitionem $X(0) \sim \mathcal{N}(0,1)$, it follows that $\rho X(0) \sim \mathcal{N}(0,\rho^2)$, and analogously $\sqrt{1-\rho^2}Z(1) \sim \mathcal{N}(0,1-\rho^2)$. Furthermore, since $\rho X(0)$ and $\sqrt{1-\rho^2}Z(1)$ are independent Gaussian random variables, it follows that

$$X(1) = \rho X(0) + \sqrt{1-\rho^2}Z(1) \sim \mathcal{N}(0,\rho^2+1-\rho^2) = \mathcal{N}(0,1).$$

Now, since it has been shown that $X(1) \sim \mathcal{N}(0,1)$, the same argument holds for all $n > 0$, and consequently it follows that all $X(n) \sim \mathcal{N}(0,1)$.

For the following investigation, an input signal $x(n)$ of length 11 (i.e., $n = 0,1,\ldots,10$) is used. Note that the first sample $x(0)$ plays a special role here, as there is no prediction signal available (it is assumed that $\hat{x}(0) = 0$, because $x(0)$ is of zero mean). Speaking in the terms of video coding, this first sample could be interpreted as the counterpart of an I frame, whereas the subsequent, predicted samples act like P frames. Consequently, for $\rho > 0$ and under a high bit rate assumption, the variance of the quantizer input signal $d(n)$ will be larger for $n = 0$ than for the remaining $n > 0$. Accordingly, this would have to be considered in the selection of the quantization step sizes for $n = 0$ and $n > 0$. In order to simplify matters, in the following only the predicted samples (corresponding to $n > 0$) are considered and it is assumed that $\hat{x}(1) = x(0)$, i.e. the prediction signal $\hat{x}(1)$ for the first predicted sample is the original (unquantized) input sample $x(0)$. This simplification is based on the assumption that the initial “intra” sample has been encoded at high reconstruction quality and can therefore serve as a good predictor for $x(1)$.

For the following experiments, 10 000 realizations of the first 11 outcomes of a first-order Gauss-Markov source with a very high correlation of $\rho = 0.99$ have been generated. For notational convenience, the number of realizations is denoted as $M = 10\,000$ and the number of predicted samples as $L = 10$. The evaluation is based on the empirical per-sample entropy rate R and signal-to-noise ratio (SNR) Q, which will be defined as follows. The notation as used in Fig. 3.9 and introduced above is slightly extended by a subscript i which refers to the actual realization of the random process ($i \in \{0,1,\ldots,M-1\}$). For example, $x_i(n)$ refers to the nth sample of the ith realization of the input signal. First, the mean squared error for a given sample position n is defined as the average squared difference between original and reconstruction:

$$MSE(n) = \frac{1}{M}\sum_{i=0}^{M-1}\left(x_i(n) - \tilde{x}_i(n)\right)^2 \tag{3.28}$$

Then, mean value and variance of the original signal are defined as:

$$mean(n) = \frac{1}{M}\sum_{i=0}^{M-1} x_i(n) \tag{3.29}$$

$$var(n) = \frac{1}{M}\sum_{i=0}^{M-1} \left(x_i(n) - mean(n)\right)^2 \tag{3.30}$$

Based on this, the SNR for a sample position n is defined as:

$$SNR(n) = 10 \cdot \log_{10} \frac{var(n)}{MSE(n)} \tag{3.31}$$

Now, the empirical per-sample signal-to-noise ratio Q is defined as:

$$Q = \frac{1}{L}\sum_{n=1}^{L} SNR(n) \tag{3.32}$$

For the definition of the empirical entropy, first the empirical probability (or relative frequency) $\hat{p}_n(x)$ of the nth quantization index $s(n)$ taking the value x is defined with the help of the cardinality of a set:

$$\hat{p}_n(x) = \frac{1}{M}\left|\{i|s_i(n) = x\}\right| \tag{3.33}$$

Based on this, the empirical entropy $H(n)$ of the nth sample is defined as follows (with $0 \cdot \log 0 = 0$):

$$H(n) = -\sum_{x\in\mathbb{Z}} \hat{p}_n(x) \cdot \log_2 \hat{p}_n(x) \tag{3.34}$$

Then, the empirical per-sample entropy rate R is defined as:

$$R = \frac{1}{L}\sum_{n=1}^{L} H(n) \tag{3.35}$$

In a first simulation experiment, no ℓ_1-regularization based optimization is employed, but instead the impact of the rounding control parameter f (Eq. 3.25) on the rate distortion performance is studied. The results are shown in Fig. 3.10 (top). The individual curves have been obtained by a variation of the quantization step size Δ. It can be seen that in particular at lower entropy rates, rounding to the nearest integer (which corresponds to using $f = 1/2$) is clearly outperformed by choosing $f = 1/6$ or

$f = 1/3$. This is a well-established fact [Sul96, SS05], and consequently, in the H.264/AVC and H.265/HEVC reference encoders, $f = 1/6$ and $f = 1/3$ are used for inter- and intra-predicted blocks, respectively. Because the input signal has a high correlation of $\rho = 0.99$ and because only the 10 predicted samples $x(1), \dots, x(10)$ are considered, excluding the initial "intra" sample $x(0)$, the SNR for $R = 0$ is larger than zero here, which manifests in an y-intercept of about 10.5 dB. This is the reconstruction quality which the predicted samples gain solely by continued prediction from $x(0)$, i.e. $\tilde{x}(n) = \hat{x}(n) = x(0)$ is used in this case for $n > 0$. The curve denoted as "convex hull" shows the optimal rate distortion performance of scalar quantization, if f is varied within $[0; 0.5]$.

In the next simulation experiment, this envelope of sweeping the rounding control parameter f serves as the reference for comparison with ℓ_1-regularized optimization. The optimization is done such that in the selection of the quantization index $s(n)$, the impact not only on the current reconstructed sample $\tilde{x}(n)$, but also on the subsequent sample $\tilde{x}(n+1)$ is taken into account. For that purpose, the quantization indices $s(n)$ and $s(n+1)$ are sought after, which minimize the following weighted sum of the squared error distortion and the ℓ_1-norm of the quantization indices for a given value of the regularization parameter μ:

$$\left\| \begin{bmatrix} x(n) \\ x(n+1) \end{bmatrix} - \begin{bmatrix} \tilde{x}(n) \\ \tilde{x}(n+1) \end{bmatrix} \right\|_2^2 + \mu \left\| \begin{bmatrix} s(n) \\ s(n+1) \end{bmatrix} \right\|_1 \tag{3.36}$$

Since the quantization indices are constrained to be integer-valued, this problem is difficult to solve (see Sec. 3.2.4). Therefore, the integrality requirement is relaxed in a first step, but instead an intermediate, real-valued $r(n)$ is determined, from which the corresponding integer-valued $s(n)$ is derived by rounding to its closest integer. Then, the optimal $r_{opt}(n)$, $r_{opt}(n+1)$ are derived as

$$\begin{bmatrix} r_{opt}(n) \\ r_{opt}(n+1) \end{bmatrix} = \arg \min_{[r(n)\; r(n+1)]^T} \left\{ \left\| \begin{bmatrix} x(n) - \tilde{x}(n) \\ x(n+1) - \tilde{x}(n+1) \end{bmatrix} \right\|_2^2 + \mu \left\| \begin{bmatrix} r(n) \\ r(n+1) \end{bmatrix} \right\|_1 \right\} \tag{3.37}$$

The relationship between the $r(n)$, $r(n+1)$ and the $\tilde{x}(n)$, $\tilde{x}(n+1)$ is as follows:

$$\begin{aligned}
\tilde{x}(n) &= \hat{x}(n) + u(n) && (3.38)\\
&= \hat{x}(n) + \Delta \cdot r(n) && (3.39)\\
\tilde{x}(n+1) &= \hat{x}(n+1) + u(n+1) && (3.40)\\
&= \tilde{x}(n) + u(n+1) && (3.41)\\
&= \hat{x}(n) + \Delta \cdot r(n) + \Delta \cdot r(n+1) && (3.42)
\end{aligned}$$

Based on $r_{opt}(n)$, the integer-valued $s(n)$ is obtained as:

$$s(n) = \operatorname{sgn}(\, r_{opt}(n)\,) \left\lfloor \frac{|r_{opt}(n)|}{\Delta} + \frac{1}{2} \right\rfloor \tag{3.43}$$

The optimization operates using a sliding window approach, i.e. in the first step, the optimal real-valued $r_{opt}(1)$ and $r_{opt}(2)$ are determined according to Eq. 3.37, and the integer-valued quantization index $s(1)$ is obtained according to Eq. 3.43. Then, the sliding window proceeds by one step, and in the next round $r_{opt}(2)$ and $r_{opt}(3)$ are determined, leading to the integer-valued $s(2)$ and so on. Using a sliding window is advantageous over optimizing several quantization indices at once as there would be no feedback of the quantization error introduced in Eq. 3.43 otherwise. The simulation results are shown in Fig. 3.10 (bottom). The curve denoted as "N=2, Δ=0.5" is obtained by varying the regularization parameter μ and using a fixed quantization step size $\Delta = 0.5$. Similar curves have been generated for $\Delta = 0.33$ and $\Delta = 0.22$. It can be seen that, on each of these three curves, for a certain range of the regularization parameter μ, the resulting SNR is higher than that of the convex hull curve of Fig. 3.10 (top) for the same entropy rate. The envelope of the individual curves which are obtained by sweeping the regularization parameter μ for a fixed value of the quantization step size Δ is shown in gray and denoted as "convex hull (N=2)." In Fig. 3.11 (top), the corresponding envelopes are shown for the cases when two, three, and five samples are considered jointly in the optimization, denoted as $N = 2$, $N = 3$, and $N = 5$. At lower to medium bit rates, a significant improvement over scalar quantization with optimal adaptation of the rounding control parameter f is observed, with higher gains for a larger value of N. For Fig. 3.11 (bottom), the simulations have been repeated using a correlation coefficient of $\rho = 0.8$. Besides the lower SNR for $R = 0$, which is even negative in this case and which is caused by the smaller correlation, making $x(0)$ a worse predictor, the general behaviour is qualitatively very similar.

Spectral analysis of ℓ_1-regularization

In this section, the impact of the ℓ_1-regularization on the power spectral density (psd) of the residual signal is studied. For that purpose, two modifications to the DPCM system as described above are made:

- The quantization is omitted, meaning that the ℓ_1-regularization is the only source of distortion.
- MSE-optimal first order linear prediction is used, i.e. $\hat{x}(n) = \rho \cdot \tilde{x}(n-1)$.

Also, a slightly different definition of an autoregressive input signal $x(n)$ is used here:

$$x(n) = \begin{cases} 0 & \text{if } n < 0 \\ \rho \cdot x(n-1) + z(n) & \text{if } n \geq 0 \end{cases} \tag{3.44}$$

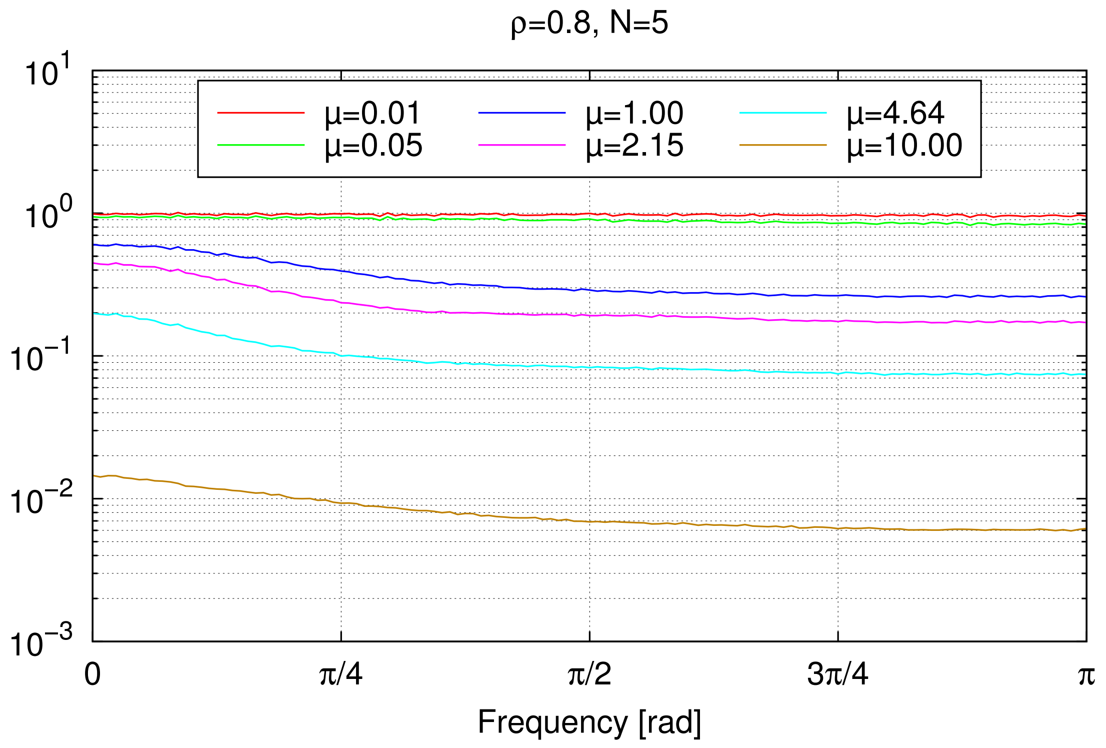
The $z(n)$ are assumed to be independent, standard Gaussian distributed, i.e. $z(n) \sim \mathcal{N}(0,1)$, which is different to the previously used autoregressive process, where the $X(n)$ where assumed to be of unit variance.

The reconstruction is still derived as $\tilde{x}(n) = u(n) + \hat{x}(n)$. The residual signal $u(n)$ is obtained by solving Eq. 3.37 and taking $u(n) = r_{opt}(n)$. Again, a sliding window approach is taken, i.e. a number N of consecutive samples $n, \ldots, n+N-1$ is considered jointly[1], then the optimization result for the first sample n is kept fixed and the sliding window proceeds by one step, such that in the next iteration the samples $n+1, \ldots, n+N$ are optimized jointly and so on.

Note that for regularization parameter $\mu = 0$, it follows that $u(n) = d(n) = x(n) - \hat{x}(n)$ and therefore $\tilde{x}(n) = u(n) + \hat{x}(n) = x(n) - \hat{x}(n) + \hat{x}(n) = x(n)$, i.e. the reconstruction is identical to the original. This is consistent with the claim that the ℓ_1-regularization shall be the only source of distortion for this investigation. Consequently, in this case, it holds that

$$\begin{aligned} u(n) &= x(n) - \hat{x}(n) && (3.45) \\ &= x(n) - \rho \cdot \tilde{x}(n-1) && (3.46) \\ &= x(n) - \rho \cdot x(n-1) && (3.47) \end{aligned}$$

[1]Eq. 3.37 represents the case for $N = 2$



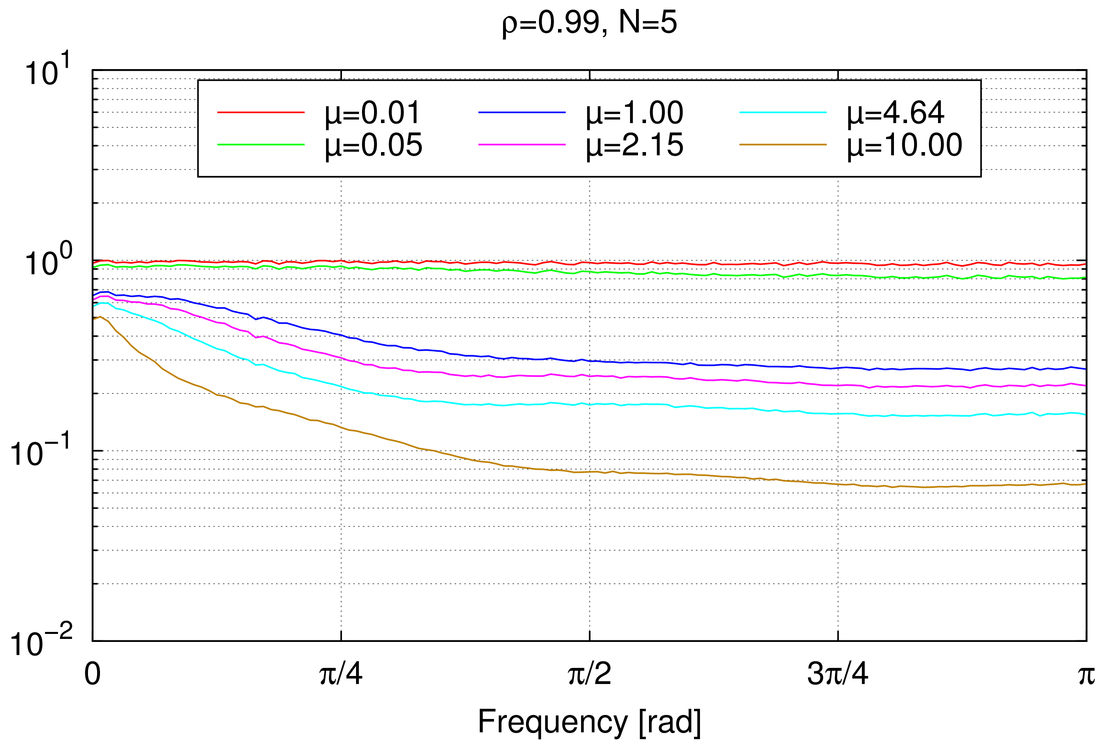


Figure 3.12: Power spectral density of the residual signal.

Furthermore, since an autoregressive input signal $x(n)$ is assumed, according to Eq. 3.44, it follows that:

$$\begin{aligned} u(n) &= (\rho \cdot x(n-1) + z(n)) - \rho \cdot x(n-1) && (3.48) \\ &= z(n) && (3.49) \end{aligned}$$

Therefore, for $\mu = 0$, the residual signal $u(n)$ will be Gaussian white noise (i.e., having flat spectrum) of unit variance.

For $\mu > 0$, $u(n) \not\equiv d(n)$ and, accordingly, $u(n) \not\equiv z(n)$. In order to determine the power spectral density (psd) of $u(n)$ for $\mu > 0$, an input sequence $x(n)$ of 10^6 samples has been generated. The residual signal samples $u(n)$ have been derived as described above, using consecutive application of the ℓ_1-regularized least squares minimization for different values of μ, ranging from 0.01 to 10. In order to study the steady state behaviour, the first 1000 samples as well as the last 10 samples of $u(n)$ have been omitted for the following explorations. For notational convenience, the sub-sequence $u'(n)$ is defined as $u'(n) = u(n + 1000)$, and the number of samples is defined as $M = 10^6 - 1000 - 10$. Then, the autocovariance sequence of $u'(n)$ is estimated as:

$$R_{u'u'}(k) = \frac{1}{M - |k|} \sum_{n=0}^{M-|k|-1} u'(n)u'(n + |k|) \qquad (3.50)$$

According to the Wiener–Khintchine theorem, the psd of $u'(n)$ is then determined as the discrete-time Fourier transform of the autocovariance sequence $R_{u'u'}(k)$ [JN84, Sec. 2.3.6, pp. 55]:

$$S_{u'u'}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} R_{u'u'}(k)e^{-jk\omega} \qquad (3.51)$$

The resulting psd plots are shown in Fig. 3.12 for $\rho = 0.8$ (top) and $\rho = 0.99$ (bottom). For both experiments a group of $N = 5$ consecutive samples has been considered jointly in the optimization. It can be seen that for small values of μ, the spectrum is almost flat, which is as expected, since for $\mu = 0$ the spectrum will be constant, corresponding to white noise. For increasing values of μ, the spectrum becomes more and more low-pass, and the energy of the signal decreases. The latter observation is also as expected, because for a large value of μ, all the $u(n)$ will be equal to zero. The low-pass characteristic of the residual signal for intermediate values of μ is consistent with the ideas presented by Guleryuz and Orchard in [GO01] for rate distortion optimized DPCM encoding of autoregressive sources at low bit rates.

Their method, which has been briefly discussed in Sec. 2.3, p. 21, basically relies on a low-pass prefiltering of the innovation signal at the encoder side. According to the findings of this section, it can be argued that by usage of ℓ_1-regularized least squares minimization, a similar effect is achieved. The ℓ_1-based method, however, is easier applicable to block-based hybrid coding as used in state-of-the-art video coding standards, because the impact of each transform coefficient on its affected reconstructed signal samples is directly captured by the linear signal model, whereas by a simple temporal low-pass filtering in the spatial domain, the impact of the block transform would be neglected.

3.4 Comparison of different solution algorithms

In this section, the question of how to solve the optimization problem 3.18 is tackled. First it is shown, that for the case of orthogonal $\mathbf{A}$, 3.18 reduces to simple *soft thresholding.* Then, for a general $\mathbf{A}$, several numerical solution algorithms are discussed. Finally, the decision to use the *iterative shrinkage/thresholding algorithm (ISTA)* in the context of multi-frame transform coefficient optimization is justified.

3.4.1 Orthogonal case: Soft thresholding solution

For the non-trivial case $\mu \neq 0$, an explicit expression for $\mathbf{c}_{opt}$ in Eq. 3.18 can only be given if $\mathbf{A}^T\mathbf{A}$ is a diagonal matrix [Mal08, p. 668]. Otherwise, for a general $\mathbf{A}$, the solution $\mathbf{c}_{opt}$ can only be determined numerically. In the context of multi-frame transform coefficient optimization, $\mathbf{A}^T\mathbf{A}$ will be diagonal only for the corner case of $N = 1$, i.e. if there is only one single frame under consideration, because in this case $\mathbf{M} = \mathbf{0}$ and consequently $\mathbf{A} = \mathbf{T}$, which is (up to a QP-dependent scaling) an orthogonal matrix if an orthogonal transform is used.

If the objective function to be minimized in Eq. 3.18 is denoted as $J(\mathbf{c})$ and $\mathbf{A}^T\mathbf{A}$ is the diagonal matrix $\operatorname{diag}(\alpha_0, \dots, \alpha_{K-1})$, the following is obtained:

$$\begin{aligned}
J(\mathbf{c}) &= \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 + \mu\,\|\mathbf{c}\|_1 && (3.52)\\
&= (\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c})^T\,(\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}) + \mu\,\|\mathbf{c}\|_1 && (3.53)\\
&= \mathbf{c}^T\,\mathbf{A}^T\,\mathbf{A}\,\mathbf{c} + \tilde{\mathbf{y}}^T\,\tilde{\mathbf{y}} - 2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})^T\,\mathbf{c} + \mu\,\|\mathbf{c}\|_1 && (3.54)\\
&= \sum_{\nu=0}^{K-1} \left(\alpha_\nu\,c_\nu^2 + y_\nu^2 - 2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_\nu\,c_\nu + \mu\,|c_\nu|\right) && (3.55)
\end{aligned}$$

From Eq. 3.55, it can be seen that this is a separable optimization problem, since in each term of the sum, there occurs only one element c_ν of the solution vector $\mathbf{c}$ (or, in other words, there are no cross-terms consisting of different c_i, c_j). Note that $J(\mathbf{c})$ is nondifferentiable for every $\mathbf{c}$ with $\exists_i c_i = 0$ because of the absolute value operator. Thus, in order to minimize $J(\mathbf{c})$, the set-valued subdifferential has to be used instead [UL01, p. 163]. The subdifferential $\partial f(\mathbf{x})$ of the convex function $f : \mathbb{R}^K \to \mathbb{R}$ at the point $\mathbf{x}$ is defined as [UL01, p. 165, Definition 1.1.4]:

$$\partial f(\mathbf{x}) := \left\{ \mathbf{s} \in \mathbb{R}^K : \langle \mathbf{s}, \mathbf{d} \rangle \le \inf_{t>0} \left\{ \frac{f(\mathbf{x} + t\,\mathbf{d}) - f(\mathbf{x})}{t} \right\} \text{for all } \mathbf{d} \in \mathbb{R}^K \right\} \qquad (3.56)$$

The vector $\mathbf{c}_{opt}$ is a global minimum of the convex function $J(\mathbf{c})$ iff $0 \in \partial J(\mathbf{c}_{opt})$ [UL01, p. 177, Theorem 2.2.1]. The ith component of the subdifferential $\partial J(\mathbf{c})$ is:

$$(\partial J(\mathbf{c}))_i = 2\,\alpha_i\,c_i - 2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i + \mu\,\partial|c_i| \qquad (3.57)$$

The subdifferential $\partial|c_i|$ of the absolute value function $|c_i|$ is:

$$\partial|c_i| = \begin{cases} \{-1\} & \text{if } c_i < 0 \\ [-1, 1] & \text{if } c_i = 0 \\ \{1\} & \text{if } c_i > 0 \end{cases} \qquad (3.58)$$

For $\mathbf{c}_{opt}$ to be a global minimum of $J(\mathbf{c})$, the following has to be fulfilled for every $i \in \{0, \dots, K-1\}$:

$$\begin{aligned}
& 0 \in (\partial J(\mathbf{c}_{opt}))_i && (3.59)\\
\Leftrightarrow\ & 0 \in 2\,\alpha_i\,c_{opt,i} - 2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i + \mu\,\partial|c_{opt,i}| && (3.60)
\end{aligned}$$

According to Eq. 3.58, three cases have to be distinguished.

Case 1: $c_{opt,i} > 0$

$$\begin{aligned} 0 &= 2\,\alpha_i\,c_{opt,i} - 2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i + \mu && (3.61)\\ c_{opt,i} &= \frac{(\mathbf{A}^T\,\tilde{\mathbf{y}})_i - \frac{\mu}{2}}{\alpha_i} && (3.62)\\ \text{subject to } (\mathbf{A}^T\,\tilde{\mathbf{y}})_i &> \frac{\mu}{2} \qquad (\text{since } \alpha_i > 0) && (3.63) \end{aligned}$$

Case 2: $c_{opt,i} < 0$

$$\begin{aligned} 0 &= 2\,\alpha_i\,c_{opt,i} - 2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i - \mu && (3.64)\\ c_{opt,i} &= \frac{(\mathbf{A}^T\,\tilde{\mathbf{y}})_i + \frac{\mu}{2}}{\alpha_i} && (3.65)\\ \text{subject to } (\mathbf{A}^T\,\tilde{\mathbf{y}})_i &< -\frac{\mu}{2} \qquad (\text{since } \alpha_i > 0) && (3.66) \end{aligned}$$

Case 3: $c_{opt,i} = 0$

$$\begin{aligned} 0 &= -2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i + \mu\,\beta \qquad \text{with } \beta \in [-1, 1] && (3.67)\\ \beta &= \frac{2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i}{\mu} && (3.68)\\ -1 \leq &\frac{2\,(\mathbf{A}^T\,\tilde{\mathbf{y}})_i}{\mu} \leq 1 && (3.69)\\ -\frac{\mu}{2} \leq &(\mathbf{A}^T\,\tilde{\mathbf{y}})_i \leq \frac{\mu}{2} && (3.70) \end{aligned}$$

Putting it all together, the following is obtained for $c_{opt,i}$:

$$\begin{aligned} c_{opt,i} &= \begin{cases} \frac{(\mathbf{A}^T\,\tilde{\mathbf{y}})_i - \frac{\mu}{2}}{\alpha_i} & \text{if } (\mathbf{A}^T\,\tilde{\mathbf{y}})_i > \frac{\mu}{2} \\ \frac{(\mathbf{A}^T\,\tilde{\mathbf{y}})_i + \frac{\mu}{2}}{\alpha_i} & \text{if } (\mathbf{A}^T\,\tilde{\mathbf{y}})_i < -\frac{\mu}{2} \\ 0 & \text{if } -\frac{\mu}{2} \leq (\mathbf{A}^T\,\tilde{\mathbf{y}})_i \leq \frac{\mu}{2} \end{cases} && (3.71)\\ &= \frac{\operatorname{sgn}((\mathbf{A}^T\,\tilde{\mathbf{y}})_i)\,\max\left\{\left|(\mathbf{A}^T\,\tilde{\mathbf{y}})_i\right| - \frac{\mu}{2},\, 0\right\}}{\alpha_i} && (3.72)\\ &= \frac{\mathcal{S}_{\frac{\mu}{2}}((\mathbf{A}^T\,\tilde{\mathbf{y}})_i)}{\alpha_i} && (3.73) \end{aligned}$$

Here, $\mathcal{S}_{\frac{\mu}{2}}$ is the *soft thresholding* operator as introduced by Donoho and Johnstone in

[DJ94] and defined by

$$\mathcal{S}_{\frac{\mu}{2}}(x) = \begin{cases} x - \frac{\mu}{2} & \text{if } x > \frac{\mu}{2} \\ 0 & \text{if } -\frac{\mu}{2} \leq x \leq \frac{\mu}{2} \\ x + \frac{\mu}{2} & \text{if } x < -\frac{\mu}{2}. \end{cases} \tag{3.74}$$

Note that if $\mathbf{A}$ is an orthogonal matrix, then $\mathbf{A}^T = \mathbf{A}^{-1}$ and $\alpha_i = 1$. Since $\mathbf{A}$ is the reconstruction operator that generates the reconstructed samples from the transform coefficients, $\mathbf{A}^{-1}$ can be interpreted as a kind of analysis or forward transform operator, that generates transform coefficients $\tilde{\mathbf{c}}$ from the desired (modified) original signal samples $\tilde{\mathbf{y}}$, with $\tilde{\mathbf{c}} := \mathbf{A}^{-1}\,\tilde{\mathbf{y}}$. In this case, the ith component of the transform coefficient vector $\mathbf{c}_{opt}$ would simply be

$$c_{opt,i} = \mathcal{S}_{\frac{\mu}{2}}(\tilde{c}_i), \tag{3.75}$$

which is the original transform coefficient (as obtained by forward transform), shrinked by an amount of $\frac{\mu}{2}$ towards zero if its absolute value is larger than $\frac{\mu}{2}$, and clipped to zero otherwise.

Noteworthy, without explicit reference to the soft thresholding operator, this has recently been proposed in [HSK+11, HKC13] as a simple, yet effective alternative to the rate distortion optimized quantization (RDOQ) method [KYC08, KCYJ09] which is implemented in the H.264/AVC reference encoder. The rate distortion gains are reportedly smaller than those of RDOQ (5.65 % bit rate reduction compared to 7.80 %, for H.264/AVC using CAVLC entropy coding), but the computational complexity, measured as an average increase of encoder runtime, is also lower (0.09 % for soft thresholding compared to 6.75 % for RDOQ).

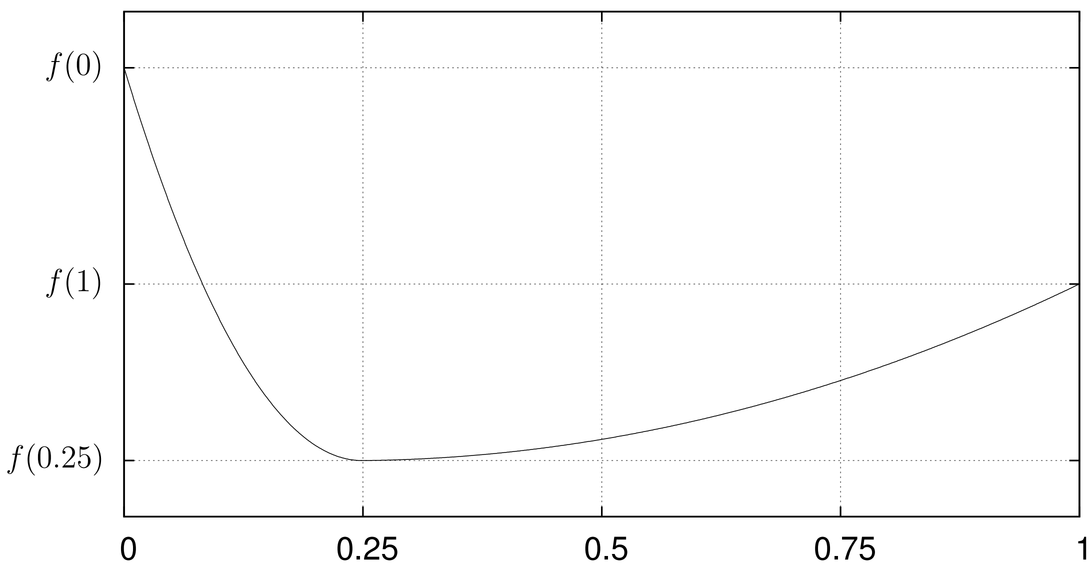


Figure 3.13: Example of a one-dimensional convex function where rounding does not lead to the optimal integer solution.

3.4.2 Orthogonal case: Optimal integer solution by rounding

In this section, it will be shown that in the separable setting of the previous section (i.e., diagonal $\mathbf{A}^T\mathbf{A}$), rounding of the optimal real-valued solution leads to the optimal integer-valued solution. Note that this is not a general property of a one-dimensional convex function, as illustrated in Fig. 3.13. In this example, the optimal real-valued solution is obtained at $x = 0.25$, which would be rounded to $x = 0$. But, since $f(1) < f(0)$, the optimal integer-valued solution is found at $x = 1$. It is true, however, that for one-dimensional convex functions with real-valued optimum at x, the optimal integer-valued solution must be either $\lfloor x \rfloor$ or $\lceil x \rceil$ [HS11].

In order to simplify matters, the following one-dimensional optimization problem is assumed:

$$c_{opt,int} = \arg\min_{c\in\mathbb{Z}} J(c) = (y-c)^2 + \mu\,|c| \tag{3.76}$$

The following inequality shows the condition for zero to be a better solution than one:

$$J(0) < J(1) \tag{3.77}$$

$$y^2 < y^2 + 1 - 2y + \mu \tag{3.78}$$

$$y < \frac{1}{2} + \frac{\mu}{2} \tag{3.79}$$

Note that for $y = {}^1\!/_2 + {}^\mu\!/_2$, there is a tie, i.e. both zero and one lead to the same value of J. More generally, $k \geq 0$ is a better solution than $k+1$, if

$$J(k) < J(k+1) \tag{3.80}$$

$$y^2 + k^2 - 2yk + \mu k < y^2 + (k+1)^2 - 2y(k+1) + \mu k + \mu \tag{3.81}$$

$$y < \frac{1}{2} + k + \frac{\mu}{2} \tag{3.82}$$

For $k \leq 0$, the condition for k to be better than $k-1$ is

$$J(k) < J(k-1) \tag{3.83}$$

$$y^2 + k^2 - 2yk - \mu k < y^2 + (k-1)^2 - 2y(k-1) - \mu k + \mu \tag{3.84}$$

$$y > -\frac{1}{2} + k - \frac{\mu}{2} \tag{3.85}$$

Consequently, the optimal integer-valued solution is obtained as follows (with $n \in \mathbb{N}^+ = \{1, 2, \ldots\}$):

$$c_{opt,int} \in \begin{cases} \{-n, -n-1\} & \text{if } y = -\frac{1}{2} - n - \frac{\mu}{2} \\ \{-n\} & \text{if } -\frac{1}{2} - n - \frac{\mu}{2} < y < -\frac{1}{2} - (n-1) - \frac{\mu}{2} \\ \vdots & \\ \{0\} & \text{if } -\frac{1}{2} - \frac{\mu}{2} < y < \frac{1}{2} + \frac{\mu}{2} \\ \{0, 1\} & \text{if } y = \frac{1}{2} + \frac{\mu}{2} \\ \{1\} & \text{if } \frac{1}{2} + \frac{\mu}{2} < y < \frac{1}{2} + 1 + \frac{\mu}{2} \\ \{1, 2\} & \text{if } y = \frac{1}{2} + 1 + \frac{\mu}{2} \\ \{2\} & \text{if } \frac{1}{2} + 1 + \frac{\mu}{2} < y < \frac{1}{2} + 2 + \frac{\mu}{2} \\ \vdots & \\ \{n\} & \text{if } \frac{1}{2} + (n-1) + \frac{\mu}{2} < y < \frac{1}{2} + n + \frac{\mu}{2} \\ \{n, n+1\} & \text{if } y = \frac{1}{2} + n + \frac{\mu}{2} \end{cases} \tag{3.86}$$

This can be written as:

$$c_{opt,int} \in \begin{cases} \{0\} & \text{if } |y| < \frac{1}{2} + \frac{\mu}{2} \\ \{\operatorname{sgn}(y) n\} & \text{if } \frac{1}{2} + (n-1) + \frac{\mu}{2} < |y| < \frac{1}{2} + n + \frac{\mu}{2} \\ \{\operatorname{sgn}(y) n, \operatorname{sgn}(y)(n+1)\} & \text{if } |y| = \frac{1}{2} + n + \frac{\mu}{2} \end{cases} \tag{3.87}$$

The tie can be broken as follows:

$$c_{opt,int} = \begin{cases} 0 & \text{if } |y| < \frac{1}{2} + \frac{\mu}{2} \\ \operatorname{sgn}(y) n & \text{if } \frac{1}{2} + (n-1) + \frac{\mu}{2} \le |y| < \frac{1}{2} + n + \frac{\mu}{2} \end{cases} \tag{3.88}$$

For $n \in \mathbb{N}^+ = \{1, 2, \ldots\}$ holds

$$\frac{1}{2} + (n-1) + \frac{\mu}{2} \le |y| < \frac{1}{2} + n + \frac{\mu}{2} \tag{3.89}$$

$$n \le |y| - \frac{\mu}{2} + \frac{1}{2} < n + 1 \tag{3.90}$$

$$n = \left\lfloor |y| - \frac{\mu}{2} + \frac{1}{2} \right\rfloor \tag{3.91}$$

Consequently Eq. 3.88 can be written as (using the soft thresholding operator of Eq. 3.74):

$$c_{opt,int} = \operatorname{sgn}(y) \max\left\{ \left\lfloor |y| - \frac{\mu}{2} + \frac{1}{2} \right\rfloor, 0 \right\} \tag{3.92}$$

$$= \operatorname{round}(\mathcal{S}_{\frac{\mu}{2}}(y)) \tag{3.93}$$

$$\text{with} \qquad \operatorname{round}(x) = \operatorname{sgn}(x) \left\lfloor |x| + \frac{1}{2} \right\rfloor \tag{3.94}$$

3.4.3 Quadratic Program approach

Schumitsch proposes to cast 3.18 in the form of a Quadratic Program (QP), which can then be solved by usage of a general-purpose numerical optimization toolbox (e.g., MOSEK). A QP is an optimization problem with a quadratic objective function and linear inequality and/or equality constraints. The generic form of a QP is as follows:

$$\begin{aligned} \min_{\mathbf{x}} \; & \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x} \\ \text{subject to } & \mathbf{B} \mathbf{x} \le \mathbf{u} \\ & \mathbf{E} \mathbf{x} = \mathbf{d} \end{aligned} \tag{3.95}$$

With the two helper variables $\mathbf{s}$, which is the vector of reconstructed sample values, and $\mathbf{k}$, where $k_i = |c_i|$, Schumitsch states the optimization problem in the following form ($\mathbf{1}$ represents the $K \times 1$ column vector consisting only of ones) [SSW04, SSW05]:

$$\begin{aligned} &\min_{\mathbf{s},\mathbf{c},\mathbf{k}} \frac{1}{2} \|\mathbf{y} - \mathbf{s}\|_2^2 + \mu\, \mathbf{1}^T \mathbf{k} \\ \text{subject to } &\mathbf{s} = \mathbf{p} + \mathbf{M}\,\mathbf{s} + \mathbf{T}\,\mathbf{c} \\ &\mathbf{k} \geq \mathbf{c} \\ &\mathbf{k} \geq -\mathbf{c} \\ &\mathbf{k} \geq 0 \end{aligned} \tag{3.96}$$

Schumitsch gives the following mapping, with the help of which 3.96 can be transformed into the standard form 3.95. ($\mathbf{I}$ is the $K \times K$ identity matrix):

$$\begin{aligned} \mathbf{x} &:= \begin{bmatrix} \mathbf{s} \\ \mathbf{c} \\ \mathbf{k} \end{bmatrix}, \quad \mathbf{H} := \begin{bmatrix} \mathbf{I} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{f} := \begin{bmatrix} -2\,\mathbf{y} \\ 0 \\ \mu\,\mathbf{1} \end{bmatrix} \\ \mathbf{B} &:= \begin{bmatrix} 0 & \mathbf{I} & -\mathbf{I} \\ 0 & -\mathbf{I} & -\mathbf{I} \\ 0 & 0 & -\mathbf{I} \end{bmatrix}, \quad \mathbf{u} := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{E} &:= \begin{bmatrix} \mathbf{I} - \mathbf{M} & -\mathbf{T} & 0 \end{bmatrix}, \quad \mathbf{d} := \mathbf{y} \end{aligned} \tag{3.97}$$

While this approach is guaranteed to find the solution of 3.18, there are two issues, making it inapplicable especially to high resolution video sequences. First, the matrix $\mathbf{E}$, which mainly consists of the two matrices $-\mathbf{T}$ and $\mathbf{I}-\mathbf{M}$, has to be stored explicitly in the random access memory, because it specifies the equality constraints of the QP. As stated in Sections 3.1.1 and 3.1.2, $\mathbf{T}$ and $\mathbf{M}$ are sparse matrices, where the maximum number of non-zero entries is limited. For the state of the art video coding standard H.265/HEVC, $\mathbf{T}$ can have up to $1024 \cdot K$ non-zero entries, where K is the number of luma samples under consideration of the current optimization. General-purpose numerical solver toolboxes typically operate using floating-point arithmetic of double precision [IEE08], which is a binary format where each variable occupies 64 bits. Thus, storing only the values of the non-zero entries of $\mathbf{T}$ for $N = 3$ frames at SDTV resolution of 720×576 luma samples could require up to $3 \cdot 720 \cdot 576 \cdot 1024 \cdot 8$ byte, which is more than 10 gigabyte. Similarly, storing the values of the non-zero entries of $\mathbf{M}$ could require up to $3 \cdot 720 \cdot 576 \cdot 128 \cdot 8$ byte, which is 1.3 gigabyte.

Note that the position of the non-zero entries has to be stored as well which further increases the memory requirements.

Furthermore, it is reported in the pertinent literature [BJMO12, pp. 38–40] that casting the optimization problem of Eq. 3.18 in the shape of a QP and applying a general-purpose optimization toolbox is generally inefficient, because "these toolboxes are generic and blind to problem structure and tend to be too slow, or cannot even run because of memory problems" [BJMO12, p. 40]. So, it is resumed that "these toolboxes are only adapted to small-scale problems and usually lead to solution with very high precision (low duality gap), while simpler iterative methods can be applied to large-scale problems but only leads to solution with low or medium precision, which is sufficient in most applications to machine learning" [BJMO12, p. 38]. Note that in the context of multi-frame transform coefficient optimization, high precision of the solution is not required as well. This is because finally sought are the integer-valued transform coefficients, which have to be derived (e.g. by rounding) from the real-valued solution of the optimization problem, and consequently a larger number of decimal places brings no benefit.

3.4.4 Interior-point method

In [KKL$^+$07] an algorithm based on the interior-point method for solving the optimization problem of Eq. 3.18 is proposed. This algorithm has also become known under the name l1_ls. Interior-point methods can be used to solve convex optimization problems having inequality constraints by solving a series of equality constrained problems instead. Details on interior-point methods can be found in [BV04, Ch. 11]. l1_ls is reportedly faster than usual interior-point methods [BBC11, p. 21 Sec. 5.1.4], but still every iteration step of the algorithm requires (approximate) solution of the linear equation system $\mathbf{H}\,\Delta\mathbf{x} = -\mathbf{g}$, where $\mathbf{H}$ is the Hessian and $\mathbf{g}$ is the gradient at the current iterate and the unknown $\Delta\mathbf{x}$ gives the search direction. Since computing the exact solution of this linear equation system would be impractical for large-scale optimization problems, only an approximate solution is computed, using only matrix-vector multiplications involving $\mathbf{A}$ and $\mathbf{A}^T$. Still, it is reported in [BBC11] that application of l1_ls is "problematic for large-scale problems" due to a large number of such multiplications. Like the Quadratic Program formulation, l1_ls delivers a high precision solution, which is not necessary in the context of this thesis.

3.4.5 Active Set/Homotopy method

Active set (or homotopy) methods rely on the piece-wise linearity of the regularization path. The algorithm starts at a known solution of the optimization problem (typically the zero vector), and then moves along the regularization path until the next breakpoint (also called *kink*) where either a previously zero coefficient becomes non-zero or vice versa. The value of the regularization parameter μ corresponding to the next kink can be derived in closed form. So, basically this type of algorithm starts from the trivial all-zero solution and then jumps from kink to kink along the regularization path, corresponding to a decreasing of the regularization parameter μ. The name active set method stems from the fact that at each stage of the algorithm there is a set of active (i.e., non-zero) variables. In the context of ℓ_1-regularized least squares optimization, active set algorithms have been proposed in [OPT00a, OPT00b, EHJT04]. These methods are particularly useful if one is interested in determining the whole regularization path or if the number of non-zero components of the desired solution vector $\mathbf{c}_{opt}$ is small. Since, within the scope of this thesis, one is only interested in the solution vector $\mathbf{c}_{opt}$ for one specific value of μ, this method is also not suitable for multi-frame transform coefficient optimization, because the number of kinks which would have to be considered in order to follow the regularization path from the zero vector to the solution vector for the desired value of μ would typically be too large.

3.4.6 Iterative Shrinkage/Thresholding Algorithm (ISTA)

An iterative shrinkage/thresholding algorithm (ISTA) for solving the optimization problem of Eq. 3.18 is proposed in [DDDM04]. A generalization of this method is *proximal forward-backward splitting* [CW05]. Sparse Reconstruction by Separable Approximation (SpaRSA) [WNF09] provides an algorithmic framework based on proximal forward-backward splitting. It deals with the minimization of a composite function as follows

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \mu\, g(\mathbf{x}), \tag{3.98}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth and convex function, and $g : \mathbb{R}^n \to \mathbb{R}$ is a regularization function that may be non-smooth. The forward-backward splitting algorithm generates a sequence $(\mathbf{x}_n)_{n\geq 0}$ which converges to $\mathbf{x}_{opt}$. The update step from the current iterate $\mathbf{x}_n$ to the next iterate $\mathbf{x}_{n+1}$ can easily be derived from the second order Taylor approximation of f around $\mathbf{x}_n$ [YSGM10]:

$$f(\mathbf{x}) \approx f(\mathbf{x}_n) + \nabla f(\mathbf{x}_n)^T(\mathbf{x}-\mathbf{x}_n) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_n)^T\nabla^2 f(\mathbf{x}_n)(\mathbf{x}-\mathbf{x}_n) \tag{3.99}$$

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \mu\, g(\mathbf{x}) \tag{3.100}$$

$$\approx \arg\min_{\mathbf{x}} \nabla f(\mathbf{x}_n)^T(\mathbf{x}-\mathbf{x}_n) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_n)^T\nabla^2 f(\mathbf{x}_n)(\mathbf{x}-\mathbf{x}_n) + \mu\, g(\mathbf{x}) \tag{3.101}$$

$$\approx \arg\min_{\mathbf{x}} \nabla f(\mathbf{x}_n)^T(\mathbf{x}-\mathbf{x}_n) + \frac{\alpha_n}{2}\|\mathbf{x}-\mathbf{x}_n\|_2^2 + \mu\, g(\mathbf{x}) =: \mathbf{x}_{n+1} \tag{3.102}$$

$$\mathbf{x}_{n+1} = \arg\min_{\mathbf{x}} \frac{1}{\alpha_n}\nabla f(\mathbf{x}_n)^T(\mathbf{x}-\mathbf{x}_n) + \frac{1}{2}\|\mathbf{x}-\mathbf{x}_n\|_2^2 + \frac{\mu}{\alpha_n} g(\mathbf{x}) \tag{3.103}$$

$$= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{u}_n\|_2^2 + \frac{\mu}{\alpha_n} g(\mathbf{x}) \tag{3.104}$$

with

$$\mathbf{u}_n := \mathbf{x}_n - \frac{1}{\alpha_n}\nabla f(\mathbf{x}_n) \tag{3.105}$$

Remarks: From Eq. 3.101 to Eq. 3.102, the Hessian $\nabla^2 f(\mathbf{x}_n)$ has been approximated by the diagonal matrix $\alpha_n \mathbf{I}$. In [BB88], it is proposed to choose α_n such that the approximation error of $\alpha_n(\mathbf{x}_n - \mathbf{x}_{n-1}) \approx \nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1})$ is minimized. This so-called *Barzilai-Borwein* step size rule has become a common feature of optimization algorithms based on ISTA, e.g. [WNF09, HYZ10]. Therefore, α_n is derived as:

$$\alpha_n = \arg\min_{\alpha} \|\alpha(\mathbf{x}_n - \mathbf{x}_{n-1}) - (\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1}))\|_2^2 \tag{3.106}$$

$$= \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})^T(\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1}))}{(\mathbf{x}_n - \mathbf{x}_{n-1})^T(\mathbf{x}_n - \mathbf{x}_{n-1})} \tag{3.107}$$

Note, that for the first value α_0 some initialization, e.g. $\alpha_0 = 1$, is needed. The step from Eq. 3.103 to Eq. 3.104 can easily be verified by plugging the definition of $\mathbf{u}_n$ from Eq. 3.105 into Eq. 3.104.

In [Mor62], the *proximity operator* $\operatorname{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ is introduced as follows:

$$\operatorname{prox}_f \mathbf{x} = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + f(\mathbf{y}) \tag{3.108}$$

It can be viewed as a generalization of the Euclidean projection operator. With the help of the proximity operator, Eq. 3.104 can be written as:

$$\mathbf{x}_{n+1} = \operatorname{prox}_{\frac{\mu}{\alpha_n} g} \mathbf{u}_n \tag{3.109}$$

$$= \underbrace{\operatorname{prox}_{\frac{\mu}{\alpha_n} g}}_{\text{backward step}} \left(\underbrace{\mathbf{x}_n - \frac{1}{\alpha_n} \nabla f(\mathbf{x}_n)}_{\text{forward step}} \right) \tag{3.110}$$

According to Eq. 3.18, $\mathbf{x} = \mathbf{c}$, $f(\mathbf{c}) = \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2$, and $g(\mathbf{c}) = \|\mathbf{c}\|_1$. In Section 3.4.1, it has implicitly already been shown, that the proximity operator of the ℓ_1-norm is the elementwise soft thresholding operator:

$$\operatorname{prox}_{\mu\|\cdot\|_1} \mathbf{x} = \mathcal{S}_\mu(\mathbf{x}) \tag{3.111}$$

This can easily be verified if $\mathbf{A} = \mathbf{I}$ is assumed in Eq. 3.52.

The gradient $\nabla f(\mathbf{c})$ for $f(\mathbf{c}) = \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2$ is

$$\nabla f(\mathbf{c}) = 2\mathbf{A}^T(\mathbf{A}\mathbf{c} - \tilde{\mathbf{y}}). \tag{3.112}$$

Consequently, the iteration for solving 3.18 is (as shown in [DDDM04, p. 1429, Eq. 2.11])

$$\mathbf{c}_{n+1} = \mathcal{S}_{\frac{\mu}{\alpha_n}} \left(\mathbf{c}_n - \frac{2}{\alpha_n} \mathbf{A}^T(\mathbf{A}\mathbf{c}_n - \tilde{\mathbf{y}}) \right). \tag{3.113}$$

Convergence of $(\mathbf{c}_n)_{n\geq 0}$ to $\mathbf{c}_{opt}$ can be shown [CW05] for $\alpha_n > \|\mathbf{A}^T\,\mathbf{A}\|_2$, where $\|\cdot\|_2$ is the spectral norm of the matrix. When using a fixed value of α_n satisfying this condition, the objective function $J(\mathbf{c}_n)$ is guaranteed to decrease in each iteration. Using the Barzilai-Borwein step size from Eq. 3.107 leads to a non-monotone behaviour, where the objective function is allowed to increase after an iteration. This generally accelerates convergence. But, in order to guarantee convergence of the iterative algorithm, a line search has to be employed [WNF09]. Starting from the current iterate $\mathbf{c}_n$ and α_n, the next candidate $\mathbf{c}_{n+1}$ is computed according to Eq. 3.113. Then, it is checked whether $\mathbf{c}_{n+1}$ fulfills a given acceptance criterion. If this is not the case, α_n is increased by a factor of two (and the resulting gradient step size consequently halved), and a new candidate $\mathbf{c}_{n+1}$ is tested. This process is repeated until $\mathbf{c}_{n+1}$ fulfills the acceptance criterion. In this thesis, the non-monotone acceptance criterion from

[WNF09, p. 2483, Eq. 22] is used with $M = 5$, i.e. $\mathbf{c}_{n+1}$ is accepted if the resulting value of the objective function is slightly smaller than the largest value over the past six iterations. More specifically, $\mathbf{c}_{n+1}$ is accepted if

$$J(\mathbf{c}_{n+1}) \leq \max_{i=\max(n-M,0),\dots,n} J(\mathbf{c}_i) - \frac{\sigma}{2}\alpha_n \|\mathbf{c}_{n+1} - \mathbf{c}_n\|_2^2 \tag{3.114}$$

with $J(\mathbf{c})$ as defined in Eq. 3.10 and $\sigma = 0.01$ (as proposed in [WNF09]).

A very important aspect for the effective performance of an iterative optimization algorithm is the choice of a proper termination criterion which decides at which iteration n to terminate and to assume that $\mathbf{c}_n \approx \mathbf{c}_{opt}$. In [HYZ10, p. 181, Sec. 4.2.1], a combination of two termination criteria is proposed. The first criterion checks whether the relative size of the last step is smaller than some tolerance *xtol*:

$$\frac{\|\mathbf{c}_n - \mathbf{c}_{n-1}\|_2}{\|\mathbf{c}_{n-1}\|_2} \leq xtol \tag{3.115}$$

The second criterion is based on the following necessary and sufficient condition for $\mathbf{c}$ to be optimal, which is obtained in [KKL+07, Sec. III-A]:

$$(\mathbf{A}^T(\tilde{\mathbf{y}} - \mathbf{A}\mathbf{c}))_i \in \begin{cases} \{-\mu\} & \text{if } c_i < 0 \\ [-\mu, \mu] & \text{if } c_i = 0 \\ \{\mu\} & \text{if } c_i > 0 \end{cases} \tag{3.116}$$

This condition implies that:

$$\|\mathbf{A}^T(\tilde{\mathbf{y}} - \mathbf{A}\mathbf{c}-)\|_\infty \leq \mu \tag{3.117}$$

The second termination criterion proposed in [HYZ10, p. 181, Sec. 4.2.1] checks if Eq. 3.117 is fulfilled up to a given tolerance *gtol*:

$$\frac{\|\mathbf{A}^T(\tilde{\mathbf{y}} - \mathbf{A}\mathbf{c})\|_\infty}{\mu} - 1 \leq gtol \tag{3.118}$$

Unless explicitly stated otherwise, in this thesis it is assumed that the iterative process is stopped when either both criteria 3.115 and 3.118 are simultaneously fulfilled or when a maximum number of iterations n_{max} is reached. The resulting number of iterations highly depends on the choice of the parameters *xtol*, *gtol*, and n_{max}. In [WNF09, p. 2484] it is claimed that “it is usually (and perhaps inevitably) left to the user to tune the stopping criteria in their codes to the needs of their application" and that "the choice of stopping criteria can dramatically affect the performance of many optimization algorithms in practice." Therefore, the actual values for these three parameters will be empirically determined from the outcome of several coding experiments. This is investigated in a later chapter of this thesis.

Similarly as the termination criterion, the choice of the starting point, i.e. how to choose $\mathbf{c}_0$, has large impact on the practical performance. Since the iteration 3.113 will converge to $\mathbf{c}_{opt}$ for any $\mathbf{c}_0$, in the simplest case $\mathbf{c}_0 = \mathbf{0}$ could be used. A good starting point $\mathbf{c}_0$, however, will reduce the number of required iteration steps. In the literature [HYZ08, WNF09], a so-called continuation strategy is described, where instead of solving 3.18 directly for a given value of μ, a series of sub-problems is solved. Each sub-problem is of the same type as Eq. 3.18, but uses a different value of μ. The first sub-problem starts with a large value of μ and a simple initialization such as $\mathbf{c}_0 = \mathbf{0}$. This will quickly converge, because, due to the large value of μ, many components of the solution vector will remain zero. In the next step, the value of μ is reduced and the previous solution is used as the new starting point $\mathbf{c}_0$ (the subsequent step "continues" where the previous step stopped). This is repeated until the final value of μ is reached. Note, that this is similar in spirit to the homotopy method (Sec. 3.4.5), where starting from the all-zero solution the regularization path is explored. In the context of multi-frame transform coefficient optimization, however, using this continuation strategy is inadvisable due to two aspects. First of all, the transform coefficients that are obtained by ordinary forward transform followed by scalar quantization can already serve as a good warm start initialization. Secondly would this warm start initialization be harmed by the continuation strategy, because the large initial value of μ will force many components of the vector to zero, which then would have to be recovered again in subsequent steps of the continuation method. Consequently, no continuation strategy is employed, and the optimization problem Eq. 3.18 is directly solved using the desired value of μ, with the transform coefficient levels as obtained by ordinary forward transform/scalar quantization as a warm start initialization for $\mathbf{c}_0$.

The iteration rule 3.113 has several properties making it particularly suited to the multi-frame transform coefficient optimization problem. It is mainly composed of very simple operations which can be performed with little demands in terms of memory requirements and computational complexity. As stated in Sec. 3.2.1, the matrix $\mathbf{A}$ is the reconstruction operator, i.e. inverse transform followed by motion-compensated prediction, which has been written in the shape of one single matrix comprising both operations in order to state the optimization problem in standard form. With $\mathbf{r}_n = \mathbf{T}\,\mathbf{c}_n$, the matrix-vector product $\mathbf{A}\,\mathbf{c}_n$ can be written as:

$$\begin{aligned}
\tilde{\mathbf{s}}_n &= \mathbf{A}\,\mathbf{c}_n && (3.119)\\
&= \sum_{\nu=0}^{\nu_{max}} \mathbf{M}^{\nu}\,\mathbf{T}\,\mathbf{c}_n && (3.120)\\
&= \sum_{\nu=0}^{\nu_{max}} \mathbf{M}^{\nu}\,\mathbf{r}_n && (3.121)\\
&= \mathbf{r}_n + \mathbf{M}\,\mathbf{r}_n + \mathbf{M}^2\,\mathbf{r}_n + \ldots + \mathbf{M}^{\nu_{max}}\,\mathbf{r}_n && (3.122)
\end{aligned}$$

Furthermore, the matrix-vector products $\mathbf{T}\,\mathbf{c}_n$ and $\mathbf{M}^{\nu}\,\mathbf{r}_n$ can be performed in separable form, since both inverse transform as well as interpolation filtering for motion-compensated prediction are separable operations in state of the art video coding standards. This further reduces the number of elementary multiplication/addition operations. For example, for a 32×32 transform block, if separability is not considered, each transform coefficient has to be multiplied by all 1024 samples of its corresponding basis image. If separability is exploited, there are 32 multiplications horizontally and vertically each, summing up to 64 multiplications total. Thus, the number of multiplications has been reduced by a factor of 16. More than that, the matrix $\mathbf{T}$, whose memory requirements can become very huge as has been argued in Sec. 3.4.3, does not need to be directly stored in the random access memory. Instead, since the basis functions of the transform are fixed for a video coding standard, it can implicitly be derived on the fly. For this purpose, only the transform size (which selects the number and set of basis functions) and the quantization parameter (which selects the scaling factor) have to be stored for each transform block, which leads to very modest memory requirements. For the example of Sec. 3.4.3 with $N = 3$ frames at SDTV resolution of 720×576, storing the transform size and the quantization parameter using one byte each for every 4×4 block, would require $3 \cdot \frac{720 \cdot 576}{4 \cdot 4} \cdot 2$ byte, which is 156 kilobyte[1] (whereas, as shown, storing $\mathbf{T}$ directly could take more than 10 gigabyte, even if its sparsity is exploited).

The matrix-vector products $\mathbf{M}\,\mathbf{r}_n$, $\mathbf{M}\,(\mathbf{M}\,\mathbf{r}_n)$, etc. can also be computed in separable form, because all state of the art video coding standards use separable sub-pel

[1]Note that for H.265/HEVC, there are four different transform sizes and the quantization parameter may range from 0 to 51, such that both values might even be stored together in one byte, resulting in 78 kilobyte total.

$$Frame_0 = \boxed{\begin{matrix} A & B \\ C & D \end{matrix}} \qquad Frame_1 = \boxed{\begin{matrix} E & F \\ G & H \end{matrix}} \qquad MV = \boxed{\begin{matrix} (1/2, 1/2) & (-1,0) \\ (1/2, 0) & (-1,0) \end{matrix}}$$

Figure 3.14: Illustrating example using two 2×2 frames and half-pel interpolation

interpolation filters.

Using the vector $\tilde{\mathbf{s}}_n$, the term $\mathbf{A}\,\mathbf{c}_n - \tilde{\mathbf{y}}$ of Eq. 3.113 can be stated as $\mathbf{d}_n := \tilde{\mathbf{s}}_n - \tilde{\mathbf{y}}$. It represents the reconstruction error (difference signal between reconstruction and original) in spatial domain for the current iterate $\mathbf{c}_n$. The direction of the gradient step is then $\mathbf{g}_n := \mathbf{A}^T\,\mathbf{d}_n$. Like for the reconstruction $\mathbf{A}\,\mathbf{c}_n$, it can be written as:

$$\mathbf{g}_n = \mathbf{A}^T\,\mathbf{d}_n \tag{3.123}$$

$$= \mathbf{T}^T \sum_{\nu=0}^{\nu_{max}} (\mathbf{M}^T)^\nu\,\mathbf{d}_n \tag{3.124}$$

$$= \mathbf{T}^T\,\underbrace{(\mathbf{d}_n + \mathbf{M}^T\,\mathbf{d}_n + (\mathbf{M}^T)^2\,\mathbf{d}_n + \ldots + (\mathbf{M}^T)^{\nu_{max}}\,\mathbf{d}_n)}_{\mathbf{z}_n} \tag{3.125}$$

$$= \mathbf{T}^T\,\mathbf{z}_n \tag{3.126}$$

The matrix-vector product $\mathbf{T}^T\,\mathbf{z}_n$ can again be computed in separable form. This is possible, because $\mathbf{T}$ is a block-diagonal matrix, and the transpose $\mathbf{T}^T$ is obtained as the transpose of the individual constituent sub-matrices. For the matrix-vector products $\mathbf{M}^T\,\mathbf{d}_n$, $\mathbf{M}^T(\mathbf{M}^T\,\mathbf{d}_n)$, etc., however, this is not possible, as will be shown with the help of a very simple example. In this example, a group of $N = 2$ frames is assumed, each having a size of 2×2 samples. The individual samples of these frames are designated with the letters A,...,H, as shown in Fig. 3.14. Further it is assumed, that $Frame_1$ is predicted from $Frame_0$ using bilinear sub-pel interpolation filtering with the motion vector field MV of Fig. 3.14. Then, the prediction samples $\hat{E}$,...,$\hat{H}$ are obtained as follows:

$$\hat{E} = \frac{A + B + C + D}{4} \tag{3.127}$$

$$\hat{F} = A \tag{3.128}$$

$$\hat{G} = \frac{C + D}{2} \tag{3.129}$$

$$\hat{H} = C \tag{3.130}$$

This leads to the following matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{cccc|cccc} A & B & C & D & E & F & G & H \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \tag{3.131}$$

It can easily be verified that a matrix-vector product $\mathbf{M}\,\mathbf{x}$ using this definition of $\mathbf{M}$ can be computed in separable form. For that purpose, the 2D interpolation filter masks for the prediction samples $\hat{E}, \ldots, \hat{H}$ are derived as follows:

$$\mathbf{P}_E = \begin{bmatrix} m_{E,A} & m_{E,B} \\ m_{E,C} & m_{E,D} \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \end{bmatrix} \tag{3.132}$$

$$\mathbf{P}_F = \begin{bmatrix} m_{F,A} & m_{F,B} \\ m_{F,C} & m_{F,D} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.133}$$

$$\mathbf{P}_G = \begin{bmatrix} m_{G,A} & m_{G,B} \\ m_{G,C} & m_{G,D} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \end{bmatrix} \tag{3.134}$$

$$\mathbf{P}_H = \begin{bmatrix} m_{H,A} & m_{H,B} \\ m_{H,C} & m_{H,D} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.135}$$

Note that only the entries of the lower-left quadrant of $\mathbf{M}$ are considered here, as all the other entries are zero by definition, which is because a predicted frame can neither depend on itself nor on future frames. It is shown that all those 2D filter masks can be written as the outer product of two vectors. This allows separable computation of the interpolation results.

In the following, the transposed matrix $\mathbf{M}^T$ is considered. Here, the 2D filter mask corresponding to row C of the matrix $\mathbf{M}^T$ (which is column C of the matrix $\mathbf{M}$) is:

$$\mathbf{R}_C = \begin{bmatrix} m_{E,C} & m_{F,C} \\ m_{G,C} & m_{H,C} \end{bmatrix} = \begin{bmatrix} 1/4 & 0 \\ 1/2 & 1 \end{bmatrix} \tag{3.136}$$

Note that since $\mathbf{R}_C$ has rank two, it is impossible to write $\mathbf{R}_C$ as the outer product of two vectors. Consequently, for the matrix-vector product $\mathbf{M}^T \mathbf{x}$, the component C of the resulting vector cannot be computed in separable form. So, even when a separable sub-pel interpolation filter is used (like in all state of the art video coding standards) and the matrix-vector product $\mathbf{M}\,\mathbf{x}$ therefore can efficiently be computed in separable fashion, this is not possible for the transpose $\mathbf{M}^T$.

More than that, the non-separability of $\mathbf{M}^T$ is not caused by the sub-pel interpolation filter, as can be seen if only full-pel motion-compensation is assumed as in the following example:

$$\hat{E} = A, \qquad \hat{F} = A, \qquad \hat{G} = B, \qquad \hat{H} = A \tag{3.137}$$

Which leads to the following matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{array}{c|cccc|cccc|}
 & A & B & C & D & E & F & G & H \\
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
E & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
H & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array} \tag{3.138}$$

Here, the 2D filter mask corresponding to row A of the matrix $\mathbf{M}^T$ (which is column A of $\mathbf{M}$) is:

$$\mathbf{R}_A = \begin{bmatrix} m_{E,A} & m_{F,A} \\ m_{G,A} & m_{H,A} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{3.139}$$

Again, this matrix has rank two, such that the matrix-vector product $\mathbf{M}^T \mathbf{x}$ cannot be computed in separable form, even if no sub-pel interpolation filtering is used at all.

The iteration rule 3.113 will now be analyzed according to its complexity:

$$\mathbf{c}_{n+1} = \mathcal{S}_{\frac{\mu}{\alpha_n}} \left(\mathbf{c}_n - \frac{2}{\alpha_n} \mathbf{A}^T (\mathbf{A}\mathbf{c}_n - \tilde{\mathbf{y}}) \right) \tag{3.113 revisited}$$

The matrix-vector product $\mathbf{A}\,\mathbf{c}_n$ equals to reconstructing the current N video frames using the transform coefficient vector $\mathbf{c}_n$. It can be computed in separable fashion and its complexity is lower than that of decoding the current N frames, since none of entropy decoding, motion vector prediction/decoding, loop filtering has to be performed. The matrix multiplication by the transpose $\mathbf{A}^T$ can be split into two parts, as shown in Eq. 3.124. The first part, the multiplication by the transposed motion matrix $\mathbf{M}^T$, as has been argued, cannot be split into two separable steps and therefore is the crux of the whole iterative process. However, since $\mathbf{M}$ is a sparse matrix and the number of non-zero entries is limited to be not greater than $128 \cdot K$ for H.265/HEVC (see Sec. 3.1.2), the complexity of the non-separable matrix-vector multiplication by $\mathbf{M}^T$ is still of order $\mathcal{O}(K)$, i.e. the number of additions and multiplications grows linearly with the number of samples K. It can be reasoned that the multiplication by $\mathbf{M}^T$ is also the key part of the whole multi-frame transform coefficient optimization, since it is in this step where the inter-frame dependencies are actually taken into account. The matrix-vector product $\sum_{\nu=0}^{\nu_{max}} (\mathbf{M}^T)^\nu \, \mathbf{d}_n$ can be interpreted such that from the current difference signal $\mathbf{d}_n$ a new vector is derived, which gives the inverse of the direction (in spatial domain) into which to move from the current iterate $\mathbf{c}_n$ such that the distortion of the reconstruction is reduced. Since this direction is given in spatial domain, it has to be multiplied by $\mathbf{T}^T$ in order to obtain the differential update for the current iterate $\mathbf{c}_n$ in transform domain. This multiplication again can be computed separably and therefore allows fast implementation. The elementwise soft thresholding operator $\mathcal{S}_{\frac{\mu}{\alpha n}}$, finally, is a very cheap and simple operation, which could even be executed in parallel on all the components, and therefore contributes little to the overall complexity of the iteration.

It can be concluded, that for multi-frame transform coefficient optimization, an iterative shrinkage/thresholding algorithm based on the Sparse Reconstruction by Separable Approximation (SpaRSA) framework [WNF09] has the following advantages over the methods described in the previous sections:

- The iteration 3.113 consists of relatively simple operations, which can to a large extent even be performed separably in horizontal and vertical direction. This enables application to large-scale optimization problems with a problem size (dimension) of 10^6 and beyond.

- The sparse matrices $\mathbf{M}$ and $\mathbf{T}$ do not need to be directly stored in the random access memory, but instead they may be derived on the fly. This markedly reduces memory requirements compared to the Quadratic Program approach (Sec. 3.4.3).

- Warm start initialization is possible using the transform coefficient vector obtained by ordinary forward transform and scalar quantization.

- The forward-backward splitting approach inherently allows to exchange the ℓ_1-norm by some other regularizer, if its proximity operator is known. This is studied in a later chapter, where the performance of different regularization functions is compared by the resulting rate distortion performance.

4 Application of the optimization method to H.265/HEVC

In this chapter, the previously described multi-frame transform coefficient optimization algorithm is applied to the recent H.265/HEVC video coding standard. Some HEVC-specific issues are discussed, the influence of various optimization parameters is analyzed, and the resulting rate distortion performance is presented.

Unless explicitly otherwise stated, the test sequences and encoder settings as used by the ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) for the development of the H.265/HEVC video coding standard are used, both for determining optimal model parameters and for evaluating the performance of the method. Since this is somewhat problematic from a scientific point of view, as training set and test set are identical, the coding results are also presented using different test sequences with all model parameters fixed as empirically derived based on the JCT-VC test sequences, such that training set and test set are disjunct. The complete description of the JCT-VC encoder settings and test sequences, the so-called "common test conditions," can be found in [JCT13]. In the following, a short overview over the different encoder configurations as defined in the common test conditions is given. There are four basic configurations:

- Intra
- Random access
- Low delay using B frames
- Low delay using P frames

Each of these four configurations exists in an 8 bit variant ("main") and a 10 bit variant ("high efficiency"). The first configuration, which uses intra only, is out of scope for this thesis, since there is no inter-frame prediction and, consequently, no inter-frame dependencies. The last two configurations are also inappropriate for this work, because the computational burden required to perform the multi-frame optimization as well as the structural delay caused by the multi-frame look-ahead exclude low delay coding scenarios (such as video conferencing or screen sharing).

So, from a use case perspective, the focus will be on the random access coding scenario. Still, in the following, the optimization method is first studied using a simple IPPP... prediction structure using one reference frame, i.e. each P frame references its directly preceding frame, and infinite intra period, i.e. only the first frame is coded as an I frame, all following frames as P frames. All frames are encoded using the same quantization parameter (QP) value. The other encoder settings are kept as in the random access configuration defined in the common test conditions. This configuration is studied for the following reasons. First of all, since this is the most simple prediction structure, it can easily be studied whether and under which conditions the multi-frame optimization shows coding gains at all before proceeding to more complicated prediction structures. Furthermore, in this prediction structure, there is no structural difference between the individual frames like in the "GOP8" prediction structure used for the random access configuration, where there is a distinction between reference and non-reference frames, and some reference frames have direct impact on a larger number referring frames than others. According to the common test conditions, this hierarchy of frames within the GOP8 prediction structure is captured by a modulation of both quantization parameter and Lagrangian multiplier used for the rate distortion optimized operational encoder control. In the multi-frame transform coefficient optimization problem, two parameters have to be appropriately chosen, namely the number of frames to consider in the joint optimization N and the regularization parameter μ. In a prediction structure like GOP8, it can be expected that N and μ should by varied based on the frame hierarchy in order to achieve optimal performance. This makes the problem of determining the right values for N and μ a complicated task. Thus, the multi-frame transform coefficient optimization problem is first studied in a simple setup where there is no such distinction between the individual frames, and the optimal values for N and μ therefore can be determined globally for the whole sequence. Based on the insights derived from the analysis of this simple IPPP... prediction structure, the behaviour of the multi-frame optimization for the GOP8 prediction structure is studied.

4.1 Investigation of the optimization algorithm parameters

4.1.1 Description of the algorithm

The general structure of the algorithm follows closely the one proposed by Schumitsch in [SSW05, Sec. 4.2]. The idea is to optimize the transform coefficients of the inter-predicted blocks such that their impact on future frames is taken into account. The intra-predicted blocks are excluded from the optimization for the following reasons:

- As shown in Sec. 3.1.3 (Eq. 3.9, p. 27) as well as Sec. 3.4.6 (Eqs. 3.119–3.122, p. 65, and Eqs. 3.123–3.125, p. 66) the computational complexity of the iterate shrinkage/thresholding algorithm (ISTA) is to a large degree governed by the length of the longest prediction chain ν_{max}. If a group of N frames is jointly optimized and only inter-predicted blocks are considered, ν_{max} is limited to be no larger than $N-1$. If intra-predicted blocks are also included in the optimization, ν_{max} can become much larger due to dependencies from one block to another within one frame which are caused by intra-prediction. This will increase the complexity of each iteration of ISTA.

- The accuracy of the linear signal model in Eq. 3.9 decreases with a larger number of frames, because the impact of the non-linear effects (rounding, clipping, filtering), which are neglected in this model, accumulates over time. The same, but more severely, can be expected if intra-prediction is also considered, since a longer prediction chain, caused by block-to-block intra-frame dependencies, will lead to an even larger model inaccuracy.

The algorithm operates in a sliding window manner, i.e. a group of N frames is considered jointly, the transform coefficients for the first frame of this group are optimized, and then the optimization windows moves one step ahead, such that the next N frames are jointly optimized. Since the inter-frame dependencies are assumed to be known for the group of N frames, a look-ahead encoding has to be performed in order to determine the corresponding motion vectors. So, for the group of N frames, initially a pre-encoding is performed, such that the motion vectors are fixed. Then, the ℓ_1-regularized optimization problem 3.18 is solved and the transform coefficients of the first frame are updated by the solution of the optimization problem.

Given a regularization parameter μ and a number of frames to optimize jointly N, the optimization algorithm for HEVC proceeds as follows.

1. Since I slices are not included in the optimization, the first frame 0 is ordinarily encoded using HM and a variable M is set equal to 1.

2. The frames M, $M+1$, …, $M+N-1$ are ordinarily encoded using HM such that their block partitioning, prediction modes, motion vectors etc. are known.

3. The multi-frame transform coefficient problem 3.18 is solved for these frames using the iterative shrinkage/thresholding algorithm (ISTA), given the inter-frame dependencies from the previous step, represented in the matrix $\mathbf{A}$, and the regularization parameter μ. The iteration is initialized using the transform coefficients obtained from the HM encoder as a warm start.

4. An integer approximation for the real-valued elements of the solution vector $\mathbf{c}_{opt}$ has to be found. Note that simple rounding may not be a feasible option here, since in the common test conditions a coding tool named *sign data hiding (SDH)* [WYH+12] is used, where for a group of 16 transform coefficients, the sign of one of these coefficients is coded using the parity of the sum of the absolute values of all the 16 coefficients. Obviously, this limits the freedom to choose an arbitrary combination of transform coefficients, and therefore has to be considered. If the sum parity of the coefficients obtained by simple rounding does not match the one required for SDH, the value of one coefficient has to be modified. For this purpose, the coefficient is chosen, where this modification leads to the smallest deviation from the real-valued coefficient as obtained from the solution algorithm.

5. The integer-valued coefficients obtained in the previous step are used for the inter-coded blocks of frame M, the coefficients of the intra-coded blocks are re-estimated, since they depend on the reconstructed samples of neighboring blocks which might have changed.

6. $M := M + 1$ and execution continues at step 2.

The motivation why a temporal sliding window approach is taken can be seen from Fig. 4.1. In this diagram, the luma peak signal to noise ratio (Y-PSNR) is shown for the first 14 frames of the BlowingBubbles sequences at 416×240 resolution, which has been encoded using the simple IPPP… prediction structure with a fixed quantization parameter (QP) setting of $QP = 30$. The first frame numbered as zero is encoded as an I frame, all others as P frames. In this diagram, two curves are shown. For both curves, the transform coefficients of frames 1–10 have been derived using the above described sliding window method, where $N = 4$ frames have been considered jointly. For the frames 11–13, the red curve denoted as "sliding window stopped at frame 10" shows the Y-PSNR which results from optimizing the four frames 10–13 jointly in one step, where the green curve shows the performance when all the frames are consecutively optimized in a temporal sliding window operation. It can be seen from the red curve that there is a significant PSNR drop from frame 10 to frame 13. This occurs, because improving the reconstruction quality of frame 10 also benefits the subsequent frames 11–13, since they are affected by frame 10 via motion-compensated prediction, whereas improving the PSNR of frame 13 will only benefit this frame itself. Similarly, improving the quality of frame 11 still additionally also benefits frames 12–13. Consequently, the multi-frame optimization problem will lead to a solution, where there are more non-zero coefficients in the first frame under consideration than in the last frame, shifting more bit rate to the beginning of the group of frames under consideration, which results in the unbalanced reconstruction quality. Since this behaviour is obviously unwanted, a temporal sliding window approach is employed, such that each frame will be encoded at approximately the same quality, where the small fluctuations of the PSNR are only caused by the characteristics of the individual sequence.

4.1.2 Exploration of the regularization path

In order to evaluate the multi-frame optimization approach, coding experiments using the eight test sequences at 416×240 ("Class D") and 832×480 ("Class C") resolution have been conducted. As prescribed in the HEVC common test conditions, quantization parameter (QP) values of 22, 27, 32, 37 have been used. For every sequence and QP value, the multi-frame optimization problem has been solved for each $N \in \{2, 3, 4\}$ and the regularization parameter μ has been swept in 26 logarithmic steps from 0.05 to 30 000, such that the regularization path, as introduced in Sec. 3.3.1, is explored in

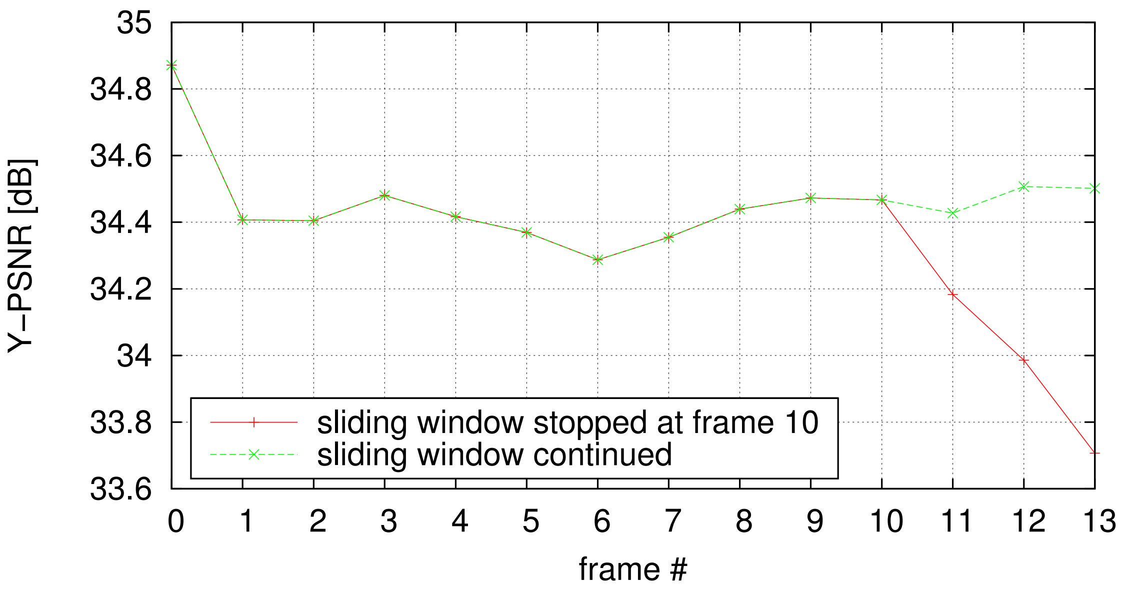


Figure 4.1: PSNR fluctuation of the first 14 frames of the BlowingBubbles sequence at 416×240 resolution, encoded using IPPP... prediction structure with $QP = 30$.

the rate distortion domain. The optimization is performed using the iterative shrinkage/thresholding algorithm (ISTA) (see Sec. 3.4.6) with a maximum number of 500 iterations, $xtol = 10^{-3}$ (see Eq. 3.115), and $gtol = 0.9$ (see Eq. 3.118). This results in an earlier termination of the algorithm than using the values of maximum 1000 iterations, $xtol = 10^{-4}$, and $gtol = 0.2$, which are proposed in [HYZ10]. However, no impact on the coding efficiency has been observed. The influence of limiting the maximum number of iterations is studied in Sec. 4.1.4. So for every sequence and QP value, $3 \times 26 = 78$ rate distortion points have been generated using the multi-frame optimization approach. The corresponding rate distortion plots are shown in Figs. 4.2–4.5, Fig. 4.6 shows the detailed view for two exemplary cases. Since the coding of the chroma color planes has not been modified, only the peak signal to noise ratio (PSNR) for the luma component (Y) is presented. Note that, as proposed by Bjøntegaard in [Bjø01], a logarithmic scale has been used for the bit rate on the x-axis. Although this presentation format is not in widespread use, it has the following advantages:

- The behaviour at lower bit rates can easier be read. With a linear scale for the bit rate, the curves in the lower bit rate region are kind of squeezed together, whereas the higher bit rate region takes a lot more space.

- In video compression, one is typically more interested in the *relative* reduction of the bit rate (i.e., method A reduces the bitstream size at the same quality by X %) than in the *absolute* reduction of the bit rate (i.e., method A reduces the bitstream size at the same quality by X kbit/s). Using a logarithmic axis for the bit rate makes it easier to read the relative bit rate reduction. Note that stepping one tick left on the x-axis of Figs. 4.2–4.5 corresponds to a bit rate reduction by $1 - 2^{-1/5} \approx 13\,\%$, whereas stepping one tick right corresponds to a bit rate increase by $2^{1/5} - 1 \approx 15\,\%$.

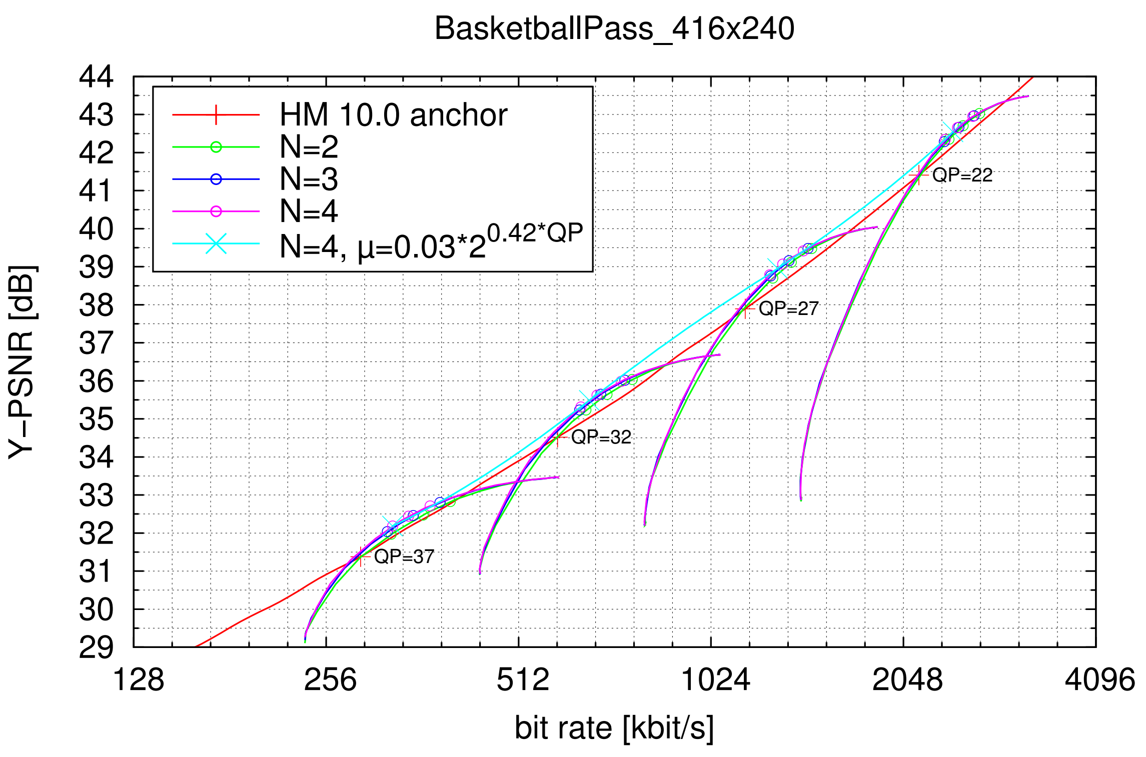
The operating points of the HM 10.0 reference encoder at the QP values of 22, 27, 32, 37 are marked by a "+" symbol in the diagrams. For each QP value, there are three curves showing the performance of the multi-frame optimization approach, one for each $N \in \{2, 3, 4\}$. The points along each of these three curves are obtained by sweeping the regularization parameter μ in 26 logarithmic steps from 0.05 to 30 000, where the lower endpoint (low bit rate, low PSNR) corresponds to $\mu = 30\,000$ and the higher endpoint (high bit rate, high PSNR) corresponds to $\mu = 0.05$. Finally, there is one curve for $N = 4$, where the regularization parameter has been chosen according to the QP-dependent rule $\mu = 0.03 \cdot 2^{0.42 \cdot QP}$ and the maximum number of iterations is limited to 50 (instead of 500). The method how this rule has been derived, as well as the meaning of the three operating points on each of the curves for $N \in \{2, 3, 4\}$ which are marked by hollow circles, will be explained in the Sec. 4.1.3.

A few things can be observed from these diagrams. Firstly, in most cases, there is a range of μ values, such that the resulting operating points of the multi-frame optimization method lie above the HM anchor curve, or in other words, there is a coding gain. Values of the regularization parameter μ outside this range lead to operating points below the HM anchor curve, indicating a coding loss. Furthermore, within the region where there is a coding gain, a higher value of N (i.e., a larger number of frames considered jointly in the optimization step) results in a higher coding gain. Finally, the starting and end points of the curves are independent of the value of N, i.e. they are the same for all the three curves. The last observation can be explained by the fact that for large μ, the optimization problem 3.18 will lead to $\mathbf{c}_{opt} = 0$. In the opposite case, for $\mu = 0$, the solution will be the same as the one obtained by ordinary forward transform, i.e. $\mathbf{c}_{opt} = \mathbf{A}^{-1}\,\tilde{\mathbf{y}}$. In both cases, the resulting transform coefficient vector $\mathbf{c}_{opt}$ is independent of the number of frames N. The observation that the end point for $\mu = 0.05 \approx 0$, which leads to the ordinary forward transform solution, lies significantly below the HM anchor curve, even though HM employs ordinary forward transform in order to obtain the transform coefficients as well, can be accounted to sophisticated rate distortion optimization techniques in HM, namely rate distortion optimized quantization (RDOQ, see Sec. 2.3 and [KYC08, KCYJ09]), where for each transform coefficient a number of candidate values are tested and the one resulting in the smallest RD cost is chosen, and zero transform block checking, where for each transform block it is checked whether setting the whole block to zero leads to a reduced RD cost.

In Fig. 4.7, the regularization path which results from setting $N = 1$ is shown for two exemplary sequences. In this scenario, no inter-frame dependencies are considered in the optimization, but instead each frame is optimized individually, using the ℓ_1-regularized least squares problem of Eq. 3.18. Since there is only one frame in this optimization, the prediction matrix $\mathbf{M}$ will be the zero matrix in this case. Consequently, the reconstruction matrix $\mathbf{A}$ is identical to the inverse transform matrix $\mathbf{T}$. The top diagram of Fig. 4.7 shows the results for the BasketballDrill sequence at 832×480 resolution, which is the best performing of all the Class C and Class D sequences for $N = 1$. The outcome for the BlowingBubbles sequence at 416×240 resolution, which is shown in the bottom diagram, is typical for the remaining sequences. There is a significant coding efficiency loss corresponding to a bit rate increase of about 4 % compared to the HM anchor configuration. The variant with $N = 1$ is also outperformed by a modified HM anchor configuration, where RDOQ has been disabled. Therefore, there are two antagonistic effects associated with the ℓ_1-based multi-frame optimization. The modelling of the resulting bit rate by the ℓ_1-norm is less accurate than the exact bit rate estimation of RDOQ, leading to a coding efficiency loss in a single frame setting. In a multi-frame setting, corresponding to $N > 1$, however, the inter-frame dependencies are considered in the distortion term, which more than compensates the loss caused by the rough rate estimate, and results in an overall coding efficiency improvement.

4.1.3 Determination of the optimal regularization parameter

In this section, the question of how to choose an appropriate value of the regularization parameter μ is addressed. Since having to sweep μ over a large range in order to find the best operating point for a given QP is very inconvenient, a simple rule similar to the one typically used for the computation of the Lagrangian multiplier λ in the



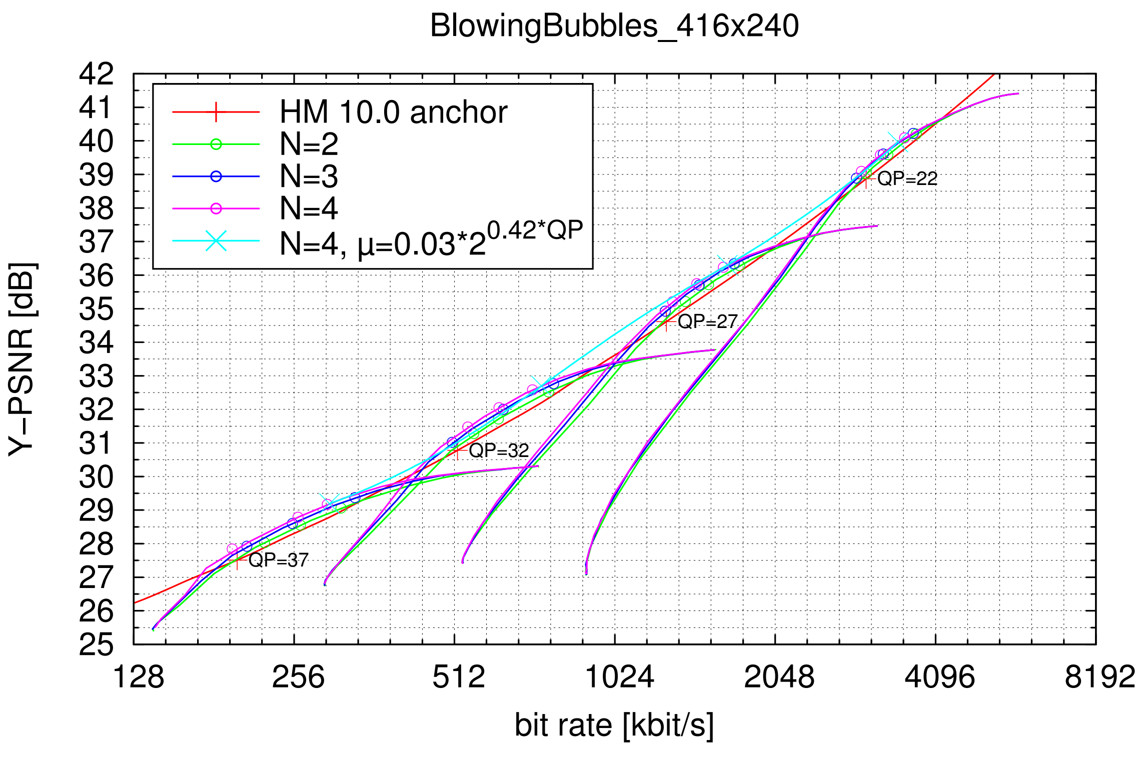
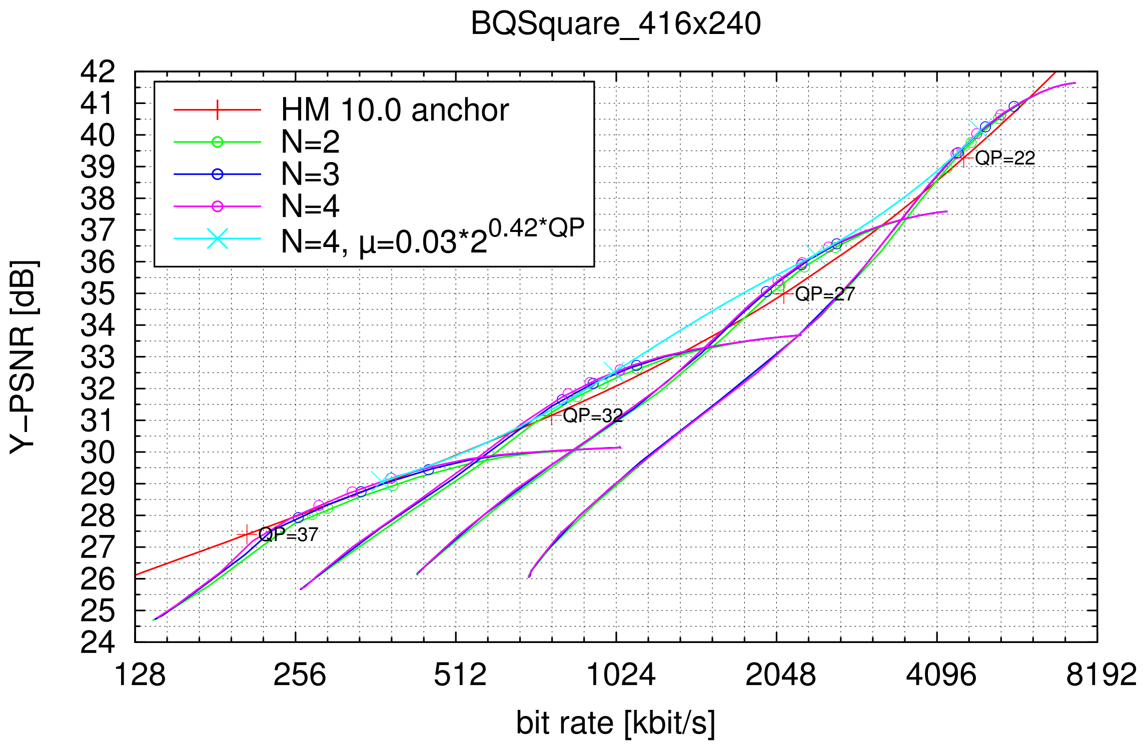


Figure 4.2: Rate Distortion results using IPPP... prediction for sequences BasketballPass (top) and BlowingBubbles (bottom) at 416×240 resolution.



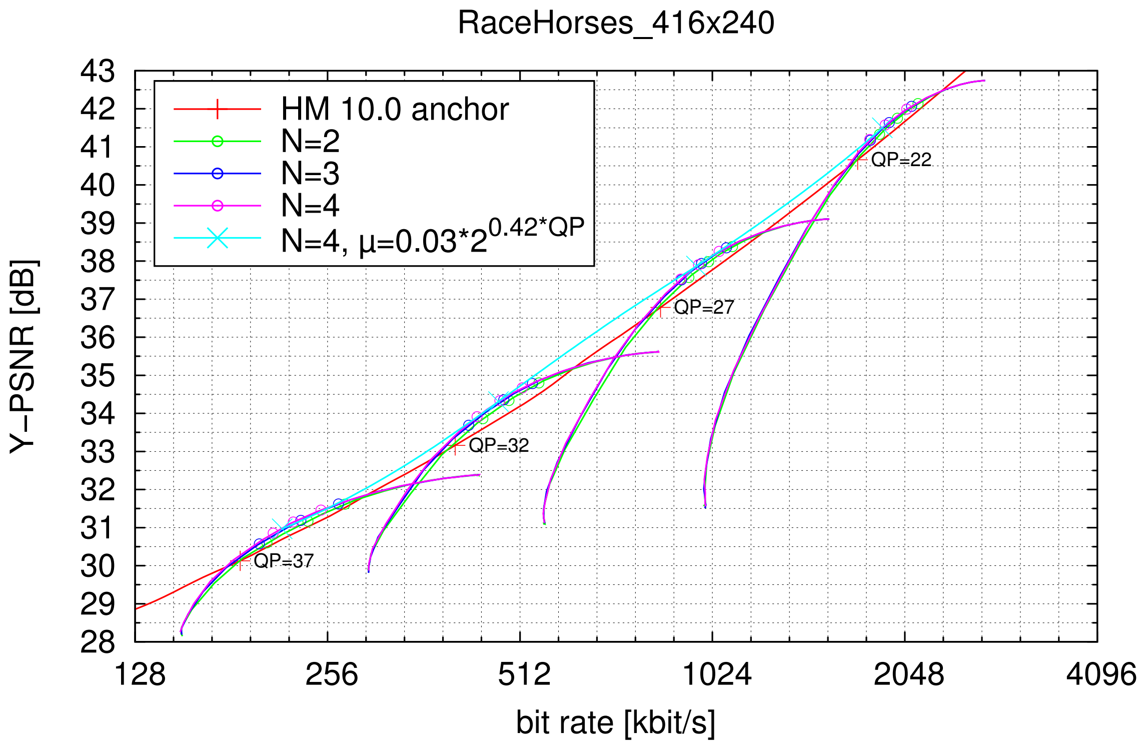
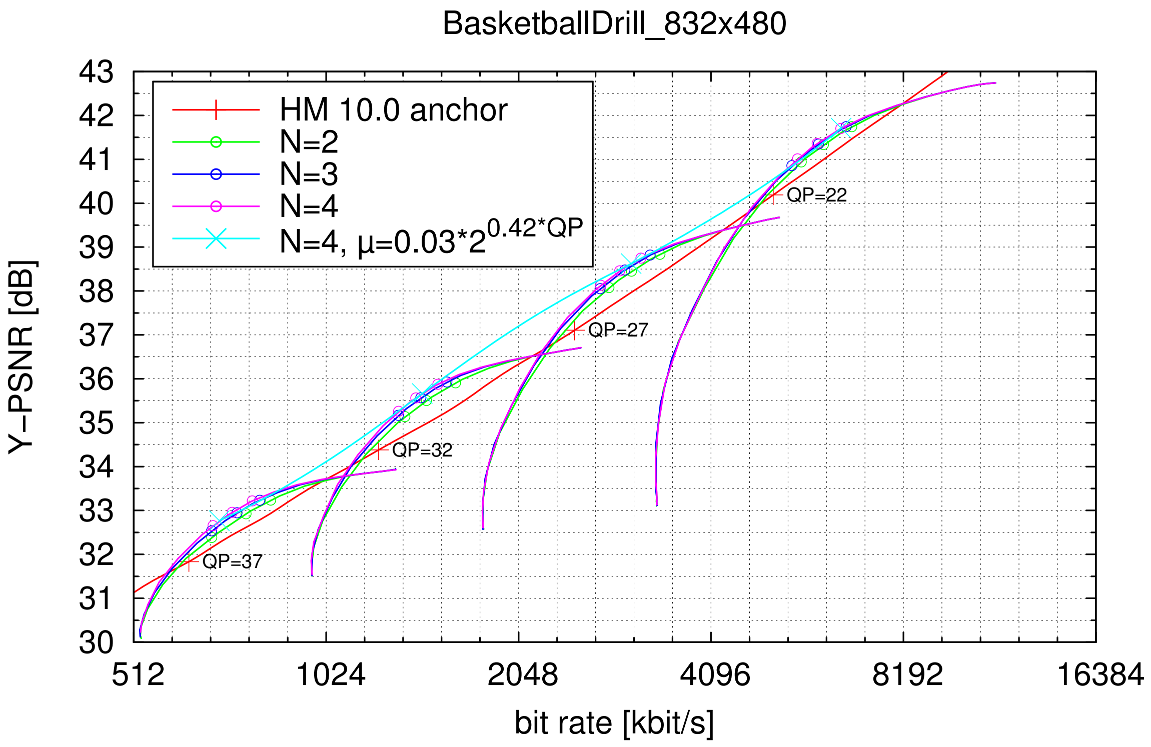


Figure 4.3: Rate Distortion results using IPPP... for sequences BQSquare (top) and RaceHorses (bottom) at 416×240 resolution.



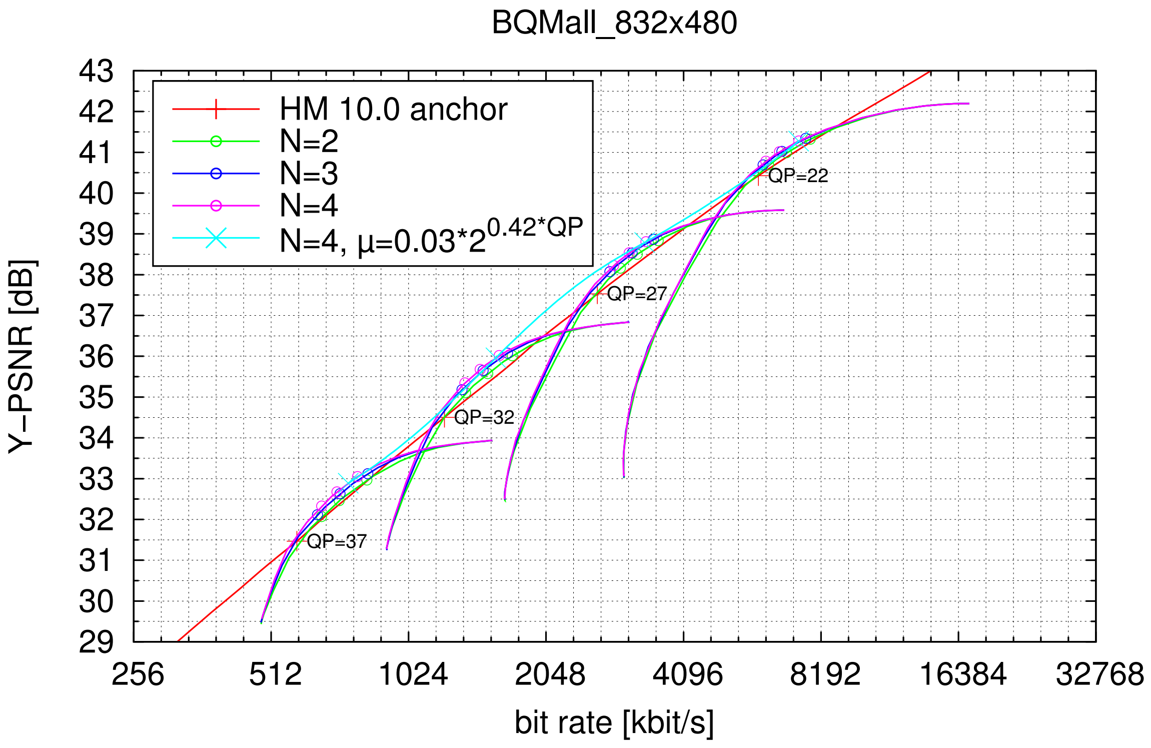
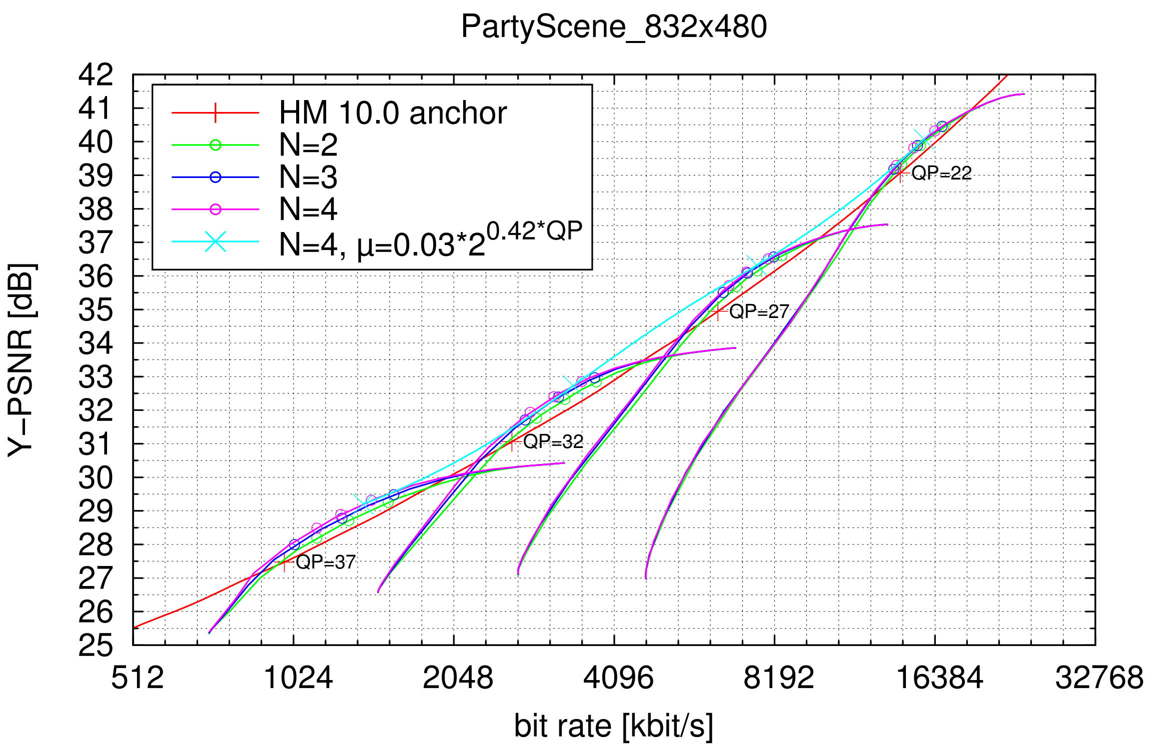


Figure 4.4: Rate Distortion results using IPPP... for sequences BasketballDrill (top) and BQMall (bottom) at 832×480 resolution.



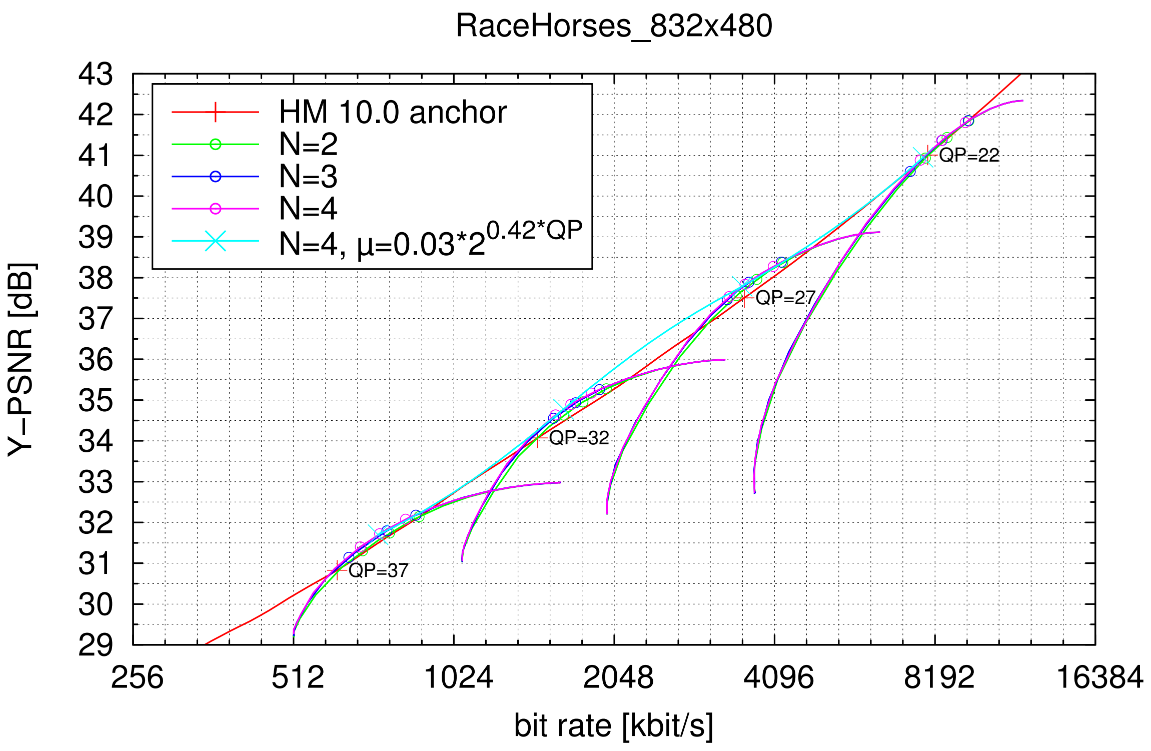
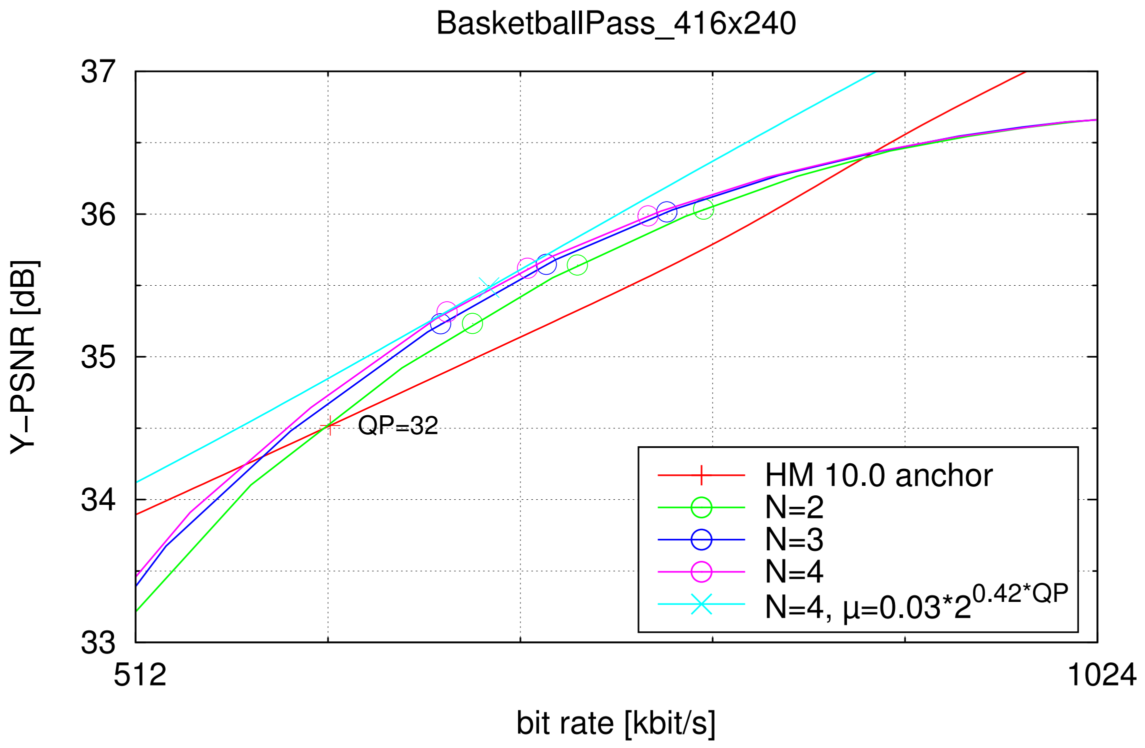


Figure 4.5: Rate Distortion results using IPPP… for sequences PartyScene (top) and RaceHorses (bottom) at 832×480 resolution.



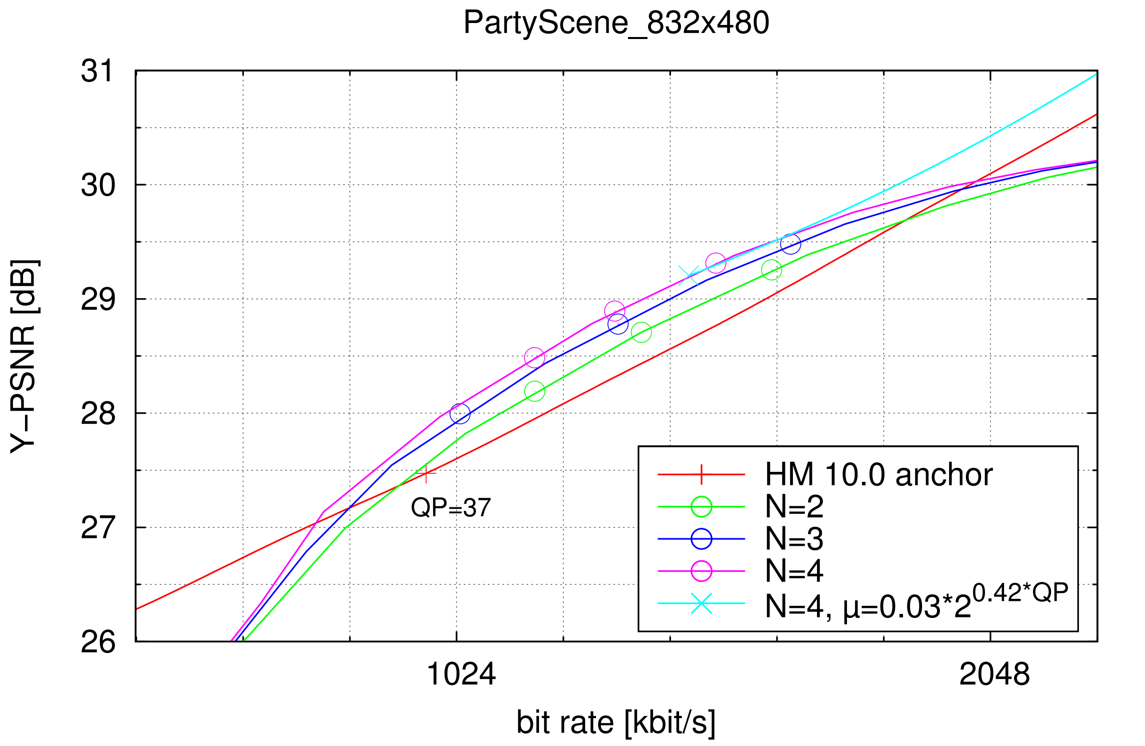
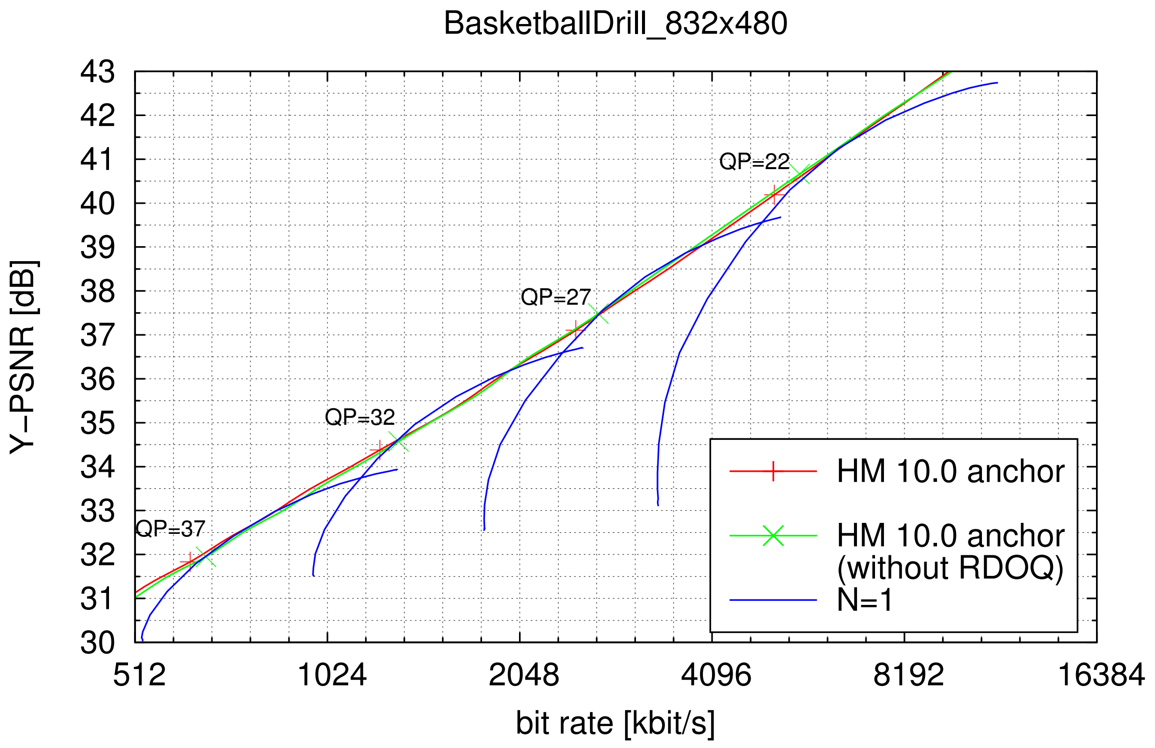


Figure 4.6: Detailed view for two exemplary cases.



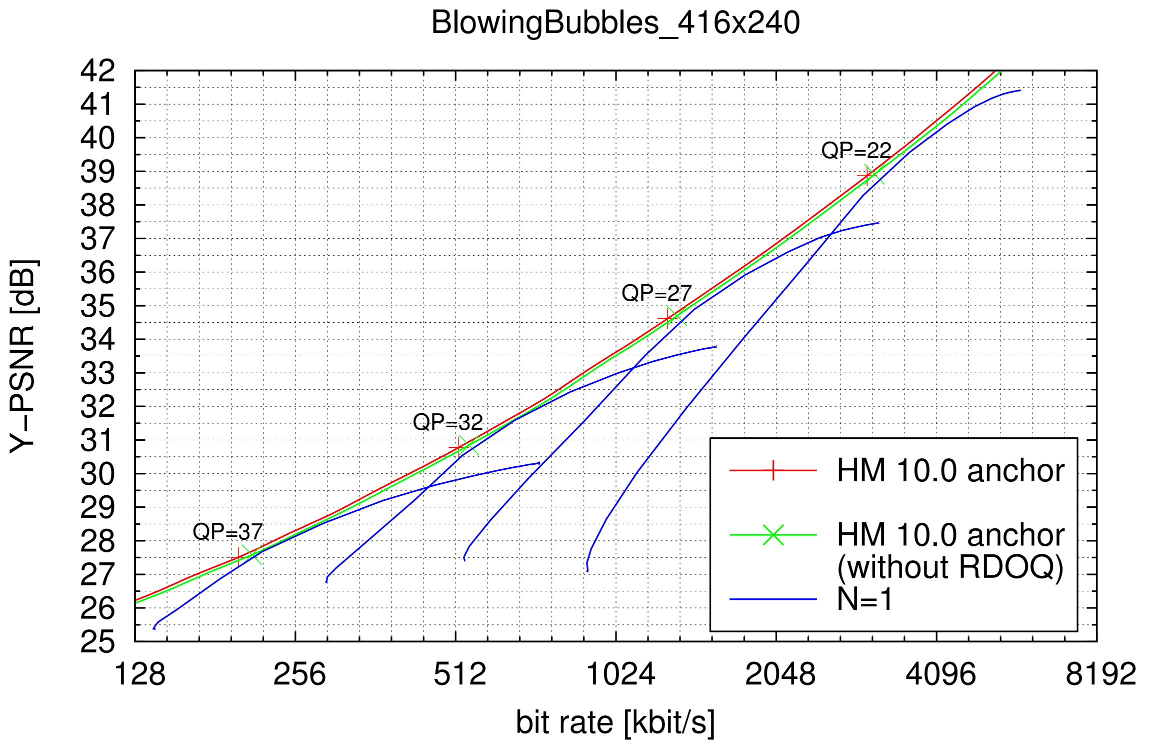


Figure 4.7: Rate Distortion results when setting $N = 1$.

rate distortion optimization of a video encoder is sought after. A similar approach has been taken in [KGFS11] in order to determine the optimal Lagrangian multiplier which controls the trade-off between foreground and background quantization step size when using automatic Sprite coding. In a first step, for each of the curves corresponding to a sweep of the regularization parameter μ in Figs. 4.2–4.5, the optimal operating point is determined. Based on this, a range of acceptable points is defined. The set of all the values of the regularization parameter μ which lead to acceptable operating points is the set of acceptable μ values for a given sequence and a given QP value. If there is an intersection of the acceptable μ values for all the sequences given a specific QP value, any value within this intersection would be an appropriate μ value for this specific QP value. If there is no such intersection, a μ value is selected instead which is as close as possible to an acceptable μ value for all the sequences. So far, for each QP value, a (possibly singleton) set of appropriate μ values has been defined. Finally, a closed-form expression for μ given the QP value can be found by the usage of regression analysis.

In the following, the individual steps are described in more detail. For the definition of the optimal operating point, two cases have to be distinguished. If there are operating points lying above the HM anchor curve (i.e., corresponding to a coding gain), then the optimal operating point is defined as the point above the anchor curve whose minimal distance to the HM anchor curve is maximal, or, simpler stated, the outermost point, maximizing the distance from the anchor curve in top-left direction. If there are no such points, then the optimal operating point is defined as the point whose minimal distance to the HM anchor curve is minimal, such that the distance from the HM anchor curve (which corresponds to the coding loss) is minimized. On each of the curves for $N = 2$, $N = 3$, and $N = 4$ in Figs. 4.2–4.5, there are three points marked by hollow circles. The central point shows the optimal operating point as defined. The outer two points mark the region of acceptable operating points. Introducing such a region is beneficial, because even though there is only one uniquely defined optimal operating point for each sequence and QP value, a range of neighboring operating points typically shows very similar coding performance. Since a QP-dependent rule for the regularization parameter μ which fits all the sequences is sought after, it is helpful to relax the requirement to hit the optimal operating point, and offer instead a range of operating points which are considered to be as equally as good as the optimal point. The range of acceptable points has been defined based on the slope of the rate distortion curve in the optimal point. Using the range of points whose slope is within $\pm 30\%$ of the slope in the optimal point has empirically been found to be expedient.

Choosing the tolerance too small will only include operating points with similar coding performance as the optimal point, however, there might be no overlap among the acceptable ranges for the individual sequences at a given QP value, complicating the issue to obtain a rule which performs well for all the sequences. Choosing the tolerance too large, on the contrary, will result in operating points being considered as acceptable which are significantly inferior in their coding performance compared to the optimal point. Note that in a flat region, where the curvature of the rate distortion curve is low, this slope-based criterion will lead to a larger range of acceptable operating points, whereas in regions with a higher curvature, the range of acceptable operating points will be smaller.

Using the distance from the anchor curve for defining the set of acceptable points, i.e. using those operating points which are within $X\,\%$ of the distance from the optimal operating point to the anchor curve, might be intuitively more appealing, because the distance to the anchor curve corresponds to the coding gain, and consequently this criterion would result in operating points with similar coding performance, which might not always be the case with the criterion based on the slopes. However, the problem with this criterion is that the range of acceptable μ values will be the smaller the closer the optimal operating point is to the anchor curve. The corner case where the μ curve touches the anchor curve is illustrated in Fig. 4.8. In this case, using the distance from the anchor curve as the defining criterion, the set of acceptable operating points would only consist of one single point, namely the optimal operating point where the μ curve touches the anchor curve.

The coding experiments in the previous section have been repeated using all the even QP values in the range 20–40, and the range of acceptable μ values has been determined for each combination of sequence and QP value as described above. The resulting intervals are shown in Fig. 4.9 for $QP \in \{22, 26, 30, 34, 38\}$. The horizontal lines represent the value which is obtained as the average of the maximum lower bound and the minimum upper bound of all the μ intervals for a given QP across the sequences. It can be seen, that in this diagram with the exception of QP 34, sequences 2 and 5, this representative value is always within the set of acceptable μ values, and even in these two exceptional cases, it is very close to the interval. Furthermore, it has to be emphasized, that the size of the intervals is driven by the criterion that the slope of the rate distortion curve in the acceptable operating points should be similar to the slope in the optimal point. For this, a deviation of $\pm 30\%$ from the slope in the optimal point has been admitted, based on empirical observations.

By allowing a larger deviation, it is possible to increase the interval sizes, at the cost of including inferior operating points into the acceptable set. The maximum lower and minimum upper interval boundaries for each QP value across all the sequences, together with their arithmetic mean, are presented in Tab. 4.1. If the maximum lower value is larger than the minimum upper value, there is no overlap of the acceptable μ ranges for all the sequences. This occurs for QPs 20, 34, and 36.

In Fig. 4.10, the μ values which are represented by a horizontal line for each QP in Fig. 4.9, are plotted over the QP. Note that a semi-logarithmic presentation using a linear x-axis and a logarithmic y-axis (so called "log-linear" plot) is used. It can be seen, that the connection of all the data points very closely resembles a straight line. A straight line in the log-linear domain corresponds to the following function in the linear domain:

$$\mu(QP) = a \cdot 2^{b \cdot QP} \tag{4.1}$$

The selection of base 2 is arbitrary, since any other base k could be achieved by multiplication of the exponent by $\log_2 k$. The values of a and b can be determined using log-linear regression [Hei68]. For $N = 3$, $a = 0.0385 \approx 0.04$ and $b = 0.4036 \approx 0.40$ have been obtained. For $N = 4$, the resulting values are $a \approx 0.03$ and $b \approx 0.42$, which is also not too far off the data points for $N = 3$ (see Fig. 4.10), such that in the following this rule has been used in order to determine the regularization parameter μ based on the quantization parameter QP:

$$\mu(QP) = 0.03 \cdot 2^{0.42 \cdot QP} \tag{4.2}$$

The resulting rate distortion curve when using this rule is also shown in Figs. 4.2–4.5 for $N = 4$. Additionally, in Tab. 4.2 the coding efficiency of using the rate distortion optimal operating point, which corresponds to using a sequence and QP specific value of μ, is compared with the case of using the rule from Eq. 4.2 across-the-board. For the comparison, the Bjøntegaard delta bit rate (BD bit rate) as proposed in [Bjø01] is used. Note that a negative value corresponds to a gain in coding efficiency. It can be seen that using the empirically derived QP dependent rule for the regularization parameter μ leads to an average loss of 0.5 % bit rate, with a maximum loss of 1.7 % for the BlowingBubbles sequences at 416×240 resolution.

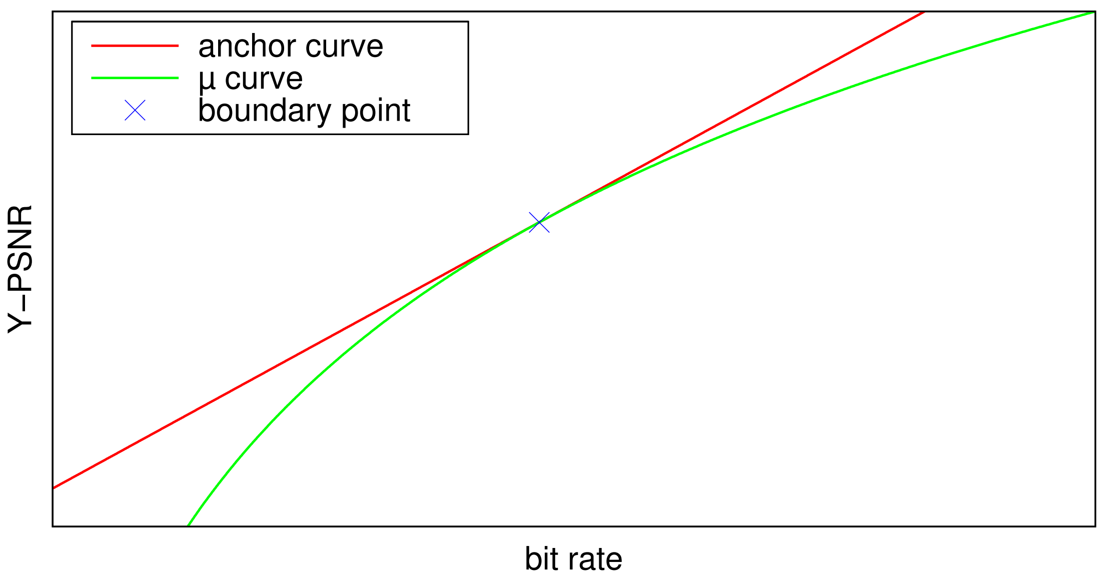


Figure 4.8: Example illustrating the case where the μ curve touches the anchor curve.

QP	Acceptable μ range for $N = 3$: max. lower value	min. upper value	midpoint	$\mu = a \cdot 2^{b \cdot QP}$: $a = 0.04$, $b = 0.40$	$a = 0.03$, $b = 0.42$
20	10.74	9.01	9.88	10.24	10.13
22	18.33	18.76	18.55	17.83	18.14
24	30.73	35.06	32.89	31.04	32.47
26	53.35	63.79	58.57	54.05	58.13
28	93.53	103.35	98.44	94.10	104.05
30	155.47	179.29	167.38	163.84	186.25
32	283.30	307.56	295.43	285.26	333.40
34	484.44	456.62	470.53	496.67	596.80
36	893.87	879.52	886.69	864.75	1068.30
38	1472.78	1751.19	1611.99	1505.62	1912.32
40	2476.40	3471.97	2974.19	2621.44	3423.14

Table 4.1: Maximum lower and minimum upper μ range value (across all the sequences), the midpoint of the two, and the value derived according to the empirical rule of Eq. 4.1 for $N = 3$ (left) and $N = 4$ (right).

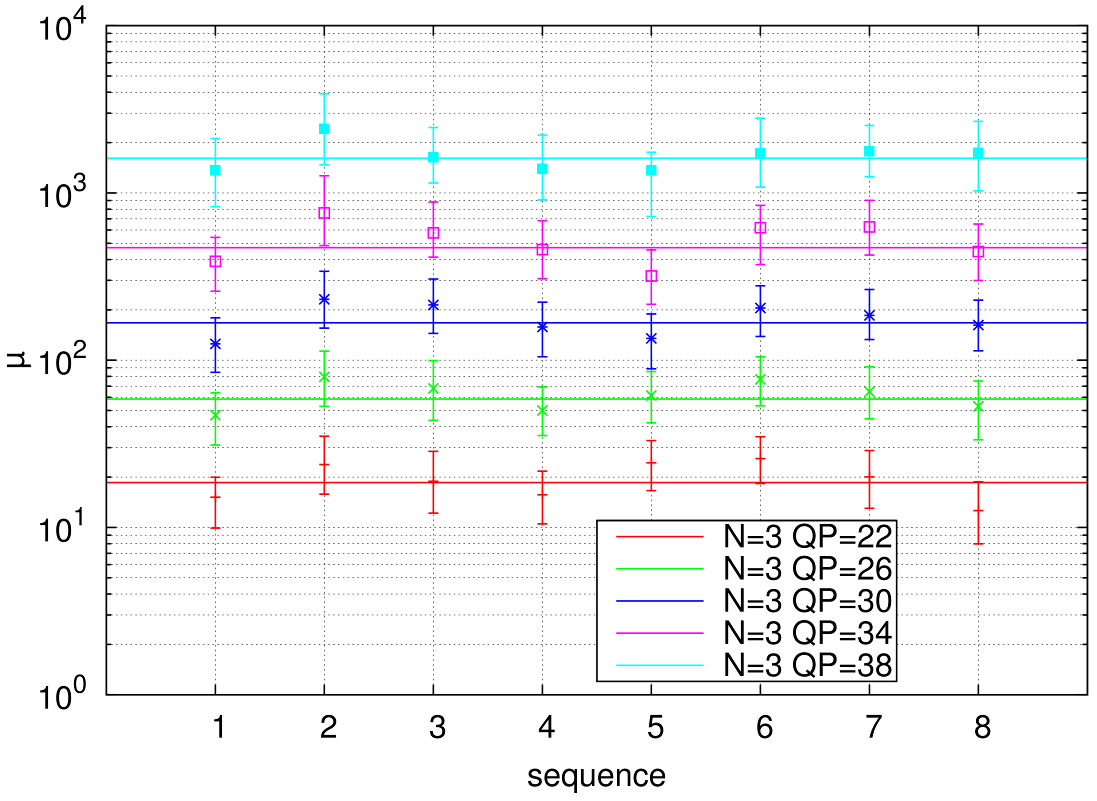


Figure 4.9: Acceptable μ values for $N = 3$ and $QP \in \{22, 26, 30, 34, 38\}$ (sequences 1–4 are at 416×240, 5–6 at 832×480 resolution, with 1: BasketballPass, 2: BlowingBubbles, 3: BQSquare, 4: RaceHorses, 5: BasketballDrill, 6. BQ-Mall, 7: PartyScene, 8: RaceHorses).

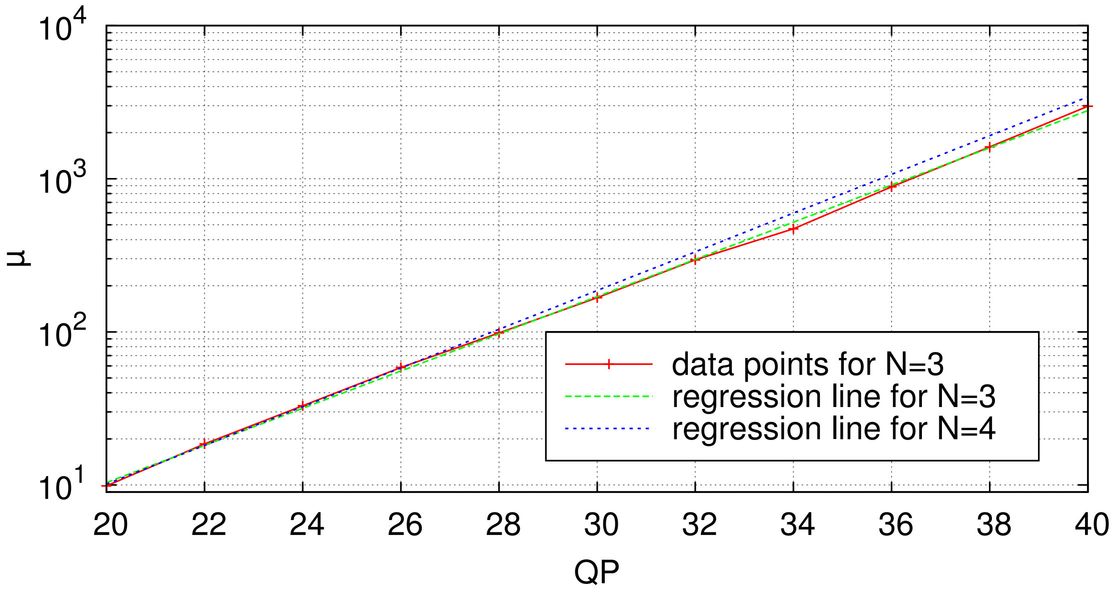


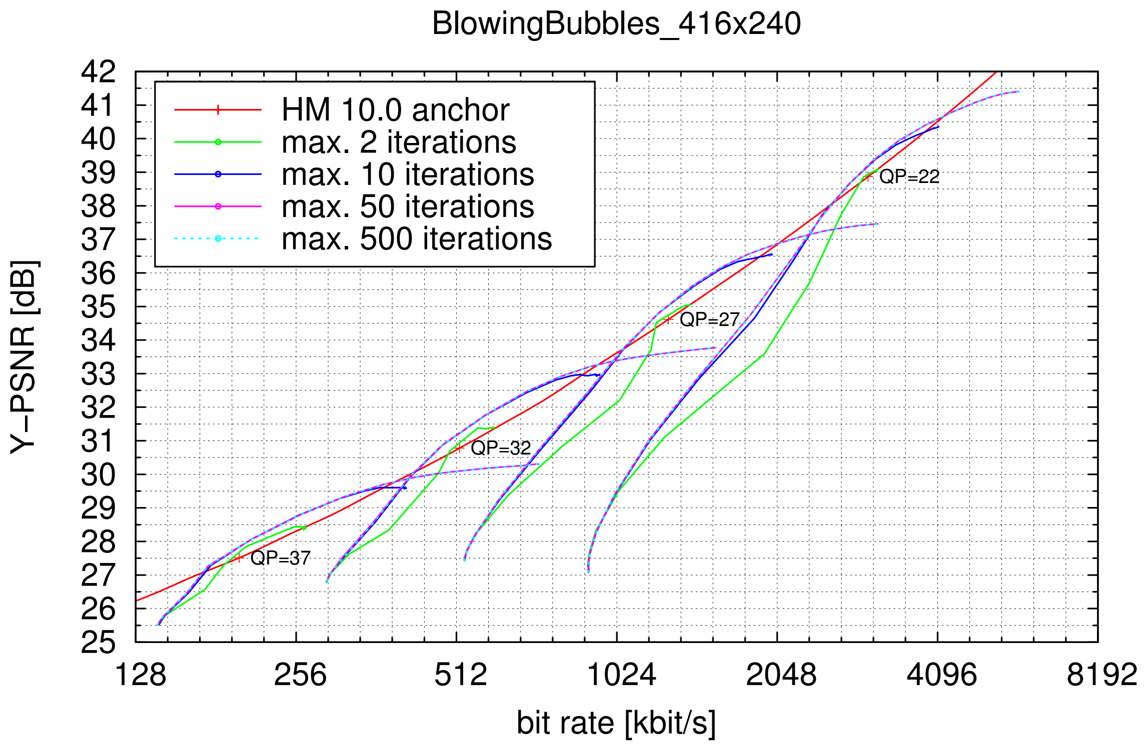
Figure 4.10: Log-linear regression analysis of the optimal μ value over the QP.

		BD bit rate [%]		
	Sequence	Optimal operating point	$\mu = 0.03 \cdot 2^{0.42 \cdot QP}$	Loss due to the μ rule
Class C (832×480)	BasketballDrill	−15.3	−15.3	0.0
	BQMall	−10.2	−9.5	0.7
	PartyScene	−12.1	−11.6	0.5
	RaceHorses	−6.5	−6.4	0.1
Class D (416×240)	BasketballPass	−8.0	−7.9	0.1
	BlowingBubbles	−11.3	−9.6	1.7
	BQSquare	−10.9	−10.3	0.6
	RaceHorses	−7.9	−7.9	0.0
	AVERAGE	−10.3	−9.8	0.5

Table 4.2: BD bit rate results for $N = 4$ comparing the performance using the optimal operating point and the QP-dependent μ rule of Eq. 4.2.

4.1.4 Analysis of limiting the maximum number of iterations

In Fig. 4.11, the impact of limiting the maximum number of ISTA iterations is shown on the basis of two exemplary sequences, BlowingBubbles at 416×240 resolution and BasketballDrill at 832×480 resolution. The curve for a maximum of 500 iterations is the same as the curve for $N = 4$ in Figs. 4.2–4.5. It can be seen that limiting the maximum number of iterations generally moves the higher end point of the regularization path (corresponding to a small μ) closer to the HM anchor operating point. This can be explained by the fact that, because the original HM transform coefficients are used as a warm start in the iterative optimization algorithm, for higher rate points, corresponding to a smaller μ, coefficients which are zero in the warm start initialization have to be recovered, which requires a larger number of iterations than forcing coefficients to zero, which happens for a larger value of μ. For a maximum of two iterations, the coding performance is clearly derogated over the whole μ range. For a maximum of ten, there is some impact on the region corresponding to smaller values of μ, making the range of acceptable μ values smaller. Using a maximum number of 50 ISTA iterations shows no coding performance losses compared to using a maximum of 500. Consequently, unless otherwise stated, in the following a maximum of 50 iterations is used.



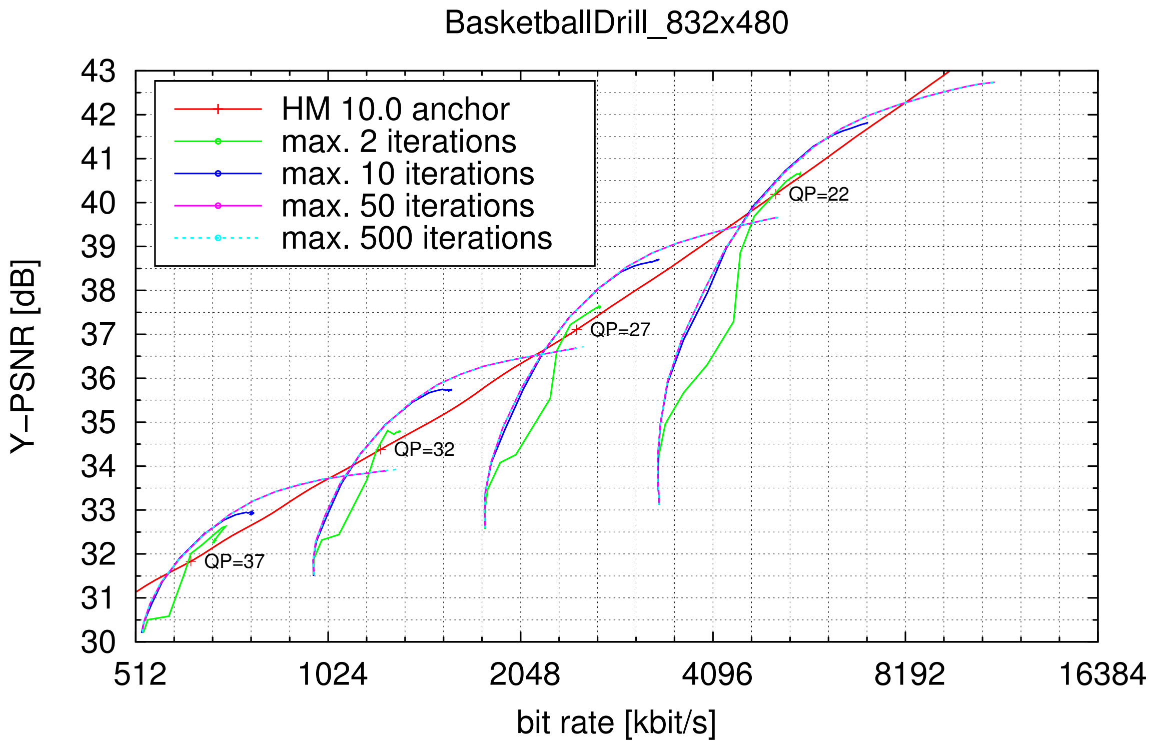


Figure 4.11: Rate Distortion results showing the impact of limiting the maximum number of iterations for two exemplary sequences.

		BD bit rate [%]			Gain of
		Zero block checking			multi frame
	Sequence	off	single frame	multi frame	zero checking
Class C (832×480)	BasketballDrill	−15.3	−11.8	−16.0	−0.7
	BQMall	−9.5	−6.1	−10.5	−1.0
	PartyScene	−11.6	−6.1	−12.5	−0.9
	RaceHorses	−6.4	−3.9	−7.0	−0.6
Class D (416×240)	BasketballPass	−7.9	−3.9	−8.8	−0.9
	BlowingBubbles	−9.6	−5.0	−11.2	−1.6
	BQSquare	−10.3	−5.0	−11.4	−1.1
	RaceHorses	−7.9	−3.1	−8.8	−0.9
	AVERAGE	−9.8	−5.6	−10.8	−1.0

Table 4.3: BD bit rate results for $N = 4$ comparing no special treatment of all-zero blocks ("off") as well as a single and multi frame variant.

4.1.5 Special consideration of all-zero blocks

The video coding standard H.265/HEVC supports very efficient signaling of coding units (CUs) using motion-compensated prediction where all the transform coefficients are equal to zero. This is done by the usage of one of the two syntax elements **cu_skip_flag** or **rqt_root_cbf**. Both are flags, i.e. they can only take the value zero or one. The first syntax element, **cu_skip_flag**, equal to one specifies that the CU is encoded in SKIP mode, i.e. neither motion vectors nor residual signal transform coefficients are transmitted. The motion parameters are derived from so-called *block merging*, which means that the motion data of a selected *merge candidate* are reused for the current CU. The list of merge candidates consists of prediction parameters from spatially neighboring or temporally co-located blocks which have been previously transmitted in the bitstream. The other syntax element, **rqt_root_cbf**, is only present when the SKIP mode is not used. If it is encoded as equal to zero, it specifies that all the transform coefficients for the current CU are zero. With the help of these two syntax elements, it is possible to set up to 6144 transform coefficients (4096 luma plus two times 1024 chroma), in the case of a 64×64 CU, to zero at the cost of very little bit rate, since only a single flag has to be transmitted. This is not captured in the multi-frame transform coefficient optimization problem 3.18, where it is assumed that each transform coefficient contributes individually to the resulting total bit rate. Due to the described highly efficient all-zero handling, changing one single transform coefficient from zero to one in a previously all-zero CU will lead to a higher bit rate increase than incrementing an already non-zero coefficient by one in absolute value.

The increase of the ℓ_1-norm, which serves as an approximation of the bit rate in Eq. 3.18, however, will be the same in both cases.

In this section, it is shown how the efficient treatment of all-zero blocks in H.265/ HEVC can also be taken into account for the multi-frame transform coefficient optimization. Three variants are compared for the Class C (832×480) and Class D (416×240) sequences in Tab. 4.3 using the Bjøntegaard delta bit rate (BD bit rate) as proposed in [Bjø01], where a negative value shows a gain in coding performance. For the multi-frame optimization, $N = 4$ frames are considered jointly. The first variant, denoted in the table as "off," refers to the case where no special handling is performed and the transform coefficients as obtained from the optimization are directly used for the HEVC bitstream. Note that this is the same as the column showing the performance of applying the QP-dependent μ rule in Tab. 4.2. Its coding performance can be viewed as a kind of reference configuration in the following. In the second variant, denoted as "single frame," two rate distortion (RD) costs are determined for each CU, namely the costs resulting from

- using the transform coefficients from the optimization, and
- setting all the transform coefficients of the CU to zero.

For $x \in \{opt, zero\}$, corresponding to encoding the optimized transform coefficients or an all-zero CU, the RD cost $J_{x,single}$ can be derived from the Lagrangian multiplier λ, the number of bits R_x and the distortion in the current frame $D_{currentCU,x}$ as follows:

$$J_{opt,single} = \lambda \cdot R_{opt} + D_{currentCU,opt} \tag{4.3}$$

$$J_{zero,single} = \lambda \cdot R_{zero} + D_{currentCU,zero} \tag{4.4}$$

If $J_{zero,single} < J_{opt,single}$, the CU is encoded as all-zero. Note that here, as in the rate distortion optimization of the HM encoder, the actual bit rate and distortion are used, not the approximations from Eq. 3.18. Further note that only the distortion of the current CU is considered, and therefore the impact on subsequent frames is neglected. While there is still an average gain of 5.6 % BD bit rate over the HM anchor configuration, compared to the "off" configuration, there is a significant loss. Since the only difference to the "off" configuration is that in the "single frame" configuration some CUs are set to zero, it can be concluded that by this means transform coefficients which would otherwise benefit subsequent frames are dropped, because the "investment" to encode those coefficients does not pay off from the perspective of a single frame.

As a consequence, the impact on subsequent frames has also to be considered in the computation of the RD cost for deciding whether a CU is to be set equal to zero. This is done by including an additional distortion term which reflects the distortion that occurs in the subsequent $N-1$ frames under consideration. Similarly as above, for $x \in \{opt, zero\}$, corresponding to encoding the optimized transform coefficients or an all-zero CU, the RD cost $J_{x,multi}$ can be derived from the Lagrangian multiplier λ, the number of bits R_x, the distortion in the current frame $D_{currentCU,x}$, and the distortion in the subsequent frames $D_{subsequentFrames,x}$:

$$J_{opt,multi} = \lambda \cdot R_{opt} + D_{currentCU,opt} + D_{subsequentFrames,opt} \tag{4.5}$$

$$J_{zero,multi} = \lambda \cdot R_{zero} + D_{currentCU,zero} + D_{subsequentFrames,zero} \tag{4.6}$$

Based on this, two related RD costs can be derived as

$$\hat{J}_{opt,multi} = \lambda \cdot R_{opt} + D_{currentCU,opt} \tag{4.7}$$

$$\hat{J}_{zero,single} = \lambda \cdot R_{zero} + D_{currentCU,zero} + \underbrace{D_{subsequentFrames,zero} - D_{subsequentFrames,opt}}_{\Delta D_{subsequentFrames,zero}} \tag{4.8}$$

where it holds that

$$J_{opt,multi} < J_{zero,multi} \Leftrightarrow \hat{J}_{opt,multi} < \hat{J}_{zero,multi}. \tag{4.9}$$

Here, $\hat{J}_{opt,multi}$ in Eq. 4.7 is equal to the RD cost for the single frame case $J_{opt,single}$, i.e. without consideration of subsequent frame dependencies, and $\Delta D_{subsequentFrames,zero}$ is a distortion difference which gives the amount by which the distortion in the subsequent frames is increased by encoding the current CU as all-zero instead of using the optimized transform coefficients.

$$\hat{J}_{opt,multi} = J_{opt,single} \tag{4.10}$$

$$\hat{J}_{zero,single} = J_{zero,single} + \Delta D_{subsequentFrames,zero} \tag{4.11}$$

Putting it all together, if the distortion difference $\Delta D_{subsequentFrames,zero}$ is added to the single frame RD cost $J_{zero,single}$, the impact on subsequent frames is considered in

		BD bit rate [%]	
	Sequence	Fixed prediction params.	Sliding window based
Class C (832×480)	BasketballDrill	−4.9	−9.2
	BQMall	−0.8	−3.4
	PartyScene	−4.4	−6.9
	RaceHorses	−1.7	−5.3
Class D (416×240)	BasketballPass	0.4	−2.9
	BlowingBubbles	−2.3	−5.9
	BQSquare	−3.9	−6.2
	RaceHorses	−2.2	−6.8
	AVERAGE	−2.5	−5.8

Table 4.4: BD bit rate results for $N = 3$ comparing the performance using fixed and sliding window based prediction parameters.

the decision between using the optimized coefficients or encoding an all-zero CU.

Note that $\Delta D_{subsequentFrames,zero}$ is a value which has to be individually computed for each CU. For that purpose, it is assumed that all other CUs are encoded using the optimized transform coefficients (*ceteris paribus* assumption), and the resulting distortion for the subsequent frames is determined, once using the optimized coefficients and once using an all-zero block for the current CU. The difference between the latter and the former gives the value of $\Delta D_{subsequentFrames,zero}$. The results for using this modified RD costs, which incorporate the impact on subsequent frames, are shown in the column denoted as “multi frame” in Tab. 4.3. The last column shows the difference between the “multi frame” and the “off” variant. It can be seen that this leads to a moderate, but consistent gain of about 1 % BD bit rate over directly using the transform coefficients which result from solving the optimization problem in Eq. 3.18.

4.1.6 Impact of using fixed prediction parameters

In the algorithm as described in Sec. 4.1.1, the prediction parameters for all but the first two frames (i.e., the initial I frame and the first P frame) are determined multiple times. In particular, the prediction parameters for the first P frame are determined one time, for the second P frame two times etc., and starting from the Nth P frame, for all subsequent P frames N times. Note that this includes all steps of motion estimation as well as determining the block partitioning for prediction and residual coding. Consequently, neglecting the initial N frames, the runtime spent therefor is increased by a factor of N. In this section, the impact of determining the prediction parameters for all the frames at once in advance is compared to the sliding window based approach. The results for the first 50 frames of the Class C and Class D sequences with $N = 3$ are shown in Tab. 4.4. Note that in both cases the transform coefficients are determined using a sliding operation, but for the results in the second column (headed as “Sliding window based”), after the transform coefficients of one frame have been optimized, the prediction parameters for the subsequent N frames have been redetermined, whereas for the results in the first column (headed as “Fixed prediction params.”), the prediction parameters are fixed. While the computational complexity is smaller using fixed prediction parameters, also are the coding gains. As shown in Tab. 4.4, employing sliding window based prediction parameter estimation improves average bit rate savings by more than 3 percentage points.

4.1.7 Comparison of different regularizers

In the multi-frame transform coefficient optimization problem in Eq. 3.18, the ℓ_1-norm term $\|\mathbf{c}\|_1$ can be viewed on the one hand as a replacement for the actual bit rate, for which there is no closed-form expression. On the other hand, it serves as a regularization term, such that, depending on the regularization parameter μ, a certain portion of the transform coefficients is set equal to zero. The ℓ_1-norm has the following advantages:

- It is a convex function, making the whole optimization problem in Eq. 3.18 convex, such that each local optimum is guaranteed to be a global optimum.
- It is sparsity inducing, resulting in a solution vector $\mathbf{c}_{opt}$ with a large number of components exactly equal to zero.
- It leads to a very simple operation in the iterative shrinkage/thresholding algorithm (ISTA), namely the so-called soft thresholding operator, where all vector components which are larger in absolute value than a given threshold are shrinked by this threshold, and all other components are set to zero.

However, it also has some disadvantages:

- For larger transform coefficients, the actual bit rate required to encode the coefficient in the bitstream, grows logarithmically with the absolute value of the coefficient, i.e. significantly slower than the ℓ_1-norm (see Fig. 2.4, p. 11).

- The ℓ_1-norm introduces a bias on the non-zero transform coefficients. This can easily be exemplified in the scalar case with $y, c \in \mathbb{R}$:

  $$c_{opt} = \arg\min_{c} (y-c)^2 + \mu|c| \tag{4.12}$$

  With the soft thresholding operator $\mathcal{S}_{\frac{\mu}{2}}$ as introduced in Sec. 3.4.1, the solution c_{opt} is obtained as:

  $$c_{opt} = \mathcal{S}_{\frac{\mu}{2}}(y) = \begin{cases} y - \frac{\mu}{2} & \text{if } y > \frac{\mu}{2} \\ 0 & \text{if } -\frac{\mu}{2} \leq y \leq \frac{\mu}{2} \\ y + \frac{\mu}{2} & \text{if } y < -\frac{\mu}{2} \end{cases} \tag{4.13}$$

  From Eq. 4.13, it can be seen, that for non-zero c_{opt}, there is a bias of $\frac{\mu}{2}$ compared to the least-squares solution $c_{opt} = y$.

Recently, iterative thresholding algorithms for other, possibly non-convex regularization functions have been proposed [XCXZ12, GZL$^+$13, VC13, ZMZ$^+$13]. In this section, the rate distortion behaviour of the following regularization functions is studied in comparison to the ℓ_1-norm:

- *capped ℓ_1-norm* [Zha08, GZL$^+$13]: This regularization function is defined as follows, with a model parameter $\theta > 0$:

  $$R_{capped,\theta}(\mathbf{c}) = \sum_{i=0}^{K-1} |\min\{c_i, \theta\}| \tag{4.14}$$

  The difference to the ℓ_1-norm is that the impact of all transform coefficients which are larger than θ in absolute value will be clipped at θ. This leads to a

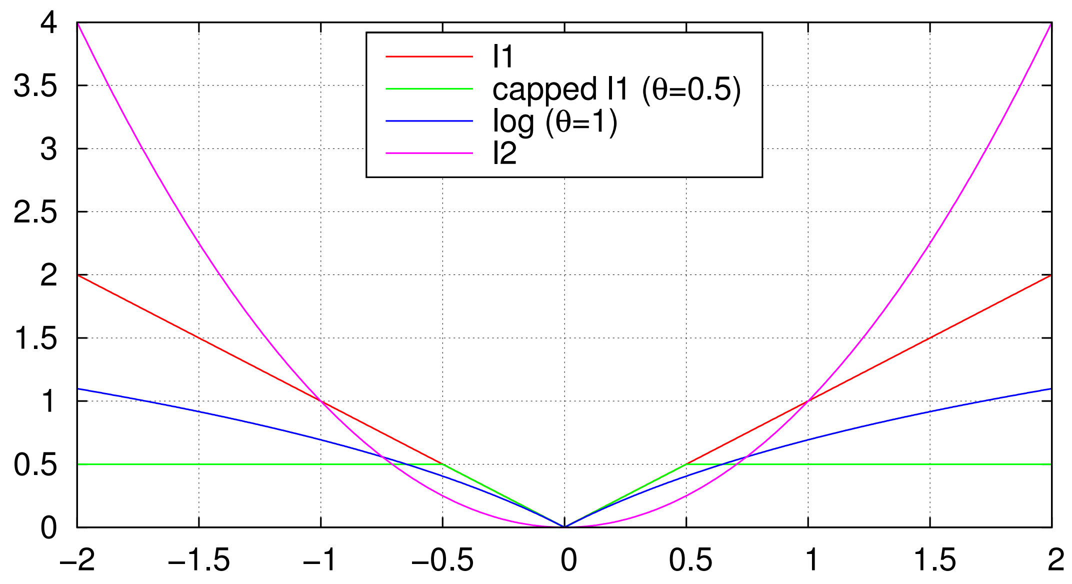


Figure 4.12: Illustration of the regularization functions ℓ_1-norm, capped ℓ_1-norm (with $\theta = 0.5$), logarithmic sum penalty (with $\theta = 1$), and ℓ_2-norm.

reduction of the bias which is imposed on the larger transform coefficients.

- *logarithmic sum penalty (LSP)* [CWB08, GZL+13]: This regularization function also has a model parameter $\theta > 0$:

$$R_{lsp,\theta}(\mathbf{c}) = \sum_{i=0}^{K-1} \log\left(1 + \frac{|c_i|}{\theta}\right) \tag{4.15}$$

For $c_i \approx 0$, it behaves very similar to the ℓ_1-norm, thus also promoting sparsity in the solution vector. For $|c_i| \gg 0$, however, it grows much slower, and therefore also reduces the bias on larger transform coefficients.

- *ℓ_2-norm*: This regularizer is also known in literature under the names Tikhonov regularization [Tik63] or ridge regression [Hoe62]. It is simply defined as the sum of the squares of the vector components:

$$R_{\ell_2}(\mathbf{c}) = \sum_{i=0}^{K-1} c_i^2 \tag{4.16}$$

Using this regularization function for $R(\mathbf{c})$ in Eq. 3.10 corresponds to minimizing the energy of both the reconstruction error and the transform coefficients.

An illustration of these three regularization functions together with the ℓ_1-norm can be found in Fig.4.12.

In order to use these regularizers in conjunction with the iterative thresholding algorithm, the corresponding thresholding functions, which are the counterpart to the soft thresholding operator for the ℓ_1-norm case, have to be known. Basically, the solution to the proximal operator problem in Eq. 3.104, as repeated below, has to be found for $g(\mathbf{x}) \in \{R_{capped,\theta}(\mathbf{x}), R_{lsp,\theta}(\mathbf{x}), R_{\ell_2}(\mathbf{x})\}$.

$$\mathbf{x}_{n+1} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{u}_n\|_2^2 + \frac{\mu}{\alpha_n} g(\mathbf{x}) \tag{3.104 revisited}$$

Since the regularization functions are separable, the following scalar problem is tackled instead of Eq. 3.104, with $x, x_{n+1}, u_n \in \mathbb{R}$.

$$x_{n+1} = \arg\min_{x} \underbrace{\frac{1}{2}(x - u_n)^2 + \frac{\mu}{\alpha_n} g(x)}_{h(x)} \tag{4.17}$$

According to [GZL+13], for $g(x) = R_{capped,\theta}(x) = \min(\theta, |x|)$, the solution to Eq. 4.17 is

$$x_{n+1,capped} = \begin{cases} w_{capped,1} & \text{if } h(w_{capped,1}) \leq h(w_{capped,2}) \\ w_{capped,2} & \text{otherwise} \end{cases} \tag{4.18}$$

with

$$w_{capped,1} = \operatorname{sgn}(u_n)\max(\theta, |u_n|) \tag{4.19}$$
$$w_{capped,2} = \operatorname{sgn}(u_n)\min(\theta, \max(0, |u_n| - \frac{\mu}{\alpha_n})), \tag{4.20}$$

which is obtained from splitting Eq. 4.17 with $g(x) = \min(\theta, |x|)$ into the two problems:

$$w_{capped,1} = \arg\min_{x} \frac{1}{2}(x - u_n)^2 + \frac{\mu}{\alpha_n}\theta \qquad \text{s.t. } |x| \geq \theta \tag{4.21}$$
$$w_{capped,2} = \arg\min_{x} \frac{1}{2}(x - u_n)^2 + \frac{\mu}{\alpha_n}|w| \qquad \text{s.t. } |x| \leq \theta. \tag{4.22}$$

For $g(x) = R_{lsp,\theta}(x) = \log(1 + {|x|}/{\theta})$, the following solution is presented in [GZL+13]:

$$x_{n+1,lsp} = \operatorname{sgn}(u_n)\, w_{lsp} \tag{4.23}$$

with

$$w_{lsp} = \arg\min_{x} \frac{1}{2}(x - |u_n|)^2 + \frac{\mu}{\alpha_n} \log\left(1 + \frac{x}{\theta}\right) \qquad \text{s.t. } x \geq 0. \tag{4.24}$$

It is stated in [GZL+13], "that the objective function above [Eq. 4.24] is differentiable in the interval $[0, +\infty)$ and the minimum of the above problem is either a stationary point (the first derivative is zero) or an endpoint of the feasible region," leading to

$$w_{lsp} = \arg\min_{x \in \mathcal{C}} \frac{1}{2}(x - |u_n|)^2 + \frac{\mu}{\alpha_n} \log\left(1 + \frac{x}{\theta}\right), \tag{4.25}$$

where $\mathcal{C}$ is a set composed of at most three elements.

If $d := \alpha_n^2(|u_n| - \theta)^2 - 4\alpha_n(\mu - \alpha_n|u_n|\theta) \geq 0$,

$$\mathcal{C} = \left\{0, \left(\frac{\alpha_n(|u_n| - \theta) + \sqrt{d}}{2\,\alpha_n}\right)_+, \left(\frac{\alpha_n(|u_n| - \theta) - \sqrt{d}}{2\,\alpha_n}\right)_+\right\}, \tag{4.26}$$

where $(x)_+ = \max(x, 0)$, otherwise

$$\mathcal{C} = \{0\}. \tag{4.27}$$

Consequently, for the LSP regularizer, Eq. 4.17 has to be evaluated for up to three candidates, and the one resulting in the smallest value is chosen.

Note that both the capped ℓ_1-norm and the LSP are non-convex regularization functions. But, as shown in [GZL+13], they can each be written as the difference of two convex functions:

$$R_{capped,\theta}(x) = \min(\theta, |x|) \quad = |x| - \max(|x| - \theta, 0) \tag{4.28}$$

$$R_{lsp,\theta}(x) = \log(1 + {|x|}/{\theta}) = |x| - (|x| - \log(1 + {|x|}/{\theta})) \tag{4.29}$$

Convergence analysis is also given in [GZL+13].

For the ℓ_2-norm, or Tikhonov regularization, the solution to the optimization problem

$$\mathbf{c}_{opt,\ell_2} = \arg\min_{\mathbf{c}} \|\tilde{\mathbf{y}} - \mathbf{A}\,\mathbf{c}\|_2^2 + \mu\,\|\mathbf{c}\|_2^2 \tag{4.30}$$

can be explicitly obtained as follows (see [DDDM04, p. 1415, Eq. 1.1]):

$$\mathbf{c}_{opt,\ell_2} = (\mathbf{A}^T\mathbf{A} + \mu\mathbf{I})^{-1}\mathbf{A}^T\,\tilde{\mathbf{y}}. \tag{4.31}$$

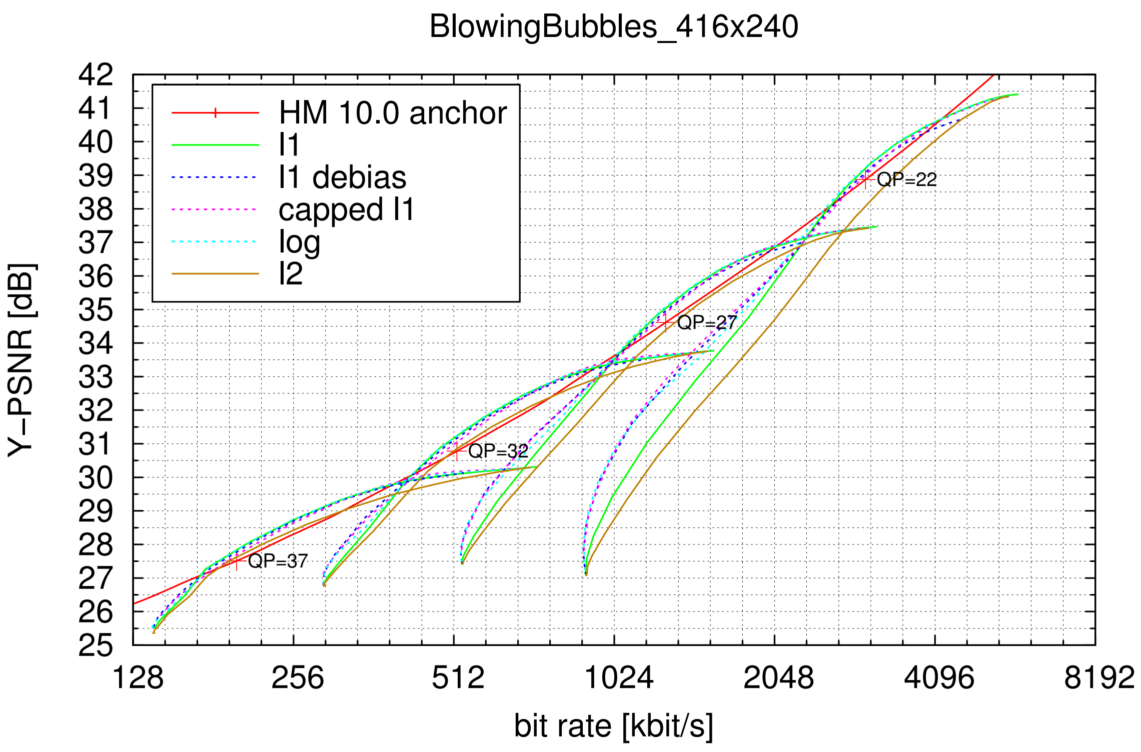
This is different to the other regularization functions discussed so far, where the optimal solution can only be obtained numerically. The problem with Eq. 4.31 in the context of multi-frame transform coefficient optimization is, however, that it requires the computation of the inverse of a very large matrix, which may be infeasible in practice. Therefore, also for the ℓ_2-norm an iterative method is employed.

For $g(x) = R_{\ell_2}(x) = x^2$ in Eq. 4.17, the following solution can easily be derived (see [DDDM04, p. 1426, Eq. 2.3]):

$$x_{n+1,\ell_2} = \frac{u_n}{1 + 2\frac{\mu}{\alpha_n}} \tag{4.32}$$

In the following, the three mentioned regularization functions are evaluated using the sequences BlowingBubbles at 416×240 resolution and BasketballDrill at 832×480 resolution, where $N = 4$ consecutive frames have been considered jointly in the optimization step. The multi-frame all-zero block checking as described in Sec. 4.1.5 is not used. For the capped ℓ_1-norm, $\theta = 0.5$ has been used, such that all transform coefficients which will not be rounded to zero, contribute equally to the regularization term. For the logarithmic sum penalty (LSP), $\theta = 1$ has been used, such that the slope in the origin is the same as for the ℓ_1-norm. The rate distortion plots showing the regularization paths are depicted in Fig. 4.13. The curves corresponding to a regularization using ℓ_1-norm, capped ℓ_1-norm, LSP, and ℓ_2-norm are denoted as “l1,” “capped l1,” “log,” and “l2” in the diagrams. The curve denoted as “l1 debias” shows the behaviour where the ordinary ℓ_1-norm regularizer is used, but in addition a so-called debiasing step [WNF09, pp. 2484] is performed on the resulting transform coefficients. By debiasing it is meant, that those coefficients which will be rounded to zero (i.e., those c_i with $|c_i| < 0.5$), are fixed at zero, whereas for the remaining coefficients, the optimization problem is solved again without the regularization term. This results in a coefficient vector having the same support as the ℓ_1-regularized solution, but without the bias on the non-zero coefficients.

Again, from Fig. 4.13 it can be seen that, except for the debiasing case, the regularization paths share common starting and end points, corresponding to the all-zero



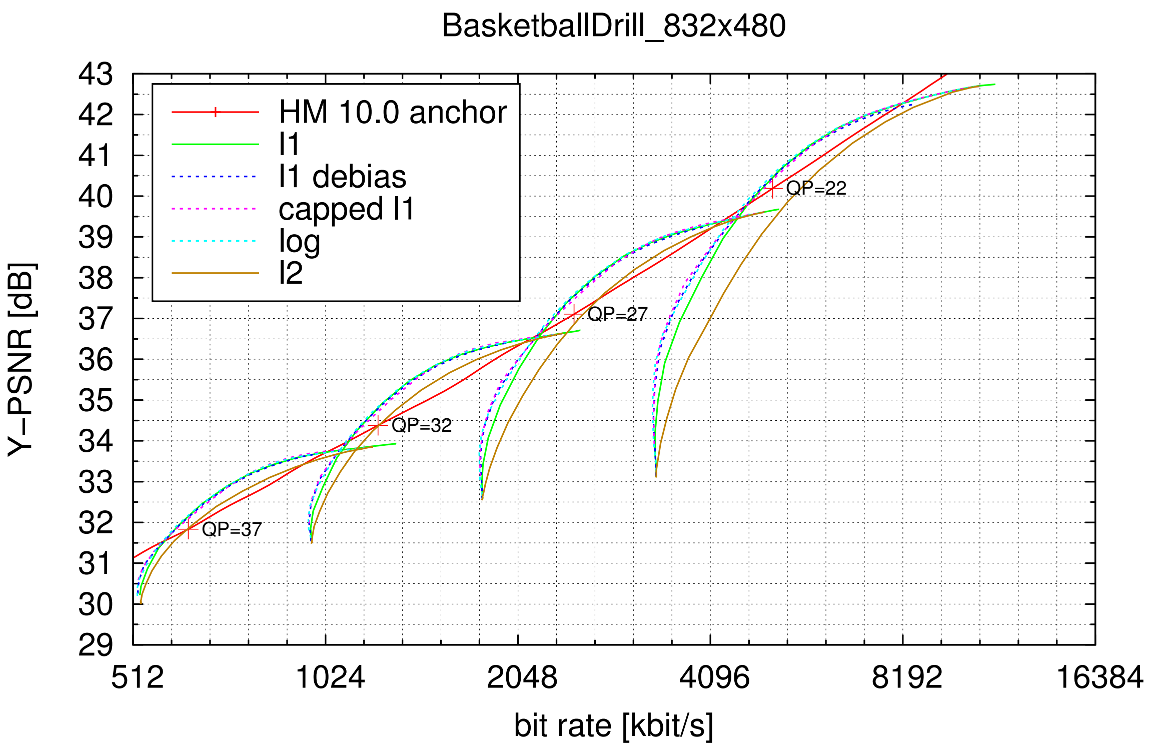


Figure 4.13: Rate Distortion results showing the regularization path for different regularization functions.

solution and the (unregularized) least squares solution. Furthermore, the variant using ℓ_2-norm regularization is significantly inferior to the other regularizers over the whole μ range. The remaining variants (ℓ_1, capped ℓ_1, debiased ℓ_1, and LSP) behave very similar in the region where the regularization path curve lies above the anchor curve. In this region, the ordinary ℓ_1-regularization is always among the two best performing variants, with the logarithmic regularization marginally ahead in some cases. In the lower end region of the regularization path, corresponding to higher μ values, all of capped ℓ_1, debiased ℓ_1, and LSP clearly outperform the ℓ_1-regularization. However, this is practically irrelevant, since it only occurs in a region where there is, even for the best performing variant, a significant loss compared to the HM anchor curve, which is caused by a mismatch of μ and QP value.

Summarizing, it has empirically been shown that in the practically relevant region, where a coding gain over HM can be observed, the ordinary ℓ_1-regularization method is always among the best performing variants. With the soft thresholding operator, the ℓ_1-regularization also leads to a very simple solution of the proximal operator problem Eq. 3.104. Consequently, there is no evidence, that using one of the other discussed regularization methods will show any benefit in a practical coding scenario, where the regularization parameter μ is matched to the quantization parameter (QP).

4.1.8 Analysis of the accuracy of the linear system model

In this section, the accuracy of the linear system model in Eq. 3.9 is studied. In particular, it is evaluated how well the distortion as obtained from the linear model corresponds to the actual distortion. The actual distortion is the distortion which results from generating a H.265/HEVC bitstream using the optimized transform coefficient vector $\mathbf{c}_{opt}$ and decoding that bitstream. In order to be able to compare the model distortion with the actual distortion using the same coefficient vector $\mathbf{c}_{opt}$, for all the experiments conducted in this section, both the sign data hiding [WYH+12] coding tool of H.265/HEVC and the all-zero block checking as introduced in Sec. 4.1.5 are disabled. The Class C (832×480) and Class D (416×240) sequences are encoded using HM for the reference and using the multi-frame optimization with a group of $N = 4$ frames. In Figs. 4.14 and 4.15, the estimated and the actual distortion for two exemplary groups of four frames are shown. The curve denoted as "actual PSNR" shows the distortion of the decoded video. As discussed in Sec. 4.1.1 and shown in

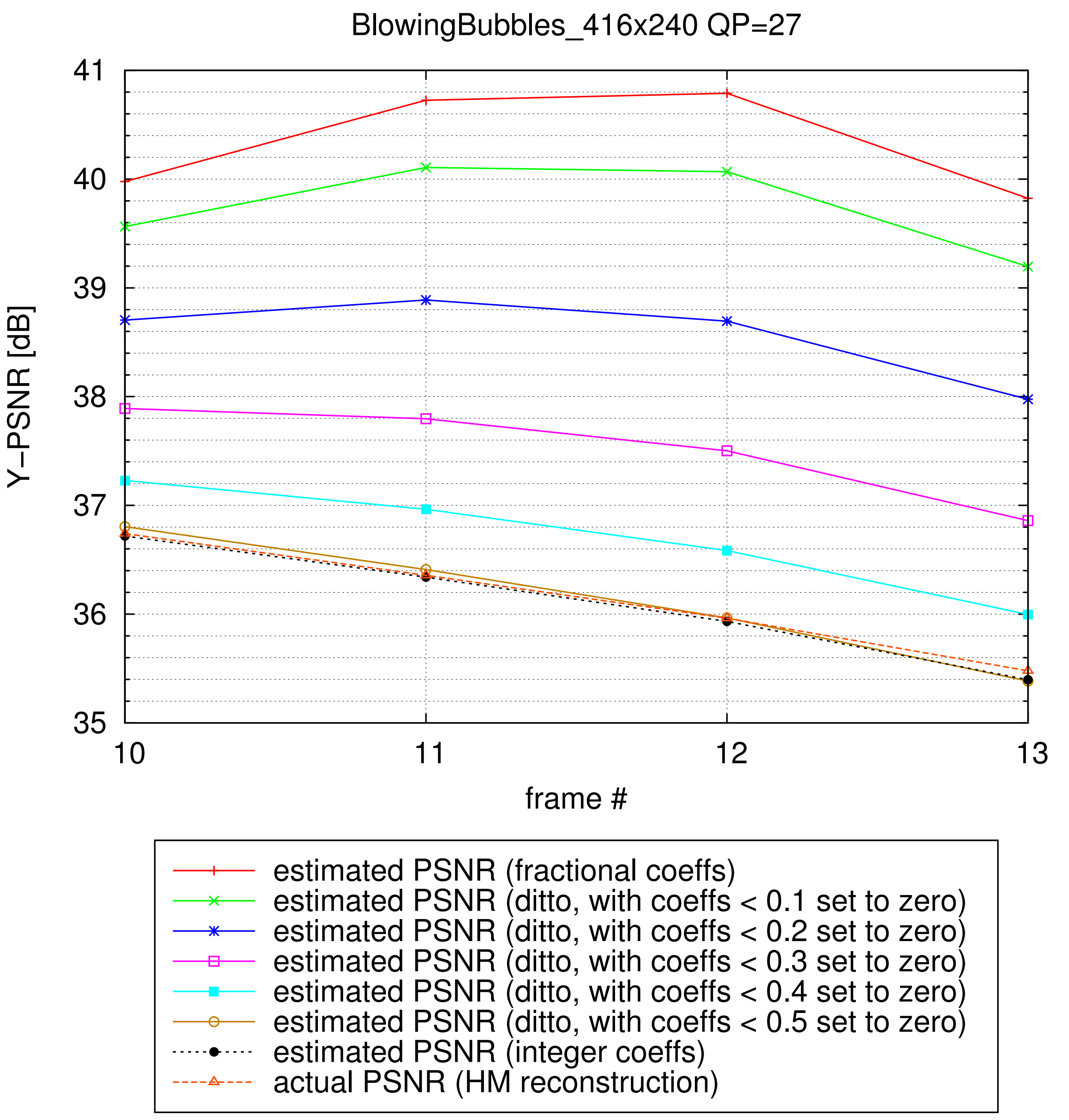


Figure 4.14: Estimated distortion from the linear signal model compared to the actual distortion from the decoded video.

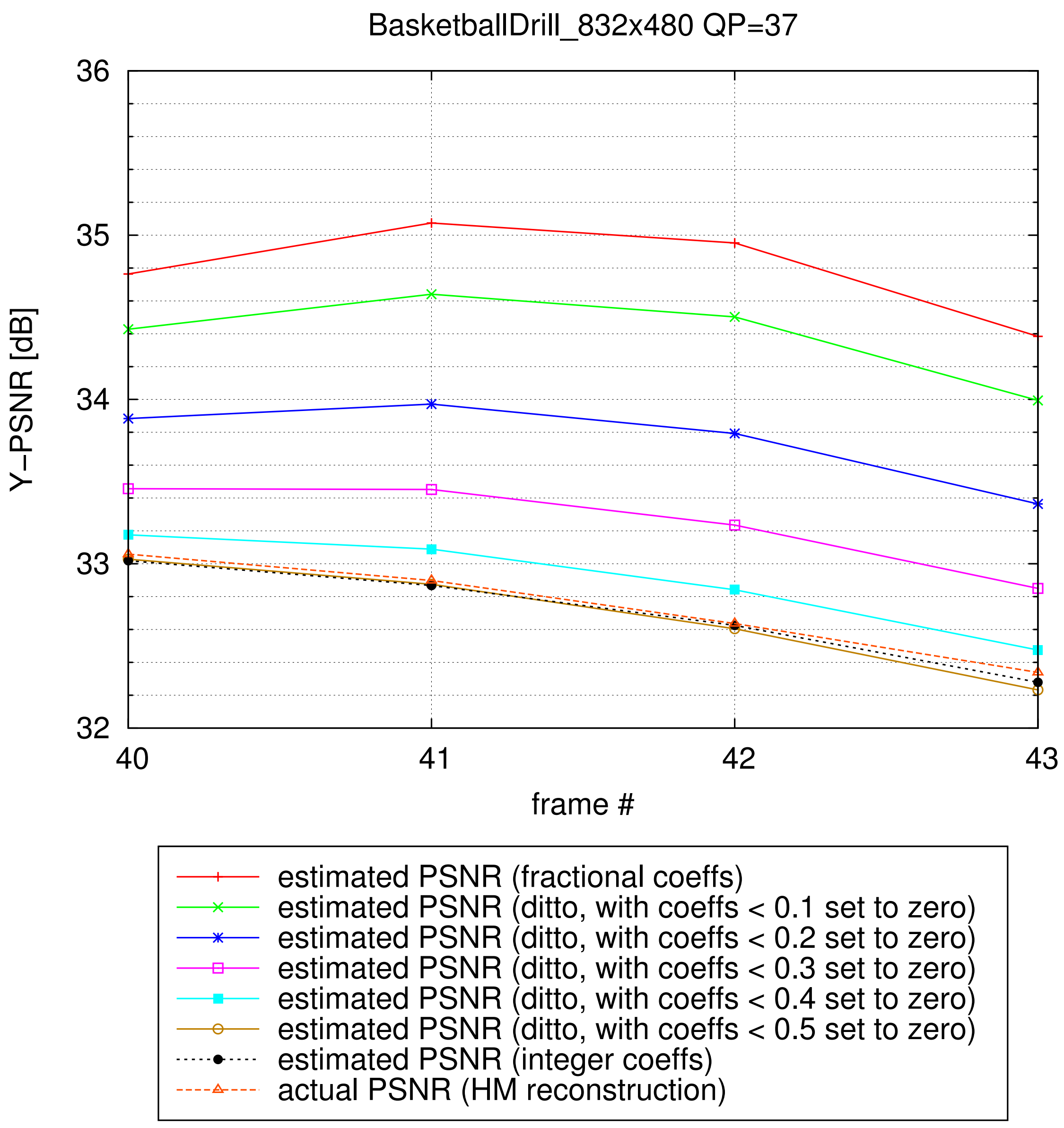


Figure 4.15: Estimated distortion from the linear signal model compared to the actual distortion from the decoded video.

Fig. 4.1, the PSNR steadily decreases from the first to last of the four frames. The red curve denoted as "estimated PSNR (fractional coeffs)" shows the distortion which results from using the real-valued transform coefficient vector $\mathbf{c}_{opt}$, as obtained from solving the optimization problem in Eq. 3.18, and reconstructing using the linear signal model. It can be seen that by using the real-valued transform coefficients, the actual PSNR is overestimated by 2–3 dB. Moreover, the fluctuation of the PSNR over the four frames is also wrongly assessed. In both examples, a PSNR increase for the second and the third frame under consideration is assumed, followed by a drop to about the same PSNR level as the first frame.

In the following, it is discovered, what leads to this huge deviation of the estimated distortion (based on the real-valued coefficients) from the actual distortion (based on integer-valued coefficients). For that investigation, five PSNR curves are derived from the real-valued transform coefficient vector $\mathbf{c}_{opt}$, where all coefficients which are smaller in absolute value than a threshold value $0.1, \ldots, 0.5$ have been set to zero. It can be seen that the curve where all vector components $c_{opt,i}$ with $|c_{opt,i}| < 0.5$ have been set to zero comes very close to the actual PSNR curve. Using a threshold of 0.4, there is still a significant gap. Note that the other vector components, which have not been set to zero, are still real-valued for these five curves. Therefore, it can be concluded, that particularly those real-valued coefficients which are smaller in absolute value than 0.5 contribute to the huge overestimation of the distortion, whereas the fractional part of the larger coefficients does not cause much harm. Finally, the curve denoted as "estimated PSNR (integer coeffs)" shows the distortion which results from using the transform coefficients where each vector component $c_{opt,i}$ has been rounded to its nearest integer (which includes setting all $c_{opt,i}$ with $|c_{opt,i}| < 0.5$ to zero). These coefficients are the same as used for the H.265/HEVC bitstreams, and therefore for the "actual PSNR" curve. It can be seen that the difference between the PSNR which is estimated using the linear signal model based on integer-valued coefficients and the actual PSNR is very small. This shows, that the linear model itself proves to be accurate, despite the negligence of non-linear effects (such as loop filtering, clipping and rounding of the reconstructed signal samples). The misassesment of the resulting PSNR is solely caused by the relaxation of the optimization problem to real-valued solutions. Since fractional coefficients which are greater than or equal to 0.5 in absolute value do not contribute much to this misassesment, it would be desirable to restrict the solution of the optimization problem Eq. 3.18, such that $c_{opt,i} \in \mathbb{R} \setminus \{x \in \mathbb{R} \mid 0 < |x| < 0.5\}$. Introducing such a constraint, however, is not a trivial task, since this would basically imply imposing an integrality constraint for a certain range of transform coefficient values. Furthermore, as shown by the experimental results presented so far (e.g., Figs. 4.2–4.5), an optimization based on the rough distortion measure calculated from the fractional coefficients still achieves coding gains for the actual distortion based on integer-valued coefficients.

4.1.9 Determination of an integer-valued solution

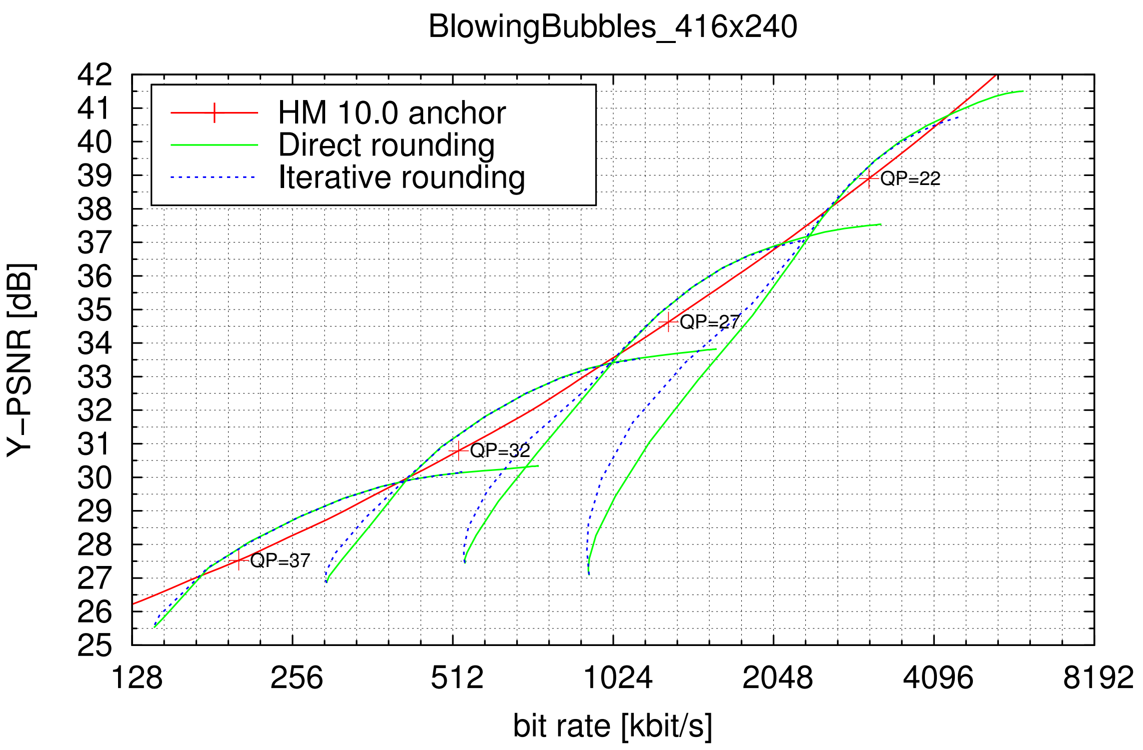
As has been shown in the previous section, the transform coefficient vector $\mathbf{c}_{opt}$ with real-valued components will lead to an overestimation of the resulting PSNR, which is caused by those coefficients $c_{opt,i}$ with $|c_{opt,i}| < 0.5$. In [SSW05], Schumitsch proposes an iterative method for obtaining an integer-valued solution. In this section, this method is studied in the context of H.265/HEVC multi-frame optimization.

The basic idea behind Schumitsch's method is to split the rounding of the transform coefficient vector $\mathbf{c}_{opt}$ into several iterations, and to round in each iteration only a subset of the transform coefficients to its closest integer value. Then, a new original signal for the next iteration is computed, such that the effect of the just rounded coefficients is considered. The optimization problem is then solved again for the remaining transform coefficients. This continues iteratively, until all transform coefficients have been rounded. Note, that this is similar in spirit to the *nulling and cancelling* method proposed in [HV05] for the solution of integer least-squares problems.

More precisely, Schumitsch proposes to round in the first iteration those coefficients $c_{opt,i}$ with $|c_{opt,i}| < 0.5$, in the second iteration those $c_{opt,i}$ with $|c_{opt,i}| < 1.5$, and so on. Generally, in iteration n with $n \in \{0, 1, 2 \ldots\}$, those coefficients $c_{opt,i}$ with $|c_{opt,i}| < n + 0.5$ will be rounded to their closest integer. Furthermore, the regularization term is also modified such that transform coefficients which are smaller in absolute value than n do not contribute, i.e.:

$$R(\mathbf{c}) = \sum_{i=0}^{K-1} \max(0, |c_i| - n) \tag{4.33}$$

Finally, the regularization parameter μ is also modified after the first iteration, i.e. the regularization parameter μ_1 which is used from iteration $n = 1$ on, is chosen as $\mu_1 = \frac{\mu_0}{4}$, where μ_0 is the regularization parameter of the first iteration ($n = 0$).



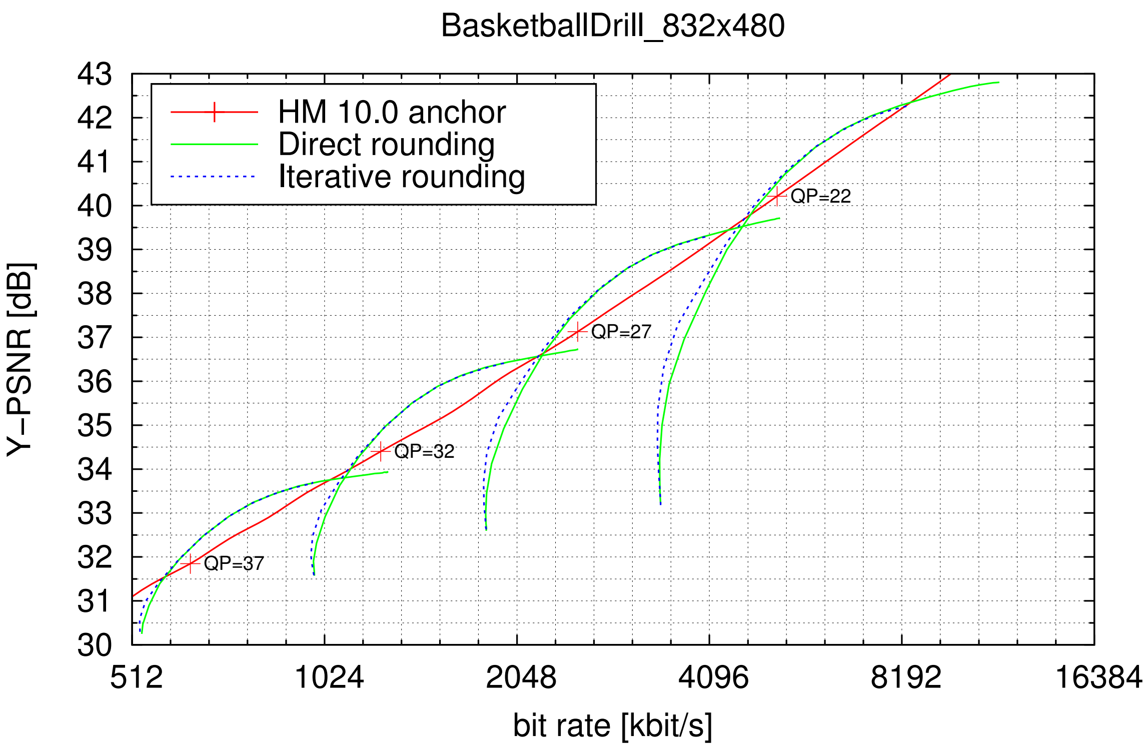


Figure 4.16: Rate Distortion results showing the regularization path using the iterative rounding method.

This method has also been implemented using the iterative shrinkage/thresholding algorithm (ISTA) framework. For that purpose, the thresholding function which corresponds to the regularization function from Eq. 4.33 has to be derived. More formally, the solution to the following proximal operator problem has to be found for $g(x) = \max(0, |x| - n)$:

$$x_{n+1} = \arg\min_x \underbrace{\frac{1}{2}(x - u_n)^2 + \frac{\mu}{\alpha_n}\, g(x)}_{h(x)} \tag{4.17 revisited}$$

$$= \arg\min_x \frac{1}{2}(x - u_n)^2 + \frac{\mu}{\alpha_n}\, \max(0, |x| - n) \tag{4.34}$$

Note that this problem is similar to the proximal operator problem for the capped ℓ_1-norm regularization (Eqs. 4.18–4.22). Accordingly, the solution can be found as

$$x_{n+1} = \begin{cases} w_1 & \text{if } h(w_1) \leq h(w_2) \\ w_2 & \text{otherwise} \end{cases} \tag{4.35}$$

with

$$w_1 = \operatorname{sgn}(u_n) \min(n, |u_n|) \tag{4.36}$$

$$w_2 = \operatorname{sgn}(u_n) \max(n, |u_n| - \frac{\mu}{\alpha_n}), \tag{4.37}$$

which is obtained from splitting Eq. 4.34 into the two problems:

$$w_1 = \arg\min_x \frac{1}{2}(x - u_n)^2 \qquad \text{s.t. } |x| \leq n \tag{4.38}$$

$$w_2 = \arg\min_x \frac{1}{2}(x - u_n)^2 + \frac{\mu}{\alpha_n}(|x| - n) \qquad \text{s.t. } |x| \geq n. \tag{4.39}$$

In Fig. 4.16, the regularization path of the iterative rounding method as proposed by Schumitsch is compared to the variant, where each real-valued transform coefficient is directly rounded to its closest integer value. Again, $N = 4$ frames have been jointly considered in the multi-frame optimization. It can be seen that while the iterative rounding method (which includes using the modified regularization term and the μ parameter variation), achieves some gains at the lower end of each regularization path (correcting too large values of μ), in the relevant region, where there is a gain over the HM anchor, the coding performance of the two methods is very similar. Note that for this experiment, again the *sign data hiding (SDH)* coding tool of H.265/HEVC has been disabled in all the three variants (HM anchor, direct rounding, and iterative rounding), because otherwise the integer-valued solution as obtained from the iterative rounding method might not be feasible due to a violation of the parity rule which is imposed when SDH is enabled. According to these results, which are representative for the whole test set, there is no benefit of using the iterative rounding method. More than that, in conjunction with SDH, a real-valued solution is actually helpful, since by the fractional part it can be decided which transform coefficient is to be modified, if the rounded solution does not match the parity rule.

4.1.10 Sign Data Hiding and multi-frame optimization

In this section, the effect of using the sign data hiding (SDH) tool of H.265/HEVC in conjunction with multi-frame transform coefficient optimization is studied. The corresponding results are shown in Tab. 4.5. In the column denoted as "A," the coding gain of the multi-frame transform coefficient optimization is shown, when SDH is enabled for the HM reference configuration, but switched off for multi-frame optimized variant. A group of $N = 4$ frames is considered jointly in the optimization, and the μ rule according to Eq. 4.2 is used. The column denoted as "B" shows the gain, when SDH is used in both the anchor and the optimized variant. The column labeled as "C" shows the difference in terms of BD bit rate between the multi-frame optimized variant without and with SDH. Note that this is not necessarily the same as the difference between the corresponding values for "A" and "B." The column denoted as "D," finally, shows the coding gain for using SDH with the anchor (i.e., the BD bit rate between an anchor configuration without and with SDH enabled). The average gain of using SDH is 1.0 % BD bit rate for the anchor (using an IPPP... prediction structure). In conjunction with multi-frame transform coefficient optimization, this drops to 0.4 %. This is the case because for the anchor and for the multi-frame optimization scenario, the adaptation of the coefficients in order to fulfill the SDH parity rule is done in different ways. In the anchor configuration, this adaptation is done by means of RDOQ, i.e. for each coefficient the effect on the actual bit rate and distortion are determined, and the coefficient with the smallest RD cost increment is modified. In the multi-frame optimization scenario, the adaptation is based on the fractional part, such that the deviation from the real-valued optimized transform coefficients is minimized. This results in a less accurate RD decision. But still, the best overall coding performance is achieved when SDH and multi-frame optimization

		BD bit rate [%]			
	Sequence	A	B	C	D
Class C (832×480)	BasketballDrill	−15.7	−16.0	−0.5	−1.0
	BQMall	−10.4	−10.5	−0.1	−0.8
	PartyScene	−12.0	−12.5	−0.5	−1.3
	RaceHorses	−6.6	−7.0	−0.4	−1.0
Class D (416×240)	BasketballPass	−8.3	−8.8	−0.5	−1.1
	BlowingBubbles	−10.9	−11.2	−0.4	−1.1
	BQSquare	−11.0	−11.4	−0.4	−1.2
	RaceHorses	−8.5	−8.8	−0.3	−0.8
	AVERAGE	−10.4	−10.8	−0.4	−1.0

Table 4.5: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) for sign data hiding (SDH), where A: gain of multi-frame optimization without SDH; B: gain of multi-frame optimization with SDH; C: BD bit rate between A and B; D: gain of SDH for the anchor configuration

	BD bit rate [%]				
Sequence	64×64+0	64×64+64	960×576+0	960×576+64	no window
BasketballDrive	−2.97	−5.40	−5.37	−5.45	−5.43
BQTerrace	−12.51	−12.84	−12.79	−12.89	−12.85
Cactus	−4.12	−4.56	−4.70	−4.59	−4.60
Kimono	−2.13	−3.45	−3.51	−3.63	−3.84
ParkScene	−7.53	−8.01	−8.04	−8.06	−8.02
AVERAGE	−5.85	−6.85	−6.88	−6.92	−6.95

Table 4.6: Rate Distortion comparison of various optimization window sizes. In the headings, $n \times n + k$ means a $(n + 2k) \times (n + 2k)$ window with an overlap of k samples.

are employed jointly.

4.1.11 Sliding window method for problem size reduction

For high resolution video sequences, the memory requirements for storing the motion matrix $\mathbf{M}$ can be too high for practical implementation. For example, for Full-HD video at 1920×1080 resolution with $N = 4$ frames in the optimization, there will be $1920 \cdot 1080 \cdot 4 \approx 8.3 \cdot 10^6$ transform coefficients. Since each inter-predicted sample may directly depend on up to 128 reference samples[1], there can be up to $1920 \times 1080 \times 3 \times 128 \approx 8 \times 10^8$ non-zero entries in $\mathbf{M}$. Note that for $N = 4$ only

[1] $8 \times 8 = 64$ because of the separable 8 tap FIR filter, $2 \times 64 = 128$ because of bi-prediction

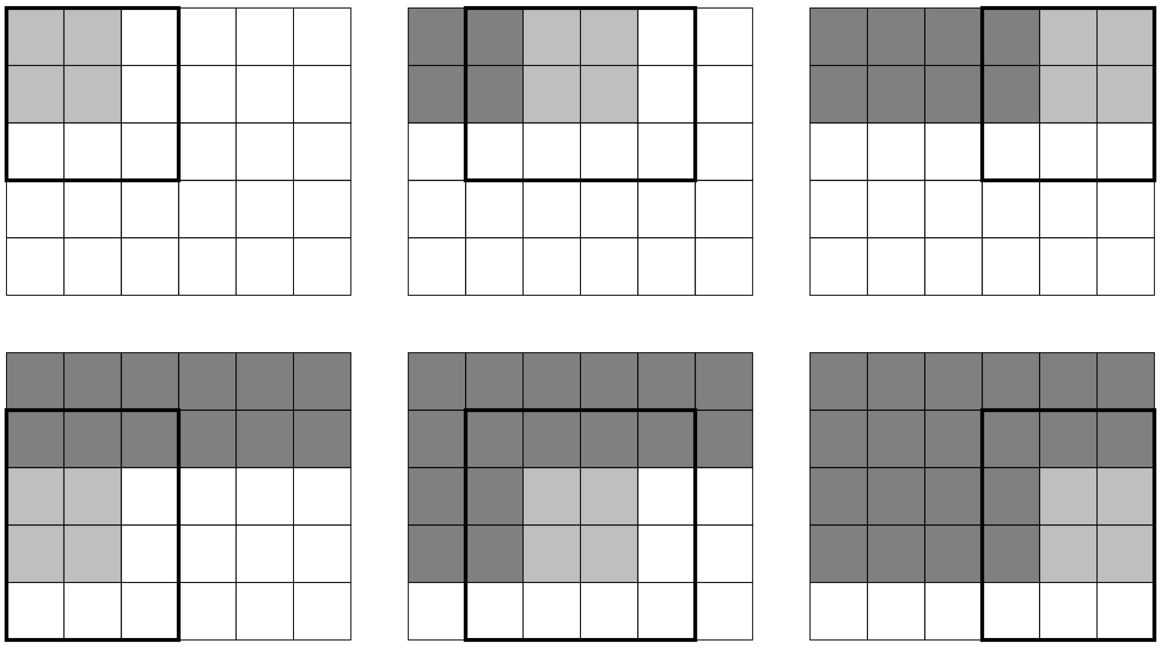


Figure 4.17: Sliding window method with optimization window (surrounded by thick lines), blocks under current optimization (light gray area) and already optimized blocks (dark gray area).

three frames have to be considered here, because the prediction signal of the first frame under consideration is assumed to be fixed and therefore not captured by the matrix $\mathbf{M}$.

In this section, a simple approach is proposed that enables application of the multi-frame transform coefficient optimization method to higher resolution video sequences and/or scenarios involving a large number N of frames considered in the optimization step (as published in [WSMW07]). For that purpose, the original optimization problem, which is too large to be solved directly, is split into a series of smaller sized sub-problems which are solved successively. This is done by usage of a spatial sliding window approach, such that each sub-problem covers a certain region of the frames to be optimized. The regions of the individual sub-problems are overlapping, as illustrated in Fig. 4.17. This is done in order to consider during the optimization of a certain region the impact of the neighboring transform coefficients. In Fig. 4.17, the first six steps of the sliding window method are shown. Each square block corresponds to a coding tree block (CTB) in H.265/HEVC. The region which is surrounded by thick lines corresponds to the optimization window. The blocks which will actually be optimized in the current step are shown in light gray, whereas the already optimized blocks of the previous steps are shown in dark gray. The transform coefficient levels of the dark gray blocks within the optimization window will also be free variables of the optimization problem in order to consider the influence of neighboring blocks. The values which are obtained for these coefficients will, however, be discarded, as they have already been determined and fixed in a previous sliding window step.

In Tab. 4.6, the BD bit rate results for the first 50 frames of the Class B (1920×1080) sequences and $N = 3$ are shown for four different optimization window sizes as well as the case where no windowing is used. The headings are denoted in the form $n \times n + k$, which corresponds to a $(n + 2k) \times (n + 2k)$ window with an overlap of k samples. With a CTB size of 64×64 luma samples, the example of Fig. 4.17 would accordingly be denoted as 128×128+64. The largest average coding gain is observed for the case with no optimization window. Using a 64×64 optimization window without overlapping results in a diminished coding gain, in particular for the high-motion sequence BasketballDrive. The behaviour for the remaining cases comes very close to the case without windowing. In the following, the variant "960×576+64" has been used for the high resolution sequences, i.e. Class A (2560×1600) and Class B (1920×1080). This variant has been chosen, because choosing the optimization window too small will result in a large number of sub-problems to be solved individually, which on the one hand reduces memory requirements, but on the other hand increases computational complexity, as the total number of ISTA iterations is higher. Therefore, a trade-off has to be made, such that the optimization window is chosen neither too small nor too large.

4.2 Multi-frame optimization in an IPPP… prediction structure

4.2.1 Analysis of the bit rate savings over the number of frames

In this section, the bit rate savings resulting from the multi-frame transform coefficient optimization are studied in more detail. For the measurement of the bit rate savings, the Bjøntegaard delta bit rate (BD bit rate) as proposed in [Bjø01] is used. The regularization parameter μ is chosen according to Eq. 4.2, the multi-frame all-zero block checking as described in Sec. 4.1.5 is employed. The number N of frames which are optimized jointly is varied in the range 2,. . . ,6. In Figs. 4.18–4.20, the bit rate savings over the number of P frames are plotted for the first 300 frames of the Class C (832×480) and Class D (416×240) sequences. Only the first 300 frames are shown, because the shortest sequence, RaceHorses, has a duration of 300 frames at both resolutions. The BD bit rate for a given numbers of P frames n has been obtained by considering the partial sub-bitstreams consisting of the first $n + 1$ frames, for both the optimized version and the HM anchor. Since the encoding of the first frame is not changed by the optimization method, all the curves start at zero BD bit rate for $n = 0$, indicating identical coding performance. For the next 2–3 frames, there is a peak, indicating a coding loss corresponding to about 10 % bit rate increase. After that, the BD bit rate curve is constantly decreasing, showing a recovery from the initial coding loss. After the first 10–20 frames, the zero line is crossed and, thus, the break even point is reached. Since the BD bit rate curve is still decreasing, the coding gain is actually increasing with the number of frames, until about frame 200, after which there is only very small additional BD bit rate improvement.

In Fig. 4.20 (bottom), the average of the BD bit rates over all the sequences is shown for $N = 2, \ldots, 6$. It can be seen that a larger value of N leads to both higher bit rate savings in the end as well as a higher initial coding loss. The number of frames needed to break even is very little affected by the value of N. The relative improvement in coding performance that comes with considering one more additional frame in the multi-frame optimization decreases with the value of N. The step from $N = 2$ to $N = 3$ results in the biggest increase in bit rate savings of about 2.5 percentage points (pp). A further increment to $N = 4$ achieves one additional pp. Increasing again to $N = 5$ still brings about 0.5 pp, whereas the step from $N = 5$ to $N = 6$ shows very little performance improvement.

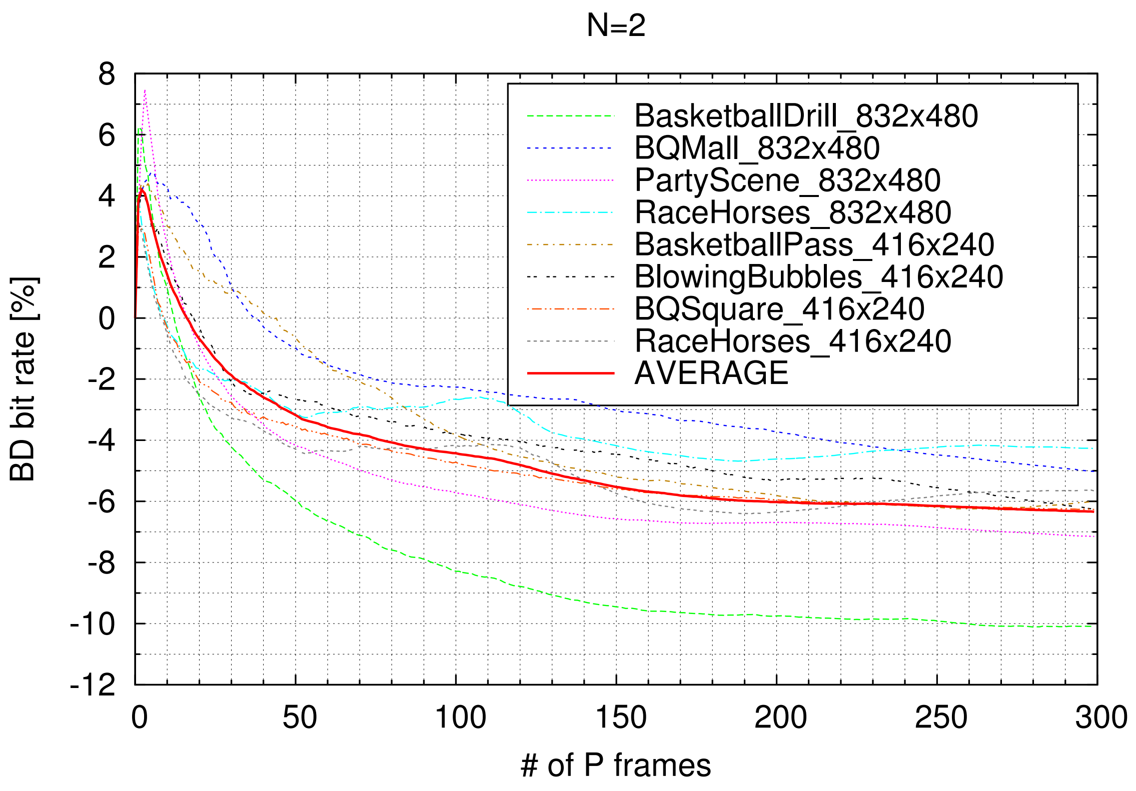
While the initial coding loss, which can be observed in all cases, might at first intuitively appear as a failure or undesirable behaviour of the algorithm, it is actually a condicio sine qua non for a functional multi-frame optimization method. If there was a consistent gain in terms of BD bit rate already from the first P frame on, then it would have been possible to achieve this gain also without consideration of the subsequent $N - 1$ frames, since the impact on these frames is not included in the BD bit rate for the first P frame. Consequently, in this case, the observed gain would, at least partially, result from correcting decisions which are suboptimal even from a single frame perspective. The initial coding loss therefore can be interpreted as an investment, which pays off after about 10–20 frames on average.

The coding experiments have been repeated for the Class B sequences (1920×1080 resolution) and $N = 2, 3, 4$. In order to reduce the memory requirements, a spatial sliding window of 960×576 samples with an overlap of 64 samples has been used. Only the first 240 frames are shown, because the two shortest sequences, Kimono and ParkScene, only have a duration of 240 frames. The resulting curves are shown in Figs. 4.21–4.22. It can be seen that the general behaviour is very similar to that of the Class C and Class D sequences. Note that the Kimono sequence has a scene change at frame 140, which explains the kink in this curve.

The corresponding curves for the Class A sequences (2560×1600 resolution) and $N = 4$ are shown in Fig. 4.23. The same optimization window parameters as for the Class B sequences have been used. For the NebutaFestival sequence, there is a coding efficiency loss of about 3 %, whereas the maximum coding gain of 18-20 % can be observed for the SteamLocomotiveTrain sequence. The poor performance for the NebutaFestival sequence can be explained from Tab. 4.7 (p. 122), where the percentage of the samples which are intra-predicted is shown for the individual sequences and QP values. The numbers are derived from the optimized bitstreams for $N = 4$ and refer only to the P frames, the initial I frame is not counted. The rightmost column denoted as “AVG” shows the average number for each sequence, across the four QP values. It can be seen that the NebutaFestival sequence has by far the largest portion of intra-predicted samples within the P frames, with more than half of the samples for $QP = 22$. A large portion of intra-predicted samples is problematic for the multi-frame transform coefficient optimization method as proposed in this thesis for two reasons:

- First, since the intra-predicted blocks are not considered in the optimization, as explained in Sec. 4.1.1, the optimization will only affect a smaller number of transform coefficients.

- Second, since the intra-predicted blocks break the temporal prediction chain, there is also less to be gained for those coefficients which are included in the optimization, because there will be a smaller number of samples referring to these.



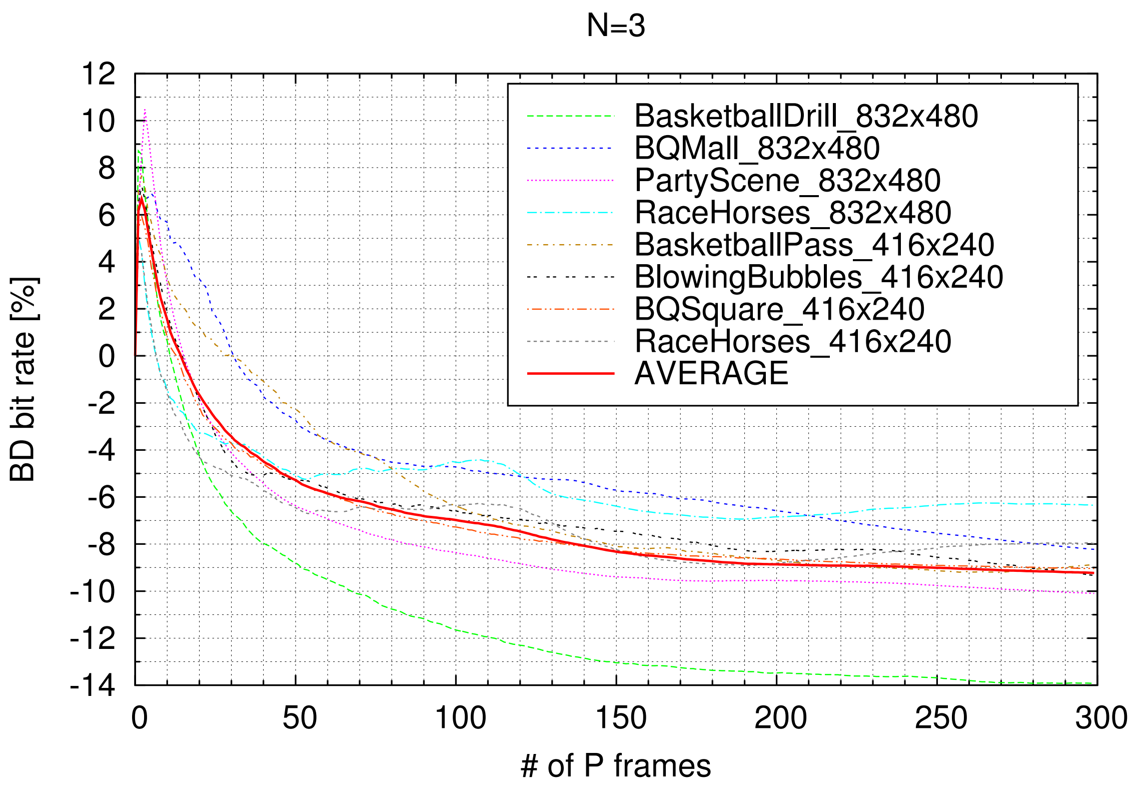
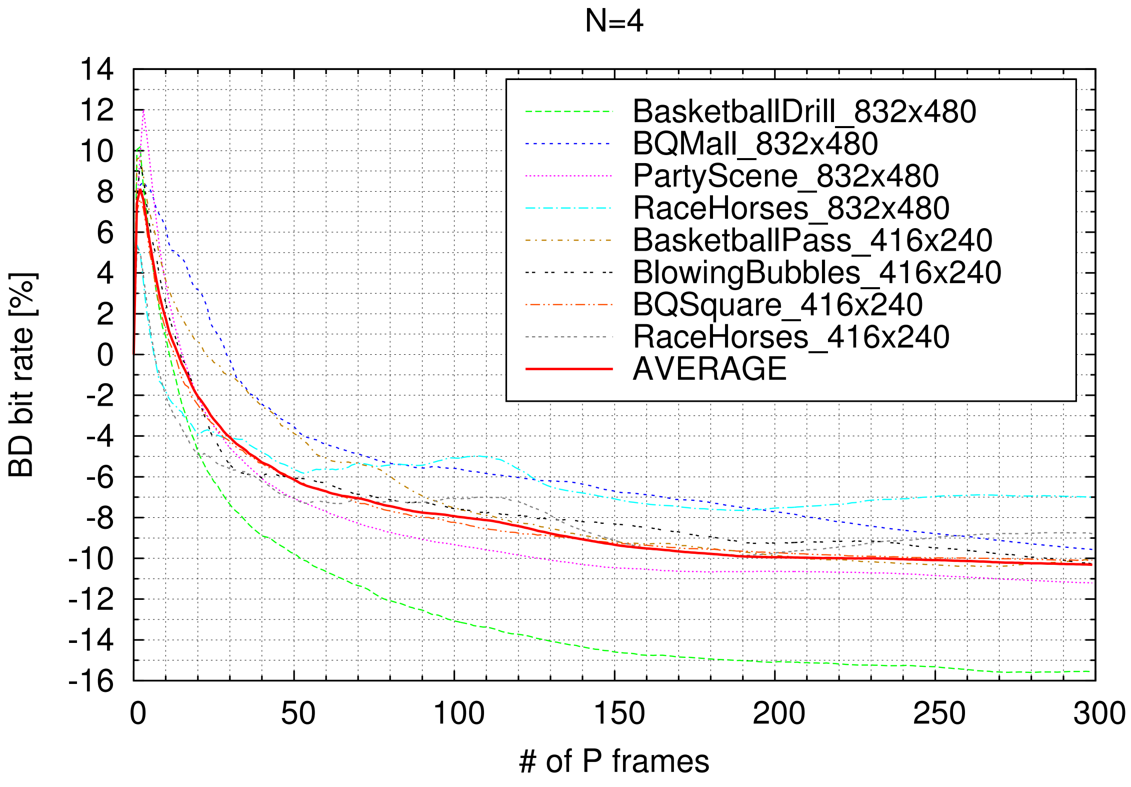


Figure 4.18: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D (top: $N = 2$, bottom: $N = 3$).



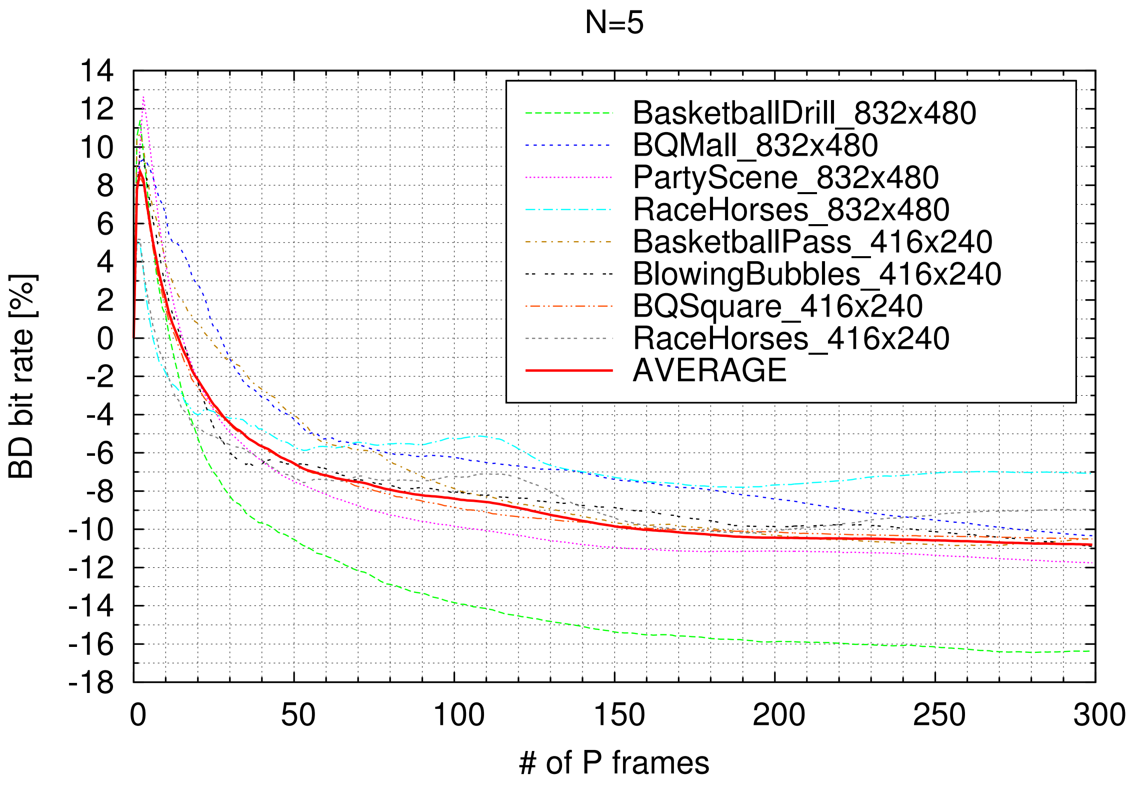
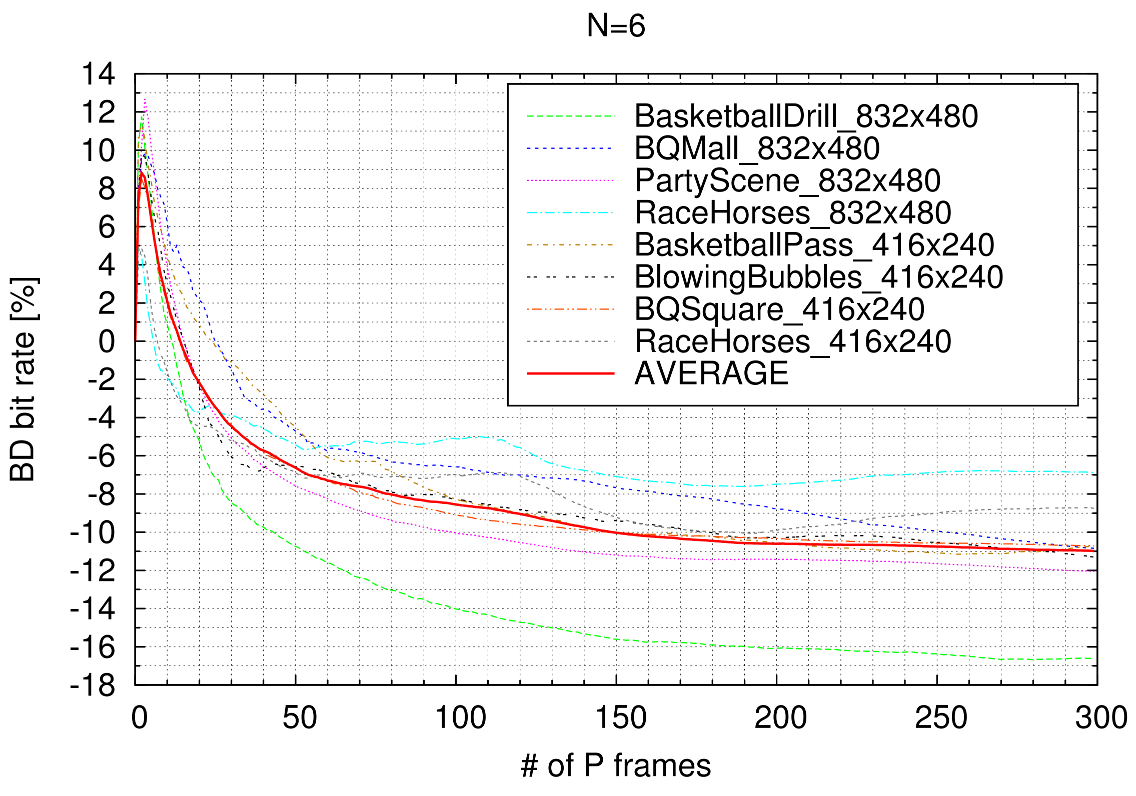


Figure 4.19: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D (top: $N = 4$, bottom: $N = 5$).



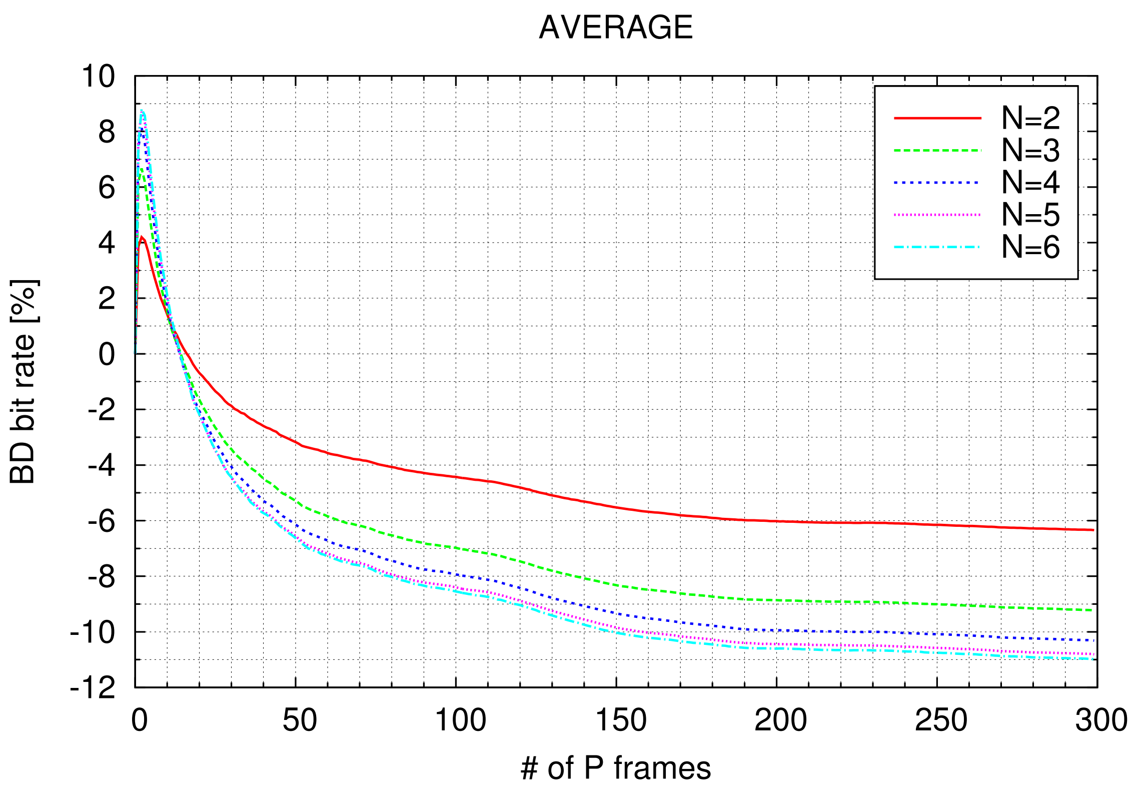
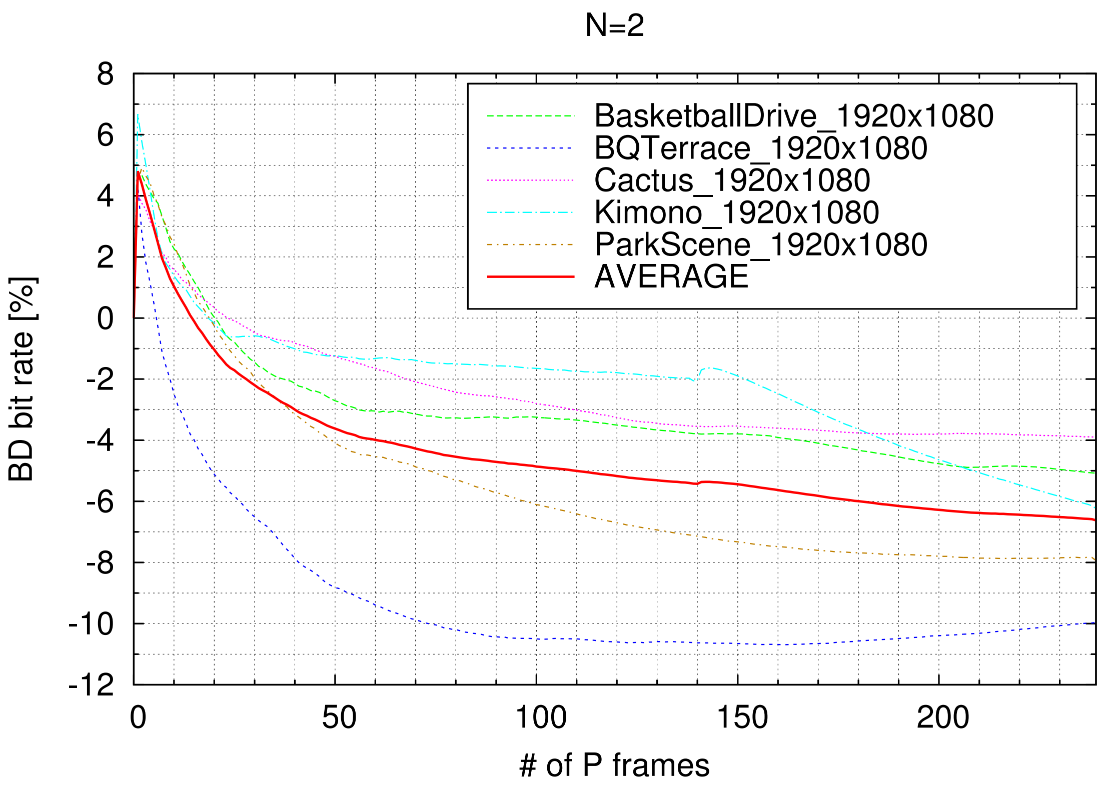


Figure 4.20: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D (top: $N = 6$, bottom: average savings for $N = 2, \ldots, 6$).



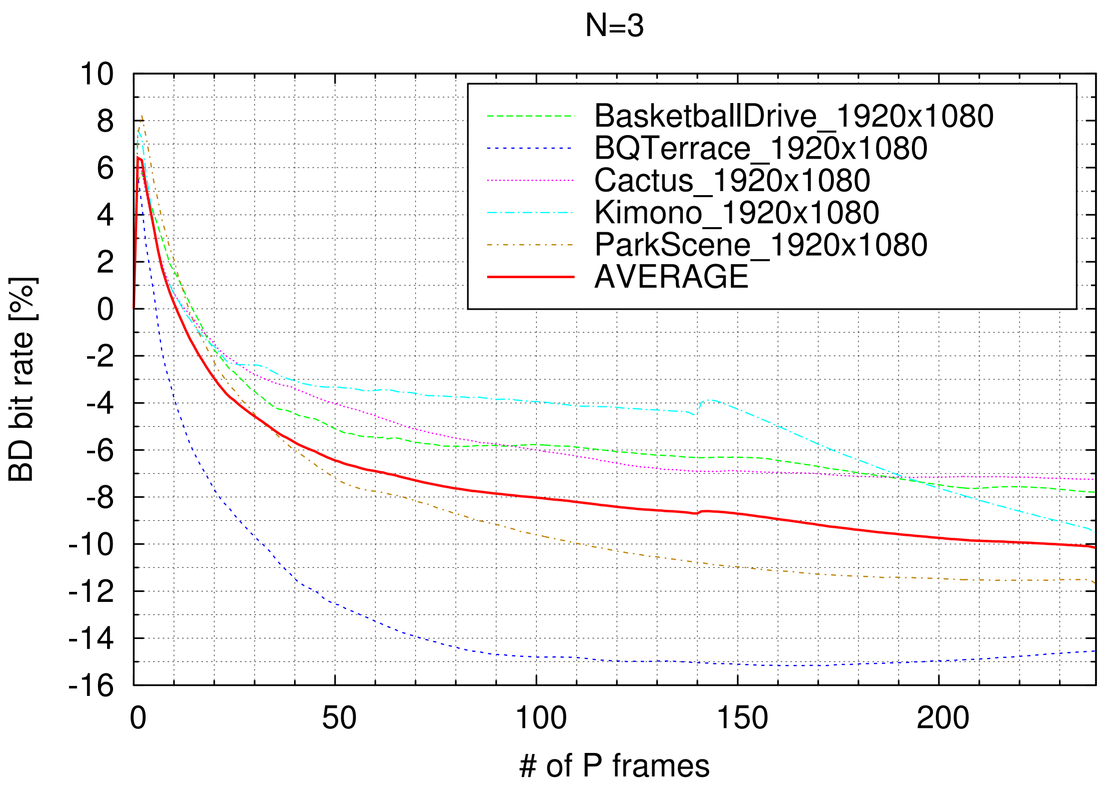
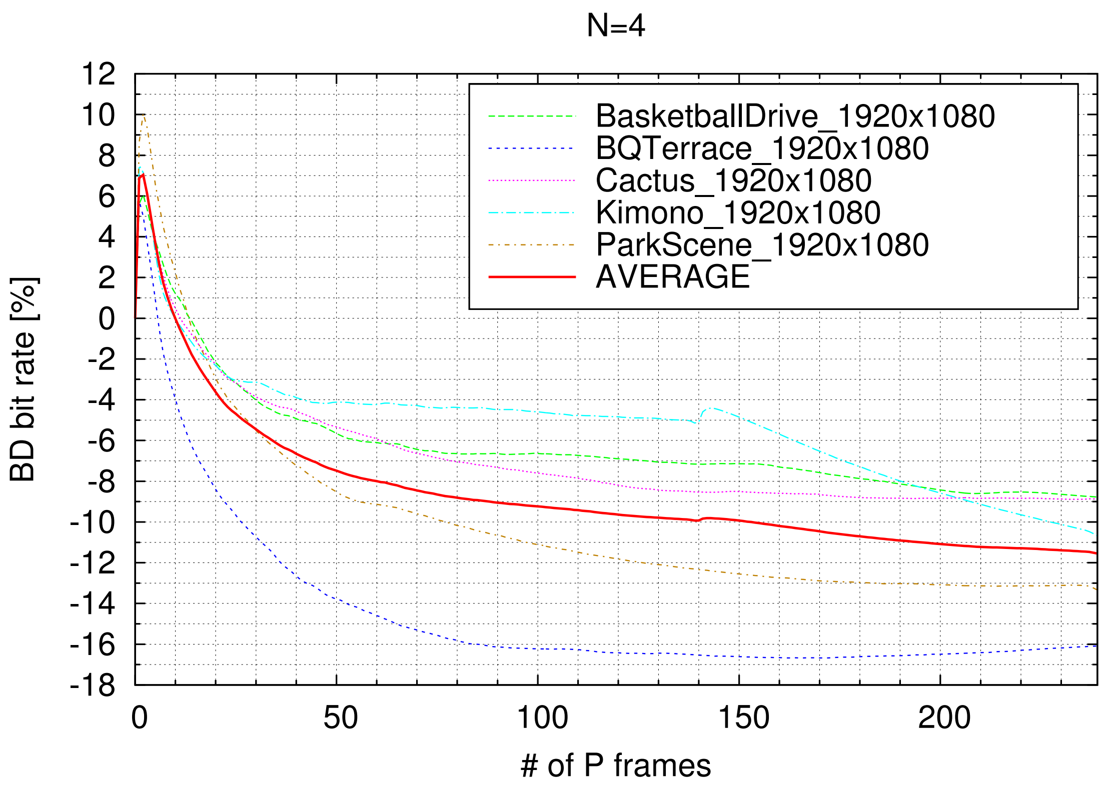


Figure 4.21: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class B (top: $N = 2$, bottom: $N = 3$).



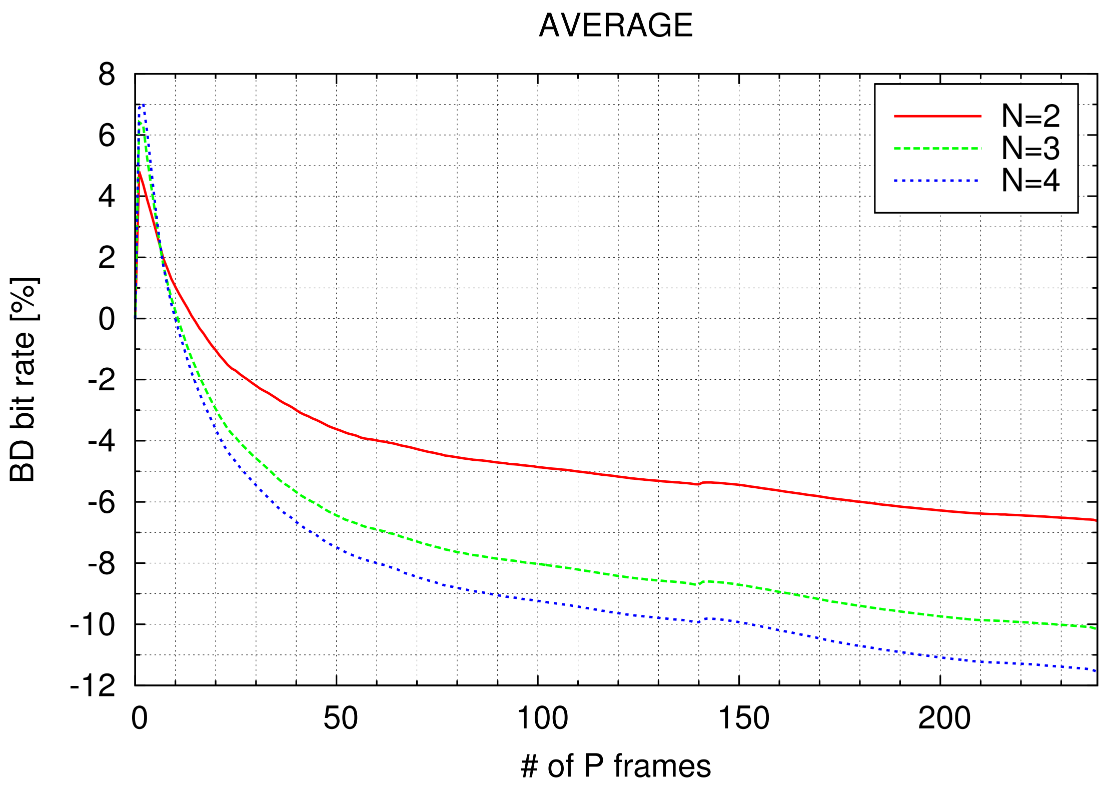


Figure 4.22: Class B bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class B (top: $N = 4$, bottom: average savings for $N = 2, 3, 4$).

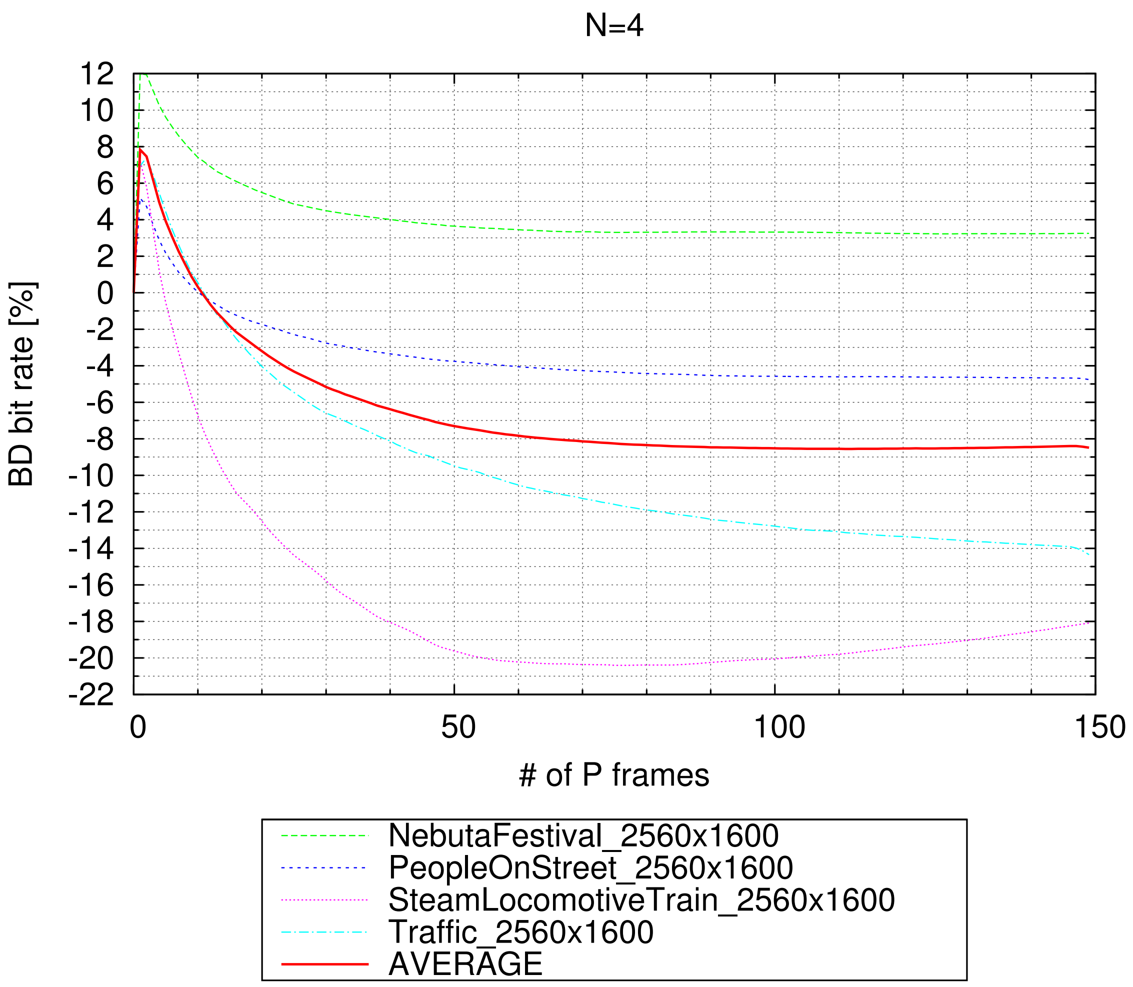


Figure 4.23: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class A ($N = 4$).

		Ratio of intra-predicted samples in P frames [%]				
	Sequence	QP=22	QP=27	QP=32	QP=37	**AVG**
Class A (2560×1600)	NebutaFestival	50.91	41.04	25.69	4.57	30.55
	PeopleOnStreet	14.28	9.09	6.53	4.76	8.66
	SteamLoco...	34.71	15.97	12.90	9.65	18.31
	Traffic	0.82	0.52	0.36	0.26	0.49
Class B (1920×1080)	BasketballDrive	13.10	8.12	6.87	6.00	8.52
	BQTerrace	23.53	0.49	0.27	0.17	6.12
	Cactus	7.21	3.57	3.31	3.25	4.33
	Kimono	4.79	3.04	2.42	2.06	3.08
	ParkScene	2.24	1.72	1.32	1.03	1.58
Class C (832×480)	BasketballDrill	7.39	6.22	5.12	4.19	5.73
	BQMall	2.61	2.11	1.78	1.50	2.00
	PartyScene	4.91	4.22	3.40	2.63	3.79
	RaceHorses	16.55	12.24	10.05	8.07	11.73
Class D (416×240)	BasketballPass	10.03	8.59	7.03	5.52	7.79
	BlowingBubbles	1.95	1.78	1.58	1.34	1.66
	BQSquare	0.05	0.03	0.03	0.04	0.04
	RaceHorses	9.78	8.14	6.42	4.53	7.22
	AVERAGE	12.05	7.46	5.59	3.50	7.15

Table 4.7: Ratio of intra-predicted samples (excluding the initial I frame).

4.2.2 Overall bit rate savings

The overall bit rate savings in terms of the Bjøntegaard delta bit rate (BD bit rate) are shown in Tab. 4.8. The results for the Class C and D sequences and $N = 4$ are the same as presented in the "multi frame" column in Tab. 4.3 and "B" column in Tab. 4.5. Note that the results refer to the full-length sequences, therefore the values might differ from the bit rate savings shown in Figs. 4.18–4.23, which show the behaviour for the initial frames of the sequences. From the Class C and D results, it can be seen that there is only modest gain, if the number of frames which are considered jointly in the optimization is increased from $N = 4$ to $N = 5$ or $N = 6$. Therefore, these results are omitted for the Class A and B sequences. For these high resolution sequences, a spatial sliding window of 960×576 samples with an overlap region of 64 samples has been used. The only case where a coding loss is observed, is the NebutaFestival sequence, which can be attributed to the large fraction of intra-predicted blocks, as discussed in the previous section. The coding results for the VCEG test set are shown in Tab. 4.9.

		BD bit rate [%]					
	Sequence	$N = 1$	$N = 2$	$N = 3$	$N = 4$	$N = 5$	$N = 6$
Class A (2560×1600)	NebutaFestival	8.1	4.7	3.5	2.9		
	PeopleOnStreet	5.2	−2.3	−4.1	−4.8		
	SteamLoco...	5.9	−5.3	−9.3	−10.8		
	Traffic	0.8	−9.1	−12.7	−14.3		
	AVERAGE	5.0	−3.0	−5.7	−6.8		
Class B (1920×1080)	BasketballDrive	6.0	−5.2	−8.1	−9.1		
	BQTerrace	5.5	−11.7	−16.5	−18.1		
	Cactus	6.4	−4.3	−7.7	−9.4		
	Kimono	5.7	−6.2	−9.5	−10.7		
	ParkScene	3.3	−7.9	−11.7	−13.3		
	AVERAGE	5.4	−7.1	−10.7	−12.1		
Class C (832×480)	BasketballDrill	−0.1	−10.5	−14.3	−16.0	−16.9	−17.2
	BQMall	4.7	−5.7	−9.0	−10.5	−11.1	−11.5
	PartyScene	4.0	−8.0	−11.3	−12.5	−13.0	−13.3
	RaceHorses	5.1	−4.3	−6.4	−7.0	−7.1	−6.9
	AVERAGE	3.4	−7.1	−10.2	−11.5	−12.0	−12.2
Class D (416×240)	BasketballPass	4.6	−5.1	−7.7	−8.8	−9.1	−9.3
	BlowingBubbles	4.4	−6.9	−10.1	−11.2	−11.8	−12.2
	BQSquare	5.6	−7.2	−10.2	−11.4	−11.8	−12.1
	RaceHorses	4.5	−5.7	−8.0	−8.8	−9.0	−8.8
	AVERAGE	4.8	−6.2	−9.0	−10.1	−10.4	−10.6

Table 4.8: Bit rate savings in terms of BD bit rate for the JCT-VC test set when using an IPPP... prediction structure.

		BD bit rate [%]			
	Sequence	$N = 1$	$N = 2$	$N = 3$	$N = 4$
VCEG 1920×1080	Crowdrun	6.0	−6.2	−8.9	−9.9
	Parkjoy	6.8	−7.1	−9.9	−10.8
	Sunflower	0.1	−13.8	−17.9	−20.2
	ToysAndCalendar	2.6	−9.5	−13.0	−14.6
	AVERAGE	3.9	−9.2	−12.4	−13.9
VCEG 832×480	Flower4	4.7	−5.0	−8.5	−9.9
	Keiba3	5.9	−1.7	−3.7	−4.2
	Nuts5	8.1	2.6	0.7	−0.6
	AVERAGE	6.2	−1.4	−3.8	−4.9
VCEG 352×288	foreman	4.0	−4.8	−7.8	−9.1
	MobileAndCalendar	1.7	−13.3	−17.4	−18.8
	Paris	1.2	−5.0	−6.6	−7.9
	Tempete	3.4	−8.8	−11.8	−12.9
	AVERAGE	2.6	−8.0	−10.9	−12.2

Table 4.9: Bit rate savings in terms of BD bit rate for the VCEG test set when using an IPPP… prediction structure.

4.2.3 Complexity evaluation

In order to assess the resulting overall complexity of the described multi-frame optimization method, a series of runtime measurements have been performed. For that purpose, the first 50 frames of the Class C and Class D test sequences have been encoded using the HM reference encoder as well as using the described optimization method with $N \in \{2, 3, 4\}$. The simulations have been performed on a workstation computer with 4 GB RAM, CPU type Intel® Xeon® X5482, running Ubuntu Linux version 11.04. For the reference configuration, HM version 10.0 has been used, the software for the multi-frame optimization is based on that version, with appropriate modifications. The individual simulations have been run successively, with no other computationally intensive jobs running in parallel. The results are shown in Tab. 4.10. The values in each row have been obtained as the averages over all the sequences within the stated class, for a given N and QP. All the values in columns denoted as "Time" are given in terms of microseconds (μs) per luma sample. In order to derive these values, the respective runtimes, which have been measured over all the 50 frames, have been divided by the total number of luma samples, i.e. $50 \times 832 \times 480$ or $50 \times 416 \times 240$, respectively. With this normalization, runtimes for different resolutions can more easily be compared. The values in columns denoted as "Factor" show the runtime as a fraction of the corresponding reference HM encoding time, e.g.

			Reference	Proposed multi-frame optimization method									
			HM Encoding	HM Encoding		Optimization (ISTA)				MF Zero Check		Total	
N	Resolution	QP	Time	Time	Factor	Time	Factor	Iters	Time/Iter	Time	Factor	Time	Factor
2	Class C (832×480)	22	36	71	1.9	17	0.5	16	1.1	137	3.7	226	6.2
		27	27	53	1.9	20	0.8	17	1.2	102	3.6	175	6.3
		32	20	40	1.9	22	1.1	16	1.4	60	2.9	122	5.9
		37	16	32	2.0	24	1.6	16	1.5	27	1.7	83	5.3
	Class D (416×240)	22	34	66	1.9	14	0.5	14	1.0	33	1.0	113	3.4
		27	26	50	1.9	16	0.7	15	1.1	24	0.9	91	3.5
		32	20	38	2.0	18	1.0	15	1.3	14	0.7	71	3.7
		37	15	29	2.0	20	1.4	14	1.5	6	0.4	56	3.9
3	Class C (832×480)	22	36	105	2.9	52	1.5	23	2.2	297	8.0	454	12.4
		27	27	78	2.8	57	2.2	23	2.4	219	7.8	354	12.8
		32	20	58	2.8	66	3.3	25	2.6	132	6.3	256	12.4
		37	16	46	2.9	72	4.6	25	2.8	64	3.9	182	11.4
	Class D (416×240)	22	34	96	2.9	46	1.4	20	2.2	77	2.3	219	6.5
		27	26	73	2.8	52	2.1	22	2.3	56	2.2	182	7.1
		32	20	56	2.8	56	2.9	21	2.6	34	1.7	145	7.5
		37	15	42	2.9	59	4.0	21	2.8	16	1.1	117	8.0
4	Class C (832×480)	22	36	139	3.8	128	3.5	31	4.0	500	13.6	768	20.9
		27	27	103	3.8	138	5.1	31	4.3	369	13.1	610	22.0
		32	20	76	3.7	146	7.3	31	4.6	221	10.5	444	21.6
		37	16	60	3.8	160	10.1	32	4.9	108	6.7	329	20.7
	Class D (416×240)	22	34	127	3.8	114	3.5	26	4.3	149	4.3	390	11.6
		27	26	97	3.7	120	4.8	28	4.3	104	4.0	321	12.5
		32	20	73	3.7	128	6.5	28	4.6	63	3.2	264	13.5
		37	15	55	3.8	134	9.0	27	4.9	30	2.0	219	14.8

Table 4.10: Complexity evaluation based on runtime measurements: "Time" is given in μs per luma sample, whereas "Factor" is given relatively to the reference HM encoding time.

a factor of two would mean twice the runtime as HM[1].

In the first data column, denoted as "Reference HM Encoding Time," the runtime of the HM encoder is shown. Note that, since these numbers are independent of N, as no multi-frame optimization is employed in this case, the same values are shown for each $N \in \{2, 3, 4\}$. It can be seen that the runtime per sample is independent of the total number of samples, or, differently stated, the runtime grows linearly with the number of samples. The runtime is higher for a smaller QP value, corresponding to a higher bit rate and a lower distortion. This may be explained by the rate distortion optimization (RDO) method employed in the operational control of the HM encoder. The RDO is based on a Lagrangian approach (see Sec. 2.2) and will therefore, for small QP, generally favor smaller block sizes, which typically require more bits to encode, but result in a smaller distortion. For QP large, however, larger block sizes will be chosen, which results in less bits being spent at the cost of a degraded reconstruction quality. Since motion estimation has to be performed for each block, and HM employs a fast encoding method using an early termination criterion, such that smaller block sizes are not considered if no further gain is expected, the resulting runtime is larger if a smaller QP value is used.

The next two columns, headed as "HM Encoding," show the runtime spent for HM in the multi-frame optimization. The significant time increase is caused by the temporal sliding window approach (as described in Sec. 4.1.1). For a total of F frames, the following holds:

- the initial I frame will be estimated[2] one time,
- the first $N-1$ P frames will be estimated $1, 2, \ldots, N-1$ times, respectively, and
- the remaining $F-N$ P frames will be estimated N times.

Consequently, on average, each frame will be estimated $\frac{1+1+\ldots+(N-1)+(F-N)\cdot N}{F}$ times.

[1]Note that the values for the "Factor" columns have also been derived as averages of the individual HM runtime factors for each sequence, and therefore may not necessarily be equal to the corresponding "Time" value divided by the reference HM encoding time, since $\frac{\frac{1}{N}\sum a_i}{\frac{1}{N}\sum b_i} \neq \frac{1}{N}\sum \frac{a_i}{b_i}$, i.e. the ratio of two averages is not identical to the average of the individual ratios.

[2]here, *estimation* referes to encoding using HM in order to determine block partitioning, prediction parameters etc.

For $F = 50$, the corresponding numbers are 1.96 for $N = 2$, 2.9 for $N = 3$, and 3.82 for $N = 4$. This matches the experimentally obtained values as shown in the column "HM Encoding Factor." Note that a speed-up could be expected if for the frames which are estimated multiple times, a warm start is used, such that the block partitioning, motion vectors etc. as obtained in the initial estimation stage are only refined in later stages.

The columns denoted as "Optimization (ISTA)" show the runtime spent for solving the multi-frame transform coefficient optimization problem Eq. 3.18, p. 30, using the iterative shrinkage/thresholding algorithm (ISTA). The two columns "Time" and "Factor" again show the absolute time in μs per luma sample as well as relatively to the reference HM runtime. The column "Iters" shows the average number of ISTA iterations until the termination criterion was fulfilled. The column "Time/Iter" shows the duration of each iteration in μs, again normalized per number of luma samples. Since the values in this column are almost identical for corresponding cases of the Class C and Class D sequences, it is found that the duration of each ISTA iteration grows linearly with the number of samples. Furthermore, the duration is higher for higher QP values, which is opposed to the behaviour of HM. This is caused by two effects:

- As stated above, for higher QP values typically larger (transform) block sizes will be used. Since for simplicity of the implementation, the inverse transform and its transpose are calculated in matrix form during the optimization, not exploiting any regular structure using butterfly computation steps etc., the total number of multiplications and additions is significantly increased for higher QP.

- The portion of intra-predicted blocks, which are not considered in the optimization, will also be lower for higher QP values, such that more blocks have to be processed.

The average runtime per iteration grows with N, which is as expected, since for a larger number of frames under consideration, both the inverse transform matrix $\mathbf{T}$ and the prediction matrix $\mathbf{M}$ (see Sec. 3.1.1 and 3.1.2) will be of larger dimension, leading to a higher number of multiplications per iteration.

The number of ISTA iterations required for convergence is approximately 10–20 % larger for the higher resolution Class C sequences, which have four times the number of samples as the Class D sequences. The number of samples therefore only has modest impact on the number of iterations. To a greater extent, the number of iterations is affected by the number N of frames considered jointly in the multi-frame optimization, indicating that the distance of the optimal transform coefficient vector $\mathbf{c}_{opt}$ to the warm start initialization vector, which are the transform coefficients as obtained from HM, is larger for a greater value of N. This is consistent with the previous observation that a larger N generally leads to improved rate distortion performance.

In the columns denoted as “MF Zero Check,” the runtime spent for the computation of the multi-frame distortion differences $\Delta D_{subsequentFrames,zero}$, as introduced in Eq. 4.8, p. 94, is shown. The runtime per luma sample grows by a factor of 3.5–4.5 if the number of samples is increased by a factor of 4, comparing the Class C and Class D results. In absolute terms, this corresponds to a growth by the square of the total number of samples. In the computation of the distortion difference, for each Coding Unit (CU), the impact on the subsequent frames of setting this CU to zero has to be determined. This requires, for each CU, performing a motion-compensation for all the samples of the CU as well as for all, directly or indirectly, referring samples in the subsequent frames. Therefore, if the number of luma samples per frame is increased, not only a larger number of calculations has to be made, corresponding to a larger number of CUs, but also each individual computation requires more multiplications, because the motion-compensated signal is obtained by multiplication by the prediction matrix $\mathbf{M}$. For large QP values, the runtime spent for this computations is significantly lower, because in this case, a large portion of CUs will have all-zero transform coefficients in the solution vector $\mathbf{c}_{opt}$, and therefore nothing has to be computed at all, because the difference of the resulting distortions for two identical transform coefficient vectors is obviously equal to zero. Note that, if time is scarce, the computation of these distortion differences could also be omitted, resulting in a reduced coding efficiency by about one percentage point BD bit rate, relative to the HM reference. according to Tab. 4.3.

In the last two columns, the total runtime in μs per luma sample as well as the relative time increase with respect to the reference HM encoding runtime is shown. It can be seen, that for a typical setting of $N = 4$, with multi-frame zero block checking enabled, the total runtime is increased by a factor of about 20 relative to HM. If the multi-frame zero block checking is disabled, the relative runtime increase will be in the order of 10.

Frame in coding order	POC (display order)	relative reference frames	relative referring frames	no. of referring frames	QP offset
0	0		1, 2, 3, 4, 5, 7, 9	7	0
1	8	-1	1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16	13	1
2	4	-1, -2	1, 2, 3, 4, 5, 6, 7	7	2
3	2	-1, -2, -3	1, 2, 3	3	3
4	1	-1, -2, -4		0	4
5	3	-2, -3, -4, -5		0	4
6	6	-3, -4, -5	1, 2, 3, 4, 5	5	3
7	5	-1, -5, -6, -7		0	4
8	7	-2, -6, -7		0	4
⋮	⋮	⋮	⋮	⋮	⋮
$8n+1$	$8n+8$	-3, -7, -8, -16	1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16	13	1
$8n+2$	$8n+4$	-1, -4, -9	1, 2, 3, 4, 5, 6, 7	7	2
$8n+3$	$8n+2$	-1, -2, -5, -10	1, 2, 3	3	3
$8n+4$	$8n+1$	-1, -2, -11		0	4
$8n+5$	$8n+3$	-2, -3, -4, -12		0	4
$8n+6$	$8n+6$	-3, -4, -5	1, 2, 3, 4, 5	5	3
$8n+7$	$8n+5$	-1, -5, -6, -14		0	4
$8n+8$	$8n+7$	-2, -6, -7		0	4

Table 4.11: Prediction structure used for the random access scenario.

4.3 Multi-frame optimization in a random access prediction structure

In this section, the multi-frame optimization method is studied for the random access (RA) configuration as defined in the common test conditions [JCT13]. For this configuration, a rather complicated prediction structure with a group of pictures (GOP) consisting of 8 frames is prescribed. The inter-frame dependencies are shown in Tab. 4.11. The first column shows the frame number in coding order. The second column shows the picture order count (POC), which is the frame number in display order. The third column shows the reference frames which can be used by the respective frame for motion-compensated prediction. Note that these values are given as offsets in coding order. Consequently, there are only negative values, since a frame cannot refer to a future frame in coding order. The fourth column shows those frames which can refer to the respective frame by motion-compensated prediction. Again, the values are given as offsets in coding order. Since a frame can only be referred by future frames in coding order, these values are all positive. The fifth column shows the number of frames which may directly predict from the given frame. The last column shows the offset that is applied to the quantizer parameter (QP) for the encoding of this particular frame. The individual GOPs are separated by horizontal bars. The differences between the rows in generic format (e.g., frame number $8n + 1$) and those of the first GOP (e.g., frame number 1) are due to the irregularity which is caused by the initial I frame, where two frames whose POC numbers are integer multiples of 8 are encoded consecutively, which only occurs at the beginning of the sequence.

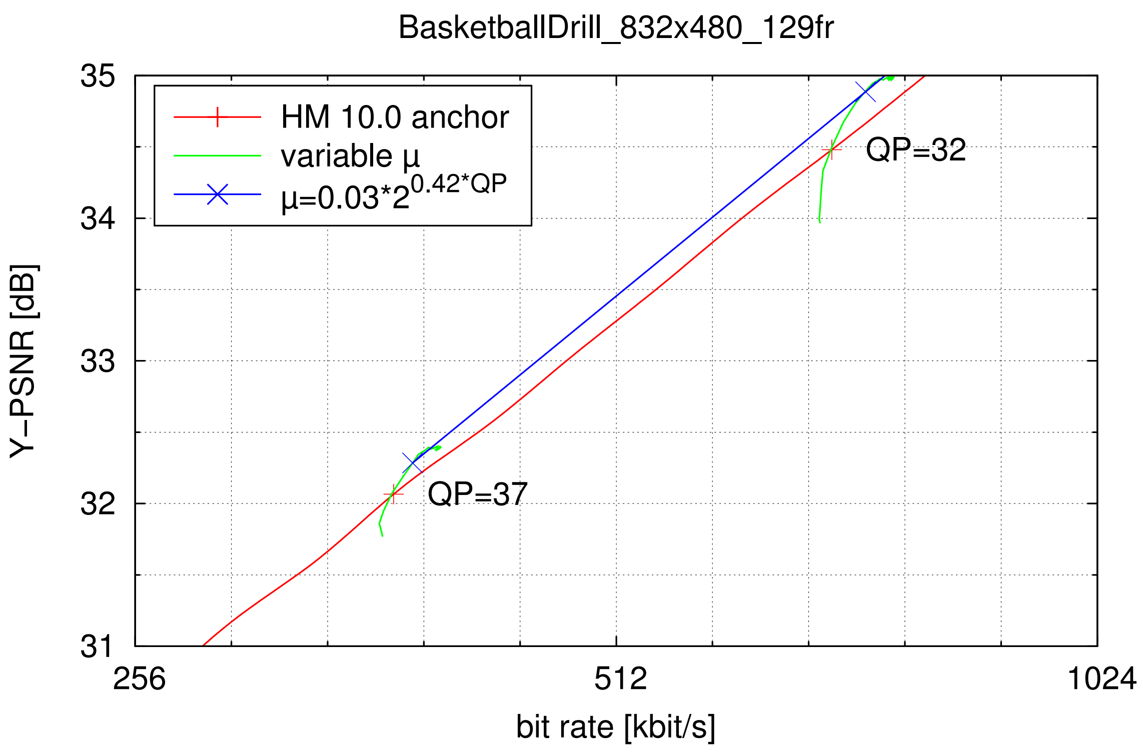
It can be seen, that every second frame in display order (i.e., those frames having an odd POC number), is a *non-reference frame*, i.e. they are not used as prediction reference by other frames. Those frames whose POC numbers are an integer multiple of 8 can be directly referenced by the largest number of frames. This is reflected by a modulation of the QP, where those frames which can be referenced by more frames will be encoded with a smaller QP value. This variation of the QP within a GOP, which is prescribed in the common test conditions, can be interpreted as a simple bit allocation method, where the frames at the top of the prediction hierarchy (i.e., those frames which are referenced most often) receive a larger share of the total bit-budget than those at the bottom, which are the non-reference frames, with a smooth transition for the frames in between these two extreme cases.

In order to allow for random access, an I frame is encoded at a regular interval for a frame with POC $= 8n$ (where $n \in \{1, 2, \ldots\}$). According to the common test conditions [JCT13], the length of the interval depends on the frame rate of the sequence and is chosen such that the temporal distance between two I frames is about one second. The intra period for a sequence with 24 frames per second (fps) is 24 frames, for 30fps it is 32 frames, for 50fps it is 48 frames, and for 60fps it is 64 frames. For the I frames, always a QP offset of zero is used. The I frames are encoded as random access points, which means that no inter-predicted frame which follows the I frame in display order uses other frames which are transmitted before the I frame as prediction reference.

4.3.1 Optimization of the frames with POC = $8n$

In this section, the multi-frame transform coefficient optimization is applied only to the frames with POC = $8n$. The transform coefficients for all the other frames are determined using the HM reference encoder with rate distortion optimized quantization (RDOQ) enabled. Again, a temporal sliding window approach is pursued, i.e. after optimization of the frames with POC = $8n$, the coding decisions for the subsequent frames are redetermined, based on the optimized transform coefficients for the "key frame" having POC = $8n$. Obviously, there is no point in applying the multi-frame optimization to the frames with POC = $2n + 1$, since these are non-reference frames, and therefore there are no inter-frame dependencies which could be exploited. Since the other three reference frames within the GOP8 prediction structure (i.e., those frames with POC = $8n + 4$, POC = $8n + 2$, and POC = $8n + 6$) differ a lot from the frames with POC = $8n$ according to their position in the prediction hierarchy, their number of referring frames, and their QP offset, the behaviour of the latter is studied isolated here. For the experiments conducted in this section, the first 129 frames (corresponding to 16 complete GOPs) of the Class C and D sequences of JCT-VC test set are used. Note that, in order to evaluate the performance of the multi-frame optimization over a large number of frames, only the first frame is encoded as an I frame, and the remaining 128 frames are encoded as B frames. The regularization parameter μ is chosen according to Eq. 4.2, where the actual QP value of the individual frame to be optimized is used (i.e., considering the QP offset as shown in Tab. 4.11). In Figs. 4.24–4.25, the rate distortion plots for two exemplary sequences are presented. The regularization path as obtained by sweeping the regularization parameter μ as well as the operating points resulting from applying the μ rule of Eq. 4.2 are shown together with the HM anchor curve. It can be seen that the μ rule, which has been derived in an IPPP... coding scenario, is also appropriate for the GOP8 case.

In a first experiment, the impact only on the reference frames within the same GOP is taken into account for the multi-frame optimization, i.e. four frames are considered jointly: For the optimization of the frame with POC = $8n{+}8$, the impact on the frames with POC = $8n{+}4$, POC = $8n{+}2$, and POC = $8n{+}6$ is considered. This experiment is denoted as "1 GOP" in the following, and the fluctuation of the BD bit rate over the frames is shown in Fig. 4.26 (top). Note that x-axis of the diagrams in Figs. 4.26–4.28 refers to frames in coding order. Again, similar to the results from Sec. 4.2.1, there is an initial coding loss at the beginning of the sequence. After 16 GOPs (or 129



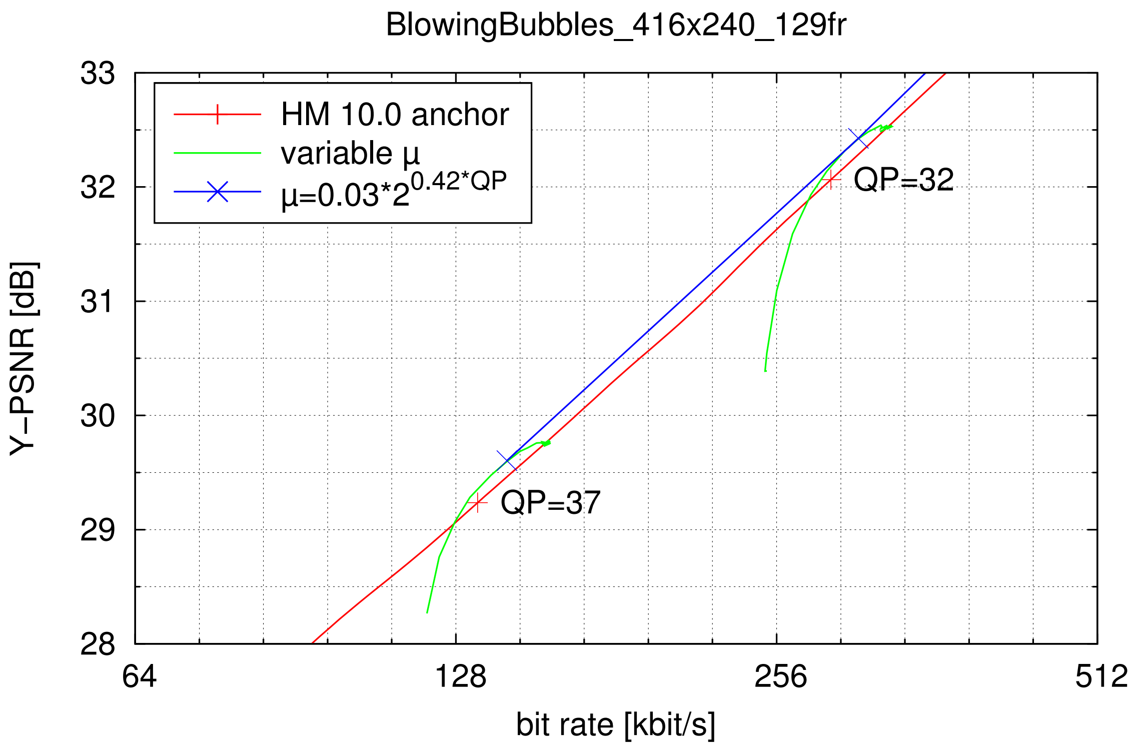
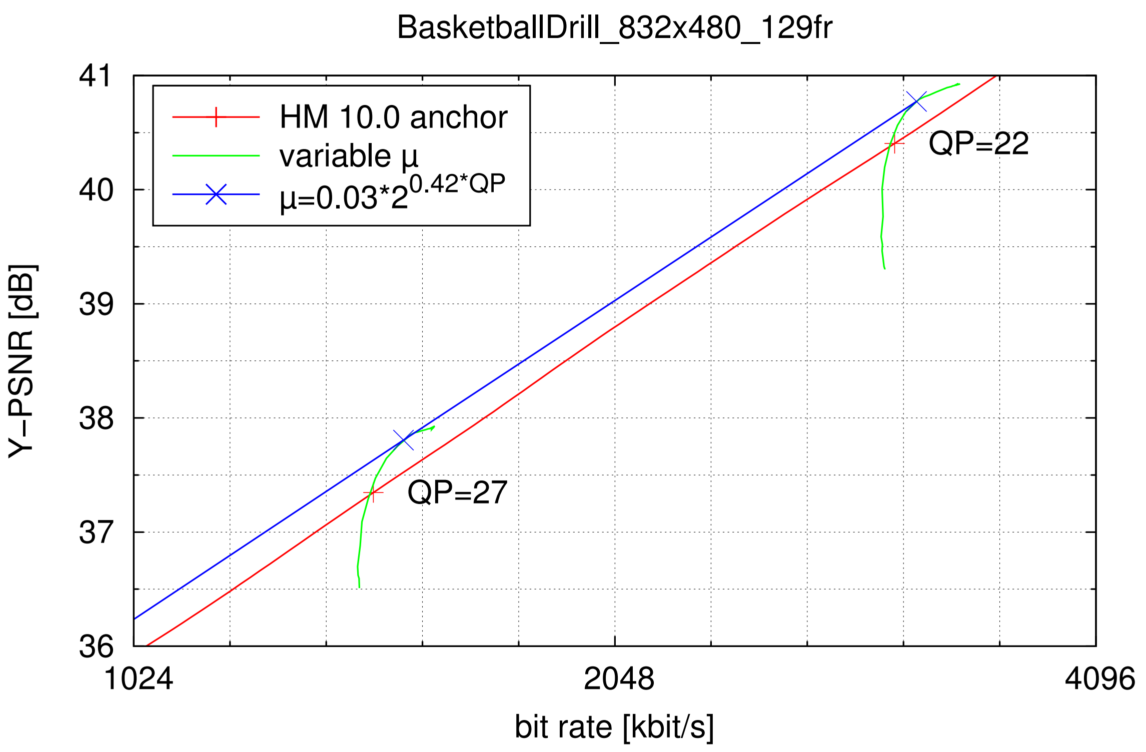


Figure 4.24: Rate Distortion curves showing the operation points according to the μ rule as well as the regularization path for QP 32 and 37.



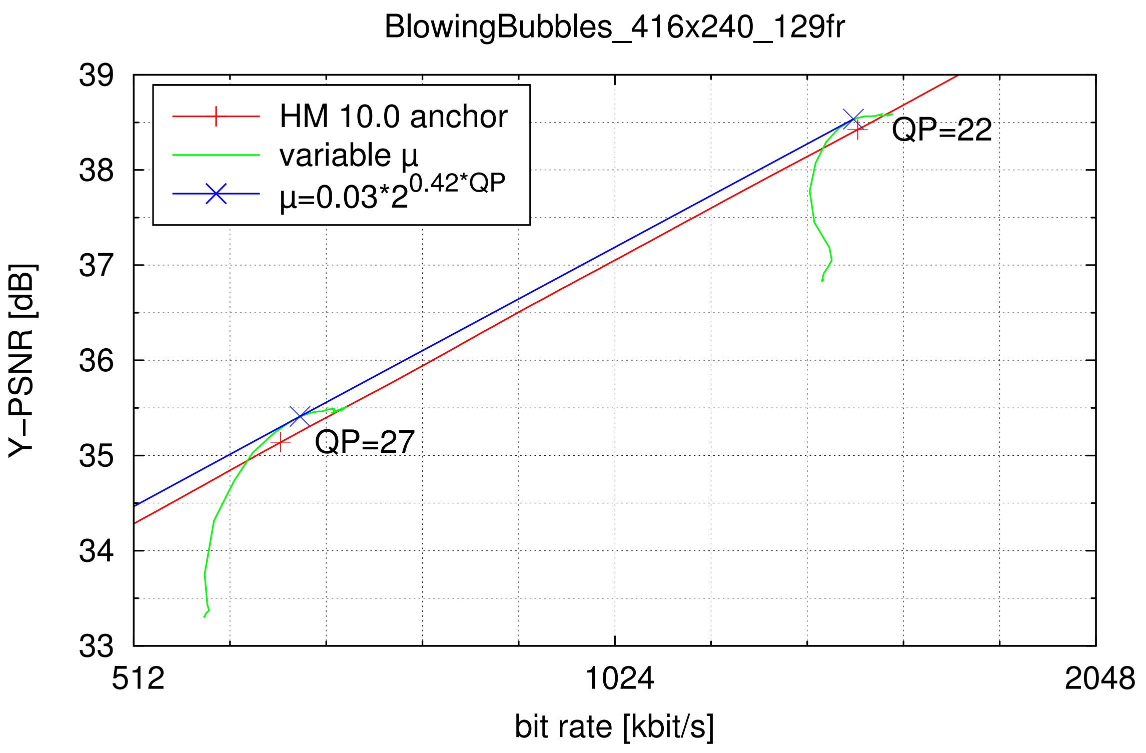


Figure 4.25: Rate Distortion curves showing the operation points according to the μ rule as well as the regularization path for QP 22 and 27.

frames), there is an average bit rate saving of 0.5 % BD bit rate, with a maximum of 2.8 % for the BasketballDrill sequence. The worst behaviour is observed for the RaceHorses sequence at 416×240, where there is a coding loss (bit rate increase) of 1.2 %. Other than in Sec. 4.2.1 (Figs. 4.18–4.23), there is not a steady decrease of the BD bit rate, corresponding to increasing bit rate savings over the number of frames. Here, the curves are somewhat “jagged,” with peaks at the frame positions $8n + 1$ in coding order (which corresponds to POC = $8n + 8$ in display order). This shows that the multi-frame optimization of these frames leads to transform coefficients which are sub-optimal from a single frame perspective, but the remaining frames within each GOP benefit from this “investment,” and the BD bit rate decreases.

In a second experiment, the impact on the non-reference frames is also taken into account, which means that all the eight frames with POC = $8n + 1, \ldots, 8n + 8$ are considered jointly for the optimization of the transform coefficients for the frame with POC = $8n + 8$. This experiment is denoted as “1 GOP (with non-refs),” and the BD bit rate results are shown in Fig. 4.26 (bottom). By inclusion of the non-reference frames into the joint optimization, the average bit rate savings after 129 frames have been increased from 0.5 % to 1.2 % BD bit rate. The maximum coding gain, which again occurs for the BasketballDrill sequence, is now 4.1 % (instead of 2.8 % as before). This improvement in coding performance comes at the cost of increased complexity, since the number of frames under consideration has been doubled.

In the next experiment, denoted as “2 GOP,” again eight frames are considered jointly in the multi-frame optimization. But differing from the previous experiment, the non-reference frames again have been excluded and instead the impact on the three reference frames of the current and the four reference frames of the subsequent GOP is taken into account. The results are shown in Fig. 4.27 (top). It can be seen that the coding performance has been further improved to an average bit rate reduction of 2.5 % after 129 frames. In comparison with the “1 GOP (with non-refs)” experiment, it can be concluded, that if the total number of frames under consideration is limited to be not greater than eight, it is favorable to consider the impact on the reference frames of the subsequent GOP, rather than the non-reference frames of the current GOP. There are two aspects contributing to this behaviour. First, the non-reference frames are “dead ends” in the prediction chain, such that spending effort to improve their reconstruction quality will only affect these frames themselves, whereas an improved reconstruction quality of a reference frame benefits all their (direct and indirect) referring frames. Second, consideration of the following GOP provides a larger look-ahead, which enables to incorporate the impact on a larger number of referring frames into the optimization problem.

In a variant of the last experiment, denoted as “2 GOP (with non-refs),” the behaviour is studied if the eight frames of the current GOP and the four reference frames of the subsequent GOP are considered jointly. Therefore, the optimization problem consists of totally 12 frames in this case. Note that in terms of computational complexity this cannot be directly compared to a setting with $N = 12$ in the IPPP... scenario of Sec. 4.2, since, due to the four non-reference frames, the longest prediction chain here has a length of 7, whereas for $N = 12$ and IPPP..., it would be 11. As has been discussed in Sec. 4.1.1 and shown in Sec. 3.4.6, the length of the longest prediction chain has great impact on the overall computational complexity of the iterative shrinkage/threshold algorithm (ISTA). The results for “2 GOP (with non-refs)” are shown in Fig. 4.27 (bottom). The average coding performance is only slightly improved compared to the previous experiment (“2 GOP”). From the results shown in Fig. 4.26 and Fig. 4.27 it can be concluded that, while in a scenario where only a single GOP is considered, the overall coding performance can be improved by inclusion of the non-reference frames into the multi-frame optimization, in a scenario where the current and the subsequent GOPs are consider, the impact on the non-reference frames can be neglected.

Finally, in an experiment is denoted as “3 GOP,” see Fig. 4.28 (top), all the reference frames of the current and the two subsequent GOPs are considered jointly, leading to a multi-frame optimization problem of 12 frames total, with a longest prediction chain of 11 frames. Again, the coding performance has been further improved to an average bit rate reduction of 2.9 % after 129 frames, with a maximum of 5.8 % for the BasketballDrill sequence and a minimum of 0.0 % for the RaceHorses sequence at 416×240 resolution.

A comparison of the sequence-wise average BD bit rates for all the five discussed experiments is shown in Fig. 4.28 (bottom). Similar as in Sec. 4.2.1, a larger initial bit rate increase coincides with a larger overall bit rate reduction at the end. The experiments denoted as “1 GOP,” “2 GOP,” and “3 GOP” have been repeated for the Class B (1920×1080) sequences. The corresponding results are shown in Figs. 4.29–4.30. Qualitatively, the general behaviour is very similar to the one observed for the Class C and D sequences. Quantitatively, the average values benefit from the good performance for the BQTerrace sequence, where a bit rate reduction of 8.7 % is measured after 129 frames for the “3 GOP” case.

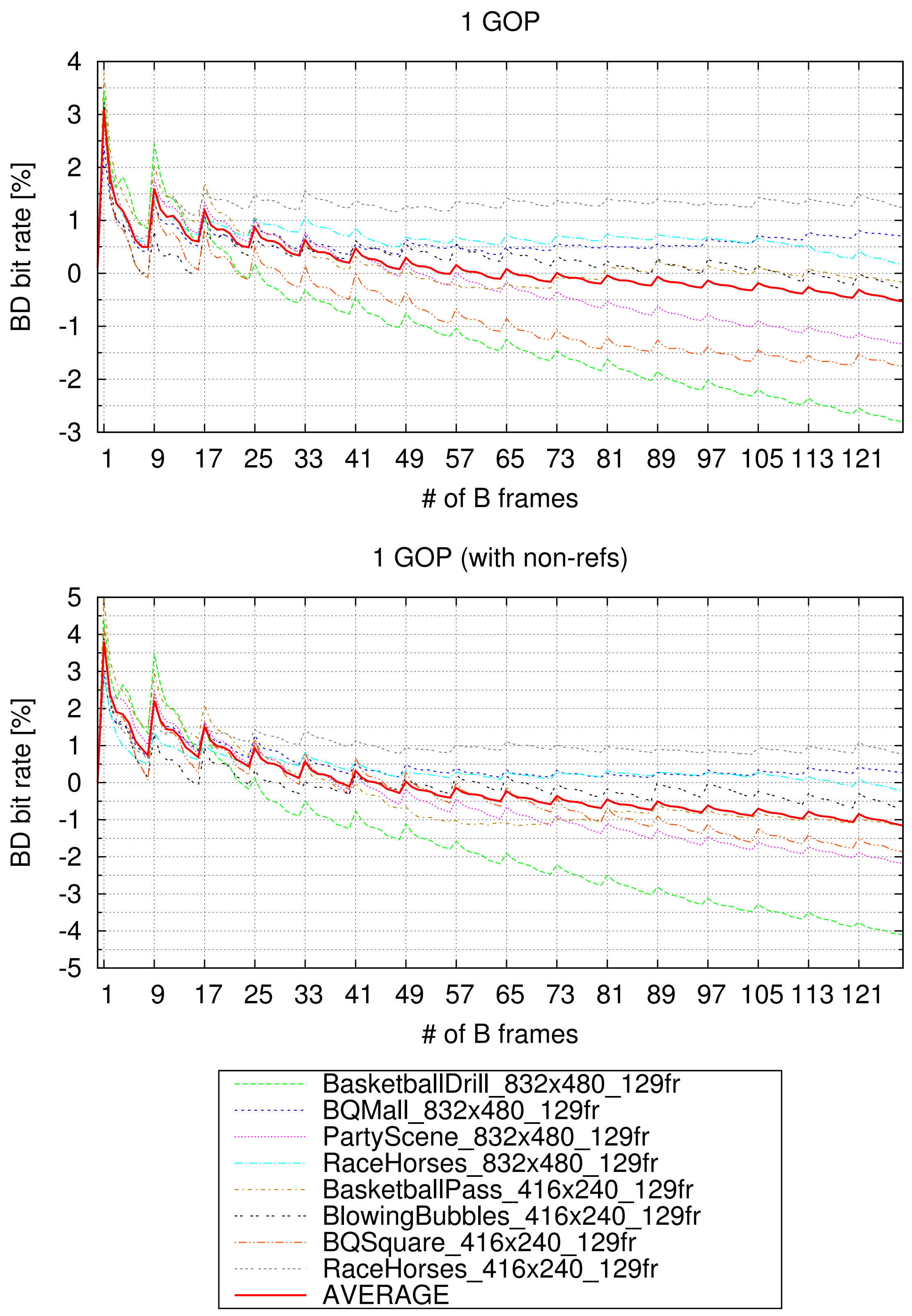


Figure 4.26: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D.

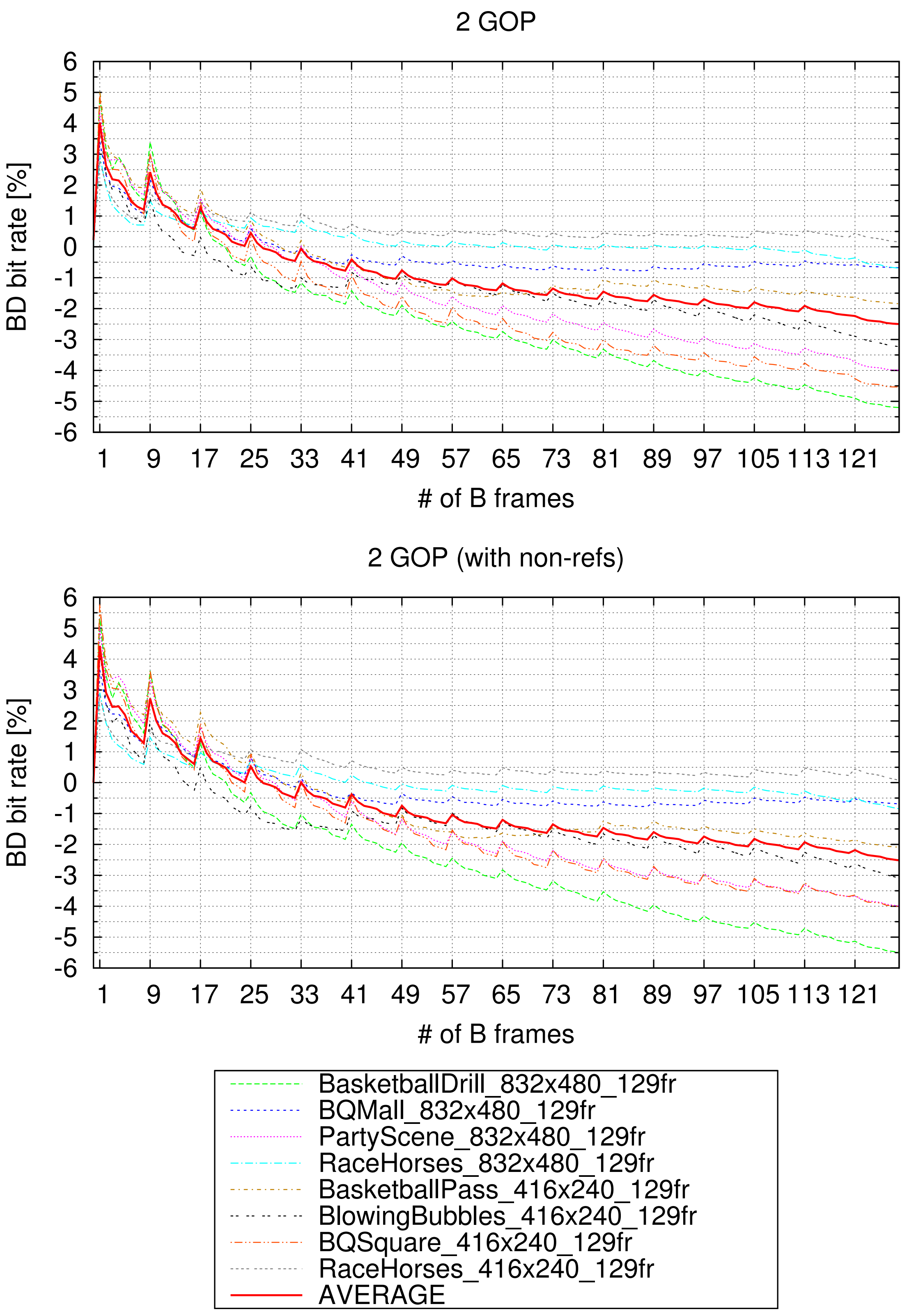


Figure 4.27: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D.

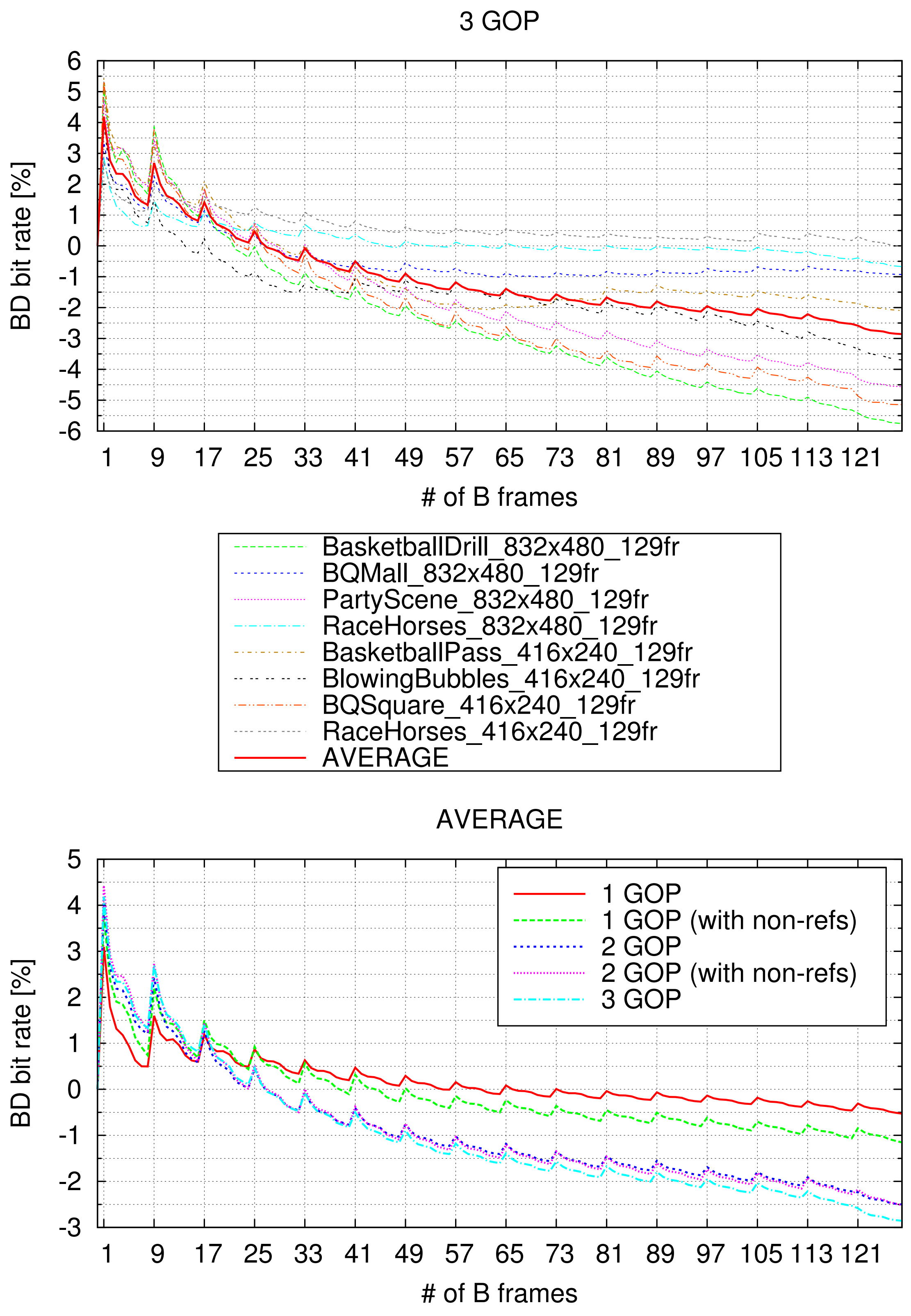


Figure 4.28: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D.

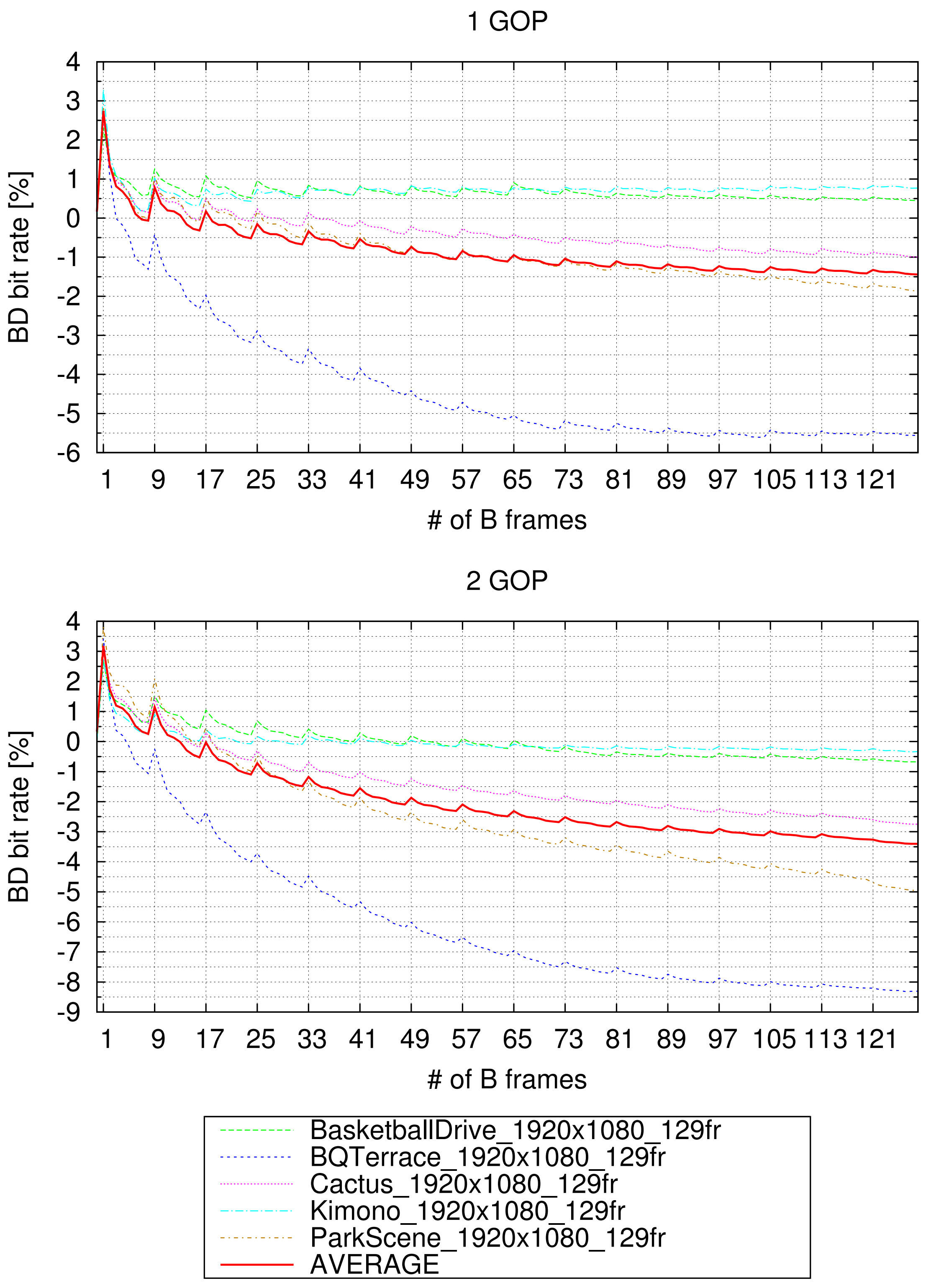


Figure 4.29: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class B.

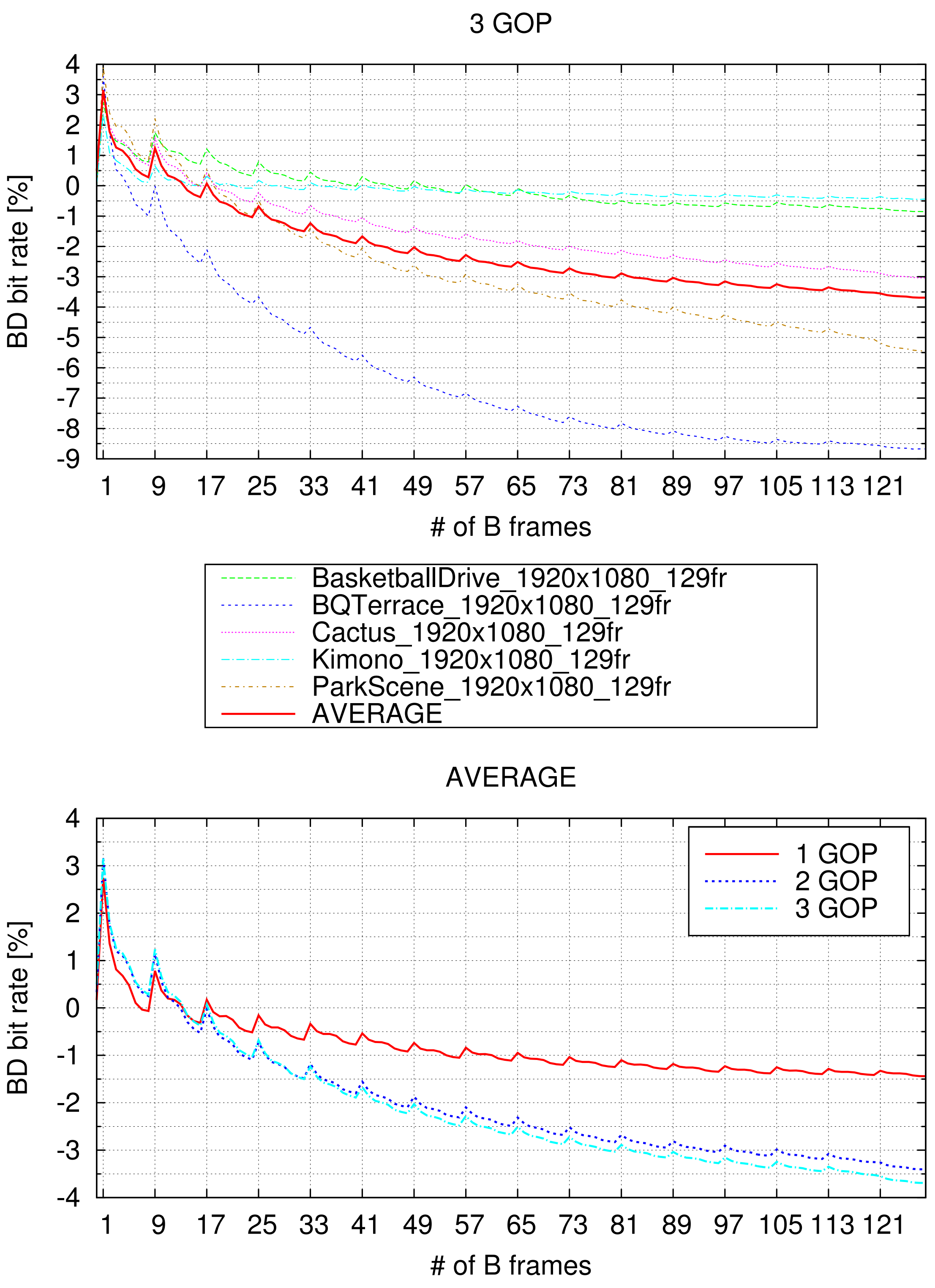


Figure 4.30: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class B.

4.3.2 Optimization of the frames with POC = $8n + 4$

The experiments conducted for this section are based on the variant "3 GOP" of the previous section, but additionally the frames with POC = $8n + 4$ are also included into the multi-frame optimization. The experimental results are shown as average values over the sequences in Fig. 4.31. Three different configurations are compared with "3 GOP."

In "3 GOP + local," for the optimization of the frames with POC = $8n+4$, the impact on all the remaining (reference and non-reference) frames within the current GOP is considered, leading to a total of 7 frames. In "3 GOP + local (same μ)," the same frames have been considered in the optimization, but the regularization parameter μ as used for the frames with POC = $8n$ has also been used for the optimization of the frames with POC = $8n+4$, whereas in all the other cases discussed in this section, the value of μ is chosen based on the individual frame QP. As can be seen, selecting μ based on the individual frame QP leads to better coding results. In "3 GOP + 2 GOP," the impact on the reference frames of the current and the direct subsequent GOP, and in "3 GOP + 3 GOP," the impact on the reference frames of the current and the two subsequent GOPs is considered, leading to a total of 7 and 11 frames, respectively. In all cases, the coding performance is inferior to the "3 GOP" configuration of the previous section. As discussed before, the multi-frame optimization generally causes an initial overhead in terms of coding efficiency, which pays off after a certain number of frames. For the frames with POC = $8n + 4$, the number of referring frames is apparently too small in order to compensate for this initial loss.

4.3.3 Discussion of the results

In the GOP8 coding scenario, the observed coding gains are significantly smaller than using an IPPP… prediction structure. This can be attributed to a coincidence of several factors. First of all, optimization of one single frame out of a group of eight frames (namely those with POC = $8n$) gave the best coding results. Since the encoding of seven out of eight frames is not modified, a significantly smaller number of transform coefficients is directly affected by the multi-frame optimization method. More than that, especially the frames having POC = $8n$ exhibit a very large share of intra-predicted samples (see Tabs. 4.12–4.14, which show the respective ratios for the

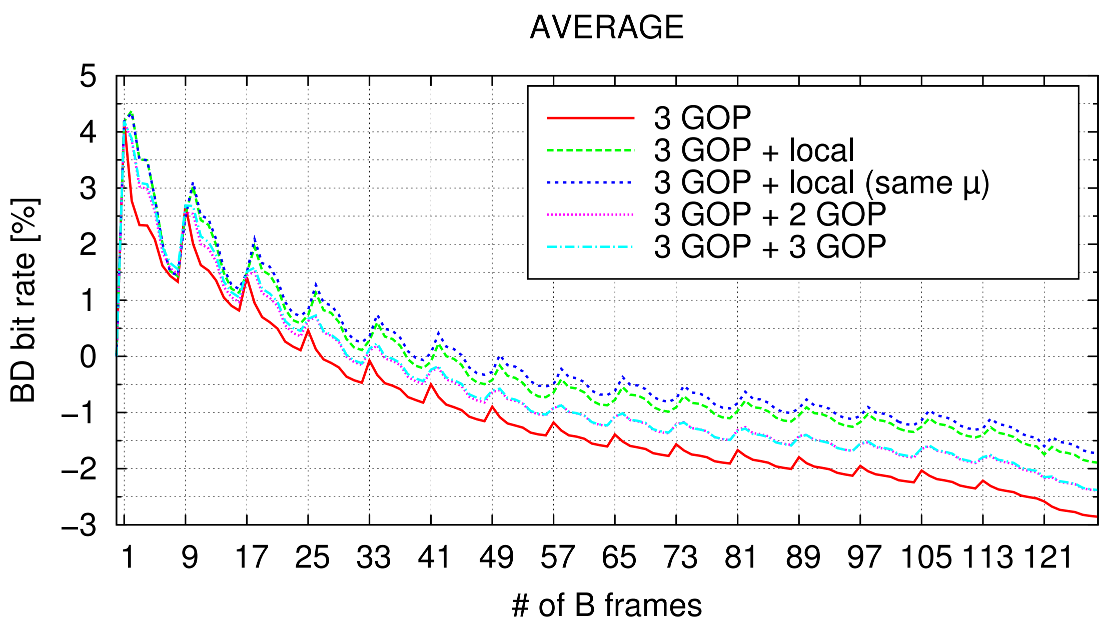


Figure 4.31: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) over the number of frames for Class C and D.

first 129 frames using the "3 GOP" configuration). As has been stated in Sec. 4.2.1, a large portion of intra-predicted samples is problematic because, on the one hand intra-predicted blocks break the prediction chain, where a longer prediction chain would be advantageous for the multi-frame optimization, and on the other hand, as intra-predicted are not included in the optimization for complexity reasons, an even smaller number of coefficients is considered in the optimization stage.

Besides that, in the IPPP... setting, the optimization of the residual signal is in some sense simpler, as the reconstructed signal has to accomplish only two objectives: To provide an appropriate reconstruction quality of the current frame and to serve as a suitable reference frame for the direct subsequent frame. In the GOP8 configuration, however, the prediction paths are more complicated. As shown in Tab. 4.11, the frames with POC $= 8n$ can be directly referred by up to 13 different subsequent frames. Besides providing an appropriate reconstruction quality for the frame itself, the corresponding residual signal therefore has to fulfill the possibly diverse requirements of the individual referring frames.

Finally, in the GOP8 configuration, bi-prediction (i.e., superposition of two prediction hypotheses) is used, unlike uni-prediction for the IPPP... setting. With bi-prediction, the prediction signal is typically of better quality, resulting in a smaller variance of

		Ratio of intra-predicted samples [%]				
	Sequence	QP=22	QP=27	QP=32	QP=37	**AVG**
Class B (1920×1080)	BasketballDrive	58.1	48.9	44.9	42.9	48.7
	BQTerrace	20.8	11.2	7.1	4.6	10.9
	Cactus	29.9	22.3	19.8	18.2	22.6
	Kimono	60.0	47.6	41.6	37.2	46.6
	ParkScene	16.4	12.6	10.6	9.1	12.2
Class C (832×480)	BasketballDrill	23.9	23.3	23.8	24.0	23.8
	BQMall	35.0	31.1	28.5	26.6	30.3
	PartyScene	21.4	20.7	19.3	17.6	19.8
	RaceHorses	69.7	61.8	55.5	51.0	59.5
Class D (416×240)	BasketballPass	30.4	28.9	27.1	24.4	27.7
	BlowingBubbles	18.1	15.4	13.3	10.8	14.4
	BQSquare	2.4	2.1	2.2	1.9	2.1
	RaceHorses	57.5	53.0	47.6	41.7	50.0
	AVERAGE	34.1	29.1	26.3	23.8	28.4

Table 4.12: Ratio of intra-predicted samples for the frames having POC = $8n$ (excluding the initial I frame).

the residual signal. Therefore, the resulting reconstruction quality is too a larger degree governed by the prediction signal, and the impact of the residual part is correspondingly smaller. Since in the multi-frame optimization approach, the selection of the encoded residual signal is addressed, using bi-prediction will result in a smaller coding gain compared to uni-prediction.

4.3.4 Overall performance for different intra periods

In this section, the coding performance of the multi-frame optimization in a random access scenario is studied based on the "3 GOP" variant of Sec. 4.3.1. The results are shown in Tab. 4.15. The column denoted as $\times k$ means that k times the intra period as prescribed in the common test conditions [JCT13] has been used, which corresponds to a random access period of approximately k seconds, with $k \in \{1, 2, \ldots, 5\}$. A random access period of 1–5 seconds represents a typical range, e.g. in [ETS14] it is stated:

> "The encoder shall place HEVC DVB_RAPs[1] in the video elementary stream at least once every 5 s. It is recommended that HEVC DVB_RAPs

[1]DVB_RAP = Digital Video Broadcasting Random Access Point

		Ratio of intra-predicted samples [%]				
	Sequence	QP=22	QP=27	QP=32	QP=37	**AVG**
Class B (1920×1080)	BasketballDrive	25.7	19.7	16.9	14.7	19.2
	BQTerrace	7.6	0.7	0.2	0.0	2.1
	Cactus	11.1	5.8	4.9	4.3	6.5
	Kimono	13.9	8.3	5.7	4.0	8.0
	ParkScene	3.2	2.0	1.4	1.0	1.9
Class C (832×480)	BasketballDrill	13.9	11.8	9.4	7.7	10.7
	BQMall	9.0	6.5	4.4	3.0	5.7
	PartyScene	13.3	11.5	9.2	6.4	10.1
	RaceHorses	36.3	24.5	18.4	11.9	22.8
Class D (416×240)	BasketballPass	12.2	9.4	7.3	4.5	8.4
	BlowingBubbles	4.7	3.8	3.1	2.9	3.6
	BQSquare	0.2	0.2	0.1	0.1	0.1
	RaceHorses	24.8	19.1	11.7	6.6	15.6
	AVERAGE	13.5	9.5	7.1	5.2	8.8

Table 4.13: Ratio of intra-predicted samples for the frames having POC $= 8n + 4$.

		Ratio of intra-predicted samples [%]				
	Sequence	QP=22	QP=27	QP=32	QP=37	**AVG**
Class B (1920×1080)	BasketballDrive	8.6	4.8	3.5	2.5	4.8
	BQTerrace	0.6	0.1	0.0	0.0	0.2
	Cactus	2.0	1.2	1.0	0.9	1.3
	Kimono	1.3	0.7	0.4	0.2	0.7
	ParkScene	0.6	0.3	0.2	0.1	0.3
Class C (832×480)	BasketballDrill	2.9	1.9	1.1	0.6	1.6
	BQMall	1.3	0.8	0.4	0.3	0.7
	PartyScene	3.9	2.7	1.6	0.9	2.3
	RaceHorses	7.0	4.0	2.1	1.2	3.6
Class D (416×240)	BasketballPass	1.8	1.2	0.6	0.4	1.0
	BlowingBubbles	1.7	1.5	1.1	0.9	1.3
	BQSquare	0.0	0.0	0.0	0.0	0.0
	RaceHorses	3.8	2.3	1.1	0.5	1.9
	AVERAGE	2.7	1.7	1.0	0.7	1.5

Table 4.14: Ratio of intra-predicted samples for the remaining frames (having neither POC $= 8n$ nor POC $= 8n + 4$).

> occur in the video elementary stream on average at least every 2 s. Where rapid channel change times are important or for applications such as PVR it may be appropriate for HEVC DVB_RAPs to occur more frequently, such as every 1 second."

Average bit rate savings of about 3 % BD bit rate can be observed, with a maximum of 10 % for the BQTerrace sequence at 1920×1080 resolution. In the random access scenario, the multi-frame optimization fails for the RaceHorses sequence, both at 832×480 and 416×240 resolution, but even in the worst test case, this results only in an overall bit rate increase of 0.3 %.

Typically, a longer intra period leads to higher bit rate savings. This is consistent with the diagrams showing the BD bit rate over the number of frames (such as Figs. 4.26–4.30). As after each intra frame the multi-frame optimization is started anew, a shorter distance between these restarts will lead to smaller bit rate savings, since the BD bit rate decreases with the number of frames.

It is reemphasized, that the encoding of only one out of eight frames (namely, those frames with POC = $8n$) is modified in order to incorporate inter-frame dependencies, whereas for the reported bit rate savings all the frames are subsumed. Therefore, the coding decisions made for those "key frames" can largely affect the overall coding performance.

		BD bit rate [%]				
	Sequence	×1	×2	×3	×4	×5
Class A (2560×1600)	NebutaFestival	−4.0	−4.8	−5.1	−5.2	−5.6
	PeopleOnStreet	0.0	−0.3	−0.5	−0.6	−0.7
	SteamLoco...	−2.0	−2.4	−2.3	−2.4	−2.4
	Traffic	−1.9	−3.7	−4.8	−5.4	−6.7
	AVERAGE	−2.0	−2.8	−3.2	−3.4	−3.8
Class B (1920×1080)	BasketballDrive	−0.1	−0.5	−0.9	−0.7	−1.0
	BQTerrace	−6.2	−8.3	−8.9	−9.4	−10.1
	Cactus	−1.3	−2.4	−2.9	−3.2	−3.4
	Kimono	−0.5	−1.8	−1.9	−1.8	−3.0
	ParkScene	−0.7	−2.7	−3.5	−4.3	−5.1
	AVERAGE	−1.8	−3.1	−3.6	−3.9	−4.5
Class C (832×480)	BasketballDrill	−2.5	−4.4	−5.4	−6.0	−6.5
	BQMall	−0.9	−1.7	−2.0	−2.2	−2.2
	PartyScene	−2.2	−4.0	−4.9	−5.5	−5.9
	RaceHorses	0.1	−0.1	0.0	−0.1	−0.2
	AVERAGE	−1.4	−2.6	−3.1	−3.4	−3.7
Class D (416×240)	BasketballPass	−0.3	−1.0	−1.2	−1.5	−1.4
	BlowingBubbles	−2.3	−3.8	−4.4	−4.9	−5.3
	BQSquare	−3.1	−4.6	−5.1	−5.4	−5.7
	RaceHorses	0.3	−0.0	−0.2	−0.1	−0.3
	AVERAGE	−1.3	−2.3	−2.7	−3.0	−3.2

Table 4.15: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) for the random access scenario with different intra periods, JCT-VC test set ($\times k$ means k times the intra period according to the common test conditions, i.e. approx. k seconds).

		BD bit rate [%]				
	Sequence	×1	×2	×3	×4	×5
VCEG (832×480)	Flower4	−1.8	−3.6	−4.3	−4.8	−4.9
	Keiba3	1.2	1.2	0.1	1.0	1.1
	Nuts5	0.3	0.1	0.4	0.2	0.2
	AVERAGE	−0.1	−0.8	−1.3	−1.2	−1.2
VCEG (352×288)	Foreman	0.2	−0.7	−1.6	−1.1	−2.8
	MobileAndCalendar	−2.3	−3.9	−4.7	−4.9	−6.1
	Paris	−0.4	−2.1	−3.1	−3.7	−4.2
	Tempete	−1.9	−3.0	−3.3	−3.7	−3.9
	AVERAGE	−1.1	−2.4	−3.2	−3.3	−4.2

Table 4.16: Bit rate savings in terms of Bjøntegaard delta bit rate (BD bit rate) for the random access scenario with different intra periods, VCEG test set ($\times k$ means k times the intra period according to the common test conditions, i.e. approx. k seconds).

5 Extension of the method for scalable video coding

In this chapter, the optimization method is extended for scalable video coding. In scalable video coding, the video signal is encoded using a layered representation, where each layer represents the signal at a given quality. Typically, three dimensions of scalability are distinguished:

- temporal scalability,
- spatial scalability, and
- fidelity or SNR scalability

In temporal scalability, each layer represents the video sequence at a given frame rate. In spatial scalability, each layer corresponds to a fixed spatial resolution of the video signal. Fidelity or SNR scalability refers to a layered coding approach, where the frame rate and the spatial resolution are identical for all the layers, but the fidelity or reconstruction quality is enhanced with each layer, which is achieved by a reduction of the quantization step size from one layer to the next.

In all variants, the lowest layer, which is also called *base layer*, can be decoded independently of all the other layers and represents the video signal at a base quality. The additional layers, which are also called *enhancement layers*, are in their decoding dependent on the base layer as well as all further subordinate layers which they refer to via *inter-layer prediction*. By the usage of inter-layer prediction, data which have already been transmitted for a lower layer, can be re-used in order to facilitate encoding of the higher layers. This makes scalable coding more efficient than simulcast, where the different quality representations of the video signal are transmitted independently of each other.

Scalable video coding techniques are supported as an extension of the H.262/MPEG-2 [SS95, Sik97] and H.264/AVC video coding standards, the latter also being referred to as SVC [SMW07]. Furthermore, under the name SHVC, there is currently standardization activity underway, which targets at extending H.265/HEVC to support scalability. In addition to the "classical" three dimensions of scalability as mentioned above, this will also include tools supporting color format and bit-depth scalability.

For the remaining part of this chapter, the focus is on spatial scalability within the scalable extension of H.264/AVC, SVC. The key insights and concepts as well as the experimental results have already been presented in [WSW08].

5.1 Spatial scalability in SVC

Spatial scalability within SVC follows a layered coding approach. Each layer contains a representation of the video sequence at a different spatial resolution, similar to an image pyramid [BA83]. Motion-compensated prediction (MCP) and intra prediction are employed at each layer as in single layer H.264/AVC coding. As a distinguished feature of SVC, for the decoding of a higher resolution layer, it is not required to fully decode the corresponding lower resolution layers. Especially, MCP only has to be performed for the highest layer during the decoding process. This property is referred to as "single-loop decoding" or "constrained interlayer prediction" [SHMW05]. In order to exploit the dependencies between the different layers, SVC allows inter-layer prediction of texture (decoded samples), motion parameters, and residual samples. For the inter-layer texture prediction, the decoded image samples of the lower resolution layer are upsampled by a separable four-tap interpolation filter. Instead of using MCP or intra prediction, the resulting upsampled signal is used as a prediction for encoding of the higher resolution input signal. Note, that due to the single-loop decoding constraint of SVC, inter-layer texture prediction can only be applied to intra-coded subordinate layer macroblocks, since otherwise MCP would also have to be performed for the lower resolution layers in order to be able to provide the corresponding decoded image samples. For inter-coded subordinate layer macroblocks, however, SVC allows inter-layer prediction of the residual signal. For that purpose, the inverse transformed residual samples of the lower resolution layer are upsampled by a bilinear interpolation filter. During the decoding process, the resulting upsampled residual signal is added to the higher resolution residual and prediction signals in order to obtain the corresponding decoded image samples. A detailed description of the inter-layer prediction process for spatial scalability within SVC can be found in [SS07]. An overview of SVC is given in [SMW07].

5.2 Matrix notation of the SVC reconstruction process

Similar to Sec. 3.1.3, the reconstruction process for spatial scalability can also be written in matrix form. In order to simplify matters, a two layer scenario is assumed, where the base layer pictures are of width W_0 and height H_0, and the enhancement layer pictures are of width W_1 and height H_1. Again, a number N of frames is considered jointly for the optimization task, leading to totally $K_0 = N \cdot W_0 \cdot H_0$ base layer samples and $K_1 = N \cdot W_1 \cdot H_1$ enhancement layer samples. Analogous to Eq. 3.4, p. 26, the reconstruction of the base layer signal $\mathbf{s}_0$ and the enhancement layer signal $\mathbf{s}_1$ can be written as:

$$\mathbf{s}_0 = \mathbf{p}_0 + \mathbf{M}_0\,\mathbf{s}_0 + \mathbf{T}_0\,\mathbf{c}_0 \tag{5.1}$$

$$\mathbf{s}_1 = \mathbf{p}_1 + \mathbf{M}_1\,\mathbf{s}_1 + \mathbf{T}_1\,\mathbf{c}_1 + \mathbf{B}\,\mathbf{s}_0 + \mathbf{R}\,\mathbf{T}_0\,\mathbf{c}_0 \tag{5.2}$$

Here, the suffixes 0 and 1 indicate base layer and enhancement layer. The reconstruction of the base layer signal is identical to the non-scalable, single layer case of Eq. 3.4. For the enhancement layer, two new $K_1 \times K_0$ matrices $\mathbf{B}$ and $\mathbf{R}$ have been introduced. The matrix $\mathbf{B}$ represents the inter-layer texture prediction, such that the matrix product $\mathbf{B}\,\mathbf{s}_0$ gives the corresponding prediction signal, whereas the matrix $\mathbf{R}$ represents the inter-layer residual prediction, with $\mathbf{R}\,\mathbf{T}_0\,\mathbf{c}_0$ being the resulting residual prediction signal. Hence, the components of $\mathbf{B}$ and $\mathbf{R}$ contain the upsampling filter coefficients at the sample positions of those macroblocks, which use the respective inter-layer prediction mechanism, and are zero for the macroblocks without usage of inter-layer prediction.

In explicit form, $\mathbf{s}_0$ and $\mathbf{s}_1$ can be obtained as follows (corresponding to Eq. 3.9):

$$\mathbf{s}_0 = \sum_{\nu=0}^{\nu_{0,max}} \mathbf{M}_0^{\nu} \left(\mathbf{p}_0 + \mathbf{T}_0\,\mathbf{c}_0\right) \tag{5.3}$$

$$\mathbf{s}_1 = \sum_{\nu=0}^{\nu_{1,max}} \mathbf{M}_1^{\nu} \left(\mathbf{p}_1 + \mathbf{T}_1\,\mathbf{c}_1 + \mathbf{B}\,\mathbf{s}_0 + \mathbf{R}\,\mathbf{T}_0\,\mathbf{c}_0\right) \tag{5.4}$$

The values of $\nu_{0,max}$ and $\nu_{1,max}$ are the length of the longest prediction chain (critical path) of the matrices $\mathbf{M}_0$ and $\mathbf{M}_1$, respectively.

With

$$\tilde{\mathbf{s}}_i = \mathbf{s}_i - \sum_{\nu=0}^{\nu_{i,max}} \mathbf{M}_i^{\nu}\,\mathbf{p}_i \qquad \text{for} \quad i \in \{0,1\} \tag{5.5}$$

and

$$\mathbf{A}_0 = \sum_{\nu=0}^{\nu_{0,max}} \mathbf{M}_0^{\nu}\,\mathbf{T}_0 \tag{5.6}$$

$$\mathbf{A}_1 = \left[\sum_{\nu=0}^{\nu_{1,max}} \mathbf{M}_1^{\nu} \left(\mathbf{B}\,\mathbf{A}_0 + \mathbf{R}\,\mathbf{T}_0\right) \quad \sum_{\nu=0}^{\nu_{1,max}} \mathbf{M}_1^{\nu}\,\mathbf{T}_1\right] \tag{5.7}$$

the following matrix products are obtained:

$$\tilde{\mathbf{s}}_0 = \mathbf{A}_0\,\mathbf{c}_0 \tag{5.8}$$

$$\tilde{\mathbf{s}}_1 = \mathbf{A}_1\,\left[\mathbf{c}_0 \quad \mathbf{c}_1\right]^T \tag{5.9}$$

With the original base layer signal $\mathbf{y}_0$ and the original enhancement layer signal $\mathbf{y}_1$, the corresponding distortion functions $D_0(\mathbf{c}_0)$ and $D_1(\mathbf{c}_0, \mathbf{c}_1)$ can be defined as follows:

$$D_0(\mathbf{c}_0) = \|\mathbf{y}_0 - \mathbf{s}_0\|_2^2 \tag{5.10}$$

$$= \left\| \underbrace{\mathbf{y}_0 - \sum_{\nu=0}^{\nu_{0,max}} \mathbf{M}_0^{\nu}\,\mathbf{p}_0}_{\tilde{\mathbf{y}}_0} - \tilde{\mathbf{s}}_0 \right\|_2^2 \tag{5.11}$$

$$= \|\tilde{\mathbf{y}}_0 - \mathbf{A}_0\,\mathbf{c}_0\|_2^2 \tag{5.12}$$

$$D_1(\mathbf{c}_0,\mathbf{c}_1) = \|\mathbf{y}_1 - \mathbf{s}_1\|_2^2 \tag{5.13}$$

$$= \left\| \underbrace{\mathbf{y}_1 - \sum_{\nu=0}^{\nu_{1,max}} \mathbf{M}_1^{\nu}\,\mathbf{p}_1}_{\tilde{\mathbf{y}}_1} - \tilde{\mathbf{s}}_1 \right\|_2^2 \tag{5.14}$$

$$= \left\| \tilde{\mathbf{y}}_1 - \mathbf{A}_1\,[\mathbf{c}_0 \quad \mathbf{c}_1]^T \right\|_2^2 \tag{5.15}$$

For the rate function $R(\mathbf{c})$, as in Eq. 3.17, p. 30, again the ℓ_1-norm of the transform coefficient vector $\mathbf{c}$ is used, i.e. $R(\mathbf{c}) = ||\mathbf{c}||_1$.

Consequently, the optimization problem in a two layer spatial scalable setting can be stated as follows:

$$\begin{bmatrix} \mathbf{c}_{0,opt} \\ \mathbf{c}_{1,opt} \end{bmatrix} = \arg \min_{[\mathbf{c}_0 \quad \mathbf{c}_1]^T} \begin{bmatrix} D_0(\mathbf{c}_0) + \mu_0\,R(\mathbf{c}_0) \\ D_1(\mathbf{c}_0,\mathbf{c}_1) + \mu_1\,(R(\mathbf{c}_0) + R(\mathbf{c}_1)) \end{bmatrix} \tag{5.16}$$

Here, μ_0 and μ_1 are the regularization parameters for the base and enhancement layer, respectively. Note that Eq. 5.16 is a multi-objective (or vector) optimization problem.

An unconstrained multi-objective optimization problem can be formulated as

$$\min\,[f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})]\,, \tag{5.17}$$

where $f_1, f_2, \ldots, f_n$ are real-valued functions. In multi-objective optimization, the scalar concept of optimality is extended to *Pareto optimality*. A vector $\mathbf{x}^*$ is called strictly Pareto optimal, if there is no vector $\mathbf{x}$ having

$$f_i(\mathbf{x}) \;\leq\; f_i(\mathbf{x}^*) \qquad \text{for all } i \in \{1,2,\ldots,n\}, \text{ and} \tag{5.18}$$

$$f_k(\mathbf{x}) \;<\; f_k(\mathbf{x}^*) \qquad \text{for at least one } k \in \{1,2,\ldots,n\}. \tag{5.19}$$

The set of all strictly Pareto optimal vectors is called *Pareto front*. A multi-objective optimization problem can be converted into a scalar optimization problem by introducing weighting factors w_i, which control the trade-off between the individual objective functions. This technique is known in literature as *scalarization* [BV04, Sec. 4.7.4] and leads to the following optimization problem:

$$\begin{aligned}
&\min \sum_{i=1}^{n} w_i \cdot f_i(\mathbf{x}) \\
&\sum_{i=1}^{n} w_i = 1 \\
&w_i > 0 \qquad \text{for all } i \in \{1, 2, \dots, n\}
\end{aligned} \tag{5.20}$$

The minimizer of Eq. 5.20 for a given weighting vector $\mathbf{w} = [w_1, w_2, \dots, w_n]^T$ is an element of the Pareto front of Eq. 5.17 [HM79].

In the case of two layer spatial scalable coding, this leads to the following aggregate objective function (with $\alpha = (W_1 \cdot H_1)/(W_0 \cdot H_0)$):

$$\begin{bmatrix} \mathbf{c}_{0,opt} \\ \mathbf{c}_{1,opt} \end{bmatrix} = \arg \min_{[\mathbf{c}_0 \quad \mathbf{c}_1]^T} \left\{ \begin{matrix} (1-w) \cdot \alpha \cdot (D_0(\mathbf{c}_0) + \mu_0 \, R(\mathbf{c}_0)) + \\ w \cdot (D_1(\mathbf{c}_0, \mathbf{c}_1) + \mu_1 \, (R(\mathbf{c}_0) + R(\mathbf{c}_1))) \end{matrix} \right\} \tag{5.21}$$

Here, the scalarization factor $w \in [0; 1]$ selects the trade-off between base and enhancement layer coding efficiency. For the special case $w = 0$, the impact of the inter-layer dependencies on the enhancement layer coding efficiency is not considered when determining the base layer transform coefficients. For the other special case $w = 1$, however, the base layer coding efficiency is not considered in the optimization. The scaling factor α accounts for the fact that base and enhancement layer have a different number of samples, and thus ensures that for $w = 0.5$ the equilibrium between base and enhancement layer coding efficiency is achieved.

For solving the scalar optimization problem Eq. 5.21, standard methods as discussed in Sec. 3.4 can be applied.

5.3 Description of the algorithm

The algorithm for the multi-layer multi-frame transform coefficient optimization is based on the single-layer multi-frame optimization algorithm described in Sec. 4.1.1. A group of N frames is considered jointly, such that the impact of both inter-frame and inter-layer prediction are considered in the determination of reference transform coefficients. The optimization is performed in the following steps:

1. Since I slices are not included in the optimization, the first frame 0 is ordinarily encoded using the SVC reference encoder software Joint Scalable Video Model (JSVM) and a variable M is set equal to 1.

2. The frames M, $M+1$, $\ldots$, $M+N-1$ are ordinarily encoded using JSVM such that their block partitioning, prediction modes, motion vectors etc. are known.

3. The multi-frame transform coefficient problem 5.21 is solved for these frames, given the inter-frame and inter-layer dependencies as obtained from the previous step.

4. An integer approximation for the real-valued elements of the base layer solution vector $\mathbf{c}_{0,opt}$ is obtained by simple rounding. Note that since H.264/AVC does not support the sign data hiding (SDH) of H.265/HEVC, no constraints on the parity of resulting vector have to be considered here.

5. The inter-layer prediction usage of the current frame M is optionally re-estimated, based on the optimized base layer transform coefficients as derived in the previous step.

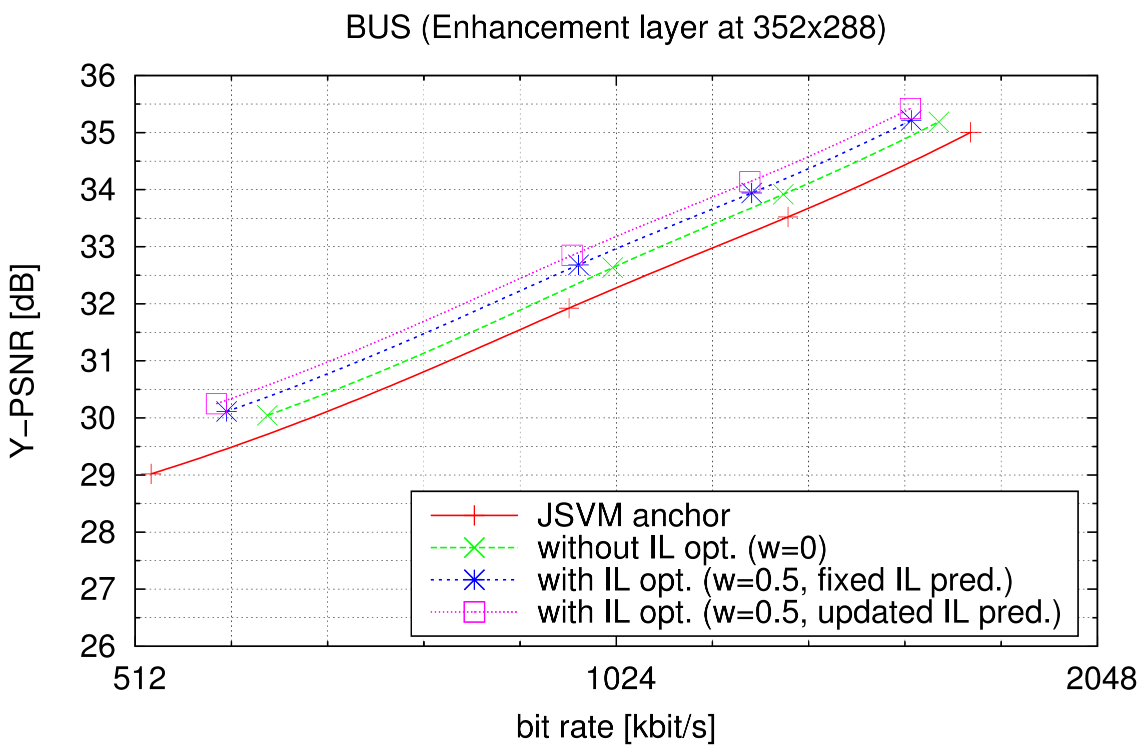
6. The multi-frame transform coefficient problem 5.21 is solved again, but this time only for the enhancement layer transform coefficient vector $\mathbf{c}_{1,opt}$, whereas for the base layer vector $\mathbf{c}_{0,opt}$ the integer coefficients as obtained in step 4 are used.

7. Analogous to step 4, an integer approximation for $\mathbf{c}_{1,opt}$ is obtained by simple rounding.

8. $M := M+1$ and execution continues at step 2.

5.4 Experimental results

The experiments were conducted using a modified version of the SVC reference encoder software Joint Scalable Video Model (JSVM), based on version JSVM_9_9. A simple



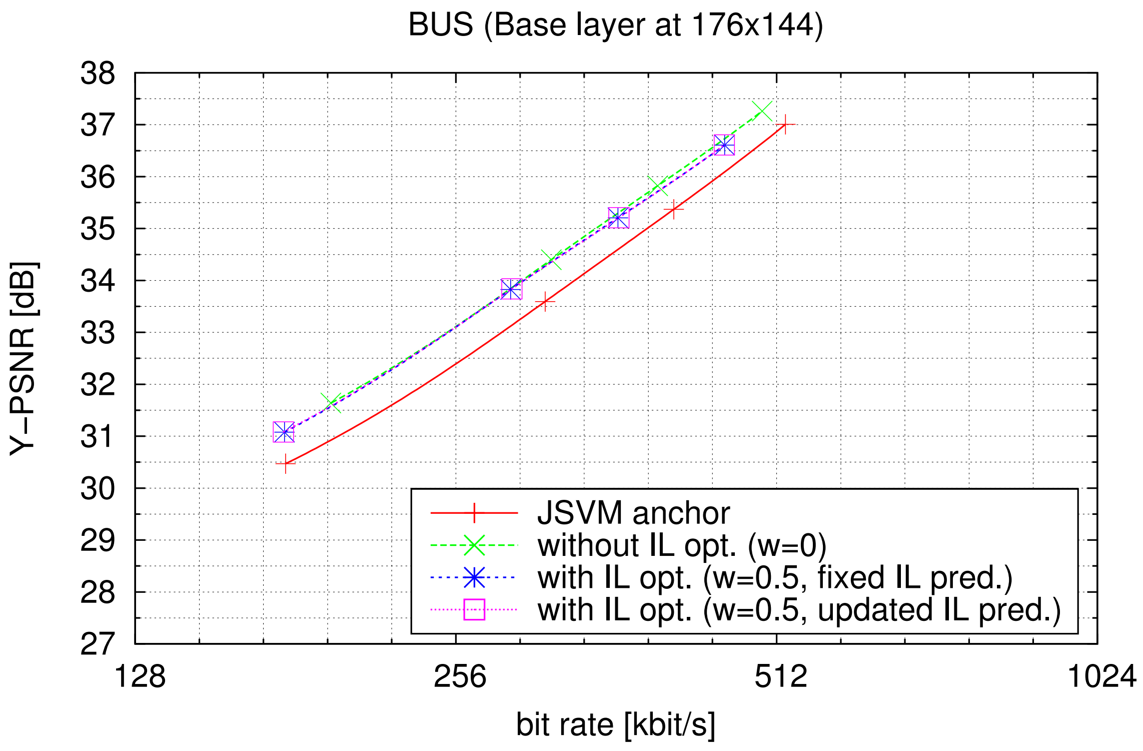
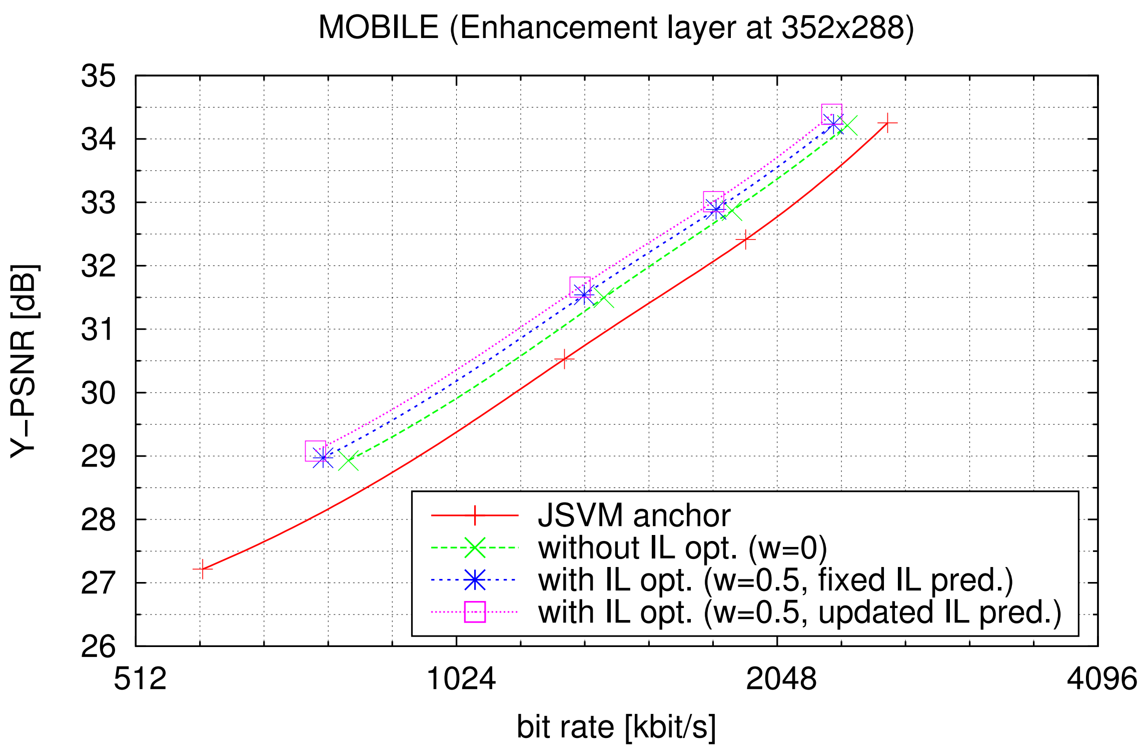


Figure 5.1: Rate distortion plots for the BUS test sequence (top: enhancement layer, bottom: base layer).



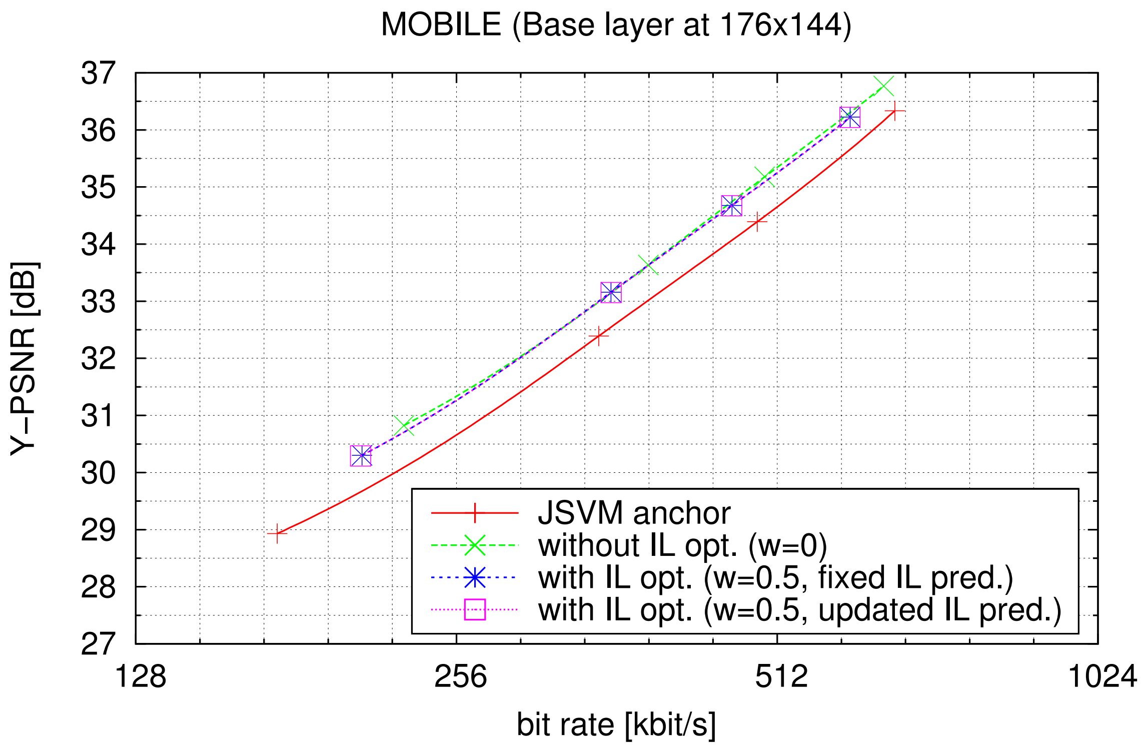


Figure 5.2: Rate distortion plots for the MOBILE test sequence (top: enhancement layer, bottom: base layer).

	BD bit rate [%]		
	without IL opt.	with IL opt.	
Sequence		fixed IL pred.	upd. IL pred.
BUS (Base layer)	−10.7	−10.2	−10.2
BUS (Enhancement layer)	−7.1	−12.4	−16.0
MOBILE (Base layer)	−10.4	−9.9	−9.8
MOBILE (Enhancement layer)	−10.6	−15.0	−17.9

Table 5.1: BD bit rate results for the two exemplary SVC test sequences.

IPPP coding structure with one reference frame, two spatial layers at QCIF (176×144) and CIF (352×288) resolution was used.

For the entropy coding, CABAC was employed for both layers. Note that since this JSVM version does not support rate distortion optimized quantization (RDOQ), the anchor uses ordinary scalar quantization. The difference between enhancement layer and base layer quantization parameter (QP) was set equal to 3. For the base layer, the following values have been used: $QP \in \{26, 28, 30, 34\}$. For the optimization algorithm, a group of $N = 3$ consecutive pictures was considered. A spatial sliding window size of 5×5 and 10×10 macroblocks for base and enhancement layer, respectively, has been used. The regularization parameter μ_i (for $i \in \{0, 1\}$) was empirically chosen as $\mu_i = 4.5 \cdot \lambda_{MODE}$, with λ_{MODE} as given in [WSJ+03, Eq. 12].

The resulting rate distortion plots are shown in Figs. 5.1 and 5.2 for the first 33 frames of two representative sequences of the test set used for SVC standardization. The corresponding BD bit rate values are shown in Tab. 5.1. In the first scenario ("without IL opt."), only the inter-frame dependencies are exploited by the optimization method for both base and enhancement layer. In a second scenario ("with IL opt., fixed IL pred."), the inter-layer (IL) dependencies have also been considered such that the impact of the base layer residual signal on the reconstruction of the enhancement layer is also taken into account. In a third scenario ("with IL opt., updated IL pred."), the inter-layer prediction usage is re-estimated after the base layer transform coefficients have been determined.

It can be seen that by inclusion of the IL dependencies into the optimization problem, the coding efficiency of the enhancement layer can be significantly improved, leading to additional bit rate savings of approximately 5 percentage points (pp). The negative impact on the base layer coding efficiency is very moderate, as the corresponding bit rate savings are reduced by 0.5 pp compared to multi-frame optimization without consideration of IL dependencies. By updating the inter-layer prediction usage after the optimized base layer transform coefficients have been determined, the bit rate savings for the enhancement layer are again further increased by about 3 pp, with no significant impact on the base layer coding efficiency.

6 Conclusions and outlook

In this thesis, a multi-frame transform coefficient optimization method for H.265/ HEVC is developed and studied. The inter-frame dependencies, which are caused by motion-compensated prediction, are considered in the encoding of the reference frames by an appropriate choice of the residual signal. The dependencies are described using a linear signal model, initially proposed by Schumitsch in [SSW04, SSW05] for H.264/ AVC optimization. Based on this model, the optimization problem is cast in the form of an ℓ_1-regularized least squares problem. For solving this problem, an optimization algorithm is developed, which is applicable to H.265/HEVC without imposing excessive demands in terms of computational complexity and memory requirements. For that purpose, a variant of the *iterative shrinkage/thresholding algorithm (ISTA)* [DDDM04, WNF09] is employed at the core of the optimization method.

The behaviour of the multi-frame optimization method is first studied in a simple IPPP... prediction structure. A simple functional relationship between the regularization parameter μ and the quantization parameter (QP) is empirically found, which is similar to the widely used rule for determining the Lagrangian multiplier λ in the operational rate distortion optimization (RDO) of a typical video encoder, as originally described in [WG01]. The performance of the multi-frame optimization method is evaluated and HEVC-specific issues, like sign data hiding (SDH) and efficient handling of all-zero blocks, are addressed. Different regularization functions are compared, and it is shown, that using the ℓ_1-norm causes no loss, if the regularization parameter μ is matched to the QP value, which is beneficial, because the ℓ_1-norm corresponds to a simple elementwise soft thresholding operation in each ISTA iteration.

The accuracy of the linear signal model is studied, and it is found, that the largest discrepancies are caused by the relaxation of the originally integer optimization problem into a real-valued one. This relaxation, however, is unavoidable from a practical point of view, as the integer optimization problem would be NP hard [HV05, VH05], and therefore practically impossible to solve. The non-linear filtering operations in H.265/HEVC, deblocking and sample adaptive offset (SAO), which are not captured by the linear signal model, have only small impact on the model accuracy.

The bit rate savings due to the proposed method are evaluated over the number of frames using the Bjøntegaard Delta bit rate (BD bit rate) metric. It is observed, that the multi-frame optimization causes an initial coding efficiency loss, which is amortized after about 10–20 frames, and after that turns into a coding gain of about 10 % BD bit rate in an IPPP... setting.

The complexity of the proposed method is assessed based on experimentally measured run-times for the individual components of the algorithm. An overall run-time increase by a factor of 10–20 relative the HM reference encoder is reported for the Class C (832×480) and Class D (416×240) sequences of the JCT-VC test set. For larger resolutions, a further run-time increase can be limited by the usage of a spatial sliding window, such that the frame is split into a series of smaller sized "optimization windows," which are processed sequentially.

The multi-frame optimization method is then applied to the random access coding scenario, as described in the JCT-VC common test conditions [JCT13]. By the outcomes of the conducted experiments, it is found that optimization of only the so-called "key frames," which is the first frame of each group of pictures (GOP) in coding order, results in the largest coding gains. The behaviour of the multi-frame optimization is evaluated for different random access (intra) periods, ranging from 1 s to 5 s. The bit rate savings are higher for a longer intra period, which matches the previous observation that it takes a certain number of frames in order to amortize the initial coding loss. In the random access scenario, bit rate savings in the order of 3 % BD bit rate are observed.

Finally, an extension of the method for spatially scalable video coding using SVC, the scalable extension of H.264/AVC, is presented. Here, in addition to the inter-frame dependencies, also the inter-layer dependencies are taken into account for the encoding of the base layer. It is shown, that the coding performance of the enhancement layer can be significantly improved by about 5 percentage points (pp) BD bit rate, with only moderate impact on the base layer performance of about 0.5 pp BD bit rate.

Outlook

Within the area of this thesis, there is opportunity for further research. In the following, a few interesting directions are pointed out:

- For this thesis, no emphasis has been put on the efficiency of the implementation. Each ISTA iteration consists of a series of matrix-vector multiplications for inverse transform and motion-compensation, which are performed in floating-point arithmetic. The run-time increase relative to HM could be reduced, if the regular structure of the matrices is exploited (e.g., using butterfly operations for the transform), and floating-point calculations are avoided.

- In this thesis, only the inter-predicted blocks are considered in the multi-frame optimization. Further gains could be expected, if also the intra-predicted blocks are included. Since this would lead to very long prediction chains, which are caused by the block-to-block prediction within each frame, a significantly higher computational complexity can be expected. In order to avoid this additional burden, each intra-predicted block could be optimized on its own, considering only the impact on the subsequent frames and neglecting the impact within its frame.

- Similar to the extension for spatial scalability based on H.264/AVC as described in Chapter 5, the described method could be adapted to the scalable extension of H.265/HEVC, called SHVC. In SHVC, the inter-layer prediction is generalized by using reconstructed and upsampled base layer frames as reference frames for the enhancement layer. This is possible because SHVC, in contrast to H.264/AVC-based SVC, follows a multi-loop decoding paradigm.

- The optimization method developed in this thesis relies on the commonly used squared error distortion measure. It is a well-known fact, that this metric only roughly corresponds to the subjective visual quality (e.g., [Gir93]). It could be countered, however, that hybrid video encoders introduce only special types of artifacts (e.g., there is typically no rotation, scaling, brightness change etc.), and that for the kind of artifact which is introduced, the squared error measure serves its purpose apparently quite well. Still, there has been considerable work on the topic of perceptual visual quality metrics, see [LK11]. In particular, perceptual distortion measures based on the total variation (TV) [ROF92] have been proposed recently [PSGF11, WZD14]. The TV is notably suited for describing the structural characteristic of an image. Consequently, by incorporating the TV as an additional regularization term into the optimization problem, it could be expected, that the edges as in the original images are better preserved, whereas introduction of annoying blocking artifacts is avoided. As has been shown in [CW05], the well established algorithm for solving TV-regularized least squares problems by Chambolle [Cha04] can be interpreted as a special case of a more generic class called *proximal forward-backward splitting*, out of which ISTA is another particular example. Since ISTA has been used for the work within this thesis, it should be possible to integrate TV-regularization into the multi-frame transform coefficient optimization approach.

Glossary

Symbols

Δ	quantization step size
λ	Lagrangian multiplier
μ	regularization parameter
ν_{max}	critical path length (longest prediction chain)
D	distortion
H	frame height (number of luma samples in vertical direction)
J	Lagrangian rate distortion cost
K	total number of luma samples under consideration
N	number of frames in joint optimization
R	bit rate
W	frame width (number of luma samples in horizontal direction)
$\mathbf{A}$	reconstruction matrix
$\mathbf{I}$	identity matrix
$\mathbf{M}$	prediction matrix
$\mathbf{T}$	inverse transform matrix
$\mathbf{c}$	vector of transform coefficient levels
$\mathbf{c}_{opt}$	vector of optimal transform coefficient levels
$\mathbf{p}$	vector of fixed prediction signal samples
$\mathbf{r}$	vector of (quantized) residual signal samples
$\mathbf{s}$	vector of reconstructed samples
$\hat{\mathbf{s}}$	vector of prediction signal samples
$\mathbf{y}$	vector of original signal samples

$\tilde{\mathbf{y}}$	vector of modified original signal samples (without the motion-compensated fixed prediction signal)

Abbreviations

AVC	Advanced Video Coding (aka H.264)
BAC	Binary Arithmetic Coding
BD bit rate	Bjøntegaard Delta bit rate
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
cbf	coded block flag
CIF	Common Intermediate Format (352×288 luma samples)
CPU	Central Processing Unit
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
dB	Decibel
DC	literally: Direct Current; here: the mean value of a waveform
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DST	Discrete Sine Transform
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
ETSI	European Telecommunications Standards Institute
FIR	Finite Impulse Response
fps	Frames per Second
GOP	Group of Pictures
HDQ	Hard Decision Quantization
HDTV	High Definition Television
HEVC	High Efficiency Video Coding (aka H.265)
HM	HEVC Test Model
IEC	International Electrotechnical Commission
IIR	Infinite Impulse Response
IL	Inter-Layer
ISO	International Organization for Standardization
ISTA	Iterative Shrinkage/Thresholding Algorithm

ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JCT-VC	Joint Collaborative Team on Video Coding
JSVM	Joint Scalable Video Model
LSP	Logarithmic Sum Penalty
MCP	Motion-Compensated Prediction
MF	Multi Frame
MPEG	Moving Picture Experts Group
MV	Motion Vector
POC	Picture Order Count
pp	percentage point
PSD	Power Spectral Density
PSNR	Peak Signal to Noise Ratio
PU	Prediction Unit
PVR	Personal Video Recorder
QCIF	Quarter CIF
QP	Quantization Parameter; Quadratic Program
RAP	Random Access Point
RD	Rate Distortion
RDO	Rate Distortion Optimization
RDOQ	Rate Distortion Optimized Quantization
RQT	Residual Quadtree
SAO	Sample Adaptive Offset
SB	Sub-block
SDH	Sign Data Hiding
SDQ	Soft Decision Quantization
SDTV	Standard Definition Television
SHVC	Scalable HEVC
SNR	Signal to Noise Ratio
SVC	Scalable Video Coding
TB	Transform Block
TU	Transform Unit
TV	Total Variation; Television
UHD	Ultra High Definition
URQ	Uniform Reconstruction Quantizer
VA	Viterbi Algorithm
VCEG	Video Coding Experts Group

VLC	Variable Length Coding
VoD	Video on Demand

Bibliography

[ANR74] N. Ahmed, T. Natarajan, and K. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[BA83] P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31(4):532–540, Apr 1983.

[Bab86] L. Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[BB88] J. Barzilai and J. M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, January 1988.

[BBC11] S. Becker, J. Bobin, and E. Candès. NESTA: A Fast and Accurate First-Order Method for Sparse Recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, 2011.

[BJMO12] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, January 2012.

[Bjø01] G. Bjøntegaard. Calculation of average PSNR differences between RD-curves. ITU-T Study Group 16 Question 6, Video Coding Experts Group (VCEG), document VCEG-M33, March 2001.

[BL02] G. Bjøntegaard and K. Lillevold. Context-adaptive VLC (CVLC) coding of coefficients. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, document JVT-C028, May 2002.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

[BWO02] M. Beermann, M. Wien, and J.-R. Ohm. Look-ahead coding considering rate/distortion-optimization. In *Proc. IEEE International Conference on Image Processing (ICIP) 2002*, volume 1, pages I–93–I–96, 2002.

[Cha04] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1-2):89–97, 2004.

[Cis14] Cisco Visual Networking Index: Forecast and Methodology, 2013–2018. Cisco® Whitepaper, June 2014.

[CLK97] P.-Y. Cheng, J. Li, and C.-C. Kuo. Rate control for an embedded wavelet video coder. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):696–702, 1997.

[CW05] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[CWB08] E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

[DDDM04] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[DJ94] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[EHJT04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[Eri85] S. Ericsson. Fixed and adaptive predictors for hybrid predictive/transform coding. *IEEE Transactions on Communications*, 33(12):1291–1302, 1985.

[ETS14] Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream. European Telecommunications Standards Institute, document ETSI TS 101 154, draft version V1.12.1, May 2014.

[Eve63] H. Everett. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 11(3):399–417, May 1963.

[FAA$^+$12] C.-M. Fu, E. Alshina, A. Alshin, Y.-W. Huang, C.-Y. Chen, C.-Y. Tsai, C.-W. Hsu, S.-M. Lei, J.-H. Park, and W.-J. Han. Sample Adaptive Offset in the HEVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1755–1764, 2012.

[For73] G. D. Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[FP85] U. Fincke and M. Pohst. Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis. *Mathematics of Computation*, 44(170):463–471, 1985.

[Gir93] B. Girod. What's Wrong with Mean-squared Error? In A. B. Watson, editor, *Visual Factors of Electronic Image Communications*, pages 207–220. MIT Press, 1993.

[GO01] O. Guleryuz and M. Orchard. On the DPCM Compression of Gaussian Autoregressive Sequences. *IEEE Transactions on Information Theory*, 47(3):945–956, Mar 2001.

[Gol66] S. Golomb. Run-length encodings (corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, 1966.

[GZL$^+$13] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Atlanta, Georgia, USA, 2013.

[Hei68] D. M. Heien. A note on log-linear regression. *Journal of the American Statistical Association*, 63(323):1034–1038, September 1968.

[HKC13] T.-Y. Huang, C.-K. Kao, and H. H. Chen. Acceleration of rate-distortion optimized quantization for H.264/AVC. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS) 2013*, pages 473–476, 2013.

[HM79] C. L. Hwang and A. S. M. Masud. *Multiple objective decision making, methods and applications: a state-of-the-art survey.* Lecture notes in economics and mathematical systems. Springer-Verlag, 1979.

[Hoe62] A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58(3):54–59, 1962.

[HS63] J. Huang and P. Schultheiss. Block quantization of correlated gaussian random variables. *IEEE Transactions on Communications Systems*, 11(3):289–296, 1963.

[HS11] R. Hübner and A. Schöbel. When is rounding allowed? A level set approach to integer nonlinear optimization. *International Conference on Operations Research*, September 2011.

[HSK$^+$11] T.-Y. Huang, P.-Y. Su, C.-K. Kao, T.-S. Ou, and H. H. Chen. Quality improvement of video codec by rate-distortion optimized quantization. In *Proc. IEEE International Symposium on Multimedia (ISM) 2011*, pages 482–487, 2011.

[Huf52] D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[HV05] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I. Expected complexity. *IEEE Transactions on Signal Processing*, 53(8):2806–2818, 2005.

[HYZ08] E. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[HYZ10] E. T. Hale, W. Yin, and Y. Zhang. Fixed-Point Continuation Applied to Compressed Sensing: Implementation and Numerical Experiments. *Journal of Computational Mathematics*, 28(2):170–194, 2010.

[IEE08] IEEE Task P754. *IEEE 754-2008, Standard for Floating-Point Arithmetic.* IEEE, New York, NY, USA, August 2008.

[JCT13] Common test conditions and software reference configurations. Joint Collaborative Team on Video Coding (JCT-VC), document JCTVC-L1100, January 2013.

[JN84] N. S. Jayant and P. Noll. *Digital Coding of Waveforms, Principles and Applications to Speech and Video.* Prentice-Hall, Englewood Cliffs NJ, USA, 1984. ISBN 0-13-211913-7.

[KCYJ09] M. Karczewicz, P. Chen, Y. Ye, and R. Joshi. R-D based quantization in H.264. *Proc. SPIE*, 7443:744314–744314–8, 2009.

[KGFS11] A. Krutz, A. Glantz, M. Frater, and T. Sikora. Rate-Distortion Optimized Video Coding Using Automatic Sprites. *IEEE Journal of Selected Topics in Signal Processing*, 5(7):1309–1321, Nov 2011.

[KK98] W.-J. Kim and S.-D. Kim. Novel bit allocation method for the motion-compensated interframe coding in the sense of optimality. *Proc. SPIE*, 3653:1009–1017, 1998.

[KKL$^+$07] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[KML$^+$12] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park. Block Partitioning Structure in the HEVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1697–1706, 2012.

[KR07] A. N. Kim and T. A. Ramstad. Improving the Rate-Distortion Performance of DPCM Using Multirate Processing With Application in Low-Rate Image Coding. *IEEE Transactions on Signal Processing*, 55(10):4958–4968, Oct 2007.

[KYC08] M. Karczewicz, Y. Ye, and I. Chong. Rate Distortion Optimized Quantization. VCEG-AH21, January 2008. 34th Meeting of the ITU-T Video Coding Experts Group (VCEG), Antalya, Turkey.

[LK11] W. Lin and C.-C. J. Kuo. Perceptual visual quality metrics: A survey. *Journal of Visual Communication and Image Representation*, 22(4):297–312, 2011.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LXSW12] C. Lan, J. Xu, G. J. Sullivan, and F. Wu. Intra transform skipping. Joint Collaborative Team on Video Coding (JCT-VC), document JCTVC-I0408, April 2012.

[Mal08] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way.* Academic Press, 3rd edition, 2008.

[Max60] J. Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1):7–12, 1960.

[Mor62] J.-J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertian. *Reports of the Paris Academy of Sciences, Series A*, 255:2897–2899, 1962.

[MSW03] 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*, 13(7):620–636, 2003.

[NHW$^+$13] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and

T. Wiegand. Transform Coding Techniques in HEVC. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):978–989, Dec 2013.

[OPT00a] M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

[OPT00b] M. R. Osborne, B. Presnell, and B. A. Turlach. On the LASSO and Its Dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

[OSS$^+$12] J. Ohm, G. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand. Comparison of the Coding Efficiency of Video Coding Standards — Including High Efficiency Video Coding (HEVC). *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1669–1684, Dec 2012.

[PLXS12] X. Peng, C. Lan, J. Xu, and G. J. Sullivan. Inter transform skipping. Joint Collaborative Team on Video Coding (JCT-VC), document JCTVC-J0237, July 2012.

[Poh81] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bull.*, 15(1):37–44, February 1981.

[PSGF11] M. Pedersen, G. Simone, M. Gong, and I. Farup. A total variation based color image quality metric with perceptual contrast filtering. *International conference on Pervasive Computing, Signal Processing and Applications*, 2011.

[RCL00] J. Ribas-Corbera and S.-M. Lei. A frame-layer bit allocation for H.263+. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(7):1154–1158, 2000.

[RL79] J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162, March 1979.

[RO06] T. Rusert and J.-R. Ohm. Macroblock Based Bit Allocation for SNR

Scalable Video Coding with Hierarchical B Pictures. In *Proc. IEEE International Conference on Image Processing (ICIP) 2006*, pages 177–180, Oct 2006.

[ROF92] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear Total Variation Based Noise Removal Algorithms. *Physica D*, 60(1-4):259–268, November 1992.

[ROV93] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to MPEG video coders. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1993*, volume 5, pages 381–384 vol.5, 1993.

[ROV94] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to multiresolution and mpeg video coders. *IEEE Transactions on Image Processing*, 3(5):533–545, 1994.

[RV94] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. *IEEE Transactions on Image Processing*, 3(5):700–704, 1994.

[Seg76] A. Segall. Bit allocation and encoding for vector sources. *IEEE Transactions on Information Theory*, 22(2):162–169, 1976.

[SG88] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(9):1445–1453, 1988.

[SHMW05] H. Schwarz, T. Hinz, D. Marpe, and T. Wiegand. Constrained inter-layer prediction for single-loop decoding in spatial scalability. In *Proc. IEEE International Conference on Image Processing (ICIP) 2005*, volume 2, pages II–870–3, Sept 2005.

[Sik97] T. Sikora. MPEG Digital Video-Coding Standards. *IEEE Signal Processing Magazine*, 14(5):82–100, Sep 1997.

[SJN$^+$12] J. Sole, R. Joshi, N. Nguyen, T. Ji, M. Karczewicz, G. Clare, F. Henry, and A. Duenas. Transform Coefficient Coding in HEVC. *IEEE Transac-*

tions on Circuits and Systems for Video Technology, 22(12):1765–1777, 2012.

[SMW07] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, Sept 2007.

[SOHW12] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012.

[SS95] R. Schäfer and T. Sikora. Digital Video Coding Standards and Their Role in Video Communications. *Proceedings of the IEEE*, 83(6):907–924, Jun 1995.

[SS05] G. J. Sullivan and S. Sun. On dead-zone plus uniform threshold scalar quantization. *Proc. SPIE*, 5960:596033–596033–12, 2005.

[SS07] C. Segall and G. Sullivan. Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1121–1135, Sept 2007.

[SSW04] B. Schumitsch, H. Schwarz, and T. Wiegand. Inter-frame optimization of transform coefficient selection in hybrid video coding. *Proc. of Picture Coding Symposium*, 3(4):59–64, 2004.

[SSW05] B. Schumitsch, H. Schwarz, and T. Wiegand. Optimization of transform coefficient selection and motion vector estimation considering interpicture dependencies in hybrid video coding. *Proc. SPIE*, 5685:327–334, 2005.

[Sul96] G. Sullivan. Efficient scalar quantization of exponential and Laplacian random variables. *IEEE Transactions on Information Theory*, 42(5):1365–1374, Sep 1996.

[SW98] G. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, 1998.

[SW07] H. Schwarz and T. Wiegand. R-D Optimized Multi-Layer Encoder Control for SVC. In *Proc. IEEE International Conference on Image Processing (ICIP) 2007*, volume 2, pages 281–284, Sept 2007.

[Teu78] J. Teuhola. A compression method for clustered bit-vectors. *Information Processing Letters*, 7(6):308 – 311, 1978.

[Tib94] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[Tik63] A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. In *[Translated] Soviet Mathematics*, volume 4, pages 1035–1038, 1963.

[TMBR14] T. Tan, M. Mrak, V. Baroncini, and N. Ramzan. Report on HEVC compression performance verification testing. Joint Collaborative Team on Video Coding (JCT-VC), document JCTVC-Q1011, May 2014.

[UL01] J. Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis.* Grundlehren Text Editions. Springer Berlin Heidelberg, 2001.

[USC93] K. Uz, J. Shapiro, and M. Czigler. Optimal bit allocation in the presence of quantizer feedback. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1993*, volume 5, pages 385–388, 1993.

[VC13] S. Voronin and R. Chartrand. A new generalized thresholding algorithm for inverse problems with sparsity constraints. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2013*, pages 1636–1640, 2013.

[VH05] H. Vikalo and B. Hassibi. On the sphere-decoding algorithm II. Generalizations, second-order statistics, and applications to communications. *IEEE Transactions on Signal Processing*, 53(8):2819–2834, 2005.

[WG01] T. Wiegand and B. Girod. Lagrange multiplier selection in hybrid video coder control. In *Proc. IEEE International Conference on Image Processing (ICIP) 2001*, volume 3, pages 542–545, 2001.

[WH85] Z. Wang and B. R. Hunt. The discrete W transform. *Applied Mathematics and Computation*, 16(1):19–48, 1985.

[WLV00] J. Wen, M. Luttrell, and J. Villasenor. Trellis-based R-D optimal quantization in H.263+. *IEEE Transactions on Image Processing*, 9(8):1431–1434, 2000.

[WNC87] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[WNF09] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, July 2009.

[WSJ$^+$03] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, 2003.

[WSMW07] M. Winken, H. Schwarz, D. Marpe, and T. Wiegand. Joint Optimization of Transform Coefficients for Hierarchical B Picture Coding in H.264/AVC. In *Proc. IEEE International Conference on Image Processing (ICIP) 2007*, volume 4, pages 89–92, 2007.

[WSW08] M. Winken, H. Schwarz, and T. Wiegand. Joint rate-distortion optimization of transform coefficients for spatial Scalable Video Coding using SVC. In *Proc. IEEE International Conference on Image Processing (ICIP) 2008*, pages 1220–1223, Oct 2008.

[WYH$^+$12] J. Wang, X. Yu, D. He, F. Henry, and G. Clare. Multiple sign bits hiding for High Efficiency Video Coding. In *2012 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–6, 2012.

[WZD14] Y. Wu, H. Zhang, and R. Duan. Total Variation Based Perceptual Image Quality Assessment Modeling. *Journal of Applied Mathematics*, 2014.

[XCXZ12] Z. Xu, X. Chang, F. Xu, and H. Zhang. $L_{1/2}$ Regularization: A Thresholding Representation Theory and a Fast Solver. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1013–1027, 2012.

[YSGM10] A. Yang, S. Sastry, A. Ganesh, and Y. Ma. Fast ℓ_1-minimization algorithms and an application in robust face recognition: A review. In *Proc. IEEE International Conference on Image Processing (ICIP) 2010*, pages 1849–1852, 2010.

[YY07] E.-H. Yang and X. Yu. Rate Distortion Optimization for H.264 Interframe Coding: A General Framework and Algorithms. *IEEE Transactions on Image Processing*, 16(7):1774–1784, 2007.

[YY09] E.-H. Yang and X. Yu. Soft Decision Quantization for H.264 With Main Profile Compatibility. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(1):122–127, 2009.

[ZE10] M. Zibulevsky and M. Elad. L1-L2 Optimization in Signal and Image Processing. *IEEE Signal Processing Magazine*, 27(3):76–88, 2010.

[Zha08] T. Zhang. Multi-stage convex relaxation for learning with sparse regularization. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1929–1936. Curran Associates, Inc., 2008.

[ZMZ$^+$13] W. Zuo, D. Meng, L. Zhang, X. Feng, and D. Zhang. A generalized iterated shrinkage algorithm for non-convex sparse coding. In *Proc. IEEE International Conference on Computer Vision (ICCV) 2013*, pages 217–224, Dec 2013.