Efficient Coding of Transform Coefficient Levels in Hybrid Video Coding

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von
Tung Nguyen

Berlin 2023

Erstgutachter/in: Prof. Dr.-Ing. Heiko Schwarz

Zweitgutachter/in: Prof. Dr.-Ing. Jens-Rainer Ohm

Tag der Disputation: 03.05.2024

Selbstständigkeitserklärung

Name: Nguyen
Vorname: Tung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Datum: Unterschrift:

Diese Arbeit ist meiner Frau und meinen beiden Kindern gewidmet, deren unerschütterliche Unterstützung die Grundpfeiler für den erfolgreichen Abschluss dieser Arbeit sind.

Mein besonderer Dank gilt Heiko Schwarz und Jonathan Pfaff für ihre fachliche Anleitung und wertvollen Einsichten, die entscheidend zum Gelingen dieser Arbeit beigetragen haben. Ich möchte auch Detlev Marpe, Thomas Wiegand, Thomas Schierl und Ralf Schäfer für ihre jahrelange Unterstützung danken, die maßgeblich zur Vollendung dieser Arbeit beigetragen hat. Weiterhin danke ich allen meinen Kollegen, deren Namen hier nicht alle aufgeführt werden können, deren Beitrag jedoch unermesslich wertvoll war. Jeder von Ihnen hat auf einzigartige Weise zu diesem Werk beigetragen.

Abstract

All video coding standards of practical importance, such as Advanced Video Coding (AVC), its successor High Efficiency Video Coding (HEVC), and the state-of-the-art Versatile Video Coding (VVC), follow the basic principle of block-based hybrid video coding. In such an architecture, the video pictures are partitioned into blocks. Each block is first predicted by either intra-picture or motion-compensated prediction, and the resulting prediction errors, referred to as residuals, are compressed using transform coding. This thesis deals with the entropy coding of quantization indices for transform coefficients, also referred to as transform coefficient levels, as well as the entropy coding of directly quantized residual samples. The entropy coding of quantization indices is referred to as level coding in this thesis. The presented developments focus on both improving the coding efficiency and reducing the complexity of the level coding for HEVC and VVC. These goals were achieved by modifying the context modeling and the binarization of the level coding.

The first development presented in this thesis is a transform coefficient level coding for variable transform block sizes, which was introduced in HEVC. It exploits the fact that non-zero levels are typically concentrated in certain parts of the transform block by partitioning blocks larger than 4×4 samples into 4×4 sub-blocks. Each 4×4 sub-block is then coded similarly to the level coding specified in AVC for 4×4 transform blocks. This sub-block processing improves coding efficiency and has the advantage that the number of required context models is independent of the set of supported transform block sizes.

The maximum number of context-coded bins for a transform coefficient level is one indicator for the complexity of the entropy coding. An adaptive binarization of absolute transform coefficient levels using Rice codes is presented that reduces the maximum number of context-coded bins from 15 (as used in AVC) to three for HEVC. Based on the developed selection of an appropriate Rice code for each scanning position, this adaptive binarization achieves virtually the same coding efficiency as the binarization specified in AVC for bit-rate operation points typically used in consumer applications. The coding efficiency is improved for high bit-rate operation points, which are used in more advanced and professional applications.

In order to further improve the coding efficiency for HEVC and VVC, the statistical dependencies among the transform coefficient levels of a transform block are exploited by a template-based context modeling developed in this thesis. Instead of selecting the context model for a current scanning position primarily based on its location inside a transform block, already coded neighboring locations inside a local template are utilized. To further increase the coding efficiency achieved by the template-based context modeling, the different coding phases of the initially developed level coding are merged into a single coding phase. As a consequence, the template-based context modeling can utilize the absolute levels of the neighboring frequency locations, which provides better conditional probability estimates and further improves coding efficiency.

This template-based context modeling with a single coding phase is also suitable for trellis-coded quantization (TCQ), since TCQ is state-driven and derives the next state from the current state and the parity of the current level. TCQ introduces different context model sets for coding the significance flag depending on the current state. Based on statistical analyses, an extension of the state-dependent context modeling of TCQ is presented, which further improves the coding efficiency in VVC. After that, a method to reduce the complexity of the level coding at the decoder is presented. This method separates the level coding into a coding phase exclusively consisting of context-coded bins and another one consisting of bypass-coded bins only. For retaining the state-dependent context selection, which significantly contributes to the coding efficiency of TCQ, a dedicated parity flag is introduced and coded with context models in the first coding phase. An adaptive approach is then presented that further reduces the worst-case complexity, effectively lowering the maximum number of context-coded bins per transform coefficient to 1.75 without negatively affecting the coding efficiency.

In the last development presented in this thesis, a dedicated level coding for transform skip blocks, which often occur in screen content applications, is introduced for VVC. This dedicated level coding better exploits the statistical properties of directly quantized residual samples for screen content. Various modifications to the level coding improve the coding efficiency for this type of content. Examples for these modifications are a binarization with additional context-coded flags and the coding of the sign information with adaptive context models.

Zusammenfassung

Alle bedeutenden Videokodierungsstandards basieren auf einer blockbasierten hybriden Kodierungsarchitektur. Zu diesen Standards gehört Advanced Video Coding (AVC) und dessen Nachfolger, High Efficiency Video Coding (HEVC). Auch die neueste Entwicklung, Versatile Video Coding (VVC), welches der Nachfolger von HEVC ist, basiert auf diesem Konzept. Im ersten Schritt der blockbasierten hybriden Videokodierung werden die Videobilder in Blöcke unterteilt. Jeder Block wird zunächst prädiziert, und die resultierenden Restfehler werden mittels Transformationskodierung komprimiert. Dabei werden die Restfehler zuerst transformiert, anschließend die resultierenden Transformationskoeffizienten quantisiert und die Quantisierungsindizes mittels Entropiekodierung signalisiert. Diese Dissertation befasst sich mit der Entropiekodierung von Quantisierungsindizes für Transformationskoeffizienten sowie mit der Entropiekodierung von direkt quantisierten Restfehlern. Die vorgestellten Ansätze konzentrieren sich sowohl auf die Verbesserung der Kodierungseffizienz als auch auf die Reduktion der Komplexität für HEVC und VVC. Diese Ziele wurden durch die Modifikation der Kontextmodellierung und der Binarisierung erreicht.

Im ersten Ansatz wird eine Unterstützung für variable Transformationsblockgrößen, die in HEVC eingeführt wurde, in der Entropiekodierung vorgestellt. Dieser Ansatz nutzt die Beobachtung, dass Quantisierungsindizes, die von null verschieden sind, typischerweise in bestimmten Teilen des Transformationsblocks, unabhängig von der eigentlichen Blockgröße, konzentriert sind. In diesem Ansatz werden Blöcke, die größer als 4×4 sind, in kleinere 4×4 Unterblöcke unterteilt. Die Entropiekodierung für 4×4 Blöcke, wie sie in AVC verwendet wird, kann dann direkt für jeden 4×4 Unterblock angewendet werden. Durch diese Unterteilung und eine angepasste Kontextmodellierung verbessert sich die Kodierungseffizienz. Ein weiterer entscheidender Vorteil dieses Konzepts ist, dass die Anzahl der erforderlichen Kontextmodelle unabhängig von der Anzahl unterstützter Transformationsblockgrößen wird.

Ein Bewertungsmaß für die Komplexität der Entropiekodierung ist die maximale Anzahl der binären Symbole, die mit adaptiven Kontextmodellen kodiert werden. Hierbei stellen die binären Symbole zur Signalisierung der Quantisierungsindizes den größten Anteil dar. Im zweiten Ansatz wird daher eine alternative Binarisierung für die Quantisierungsindizes vorgestellt. Diese adaptive Binarisierung, im Gegensatz zur zuvor verwendeten festen Binarisierung, reduziert die maximale Anzahl an binären Symbolen pro Quantisierungsindex, die mittels Kontextmodellen signalisiert werden, von 15 auf drei für HEVC. Gleichzeitig erreicht die adaptive Binarisierung dieselbe Kodierungseffizienz wie die zuvor verwendete feste Binarisierung und verbessert die Kodierungseffizienz bei hohen Bitraten.

Die Kontextmodellierung umfasst unter anderem die Auswahl eines adaptiven Kontextmodells, mit dem ein binäres Symbol kodiert wird. Bislang nutzte die Kontextmodellierung für die binären Symbole der Quantisierungsindizes nur indirekt die Informationen von den bereits rekonstruierten Quantisierungsindizes im gleichen Block. Jedoch wurden direkte statistische Abhängigkeiten zwischen den Quantisierungsindizes innerhalb eines Blocks beobachtet. Im dritten Ansatz wird daher eine alternative Kontextmodellierung vorgestellt, die bereits rekonstruierte Quantisierungsindizes in der Nachbarschaft auswertet, um die Kodierungseffizienz für HEVC und VVC weiter zu verbessern.

Im dritten Ansatz wurde eine Kontextmodellierung entwickelt, die sich für die trellis-basierte Quantisierung (TCQ) eignet — eine zustandsbasierte Quantisierungsmethode mit zwei skalaren Quantisierern. Die Auswahl eines der beiden Quantisierer erfolgt durch den aktuellen Zustand, der aus dem Zustand sowie der Parität des vorangegangenen Quantisierungsindex abgeleitet wird. Dies ist möglich, da die im dritten Ansatz entwickelte Kontextmodellierung die Kodierreihenfolge entsprechend anpasst. Mit TCQ werden weiterhin verschiedene Kontextmodellsätze für die Kodierung der Signifikanz abhängig vom aktuellen Zustand verwendet. Im vierten Ansatz wird, basierend auf statistischen Analysen, eine Erweiterung der zustandsabhängigen Kontextmodellierung mit TCQ vorgestellt, die die Kodierungseffizienz in VVC weiter verbessert. Zudem wird der Ansatz so erweitert, dass er die Komplexität reduziert, ohne die Kodiereffizienz signifikant zu beeinträchtigen.

Der zuletzt vorgestellte Ansatz betrifft die Kodierung von Quantisierungsindizes, die keiner Transformation unterzogen wurden. Der sogenannte “Transform Skip Mode” wird oft bei Signalen verwendet, die nicht mit einer Kamera aufgezeichnet wurden. Aufgrund der statistischen Eigenschaften solcher Signale bieten Modifikationen der Kontextmodellierung und der Binarisierung eine signifikant verbesserte Kodierungseffizienz.

Contents

CHAPTER 1

Introduction

Contents

Video coding describes the compression and decompression of raw digital video content so that the amount of data is reduced while the reconstructed content provides as much fidelity as possible. A video coding system comprises an encoder to compress the raw data into a bitstream and a decoder to decompress the bitstream and reconstruct the video content. Figure 1.1 illustrates an example of an application that requires video coding, the live broadcasting of an event [1]. In this example, the scene is captured by a camera, which generates the raw digital video content. This raw signal serves as the input to the video encoder, which compresses the signal into a bitstream, a digital description of the video content with less data than the original raw signal. The bitstream could be stored, for example, for archiving purposes. In most applications, video signals are transmitted to a receiving side, for example, over a broadcast channel or the Internet. The practical importance of video coding can be seen when considering the amount of data for raw video sequences. A high-definition (1920×1080 spatial resolution) raw video sequence (4:2:0 subsampled $Y'C_BC_R$) with a frame rate of 50 Hz and 8-bit precision per sample requires about 1244 *megabits per second* (Mbps). For an ultra-high-definition raw video sequence (3840×2160) with the same frame rate and bit depth, one second of video requires about 4977 megabits. At the time of this thesis, the size of high-end consumer hard drives is 20 terabytes; with such a hard disk, about 67 minutes of raw ultra-high-definition video or four and a half hours of raw high-definition video could be stored. An alternative example demonstrating the necessity of video coding is the limited bandwidth available for typical end-consumer applications. At the time of this thesis, the median internet connection speed in Germany is about 83 Mbps [2]. This bandwidth is insufficient to transmit raw high-definition content for real-time playback. Furthermore, it is even lower in practical applications because the bandwidth has to be shared between multiple clients.

The progress in video coding, in the form of new video coding standards specifying new and refined algorithms, is mainly driven by the demand for higher coding efficiency, i.e., fewer bitstream data for the same perceptual quality or, vice versa, higher perceptual quality for the same amount of bitstream data. The emergence of new video applications often causes this demand for higher coding efficiency. Examples of video applications that appeared in the last decade are the introduction of new video formats (high-definition, ultra-high-definition), internet-based video-on-demand services, the usage of high-dynamic range, computer-generated, panorama, and 360-degree content, live streaming of video games, the usage of smartphones, and more. Another trend that has contributed to the demand for higher coding efficiency in recent years is the increase in the relative share of video data distributed over the Internet. As reported in [3], the share of video data transmitted over the Internet has increased significantly. A reason for the observed growth of video data is the increase in the number of video applications, such as watching video content using smartphones and Tablet computers. The report in [3] predicts that this trend will continue, emphasizing the importance of higher coding efficiency.

State-of-the-art video compression standards, such as *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC), its successor *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC), and the latest development at the time of this thesis, the *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC) standard, are based on a block-based hybrid video coding architecture. Each video picture is initially partitioned into fixed-sized blocks. These fixed-sized blocks are coded line-wise from left to right, and a block may be further partitioned into smaller blocks of variable size. Each block is first predicted either using already reconstructed samples of the same frame (intra-picture prediction) or samples of already reconstructed frames (inter-picture or motion-compensated prediction). The resulting prediction errors, also referred to as residuals, are transformed and quantized, and the resulting quantization indices are then coded into the bitstream

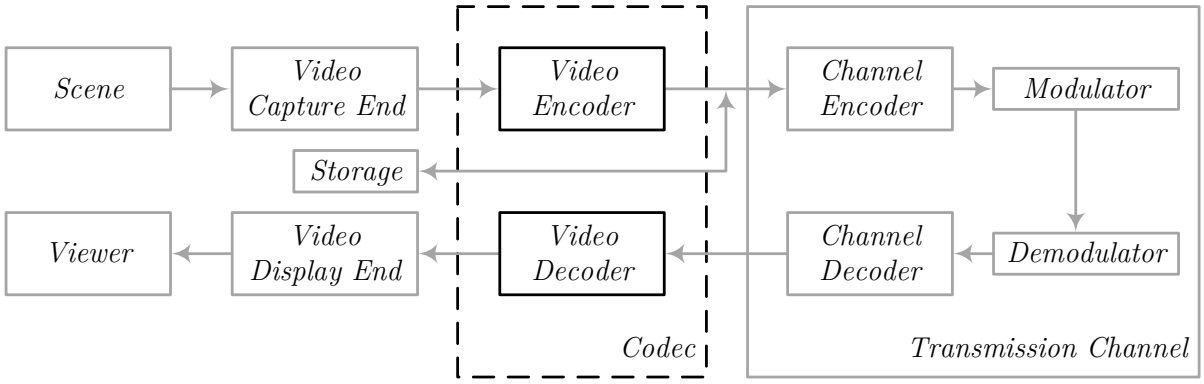

Figure 1.1
An example application that requires video compression: The live broadcasting of an event. In this example, the camera captures the scene and provides the raw digital video signal to a video encoder. The video encoder compresses the signal into a bitstream, which is a description of the video signal, but with less data than the original input video. The bitstream can be stored or transmitted over a transmission channel like a broadcast channel or the Internet.

using entropy coding. The quantization indices are referred to as transform coefficient levels and the coding of transform coefficient levels is referred to as transform coefficient level coding. Besides the transform coefficient levels, partitioning information, prediction and motion data, quantization step sizes, and more are entropy coded in the bitstream. The transform coefficient levels typically have a significant share of the bitstream. This share increases with a higher reconstruction fidelity and bit-rate. A well-designed entropy coding architecture, such as arithmetic coding, can achieve an efficiency close to optimal block codes when the conditional probabilities are known. In most cases, however, these conditional probabilities highly depend on the input signal, the chosen quantization step size, and the applied coding tools before the entropy coding. As a consequence, these conditional probabilities are not known with sufficient accuracy. Therefore, this thesis focuses on improving the estimation of suitable conditional probabilities for transform coefficient levels, which finally results in coding efficiency improvements.

1.1 | Organization of the Thesis

This thesis starts with a brief overview of hybrid video coding with a particular focus on entropy coding. After that, the developed context modeling and binarization techniques are presented in five chapters, each concentrating on a different concept for increasing the coding efficiency or reducing the implementation complexity. The following list briefly summarizes the chapters of the thesis:

- **Chapter 2 – Video and Entropy Coding:** This chapter starts with a brief overview of block-based hybrid video coding, followed by a review of entropy coding with a focus on the framework of *context-based adaptive binary arithmetic coding* (CABAC). Theoretical and practical assumptions are discussed, the blocks responsible for modeling the conditional probabilities in CABAC are identified, and the basic notations and configurations used for the experiments throughout this thesis are introduced.
- **Chapter 3 – Transform Coefficient Level Coding for Variable Block Sizes:** The introduction of transform block sizes larger than 4×4 samples in HEVC significantly improves the coding efficiency. However, a level coding that supports multiple transform sizes is required for entropy coding, because the level coding in AVC is only specified for 4×4 and 8×8 transform blocks. This chapter describes how the transform coefficient level coding can support variable transform block sizes without a linear dependency between the number of supported transform sizes and the number of context models. The independency between these two aspects is achieved by a partitioning of larger transform blocks into 4×4 sub-blocks and a sub-block-wise coding coupled with a sub-block-wise context modeling.
- **Chapter 4 – Adaptive Binarization of Transform Coefficient Levels:** An indicator for the complexity of entropy coding is the maximum number of context-coded symbols per sample or transform coefficient. The coding of binary symbols with adaptive context models is comparably complex due to

the feedback loop, which impedes efficient parallel hardware implementations. Therefore, the number of context-coded bins should be kept as low as possible without sacrificing coding efficiency. This chapter describes a method to reduce the maximum number of context-coded bins by utilizing an adaptive binarization for absolute transform coefficient levels with Rice codes.

- **Chapter 5 – Template-Based Context Modeling:** This chapter presents a template-based context modeling that improves coding efficiency since the evaluation of the neighboring frequency locations inside the template enables a suitable estimation of conditional probabilities. Further compression efficiency improvements are then achieved by changing the coding order within a sub-block so that absolute levels inside the template can be utilized for context modeling. This modification is equivalent to enabling more conditionals for context modeling.
- **Chapter 6 – Level Coding Suitable for Trellis-Coded Quantization:** An optional low-complex vector quantizer, referred to as *trellis-coded quantization* (TCQ), that improves coding efficiency significantly was included in the specification of VVC. The first part of this chapter briefly describes why the level coding developed in the preceding chapter is suitable for TCQ, and how the so-called state-dependent context modeling can be extended to improve coding efficiency further. The main topic of this chapter is throughput increase, which is achieved by separating the context- and bypass-coded bins of a sub-block and organizing them into two coding phases. Since this separation would break the compatibility of the level coding with TCQ, the binarization is modified by including a parity flag, which finally decreases the implementation complexity with a negligible impact on coding efficiency.
- **Chapter 7 – Transform Skip Residual Coding:** The increased use of screen content has led to the development of dedicated screen content coding tools that significantly improve coding efficiency for this type of content. An existing and straightforward technique to improve screen content coding is quantizing the residual samples directly, i.e., the transform is bypassed and the quantization indices are entropy coded using the conventional level coding. This chapter describes a new dedicated level coding for directly quantized residuals that further improves the coding efficiency for screen content.

1.2 | State-of-the-Art

In block-based hybrid video coding, the input pictures are partitioned into blocks, the blocks are predicted, and the resulting prediction error blocks are transmitted using the concept of transform coding. Transform coding [4] is a lossy data compression method widely used in multimedia applications such as image, video, and audio coding. Firstly, the input signal is mapped from the spatial or time domain to a transform domain using a unitary transformation, such as the Fourier transform [5] or the *discrete cosine transform* (DCT) [6]. The transformed data, referred to as transform coefficients, are quantized, and the resulting quantization indices (also called transform coefficient levels) are entropy-coded to generate a compressed representation of the original signal, which can be efficiently stored and transmitted. The first approaches of transform coding for images, which included Fourier and Hadamard transforms [7, 8], were proposed in 1968. Transform coding is based on the idea that most of the energy in a signal is concentrated in a few transform coefficients, and each transform coefficient can be coded independently. In practice, it turned out that statistical dependencies between transform coefficients of a block exist, which can be exploited by joint or conditional entropy coding techniques to achieve higher coding efficiency.

An entropy coding method widely used in the early days of image and video coding is run-length coding, with its first application dating back to 1967 [9]. As its name indicates, the basic concept of run-length coding is to signal the number of repeating zeros by a single value. Run-length coding is used in various block-based image and video coding standards that are no longer state-of-the-art, such as JPEG [10], *H.262/MPEG-2 Part 2* (MPEG-2) [11, 12], H.263 [13], and *MPEG-4 Part 2* (MPEG-4 Visual) [14]. In those image and video coding standards, a special variant of run-length coding, referred to as run-level coding, is used for coding the transform coefficient levels inside a block. In Baseline JPEG [15], the DC level of a block is coded differentially to the DC level of the previous block. The difference is coded as a pair $(category, value)$, where *category* specifies a range of values for the difference and is transmitted using a variable-length code, whereas *value* specifies the actual difference inside this range and is transmitted with a fixed-length code. The AC

levels of a block are mapped to a sequence using the zigzag scanning pattern. The sequence of scanned AC levels is represented by a sequence of triples $(run, category, value)$. The *run* specifies the number of zero-valued levels before the next non-zero level, *category* specifies a range for the next non-zero level, and *value* specifies the actual value of the next non-zero level inside this range. The pairs $(run, category)$ are coded using a variable-length code and the value is coded using a fixed-length code. A special codeword in the variable-length code is reserved for an end-of-block symbol, which signals that all remaining AC levels of the block are equal to zero. Two variable-length codes, which are not specified in the standard but have to be selected in the encoder, are used in JPEG: One for coding the category for DC levels and the other for coding the $(run, category)$ pairs for the scanned sequence of AC levels. A similar run-level coding is specified for MPEG-2 [16], but here $(run, level)$ pairs, with level being the value of the next non-zero level, are coded directly with a variable-length code specified in the standard. In H.263 and MPEG-4 Visual, an extension of the run-level coding is used; the variable-length code is specified for triples of the form $(last, run, level)$. The entry *last* specifies whether the next non-zero level is the last non-zero level inside the transform block. This concept does not require the signaling of the end of the block, but requires a so-called *coded block flag* to signal whether or not all coefficients in a block are zero-valued.

The concept of run-level coding can also be found in the AVC standard [17] with *context-based adaptive variable length coding* (CAVLC) [18]. However, the run-level coding in AVC significantly differs from the variants used in the previous image and video coding standards in the aspect that it exploits joint entropy coding to a larger degree and additionally incorporates conditional entropy coding to increase the coding efficiency. First, the number of non-zero levels and the trailing ones, i.e., the number of absolute levels equal to one at the end of the sequence of the non-zero levels, are coded using a single codeword. Furthermore, the variable-length code table used is chosen depending on the data in the left and top neighboring transform blocks. This approach is based on the observations that there are statistical dependencies between the numbers of non-zero transform coefficient levels in neighboring blocks and that the non-zero levels at the end of the scanned sequence typically have an absolute value equal to one. After that, the signs for the trailing ones are coded in reverse scanning order. The level magnitudes of non-zero levels that are not included in the trailing ones (their number can be derived by the total number of non-zero levels minus the number of trailing ones) are coded next. The last bit (*least-significant bit* (LSB)) of the codeword that represents a level magnitude indicates the sign for the corresponding level. Because the level magnitudes typically decrease towards the high-frequency scanning positions, these magnitudes are coded in reverse scanning order and the variable-length code table is adaptively selected for each scanning position depending on the previously coded level magnitude. Finally, after coding the total number of zero-valued levels from the DC location to the last non-zero level, the runs between two successive non-zero levels are coded for specifying the positions of the non-zero levels (also in reverse scanning order).

During the 1990s, wavelet-based transform coding gained popularity as an alternative architecture for image coding, with influential publications in [19, 20, 21, 22, 23]. These wavelet approaches commonly employ adaptive context models combined with a binary arithmetic coding engine for entropy coding. After applying a wavelet transform, the resulting wavelet coefficients of the subbands are entropy coded utilizing bit planes. When wavelet coefficients are represented using a fixed-length binary code, each bit plane denotes the bit values at a specific position within the codewords associated with the wavelet coefficients of the subband. In plain bit-plane coding, the coding process begins with the *most-significant bit* (MSB). The *embedded zerotree wavelet* (EZW) algorithm, an extension of plain bit-plane coding, uses a quadtree structure spanning over coefficients in different sub-bands of the same orientation that share the same spatial region [20]. This algorithm takes advantage of the high likelihood of zero-valued coefficients in higher sub-bands of the same orientation when a zero-valued coefficient is coded in the lower sub-band for the same spatial region. A further development of EZW is the *set partitioning in hierarchical trees* (SPIHT) technique that also utilizes similar statistical properties [22]. Tree construction and signaling are arranged using three distinct sets of wavelet coefficients, which are adjusted during encoding and decoding in SPIHT. The JPEG 2000 image coding standard [23] bases its wavelet coefficient coding on bit-plane coding without zerotrees, a method known as *embedded block coding with optimised truncation* (EBCOT) [24]. EBCOT organizes the coding of each bit plane into three coding passes, called significance propagation, magnitude refinement, and cleanup. A distinction is made between significant and insignificant positions, where significant positions are those

positions that have any binary value equal to 1 signaled in a previous bit plane. As no significant positions are available for the first bit plane, the coding begins with the cleanup pass, which is always invoked when insignificant locations without significant neighbors exist in the bit plane. The cleanup pass uses run-length coding, while the significance propagation pass individually codes the binary value of insignificant positions with significant neighbors. In the magnitude refinement pass, the binary symbols of significant positions in the current bit plane are coded.

The three wavelet-based image coding algorithms mentioned—EZW, SPIHT, and EBCOT—utilize template-based context modeling. In EZW, adaptive context models are chosen based on surrounding locations and the parent node [25]. In contrast, EBCOT determines the context model for coding an insignificant position according to the significance of the eight neighboring locations in the significance propagation pass. During the early stages of the AVC development, a wavelet-based video coding proposal featuring an architecture called *partitioning, aggregation, and conditional coding* (PACC) [26] was submitted in response to the *Call for Proposals* [27]. PACC employs zerotrees and template-based context modeling, where the locations covered by the template depend on the current bit plane level and potentially including the nearest neighbors within the causal neighborhood. The PACC concept laid the groundwork for the development of CABAC, an arithmetic coding architecture designed for block-based transform coding and first specified in AVC. CABAC offers higher coding efficiency compared to CAVLC [28], which is why it has been specified as the entropy coding method in both HEVC and VVC.

The usage of binary arithmetic coding with context modeling, a common approach in wavelet-based image and video coding, is also applicable to block-based hybrid video coding. Separating conditional probability estimation (context modeling) from codeword construction (binary arithmetic coding engine) offers modeling and implementation advantages, which are discussed in more detail in chapter 2. However, incorporating wavelet-based concepts into conventional block-based hybrid video coding presents several challenges. Conventional block-based video coding employs a DCT or similar block transforms rather than wavelet transforms. When applying a wavelet transform, the image structure is retained in the subbands, meaning that the statistical properties of the wavelet coefficients are relatively stable across the subbands. Furthermore, statistical dependencies between wavelet coefficients in different subbands are, to a large extent, predetermined by the structure of the wavelet transform. Strong dependencies typically exist between wavelet coefficients associated with the same orientation and spatial region of the image. These facts can be exploited by designing a suitable context modeling that considers these aspects. In block-based hybrid video coding, on the other hand, the transform is applied to a small section of a frame, and the input is a residual signal that highly depends on the original signal inside the block and the used predictor. Therefore, the statistical properties of transform coefficient levels may vary significantly from one block to another within the same frame. Moreover, in modern video coding standards, the differences in statistical properties of transform coefficient levels for blocks within the same frame are further increased by the usage of varying transform types and sizes. Beyond context modeling aspects, video coding must also address stringent complexity constraints required for real-time decoding and cater to a broad range of applications. For example, a video codec should be capable of efficiently handling both screen content and camera-captured content These challenges were successfully addressed for CABAC in AVC [28], but have to be taken into account for subsequent developments. In particular, HEVC and VVC introduce additional challenges due to the wider variety of block sizes and supported transforms. This thesis aims to identify the statistical dependencies arising from the new coding and partitioning tools introduced in HEVC and VVC and leverage them by employing context modeling inspired by wavelet approaches.

A significant portion of research on coding transform coefficient levels took place during the standardization phases of HEVC and VVC. Examples of these studies include [29], which proposed modifying the signaling of the last significant scanning position for HEVC; [30], which recommended signaling the coded sub-block flag for each sub-block; [31], which suggested signaling the area within the block consisting of non-zero coefficients as a rectangle; and [32], which suggested reducing the number of context models for VVC. Outside the standardization bodies, there is comparably less research on the topic of transform coefficient coding, one example being [33]. In this work, the authors developed a template-based context modeling approach based on an early version of the template-based context modeling discussed in chapter 5. Unlike

the approach discussed in chapter 5, they employed a different configuration for the position-dependent context model set. Additional research was carried out for AV1 [34, 35], a video codec developed by the industry consortium AOMedia. In AV1, the transform coefficient level coding combines existing approaches, such as the interleaved signaling of non-zero positions and the last significant scanning position of AVC, and a bit plane coding similar to those used for wavelet coefficients [36].

The standardization activities are also the driving factor for the context modeling and binarization techniques investigated in this thesis. The presented techniques were developed during the standardization phases of HEVC and VVC with the challenges mentioned above in mind. As a result, these techniques offer improved coding performance and versatility in a variety of coding conditions. Below is a brief summary of the development of the individual coding techniques presented in this thesis in the context of the standardization of HEVC and VVC:

- **Chapter 3 – Transform Coefficient Level Coding for Variable Block Sizes:** The development of the level coding presented in this chapter started with the development of HEVC. The level coding in AVC was specified for 4×4 transform blocks in the first published version in 2003. With the introduction of 8×8 transform blocks for the High profiles in AVC, the level coding specified for 4×4 transform blocks was extended to 8×8 transform blocks in a rather straightforward way. This extension could be generalized and applied to larger transform block sizes. However, this would result in the disadvantage that the number of context models linearly increases with the number of supported transform sizes, because each transform block size uses dedicated context model sets. Besides the linear increase in the number of context models, the coding efficiency may also suffer from context dilution, since the resulting context models are used more infrequently and, thus, adapt slower to the actual symbol statistics. The 4×4 sub-block processing presented in this chapter provides a solution to the undesired dependency between the number of supported transform sizes and the number of context models [37].
- **Chapter 4 – Adaptive Binarization of Transform Coefficient Levels:** When HEVC was in its early stages of development, the binarization of absolute transform coefficient levels in CABAC was inherited from the variant specified in AVC. However, there were concerns regarding the implementation complexity of context-adaptive coding, particularly for high bit rates. The main reason for this concern was that, in contrast to AVC, HEVC aimed to specify a single entropy coding method, which eventually led to the selection of CABAC. To address these complexity concerns, an adaptive binarization technique with Rice codes was developed, which is presented in this chapter. This technique reduces the maximum number of context-coded bins per level, and thereby decreases the complexity of CABAC [38].
- **Chapter 5 – Template-Based Context Modeling:** A first version of the template-based context modeling for transform coefficient levels [38] was proposed in the Fraunhofer HHI submission [39] to the *Call for Proposals* [40] that initiated the development of HEVC. This template-based context modeling was utilized for the coding of the significance flags and was included in the initial reference software for the HEVC development, which was called Test Model under Consideration (TMuC) [41]. The template-based context modeling presented in this chapter was developed for VVC; it represents an advanced version of the template-based context modeling in [42], which itself is an improved version of the variant initially presented in [39].
- **Chapter 6 – Level Coding Suitable for Trellis-Coded Quantization:** TCQ was initially proposed for VVC in combination with the level coding presented in chapter 5 [43]. However, in the level coding that utilizes absolute levels, context- and bypass-coded bins are interleaved. During the VVC standardization process, this interleaved transmission of context- and bypass-coded bins was considered a potential complexity issue. Therefore, the level coding presented in this chapter was developed, where the context- and bypass-coded bins are coded separately so that bypass-coded bins of a sub-block are transmitted successively. Because the parity information of the previously coded level, required for the state-dependent context model sets of TCQ, would become unavailable, the parity information is transmitted as a dedicated context-coded flag [44].
- **Chapter 7 – Transform Skip Residual Coding:** The transform skip residual coding presented in

this chapter was developed during the VVC development. The first concepts to improve the coding efficiency by adapting the context modeling were developed in HEVC *Range Extensions* (REext) [45]. The approach presented in this chapter further improves the coding efficiency for screen content by designing a dedicated transform skip residual coding that considers the statistical properties of transform skip residuals [46].

1.3 | Main Contributions

The developed context modeling and binarization techniques for the level coding presented in this thesis improve compression efficiency or reduce the implementation complexity. That is achieved by studying the statistical properties of transform coefficient levels. Based on the findings of these investigations, either the context modeling or the binarization, or both of them, are refined to achieve compression efficiency improvements or a reduction of implementation complexity.

The underlying entropy coding for the level coding considered in this thesis is the CABAC framework that employs context models to represent conditional probabilities for the binary symbols. These context models are adaptive, i.e., the binary probability of the employed context model is updated after coding a binary symbol. This property, in turn, means that a more suitable context modeling can better distinguish between different conditional binary distributions for the input symbols. Therefore, the objective of optimizing the level coding regarding compression efficiency or implementation complexity can be reduced to context modeling and binarization. Note that context modeling also includes changes in the scanning or coding order, because these modifications either alter the availability of variables that can be used for designing suitable conditions or enable new conditionals for the probability estimation.

The context modeling and binarization techniques presented in this thesis have contributed to more efficient video coding standards. The main contributions are summarized in the following list:

- The introduction of variable transform block sizes improves coding efficiency, but it requires a level coding that supports various transform block sizes. It is demonstrated that a level coding supporting transform block sizes larger than 4×4 can be realized efficiently by partitioning the block into 4×4 sub-blocks and coding the transform coefficient levels sub-block-wise.
- The sub-block-wise coding can be extended by a sub-block-wise context modeling, where the context model set used for the context-coded symbols of a sub-block is chosen depending on the preceding sub-block within the same transform block. It is demonstrated that such a sub-block-wise context modeling allows for sharing context models among different block sizes, which decouples the number of context models from the actual number of supported transform sizes and simultaneously improves the coding efficiency.
- A binarization for non-binary symbols is necessary for binary arithmetic coding. In the AVC standard, the binarization used for absolute transform coefficient levels represents the concatenation of two prefix codes, where the binary symbols of the first prefix code are coded with adaptive probability models. In contrast, the remaining symbols are coded in the low-complex bypass mode. However, the value for which the transition between the two prefix codes occurs (referred to as the cut-off value) results in a relatively high number of context-coded symbols per level. It is demonstrated that by using an adaptive binarization with Rice codes the cut-off value and, thus, the implementation complexity can be significantly decreased without harming the coding efficiency.
- It is further demonstrated that the adaptive binarization of absolute transform coefficient levels with Rice codes improves coding efficiency for high and very high bit-rate operation points.
- The statistical dependencies between the absolute levels within a given transform block can be utilized to improve the estimation of conditional probability models (context modeling). It is demonstrated that a local template that considers the already reconstructed significance information of the neighboring frequency locations improves the context modeling, which directly results in an improved coding efficiency.

- The context modeling highly depends on the available conditionals. For example, when the significance flags for all levels inside a transform block are coded first, the context modeling for the significance flag can only consider reconstructed significance flags. It is demonstrated that modifying the coding order so that the absolute levels become available for the template-based context modeling improves the coding efficiency.
- The template-based context modeling of the level coding that utilizes absolute levels for context modeling is also suitable for TCQ. For improving the level coding for TCQ, state-dependent context model sets are used for the context modeling of the significance flags, where different sets are used depending on the current quantization state. It is demonstrated that adding another context model set to the state-dependent context model sets further improves the coding efficiency.
- The implementation complexity of the level coding can be decreased by grouping the bypass-coded bins within a sub-block or transform block and coding them successively. For the level coding that combines a template-based context modeling with absolute levels, this complexity decrease can be achieved by separating the context- and bypass-coded bins within a sub-block into two coding phases. It is demonstrated that, by adding a dedicated parity flag, such a separation can be achieved with a negligible impact on coding efficiency for configurations with and without TCQ enabled.
- The maximum number of context-coded bins per transform coefficient was limited to three without negatively impacting the coding efficiency by reducing the cut-off value and using the adaptive binarization with Rice codes. It is demonstrated that this maximum number of context-coded bins per transform coefficient can be further reduced without harming the coding efficiency when decreasing the cut-off value adaptively during the coding of the transform coefficient levels.
- The statistical properties of screen content are significantly different from those of traditional camera-captured content. Encoders often choose a coding mode that skips the transform and directly quantizes the residual samples. The resulting levels are then entropy coded with the conventional level coding. It is demonstrated that modifying the context modeling for the directly quantized residual samples improves the coding efficiency of screen content.
- In order to avoid interference with the conventional level coding, the coding of directly quantized residual samples is implemented in a dedicated level coding, referred to as transform skip residual coding. It is demonstrated that scanning the transform block sub-block-wise from the top-left to the bottom-right corner of the block for transform skip residual coding improves the coding efficiency for screen content, and that the signaling of the last significant scanning position for transform skip residual coding does not provide any benefit.
- It is demonstrated that increasing the number of context-coded flags, reducing the size of the local template, and coding the sign information with adaptive context models significantly improve the coding efficiency for screen content.
- It is further demonstrated that the transform skip residual coding contributes to the overall coding efficiency of screen content when combined with other dedicated screen content coding tools.

1.4 | Practical Importance

The context modeling and binarization techniques presented in this thesis were developed with the aim of practicability, i.e., they were conceptually designed to be included in the video coding standards HEVC and VVC. For this purpose, several input contributions covering the presented context modeling and binarization techniques were submitted to either the *Joint Collaborative Team on Video Coding* (JCT-VC) responsible for the HEVC development or the *Joint Video Experts Team* (JVET) responsible for the VVC development. Correspondingly, many of the context modeling and binarization techniques presented in this thesis can be found in the video coding standards HEVC and VVC:

- The partitioning and processing of larger residual block sizes in sub-blocks is a concept that is used in the level coding specified in both HEVC and VVC.

- The adaptive binarization of transform coefficient levels using the concatenation of the TRU code, Rice codes, and the 0^{th}-*order exponential-golomb* (EG0) code is a concept that is used for the binarization of absolute transform coefficient levels in both HEVC and VVC.
- The template-based context modeling combined with a modified coding order, so that the neighboring frequency locations inside the local template contain absolute levels, served as the starting point for the template-based context modeling employed in the level coding of VVC.
- The extension of the state-dependent context model sets for context modeling of the significance flags by one additional context model set is used in the level coding of VVC.
- The separation of context- and bypass-coded bins for a sub-block combined with a binarization that includes a dedicated parity flag is used for the level coding specified in VVC.
- The restriction of the maximum number of context-coded bins that significantly reduces implementation complexity is specified for the level coding of VVC.
- VVC includes a dedicated transform skip level coding.
- The transform skip level coding of VVC employs a sub-block-wise scanning from the top-left to the bottom-right corner of the block.
- The transform skip level coding of VVC processes all sub-blocks of the block, meaning that the last significant scanning position, which is signaled in the conventional level coding, is not transmitted.
- In the transform skip level coding of VVC, a reduced local template for context modeling is utilized, more context-coded flags than in the conventional coefficient coding are coded for a level, and the sign information is coded with adaptive probability models.

CHAPTER 2

Video and Entropy Coding

Contents

Nowadays, successful modern video coding standards build upon the hybrid coding architecture, where block-based prediction is combined with transform coding of the prediction error signals. In this architecture, lossy coding, which is essential for achieving the compression ratios required in practical video applications, is achieved by quantization. Before the quantization, the signal undergoes a 2-dimensional transform, and the quantization is performed in the transform domain. The transform reduces statistical dependencies within the input signal, allowing scalar quantization to achieve a performance closer to that of vector quantization. It should be noted that the transform can only remove linear dependencies (i.e., correlations), which implies that the application of a transform does not make any difference if the input signal is uncorrelated. Another effect of transform coding is that the quantization in the transform domain is more pleasing to human perception.

At the end of the coding pipeline, entropy coding generates the output as a binary sequence, referred to as **bitstream**. Each bitstream contains information, such as the partitioning structure and prediction data, and the quantization indices referred to as **transform coefficient levels**. Entropy coding represents a lossless coding concept where the reconstructed signal is identical to the input signal. It is an important component within the hybrid video architecture, because it can contribute to the overall coding efficiency by utilizing the statistical properties of the provided signals and exploiting remaining statistical dependencies.

This chapter starts with a brief overview of the hybrid video coding architecture and its components to understand the input provided to the entropy coding within the modern hybrid video coding framework. The second part of this chapter includes a more detailed description of entropy coding and the theoretical and practical assumptions for the coding of the transform coefficient levels, the notations, and the configurations and experimental environment for the results presented throughout this thesis.

2.1 | Hybrid Video Coding

Each input frame is partitioned into fixed-size N×N blocks, where N is the edge length of the block, and each N×N block can be further subdivided into smaller blocks for prediction and transform coding in block-based hybrid video coding. This partitioning applied to the input frame forms a (partitioning) grid, and figure 2.1 illustrates an example of such a partitioning grid for a frame of an high-definition test sequence. The processing of the partitioning grid is commonly line-wise from left to right. Each N×N grid element, which is a block of 64×64 samples in this example, can be subdivided into smaller blocks, and the final result is a partitioning tree, where the tree structure depends on the employed partitioning technique, such as the

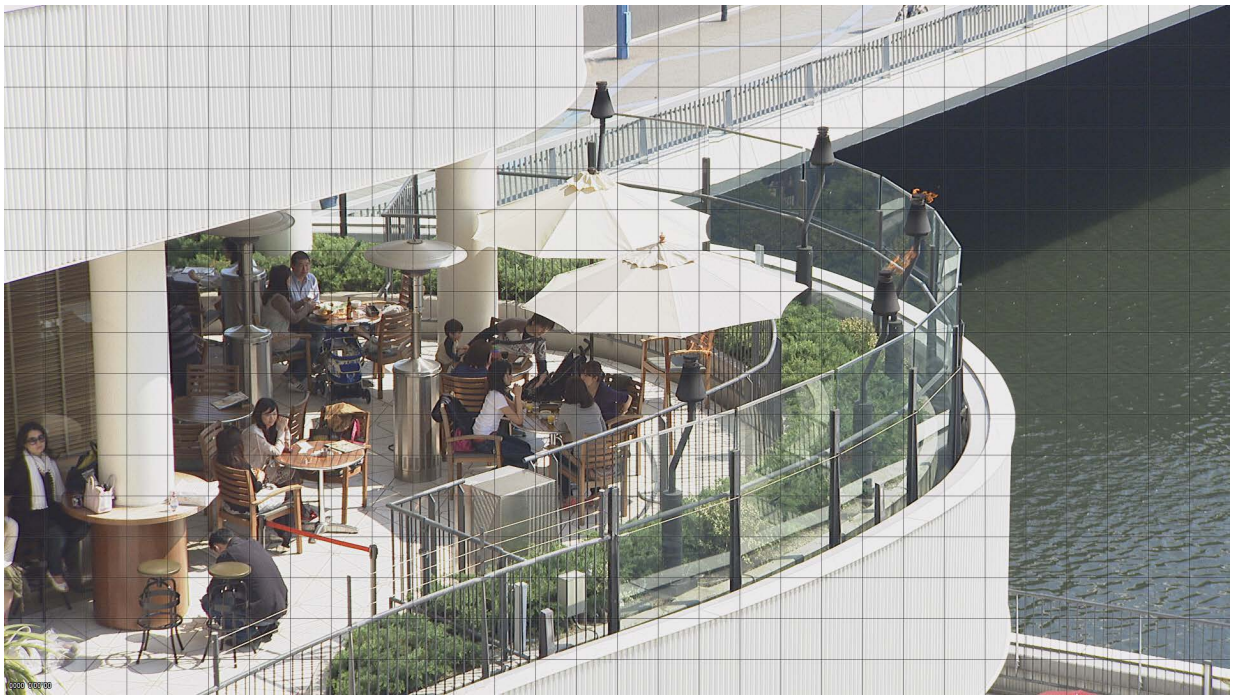

Figure 2.1
Snapshot taken from the test sequence *BQTerrace* (1920×1080) partitioned into 64×64 blocks of samples, which is also the maximum coding tree unit size in *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC). Each 64×64 block is processed individually line-wise from left to right.

quadtree approach. Prediction is performed for each block, which corresponds to a leaf of the partitioning tree, and the resulting prediction errors are referred to as the **residual signal**. A transform decorrelates the residual signal, and the following quantization reduces the signal information. Finally, the entropy coding encodes the resulting transform coefficient levels into the bitstream.

A simplified processing diagram from an encoder viewpoint is illustrated in figure 2.2 for each block, denoted as B_i that contains the original input samples. The original block samples are predicted using intra- or inter-prediction, followed by the transform of the prediction residuals and the quantization of the resulting transform coefficients. At the final stage of the pipeline, the entropy coding encodes the transform coefficient levels and other coding information into the bitstream. Further, it contributes to the overall coding efficiency by utilizing the statistical properties of the signal and exploiting the remaining statistical dependencies within the transform coefficient levels. The processing from the decoder viewpoint is inverse, i.e., the entropy decoder reconstructs the transform coefficient levels from the bitstream and provides the input to the scaling process, which is the counterpart of the quantization at the decoder side. After scaling the transform coefficients, the signal is transformed back into the spatial domain by the inverse transform, and the spatial signal is combined with the prediction signal resulting in the reconstructed signal. The difference between the original and reconstructed samples is referred to as **distortion**. The most commonly employed distortion metric is mean squared error.

Conceptually, block-based hybrid video coding combines a set of different coding tools into a single design: partitioning, prediction, transform, quantization, and entropy coding, where each part can be optimized for more efficiency (in terms of coding efficiency and complexity). However, state-of-the-art encoders need to implement advanced algorithms to exploit the whole extent provided by sophisticated coding tools. Detailed overviews for the specific video coding standards are provided in [47] for HEVC and in [48] for *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC). A broader description of the development line for the H.26x video coding standards can be found in [49].

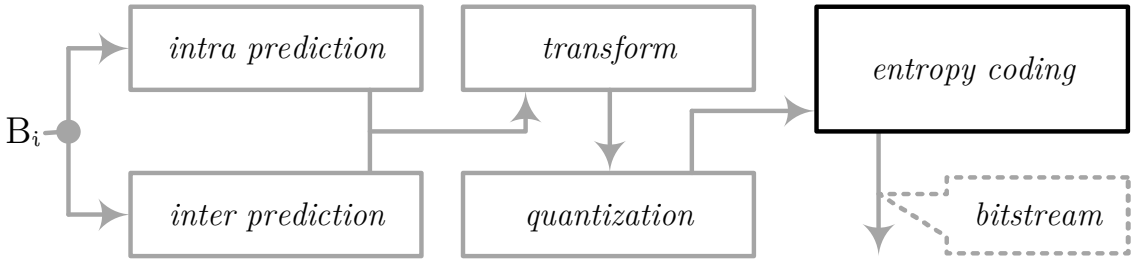

Figure 2.2
Simplified block diagram of block-based hybrid video coding architecture from the encoder viewpoint, where the input is a block B_i containing original samples. A prediction is performed for the samples in B_i using either intra- or inter-prediction. The prediction errors, referred to as residuals, are then transformed, and the obtained transform coefficients are quantized, resulting in quantization indices, also referred to as transform coefficient levels. At the end of the coding pipeline, the entropy coding utilizes statistical properties and exploits the remaining statistical dependencies for further coding efficiency improvements.

2.1.1 | Partitioning

The partitioning specifies how each grid node is subdivided into smaller blocks, where the syntax and semantics depend on the employed technique. Different alternatives are possible for the shape, the size, and the signaling, which leads to a certain partitioning tree. Those aspects form the design of the partitioning scheme. Figure 2.3 illustrates the partitioning in HEVC, which uses quadtree structures with a corresponding coding order. In this example, a flag is signaled for each block, specifying whether or not a 2N×2N block is further subdivided into four N×N blocks. More information and detailed description for partitioning in practical video standards can be found in [38] for HEVC and in [50] for VVC.

2.1.2 | Prediction

Each block employs a prediction that generates a residual signal with usually less energy than the original input signal. There are two main prediction modes: intra-prediction employing reference samples of the already reconstructed regions of the current frame and inter (or temporal) prediction employing reference samples of fully reconstructed preceding frames.

Intra Prediction

The top and the left samples adjacent to the current block form reference samples in intra-prediction, and so-called angular intra modes predict the current block samples according to an angle using extrapolation. Further, specialized intra-prediction modes suitable for a specific type of texture or content exist, such as luma-to-chroma intra prediction allowing the prediction of chroma blocks from the co-located luma samples, the planar mode, where the extrapolation is according to a plane, or the DC mode that uses the mean of the reference samples as predicted value for all sample positions. An overview of the different intra-prediction modes available in HEVC, such as the planar mode, is given in [51]. A respective overview for VVC is given in [52].

Inter Prediction

A reconstructed area in a fully reconstructed frame provides the reference samples for inter-prediction. The spatial region used as the reference is indicated by a motion vector specifying the spatial displacement relative to the current block, and a reference index specifying the employed reconstructed picture. Motion vector prediction is used for efficient signaling, and only the difference between the current motion vector relative to the predictor is coded in the bitstream. How the predictors are constructed and what kind of the special modes exist, such as skip and merge, and more, are described in-depth for HEVC in [53], and the in-depth description for VVC can be found in [54].

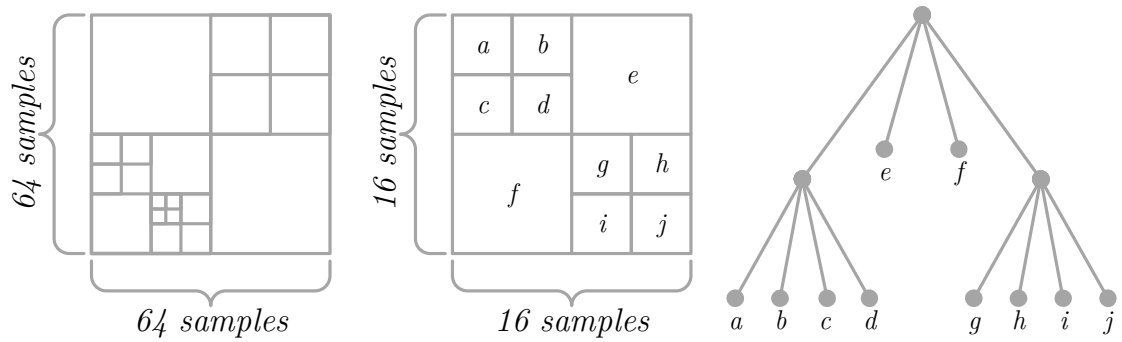

Figure 2.3
Example for the partitioning in HEVC using quadtree structures where a single flag signals the split of a 2N×2N node into four N×N child nodes. The center illustration shows the depth-first Z-scanning processing denoted by alphabet characters, and the right illustration shows the corresponding quadtree structure.

2.1.3 | Transform

A transform in video coding is, in theory, a reversible process that changes the basis of the 2-dimensional signal representing the information in the spatial domain. After the application of a suitable transform, which is usually based on either sine or cosine basis functions with the corresponding transforms being referred to as *discrete sine transform* (DST) or *discrete cosine transform* (DCT), respectively, the block signal is decorrelated, and its energy is typically concentrated into few transform coefficients. Practical video coding standards employ integer approximations of the corresponding transforms, which can be implemented more efficiently. More information for the transforms in HEVC can be found in [47], and for VVC an overview can be found in [55].

2.1.4 | Quantization

The quantization that follows the transform is the non-reversible phase of the coding pipeline, and it reduces the signal information so that the compression factor necessary for practical applications can be achieved. A simple quantization approach is a *uniform reconstruction quantization* (URQ), where the quantization step size Δ defines the distance between two reconstruction values and, thus, the accuracy of the transform coefficients. Because practical video coding standards specify the scaling process only, the counterpart of the quantization at the decoder side, practical encoders can choose the quantization implementation depending on their requirements. Adjusting the quantization step size Δ indirectly controls the bit-rate, and practical encoders usually implement a user-defined *quantization parameter* (QP), where internally the QP is mapped to a Δ value. Further information on quantization can be found in [56].

2.1.5 | Entropy Coding

From the encoder viewpoint, the entropy coder is the last component of the coding pipeline. It is responsible for the output of the transform coefficient levels and the side information as a sequence of binary symbols, referred to as bitstream. State-of-the-art video coding standards employ *context-based adaptive binary arithmetic coding* (CABAC) [57] as the entropy coding scheme, which has three major components: binarization, context memory, and an arithmetic coding engine.

2.2 | Context-Based Adaptive Binary Framework

The CABAC framework inherits an arithmetic coding engine, a concept that goes back to Shannon-Fano-Elias and block codes, which will be briefly reviewed next.

In block codes [1], a sequence of input symbols $(s_1, s_2, \ldots, s_M)$ is mapped to a codeword according to the joint probability $p(s_1, s_2, \ldots, s_M)$, where M is the sequence length. Block codes can achieve an efficiency close to the entropy rate (if it exists) for a large value of M. However, it is impossible to implement general block codes for larger sequence lengths, because of excessive memory prerequisites due to the number of codewords.

2.2.1 | Shannon-Fano-Elias Codes

A solution that avoids the need for storing the codewords is Shannon-Fano-Elias coding. Let $\mathcal{A}$ be an alphabet and $\mathcal{V} = \{v_0, v_1, \ldots\}$ be an ordered set of all possible sequences of symbols $s_i \in \mathcal{A}$ with a sequence length equal to M. Furthermore, for each $v_k \in \mathcal{V}$, let $p(v_k)$ denote the joint probability. In Shannon-Fano-Elias coding, each sequence v_k is assigned to a half-open interval $[L_k, L_k + W_k)$, where the width W_k and the lower bound L_k of the interval are derived by

$$W_k = p(v_k) \qquad L_k = \sum_{i=0}^{k-1} p(v_i)\,. \tag{2.1}$$

For each of the intevals $[L_k, L_k + W_k)$, the corresponding codeword $b_1 b_2 \ldots b_K$ can be found by selecting a real number $c_k \in [L_k, L_k + W_k)$ that has a binary representation $c_k = 0.b_1 b_2 \ldots b_K$ with $K = \lceil -\log_2 p(v_k) \rceil$.

2.2.2 | Iterative Shannon-Fano-Elias Codes

In contrast to joint probabilities, an advantage of conditional probabilities is that they can be described by simpler models while achieving a sufficient approximation of the general variant. Such an approximation eventually reduces the memory requirements to a feasible level for practical implementations. The iterative variant of Shannon-Fano-Elias codes utilizes conditional instead of joint probabilities, where each symbol s_k^m of the sequence v_k is evaluated to determine iteratively the current width W_k^m, and the current lower bound L_k^m. Exactly as in the non-iterative variant, each resulting codeword $b_1 b_2 \ldots b_K$ corresponds to a real number $c_k \in [L_k^M, L_k^M + W_k^M)$ that has a binary representation $c_k = 0.b_1 b_2 \ldots b_K$ with $K = \lceil -\log_2 W_k^M \rceil$. A further advantage of the concept is that some bits of the codeword $c_k = 0.b_1 b_2 \ldots b_K$ are settled during the iteration and can be transmitted while processing the symbols of the sequence, which makes the concept feasible for practical applications. This iterative concept forms nested intervals, i.e., the first symbol s_k^1 determines the interval $[L_k^1, L_k^1 + W_k^1)$, and the next interval $[L_k^2, L_k^2 + W_k^2)$ is within the interval $[L_k^1, L_k^1 + W_k^1)$, where in each iterative step the current interval is refined towards the final interval. In contrast to the non-iterative variant, the set of all possible sequences $\mathcal{V}$ does not have to be ordered, but the alphabet $\mathcal{A}$ has to be ordered.

Let $p\left(s_k^i | s_k^{i-1}, s_k^{i-2}, \ldots, s_k^1\right)$ denote the conditional probability which depends on all available preceding symbols of the sequence v_k, and $p\left(s_k^i | C\left(s_k^{i-1}, s_k^{i-2}, \ldots, s_k^1\right)\right)$ denotes an estimation, where $C(\cdot)$ is the conditional that represents a model. An example for simpler models using the conditional is $p\left(s_k^i | C(\cdot)\right) = p\left(s_k^i\right)$ when it is assumed that there are no statistical dependencies among the symbols of the sequence, or a first-order model with $p\left(s_k^i | C(\cdot)\right) = p\left(s_k^i | s_k^{i-1}\right)$. Algorithm 2.1 summarizes the iterative approach from the encoder viewpoint using the modeling by $C(\cdot)$, where the interval width $W_k^m := W_m$ is updated depending on the conditional probability of the current symbol $s_k^m := s_m$, and the interval bound $L_k^m := L_m$ is updated depending on the sum of conditional probabilities for all symbols in the alphabet prior to s_m.

Algorithm 2.1 Overview of the iterative Shannon-Fano-Elias codes from the encoder viewpoint. The codeword can be any value in $[L_M, L_M + W_M)$, and the algorithm suggests the binary representation of $\lceil L_M \cdot 2^K \rceil$ with K bits.

Require: $v_k = \left(s_k^1, s_k^2, \ldots, s_k^M\right) =: (s_1, s_2, \ldots, s_M)$
1: $m \leftarrow 1$, $L_0 \leftarrow 0$, $W_0 \leftarrow 1$
2: **while** $m \leq M$ **do**
3: $\quad W_m \leftarrow W_{m-1} \cdot p(s_m | C(s_{m-1}, s_{m-2}, \ldots, s_1))$
4: $\quad L_m \leftarrow L_{m-1} + W_{m-1} \cdot \sum_{\forall a \in \mathcal{A}: a < s_m} p(a | C(s_{m-1}, s_{m-2}, \ldots, s_1))$
5: $\quad m \leftarrow m + 1$
6: **end while**
7: $K \leftarrow \lceil -\log_2 W_M \rceil$
8: outputs binary representation of $\lceil L_M \cdot 2^K \rceil$ with K bits

2.2.3 | Arithmetic Codes

The presented iterative variant of Shannon-Fano-Elias codes is impossible to implement for practical application in its current form due to the high precision required for the parameters. Arithmetic coding replaces the

$\lvert x\rvert$	TRU					$\ell(x)$
0	0					*1*
1	1	0				*2*
2	1	1	0			*3*
3	1	1	1	0		*4*
4	1	1	1	1	0	*5*
5	1	1	1	1	1	*5*

Table 2.1
Example of the TRU code for $|x| \in \{0, 1, 2, 3, 4, 5\}$ and maxVal $= 5$, where blue bins denote the terminating binary symbol. Due to the truncation, the codeword length for $|x| = 4$ and $|x| = 5$ are the same with $\ell(4) = \ell(5) = 5$.

real-valued parameters L, W, and the probabilities by finite-precision arithmetic. Different configurations are possible on the number of bits for each parameter, the updating rules for L and W, and the renormalization rule required to maintain the precision. In CABAC, the arithmetic coding engine is referred to as modulo coder [28].

2.2.4 | Challenges in Entropy Coding

With arithmetic codes, entropy coding can reach an efficiency close to optimal block codes with the same block sizes, when the conditional probabilities are known, whereas using wrong probabilities leads to an increased bitstream size. The challenge in entropy coding can be identified as modeling or estimating the (conditional) probabilities as accurately as possible, so that the bitstream size is minimized.

2.2.5 | Design Principles of CABAC

As its name implies, CABAC is a framework with a binary arithmetic coding engine, which has some advantages over multi-alphabet designs. Binary arithmetic coding reduces implementation costs, because the engine has to maintain two intervals only, and the estimation of binary (conditional) probabilities tends to be more feasible than for multi-alphabets in practical applications.

Besides the binary arithmetic coding engine, the CABAC framework inherits a binarization that maps non-binary inputs to a sequence of bins. For that, a prefix code is required, but the actually used code is not relevant for known *probability mass function*s (pmfs) because the symbol probabilities can be uniquely mapped to probabilities for the bins of the prefix code. For the estimation of the pmf, the framework provides writeable context memory, where each adaptive context model represents a binary pmf that can be used to specify the probability of the bin that is encoded or decoded. The assignment of a context model to a bin depends on the design of the context modeling, and the associated task can be abstracted as determining the pmf for a given bin.

2.2.6 | Binarization

For non-binary input symbols, the binarization maps the input into a sequence of bins, referred to as bin string. Typically, only the absolute value is considered with the sign transmitted separately as a flag for integer-valued input, such as for transform coefficient levels. Two types of codes are commonly used in the CABAC framework: the *truncated unary* (TRU) code and the 0^{th}*-order exponential-golomb* (EG0) code, and the following paragraphs give a brief description of the two.

Truncated Unary

A unary code is a prefix-free code for countable but infinite alphabets where the codeword of the current symbol within the ordered codeword table is one binary symbol longer than the codeword of its preceding symbol. For $|x| \in \mathbb{N}_0$, the codeword is $|x|$ times $'1'$, and a terminating symbol $'0'$, where the bin values for the non-terminating and the terminating symbols are interchangeable. The terminating symbol for the codeword representing the last value within finite sets can be omitted, which leads to the same codeword length for the last value of the set and its predecessor. Table 2.1 lists an example for the resulting TRU code with the finite set $\{0, 1, 2, 3, 4, 5\}$, where the two last elements have the same codeword length with

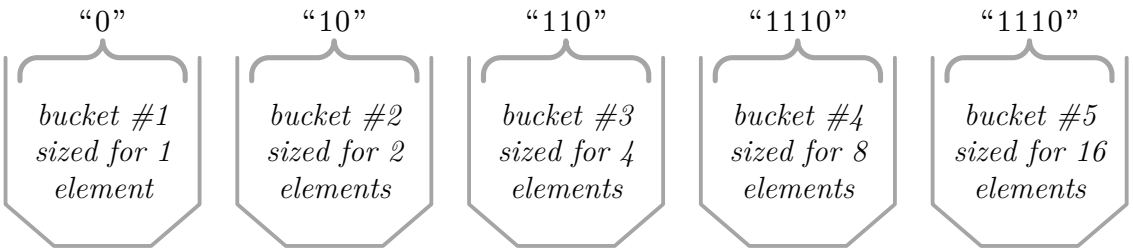

Figure 2.4
Bucket model of the EG0 or Elias-γ code, where the number of items a bucket can contain doubles for each further bucket, resulting in a variable-sized bucket concept.

$\ell(4) = \ell(5) = 5$.

0^{th}-order Exponential-Golomb (EG0)

The EG0 code is equivalent to the Elias-γ code [58], where a codeword consists of a prefix and a suffix. Each $|x| \in \mathbb{N}_0$ can be represented by

$$|x| = 2^k - 1 + r, \tag{2.2}$$

with $k \in \mathbb{N}_0$ and $r \in \{0, \ldots, 2^k - 1\}$. The prefix is the unary code of k, and the suffix is the fixed-length binary code of r with a length equal to k. An absolute offset, which is equal to $2^k - 1$, is indicated by the prefix alone, whereas the suffix indicates the difference between $|x|$ and $2^k - 1$. For this type of code, an often employed representation is the bucket model, where the ordered buckets are identifiable by k, and each bucket contains exactly k items in the case of EG0. Figure 2.4 illustrates the bucket concept for the EG0 or Elias-γ code, where the prefix specifies the bucket index and the suffix points to the concrete item within the selected bucket.

Table 2.2 lists the codewords for the first five elements of $|x| \in \mathbb{N}_0$, where the red-marked bins denote the prefix and the blue-marked bins denote the suffix. A property of the EG0 code is that the suffix length increases by one symbol for each further bucket, resulting in a doubling of the bucket size of each further bucket, i.e., the bucket sizes are equal to $\{1, 2, 4, 8, \ldots, 2^k, \ldots\}$ with k being the bucket index. Moreover, since the prefix for the indication of each further bucket increases by one symbol, the transition from one bucket to the next bucket results in an increase in codeword length by two symbols, whereas the codeword length for values within a bucket is the same.

2.2.7 | Adaptive Context Models

Each bin coded with the arithmetic coding engine requires a probability, whose exact value is usually unknown in practical applications. In the CABAC framework, so-called adaptive context models are provided, and each of them represents one out of 128 possible binary pmfs, where the parameter of the pmf, i.e., either the probability $P(x = 1) = p$ or $P(x = 0) = 1 - p$ is implemented as an internal state. After encoding or decoding a bin with an adaptive context model, the actual bin value is used to update the pmf represented by the employed context model.

$\|x\|$	EG0					$\ell(x)$
0	0					*1*
1	1	0	0			*3*
2	1	0	1			*3*
3	1	1	0	0	0	*5*
4	1	1	0	0	1	*5*
5	1	1	0	1	0	*5*

Table 2.2
Example of the EG0 or Elias-γ code with the input is $|x| \in \mathbb{N}_0$, where the prefix is the unary code and the suffix the fixed-length binary code. Since the first bucket can contain one item only, the suffix is ($\emptyset$).

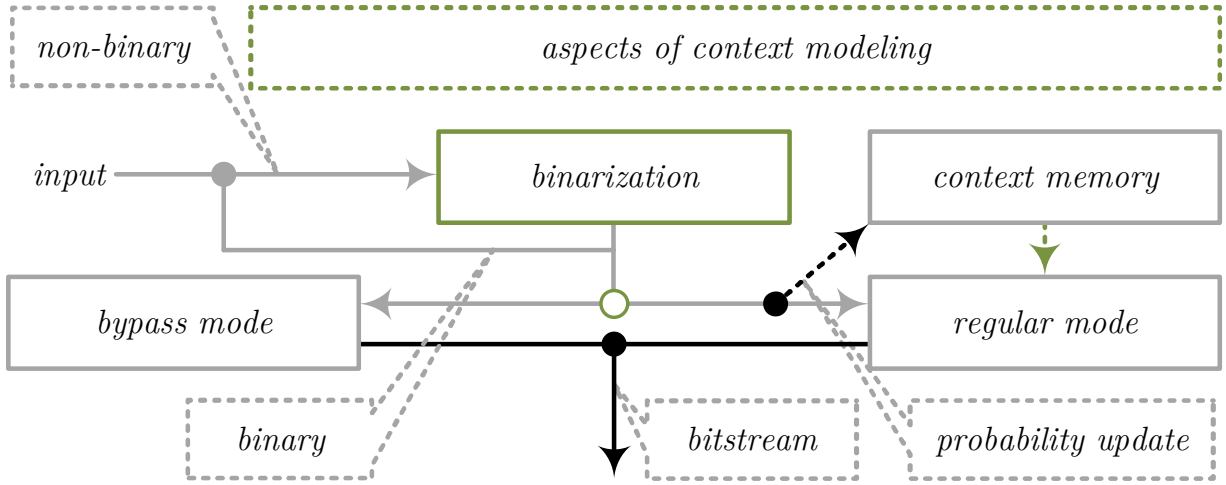

Figure 2.5
Block diagram of CABAC from the encoder viewpoint. Solid lines represent the flow of bins, where solid gray lines denote the time before encoding a bin, and solid black lines denote the time after encoding a bin. Dotted arrows represent the flow of information from and back to the context memory. Annotations contain additional information for clarification.

The simplest form of probability estimation for a binary source is counting the number of bins equal to one and dividing that number by the total number of observed bins. Let $p_{n+1} = P(b_{n+1} = 1)$ denotes the estimated probability for the next bin being equal to one, then the relative frequency approach can be formulated as:

$$p_{n+1} = \frac{1}{n+1} \sum_{i=0}^{n} b_i. \tag{2.3}$$

In this simple approach, all bins have an equal impact on the estimated probability of the next symbol. A window or weight function can be used to give more recent bins a higher influence. For a so-called exponential decay window, the weights are derived using an exponential decay, where the weight for b_{i+1} is proportionally smaller than that for b_i. This exponential decay window can be implemented recursively, where α controls the window characteristics, i.e., the weight distribution or the rate of adaptation:

$$p_{n+1} = \alpha p_n + (1 - \alpha) b_n. \tag{2.4}$$

The multiplication-free implementation in CABAC discretizes the possible values of the probabilities p_n using 128 states, which can be represented by a 7-bit integer, with α being:

$$\alpha = \sqrt[63]{\frac{0.01875}{0.5}} \approx 0.95. \tag{2.5}$$

The real-valued probability for each state q is then:

$$p(q) = \begin{cases} 0.5 \cdot \alpha^{63-q}, & \text{if } q < 64, \\ 1 - 0.5 \cdot \alpha^{q-64}, & \text{if } q \geq 64. \end{cases} \tag{2.6}$$

A state transition table, which implements a *finite-state machine* (FSM), is used to perform the probability update in CABAC, because the update only depends on the current internal state and the actual bin value. In VVC, the probability estimation is extended by a further estimator, and the final estimation is the combination of the two estimators, which, among other effects, enables a faster adaptation at the beginning [56].

2.2.8 | Context Modeling

A block diagram of CABAC from the encoder viewpoint is illustrated in figure 2.5. In this block diagram, solid lines represent the flow of the input and the bins, and dotted lines represent the flow of information

from and back to the context memory. Gray lines represent the time before encoding a bin, and black lines represent the time after encoding a bin. The coding of a non-binary input starts the binarization process. Each bin is coded with either an adaptive context model, where the bin probability is derived from the internal state of the context model, or in the bypass mode with a fixed uniform pmf. For each bin not coded in the bypass mode, the internal probability of the employed context model is updated using the actual bin value, as denoted by the feedback to the context memory in figure 2.5.

Context Modeling Task

From a general perspective, the task of context modeling is to select a conditional probability estimate $p\left(b_n|C\left(b_{n-1}, b_{n-2}, \ldots, b_0\right)\right)$ that is close to the real conditional probability $p\left(b_n|b_{n-1}, b_{n-2}, \ldots, b_0\right)$ for the bin b_n to be coded. This task can be relatively difficult when trying to estimate the conditional probabilities directly, and the decomposition into different modules, as in the CABAC framework, makes the task more feasible. Adaptive context models enable the focus of the design on apriori knowledge about the statistical properties and dependencies, rather than a direct estimation of the probability. Let x be an absolute transform coefficient level, which is binarized using the unary code. Then, each bin b_i denotes the conditional event "$x > i \| x > i-1, \ldots, x > 0$" with the associated proability $P\left(b_i\right) \coloneqq P\left(x > i | x > i-1, \ldots, x > 0\right)$. An example of apriori knowledge is the observation that $P\left(x > 0\right) > P\left(x > 1 | x > 0\right)$. The observed statistical property can be exploited by using different context models when coding b_0 and b_1. During the coding process, the context models adapt to the actual probabilities, which depend on the input signal, operation point, and more. For the above example, the probability of $P\left(x > 0\right)$ should be different from $P\left(x > 1 | x > 0\right)$, and using the same context model for both bins would increase the average codeword length, if the assumption is correct.

Though selecting an adaptive context model is the main task of the context modeling within the CABAC framework, three more aspects interact directly with the context modeling, which should be attributed to the task of the context modeling and should be considered during its design.

- The choice of a prefix code used for the binarization of the non-binary input interacts with the context model selection, because a decision on which context model should be used for each bin of the bin string has to be made. When using a different prefix code, the probabilities for the bins alter; therefore, the binarization design influences the context modeling and is considered as a part of the context modeling in this thesis.
- For each bin, a decision has to be made whether the bin should be coded with an adaptive context model or not, depending on apriori knowledge of the statistical properties.
- The coding order, which cannot be depicted in figure 2.5, is another aspect that is a part of the context modeling, because it affects the appearance order, and thus, the conditionals $C\left(\cdot\right)$ available for context modeling. An example is the scanning pattern used in the level coding that specifies the sequential processing of the 2-dimensional block, where changing the scanning pattern leads to a different coding order and the available conditionals.

Another problem within context modeling is the so-called context dilution, where too many context models lead to insufficient input symbols for each adaptive context model, resulting in problems with suitably estimating probabilities. Conversely, the coding efficiency is poor when using the same adaptive context model for bins of different pmfs to save context memory. When the logic of context model selection is too complicated, the implementation feasibility may be affected, eventually to a level making practical implementations almost impossible. In summary, the design of the context modeling has to balance the aspects mentioned above, among additional constraints not mentioned here, so that a trade-off can be found that provides sufficient coding efficiency while requiring the least context models possible.

Notation for Context Model Selection

For this thesis, the assignment of a bin b_n that represents a piece of unique information, such as $b_n \coloneqq (x > 1)$, to a context model is realized via a **context model offset**. Let $\mathcal{C}_n$ denote an array of context models used

for coding the bin b_n, and $\delta_n \in [0, |\mathcal{C}_n| - 1] \cap \mathbb{N}_0$ is a corresponding context model offset. Note that the same unique information can have different pmfs, and for the above example, the probability $P(x > 1)$ may depend on the location within the transform block, among other aspects. Therefore, bins representing the same unique information can require different context models to achieve an appropriate coding efficiency. Selecting a context model is denoted as $\mathcal{C}_n[\delta_n]$, and the main task of the context model selection in this thesis can be summarized as the determination of the context model offset δ_n.

Optimization Strategies in This Thesis

Different optimization approaches under the umbrella term context quantization based on non-parametric statistics exist in the literature, such as in [59, 60]. Conceptually, the optimal number of context models and assignments for a bin of the same type is sought, such as for $b_0 \coloneqq (x > 0)$. While the so-called context quantization approaches are generic, their focus is purely on context model selection without considering the coding order and the binarization. Furthermore, the final results require a relatively large look-up table to implement the conditional $C(\cdot)$, because the conditional or context function $C(\cdot)$ does typically not inherit any causal dependencies, which could be implemented by closed formulas or simple arithmetic.

The task of context modeling contains more aspects than context model selection, at least in this thesis, and this thesis focuses on heuristic methods, where statistical properties are first examined, and from the analyses, a model is established, from which a context modeling design is derived. Instead of developing a new overall context modeling design from scratch, the existing context modeling is optimized given the requirements and considering further constraints, such as the number of context models or the number of bins coded with context models.

2.3 | Chapter Summary

This chapter briefly described the relevant elements of a hybrid video coding architecture: partitioning, prediction, transform, quantization, and entropy coding. Because this thesis is about level coding, which is the main part of entropy coding in hybrid video coding, the concepts that led to the arithmetic coding were reviewed more detailly. Shannon-Fano-Elias codes that can achieve the efficiency of block codes without storing the codewords were presented, followed by iterative Shannon-Fano-Elias coding that utilizes conditional pmfs instead of joint pmfs. The latter property is crucial for practical applications, because conditional pmfs can be approximated by simpler models while achieving sufficient accuracy. Arithmetic coding is then the implementation of iterative Shannon-Fano-Elias coding with finite-precision arithmetic.

Next, the CABAC framework was presented and discussed, followed by the description of the context modeling task, which in the first approximation is the selection of adaptive context models, where each context model describes a binary pmf. It was clarified that context modeling in this thesis involves more aspects, such as the binarization process or the coding order of transform coefficient levels. Finally, the optimization strategies used to develop the techniques presented in this thesis were described, and why more generic approaches, commonly published under the umbrella term context quantization, were not pursued.

CHAPTER 3

Transform Coefficient Level Coding for Variable Block Sizes

Contents

The continuous advances in hardware technologies allow for faster computation and higher throughput, making cutting-edge video coding techniques feasible. Among the coding tools in *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC) that contribute to the improved coding efficiency relative to the predecessor *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC) [61, 17] are adaptive partitioning techniques. These techniques support flexible block sizes for prediction [57] and transform coding of the prediction errors [62]. But the transform coefficient level coding (or level coding) with *context-based adaptive binary arithmetic coding* (CABAC) [28] in the first version of AVC was desgined for 4×4 transform blocks only. The introduction of additional transform sizes came in the form of 8×8 transform blocks in the second version of AVC. Its support within CABAC was realized by modifying the design for 4×4 transform blocks and using different context models than for 4×4 transform blocks. Such an approach can, in principle, be generalized and applied to additional transform sizes.

This chapter presents an alternative design for the level coding for variable block sizes. During the development, particular attention was paid to implementation feasibility (in terms of complexity for hardware architectures) and coding efficiency. The main idea of the concept is to partition transform blocks larger than 4×4 samples into 4×4 sub-blocks and process each sub-block individually. That enables software and hardware implementations to employ a unified design for the different transform block sizes. Moreover, the developed design decouples the number of context models from the number of supported transform sizes by sharing context models across different block sizes. That property enables, in turn, a straightforward introduction of additional transform sizes and shapes as the development of the VVC standard proved. Compared to the approach used for 8×8 transform blocks in AVC, the alternative design provides higher coding efficiency and has a fixed number of context models, which is independent of the number of supported transform sizes. The concepts developed in this chapter form a foundation for practical implementations, because its basic architecture can be found in both the HEVC [63, 47], and its successor, the VVC [64, 49] standard.

3.1 | Problem Statement

In the first version of AVC, transform coding of prediction residuals is specified using 4×4 blocks only. However, particularly for high-resolution video formats, the coding efficiency can be improved by additionally supporting larger block sizes.

The second version of AVC introduced 8×8 transform blocks in addition to the existing 4×4 transform blocks. Entropy coding of the quantization indices, also referred to as **transform coefficient levels**, in 8×8 transform blocks was realized by extending the design specified for 4×4 transform blocks. Different

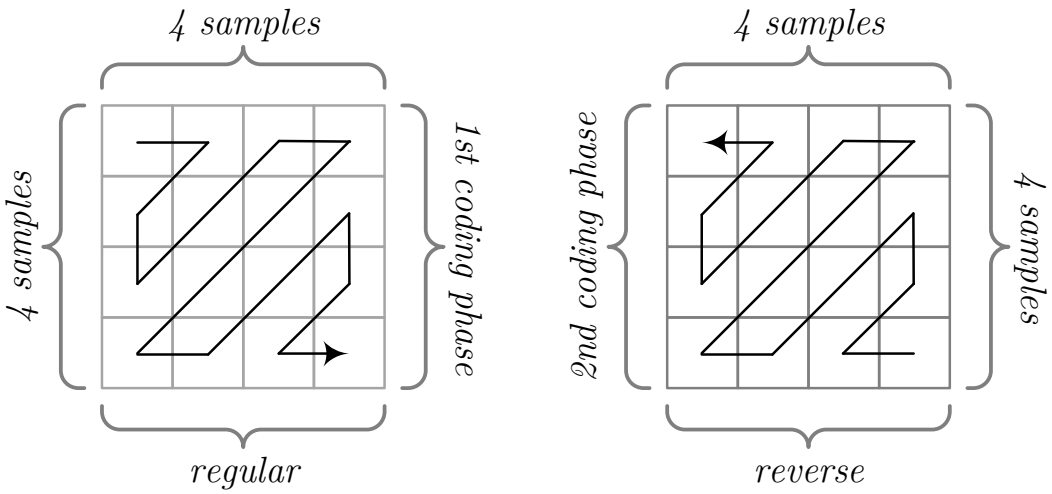

Figure 3.1
Zigzag scanning pattern used in AVC. On the left: The regular zigzag scanning pattern for 4×4 transform blocks used in the first coding phase. On the right: The corresponding reverse zigzag scanning pattern for 4×4 transform blocks used in the second coding phase.

context models than for 4×4 transform blocks are used for the entropy coding of transform coefficient levels belonging to 8×8 transform blocks, because they are characterized by different statistical properties. This approach could be generalized and applied to additional transform block sizes; it will also be used as the basis for the level coding approaches investigated in this chapter.

Nevertheless, this approach becomes infeasible for more flexible partitioning schemes that support a large variety of block sizes, because it requires a separate set of different context models for each supported transform size. That does not only increase the implementation complexity but may also lead to context dilution [65], where too many context models can result in inferior coding efficiency, because the probability estimation cannot adapt to the actual symbol statistics due to an insufficient number of input symbols for a context model.

3.2 | Transform Coefficient Level Coding in AVC

In AVC, the transform coefficient level coding for 4×4 transform blocks consists of two coding phases. In the first coding phase, the locations of non-zero valued transform coefficient levels, also referred to as **significant** levels, are transmitted. The binary mask identifying significant and insignificant transform coefficient levels is commonly called **significance map**. The actual values for the significant locations are then transmitted in the second coding phase.

Two serialization processes are involved: The **scanning pattern** and the **binarization** of transform coefficient levels. A scanning pattern specifies the mapping from a 2-dimensional array of transform coefficient levels within a transform block to a one-dimensional array. The first coding phase uses the zigzag scanning pattern, as illustrated on the left in figure 3.1. After finishing the first coding phase, the second phase starts from the highest frequency position located at the bottom-right corner of the transform block and uses the reverse zigzag scanning pattern, as illustrated on the right in figure 3.1. The use of the zigzag scanning pattern reflects the statistical property that low-frequency positions located at the top-left area of the block have a higher probability for significant transform coefficient levels due to the energy compaction of the transform.

The second serialization process decomposes the integer-valued transform coefficient levels, denoted as x, into a sequence of binary-valued symbols $(b_n)_{n\in\mathbb{N}}$, referred to as **bin string**, for coding with the binary arithmetic engine, where each binary-valued input symbol is referred to as **bin**. The absolute value $|x| \in \mathbb{N}_0$ of each transform coefficient level is binarized using a combination of *truncated unary* (TRU) and 0^{th}*-order exponential-golomb* (EG0) codes. More accurately, the binarization of $x_1 = \min(|x|, 15)$ uses the TRU code,

$\lvert x\rvert$	TRU								EG0					$\ell(x)$
0	0													*1*
1	1	0												*2*
2	1	1	0											*3*
3	1	1	1	0										*4*
4	1	1	1	1	0									*5*
...	...	...	...	...	...									...
12	1	1	1	1	...	0								*13*
13	1	1	1	1	...	1	0							*14*
14	1	1	1	1	...	1	1	0						*15*
15	1	1	1	1	...	1	1	1	0					*16*
16	1	1	1	1	...	1	1	1	1	0	0			*18*
17	1	1	1	1	...	1	1	1	1	0	1			*18*
18	1	1	1	1	...	1	1	1	1	1	0	0	0	*20*
19	1	1	1	1	...	1	1	1	1	1	0	0	1	*20*
20	1	1	1	1	...	1	1	1	1	1	0	1	0	*20*

Table 3.1
Bin strings for absolute transform coefficient levels $|x|$ up to 20, inclusively, when using the binarization process specified in AVC for transform coefficient levels. Note that values greater than 14 include an EG0 suffix, whereas the suffix does not exist for values less than 15.

whereas the remaining absolute value ($x_2 = |x| - 15$), which is only transmitted if $x_1 = 15$, is binarized by the EG0 code. All bins belonging to the TRU code are coded with adaptive context models in the CABAC framework, they are referred to as **context-coded** bins. All bins belonging to the EG0 code are coded in the bypass operation mode, and these bins are referred to as **bypass-coded** bins. When the transform coefficient level is significant ($x \neq 0$), an additional sign bin is coded in the bypass mode. The implicit assumption for all bypass-coded bins is that the source probabilities for both possible values are the same. Table 3.1 summarizes the bin strings for absolute transform coefficient levels up to 20, inclusively, in AVC. Note that the length $\ell(x)$ in table 3.1 denotes the bin string length rather than the final codeword length. Because the described binarization scheme depends on the input symbol x only and not on any further input parameters, it is referred to as static binarization. The decoder can reconstruct the final absolute transform coefficient level given only the bin string.

Before coding the actual transform coefficient levels, a b_{cbf} flag (*coded block flag*) is signaled in the bitstream for each transform block. It specifies whether the corresponding transform block contains at least one significant transform coefficient level. The b_{cbf} flag efficiently represents insignificant transform blocks that often occur at lower bit-rate operation points. Consequently, the coding process described in the following paragraphs considers only the case $b_{cbf} = 1$.

3.2.1 | Coding Phases in AVC

The transmission of the transform coefficient levels comprises two coding phases: The significance map signaling followed by the values for the significant scanning positions. For the sake of comprehensibility, the following description is from the viewpoint of the encoder.

1^{st} Coding Phase

In the first coding phase that uses the regular scanning pattern (illustrated on the left in figure 3.1), a b_{sig} flag is transmitted for each scanning position i. The b_{sig} flag specifies whether the scanning position i has a significant level ($x \neq 0$) or not and corresponds to the first bin of the TRU code ($b_{|x|>0}$). If the scanning position i is significant ($b_{sig}[i] = 1$), a b_{last} flag is additionally transmitted directly after the b_{sig} flag. This flag specifies whether the current scanning position i is the last significant scanning position ($b_{last}[i] = 1$) in the transform block. Interleaving the two flags allows for an efficient representation of insignificant areas at high-frequency locations.

Algorithm 3.1 Pseudo-code of the level coding in AVC for 4×4 transform blocks consisting of two coding phases realized by two while-loops.

```
1: i ← 0, i_last ← −1
2: while i_last < 0 do
3:     i_last ← significanceAndLast (i)
4:     i ← i + 1
5: end while
6: i ← i_last, c_1 [i] ← 1, c_2 [i] ← 0
7: while i ≥ 0 do
8:     absoluteValueAndSign (i, c_1, c_2)
9:     i ← i − 1
10: end while
```

2^{nd} Coding Phase

After establishing the significance map, the absolute values and signs for the significant scanning positions are transmitted. Note that the first bin b_{sig} of the binarization for the absolute values is already coded in the first coding phase. Only the remaining bins representing the absolute values minus one are transmitted in the second coding phase. The second coding phase uses the reverse scanning pattern starting from the last (significant) scanning position (illustrated on the right in figure 3.1), because this reverse scanning pattern allows for a more efficient context modeling as the probabilities become more predictable towards the low-frequency positions. Each bin belonging to the TRU code (up to 14) is coded with a context model, whereas the bins belonging to the EG0 code (if present) are coded in the bypass mode of the arithmetic coding engine. Finally, the sign information (b_{sign} flag) is coded in the bypass mode for the significant levels ($x \neq 0$), because of its uniform distribution.

An overview of the design is given in algorithm 3.1, where two while-loops realize the two coding phases. Note that c_1 and c_2 in algorithm 3.1 are arrays storing **tracking variables**. They are often used to implement the context modeling rules described in the next subsection in practical applications.

3.2.2 | Context Modeling

Let $\mathcal{C}_n$ denote an array of context models used for coding the bin n, where n represents actual flags, such as b_{sig} or b_{last}. Each context model is then addressed by a **context model offset** $\delta_n(i)$ at the scanning postion i as $\mathcal{C}_n[\delta_n(i)]$.

1^{st} Coding Phase

For both b_{sig} and b_{last}, each scanning position i employs a dedicated context model. Given the scanning position i, the corresponding context model is then $\mathcal{C}_{sig}[i]$ and $\mathcal{C}_{last}[i]$ (where $\delta_{sig}(i) = \delta_{last}(i) = i$). A summary for the assignment is given in algorithm 3.2 (sub procedure of coding the significance b_{sig} and last b_{last} flags). Whenever $b_{last}[i] = 1$ is signaled, the last significant scanning position is identified as being equal to i, and the first coding phase is terminated.

Algorithm 3.2 *significanceAndLast*: Coding of b_{sig} and b_{last} including the corresponding context modeling in AVC.

```
Require: scanning position i
1: δ_sig (i) ← i, δ_last (i) ← i
2: transmit b_sig [i] using C_sig [δ_sig (i)]
3: if b_sig [i] = 1 then
4:     transmit b_last [i] using C_last [δ_last (i)]
5:     if b_last [i] = 1 then
6:         return i
7:     else
8:         return −1
9:     end if
10: end if
```

2^{nd} Coding Phase

Up to 14 context-coded bins may be present for each $x \neq 0$, where the first bin $b_{|x|>1}$ uses a dedicated context model from the set $\mathcal{C}_{x>1}$. For the remaining 13 context-coded bins, the same context model from the set $\mathcal{C}_{x>2}$ is used, where the derivation of $\delta_{x>1}(i)$ and $\delta_{x>2}(i)$ differs. For the first bin $b_{|x|>1}$, the context model index is incremented by one ($\delta_{x>1}(i) = \delta_{x>1}(i+1)+1$) after the coding of $b_{|x|>1}[i+1] = 0$, but switches to a fixed context model offset ($\delta_{x>1}(i) = 0$) after the coding of a bin $b_{|x|>1}[i+1] = 1$. In the latter case, the same context model is used for all remaining scanning positions of the transform block. Let $B_{x>1}(i) = \sum_{k=i+1}^{i_{last}} b_{|x|>1}[k]$ denote the sum of the already coded $b_{|x|>1}$ flags and $B_{sig}(i) = \sum_{k=i+1}^{i_{last}} b_{sig}[k]$ the corresponding sum of the b_{sig} flags. Then, the context index derivation for the $b_{|x|>1}$ flag in AVC is given by the following formula:

$$\delta_{x>1}(i) = \begin{cases} 0, & \text{if } B_{x>1}(i) > 0, \\ \min(4, B_{sig}(i)+1), & \text{otherwise.} \end{cases} \tag{3.1}$$

The rationale for such context modeling is that the probability of $b_{|x|>1}$ increases towards the low-frequency scanning positions, and its probability of being equal to one becomes fairly predictable. With each occurence of $b_{|x|>1} = 0$, the probability for $b_{|x|>1}(i) = 1$ increases, while the occurence of $b_{|x|>1}(i+1) = 1$ indicates that the probability for $b_{|x|>1}(i)$ becomes rather independent of the values of the bins following $b_{|x|>1}(i+1) = 1$. The context model offset $\delta_{x>2}(i)$ for the remaining 13 context-coded bins (if present) is incremented by one after the coding of a bin $b_{|x|>1}(i+1) = 1$. Consequently, $\delta_{x>2}(i)$ relies on $B_{x>1}(i)$ and the following formula summarizes the context modeling:

$$\delta_{x>2}(i) = \min(4, B_{x>1}(i)). \tag{3.2}$$

Algorithm 3.3 summarizes the second coding phase and the described context modeling for the context-coded bins of x. The evaluation of the already coded $b_{|x|>1}$ flags is realized by the tracking variables c_1 and c_2 in practical implementations.

Algorithm 3.3 *absoluteValueAndSign*: Coding of absolute levels and signs together with the context modeling using the two tracking variables c_1 and c_2 in AVC.

Require: scanning position i, arrays storing tracking variable values c_1, and c_2

if $b_{sig}[i] = 1$ **then**
  $\delta_{x>1}(i) \leftarrow \min(c_1[i], 4)$
  transmit $b_{|x|>1}[i]$ using $\mathcal{C}_{x>1}[\delta_{x>1}(i)]$
  if $b_{|x|>1}[i] = 1$ **then**
    $\delta_{x>2}(i) \leftarrow \min(c_2[i], 4)$, $j \leftarrow 1$
    do
      $j \leftarrow j+1$, $b_j = b_{|x|>j}[i]$
      transmit b_j using $\mathcal{C}_{x>2}[\delta_{x>2}(i)]$
    while $b_j \wedge j < 14$
    if $b_j \wedge j = 14$ **then**
      transmit $|x_i| - 15$ in bypass mode using $EG0$
    end if
    $c_2[i-1] \leftarrow c_2[i]+1$, $c_1[i-1] \leftarrow 0$
  else if $c_1[i] \neq 0$ **then**
    $c_1[i-1] \leftarrow c_1[i]+1$
  end if
  transmit $b_{sign}[i]$ in bypass mode
end if

Number of Context Models

The described level coding requires four different arrays of context models ($\mathcal{C}_{sig}$, $\mathcal{C}_{last}$, $\mathcal{C}_{x>1}$, and $\mathcal{C}_{x>2}$). Let $|\mathcal{C}_n|$ denote the number of items in an array $\mathcal{C}_n$, then the number of total context models is equal to 42 ($|\mathcal{C}_{sig}| = 16$, $|\mathcal{C}_{last}| = 16$, $|\mathcal{C}_{x>1}| = 5$, and $|\mathcal{C}_{x>2}| = 5$).

3.2.3 | 8 × 8 Transform Blocks and Generalization

The level coding for 8×8 transform blocks in AVC is a simple extension of the design for 4×4 transform blocks. For the first coding phase, however, if one attempts to keep the number of context models for b_{sig} and b_{last} the same as for 4×4 transform blocks, at least some context models have to be used for multiple scanning positions. The solution of the extended design in AVC is assigning the same context model to four successive scanning positions and using different context models than for 4×4 transform blocks. The only change to the context modeling of the second coding phase is the usage of different context models for 8×8 transform blocks.

Using different context models when dealing with 8×8 transform blocks can be expressed by adding a corresponding offset to the context indices (δ_{sig}, δ_{last}, $\delta_{x>1}$, and $\delta_{x>2}$), and doubling the size of the context model arrays ($\mathcal{C}_{sig}$, $\mathcal{C}_{last}$, $\mathcal{C}_{x>1}$ and $\mathcal{C}_{x>2}$). When following the principles used in AVC for 8×8 transform blocks, the concept can be generalized to additional block sizes by:

- Increasing the number of successive scanning positions that use the same context model so that the number of context models necessary for b_{sig} and b_{last} is equal to 16;
- Adding 16 context models to both $\mathcal{C}_{sig}$ and $\mathcal{C}_{last}$, and five context models to both $\mathcal{C}_{x>1}$ and $\mathcal{C}_{x>2}$ for each additional transform block size;
- Using additional offsets for δ_{sig}, δ_{last}, $\delta_{x>1}$, and $\delta_{x>2}$ depending on the transform block size N.

In the following, it is assumed that the partitioning only supports square transform blocks with a power-of-two width and height greater than or equal to 4, as it is the case in both AVC and HEVC. Nonetheless, the concept can be extended to more general rectangular blocks in a straightforward way. Let $n = \log_2 \mathrm{N} - 2$ with N being the width and height of a square transform block, then the generalized version for the context index offsets is given by:

$$\begin{aligned} \delta_{sig}(i) &\leftarrow 16n + (i \gg (n \ll 1)), \\ \delta_{last}(i) &\leftarrow 16n + (i \gg (n \ll 1)), \\ \delta_{x>1}(i) &\leftarrow 5n + \min(c_1[i], 4), \\ \delta_{x>2}(i) &\leftarrow 5n + \min(c_2[i], 4). \end{aligned} \tag{3.3}$$

The first two assignments in the above formula replace the first line in algorithm 3.2, and the last two assignments replace the corresponding derivation in algorithm 3.3 (line two and line five). Note that using distinctive context models for each transform size is a disadvantage, because it introduces a linear relationship between the number of additional transform sizes and the number of context models. A reason for using different context models depending on the transform size is that the statistical properties of transform coefficient levels depend on the transform block size.

3.2.4 | Reference Implementation and Experimental Setup

The generalized AVC design is the starting point for the development in this chapter. It is referred to as *implementation 3-0* (IMP3-0) and is the initial anchor for coding efficiency comparisons. For all coding experiments in this chapter, the HEVC reference software, version 16.22 (HM-16.22), was used as the basis. The coding experiments are evaluated by measuring the *Bjøntegaard delta bit-rate* (BD-rate) [66] for the luma component between two codec versions, and the experiments mainly follow the HEVC *common test conditions* (CTC) [67]. Specifically, the used test set, coding tools configuration, and quantization parameters (QPs) controlling the operation points are described in detail in [67]. Within the used experimental environment, the encoder can choose among four different (square-shaped) transform block sizes: 4×4, 8×8, 16×16, and 32×32. Different than specified in the CTC, the performed encoder simulations did not use *sign data hiding* (SDH), *rate-distortion optimized quantization* (RDOQ), and *transform skip mode* (TSM) to simplify the conducted experiments. Furthermore, all adaptive context models used for level coding were initialized as *equi-probable* (EP) to avoid interference from initial probabilities. The total number of context models employed for the generalized AVC design is equal to 168 (4 transform sizes multiplied by 42).

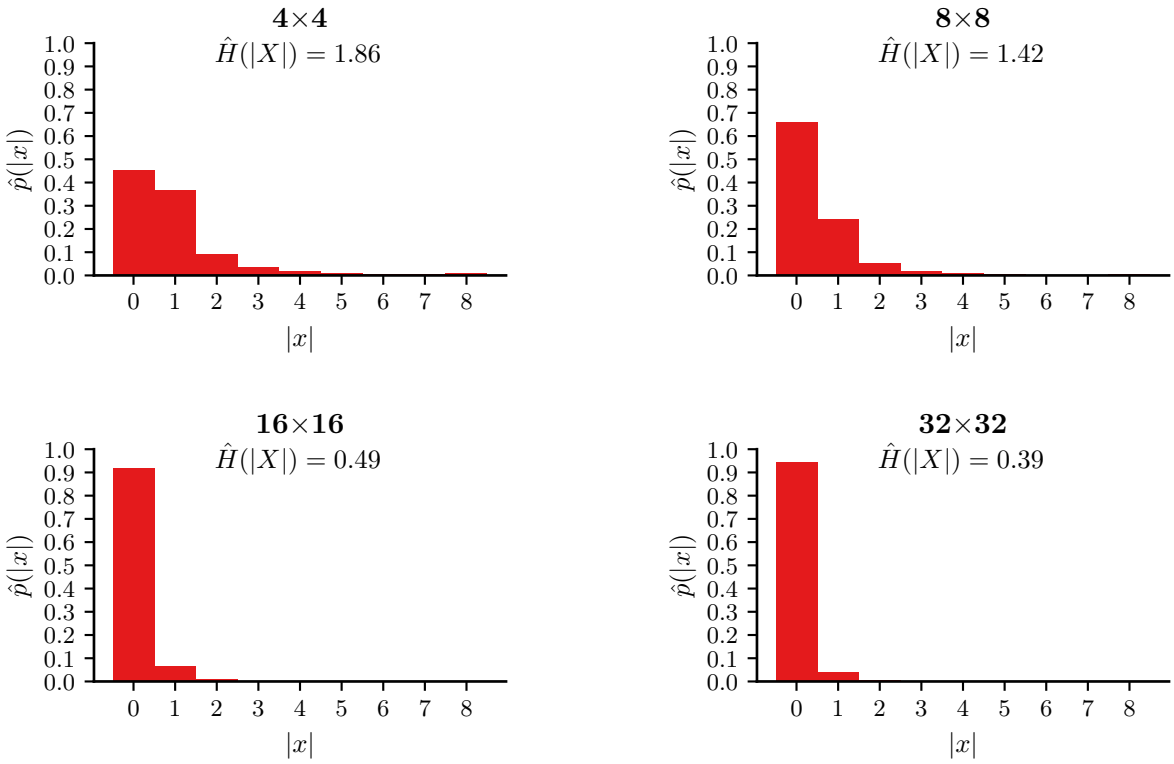

Figure 3.2
Histograms of coded absolute transform coefficient levels for the four used transform block sizes, acquired by coding the first frame of the *BQTerrace* sequence using HM-16.22. Absolute transform coefficient levels greater than eight were assigned to the last bin of the histogram, and the QP was set to 27.

3.2.5 | Properties of Variable Transform Sizes

To demonstrate that different transform sizes lead to different probability distributions, figure 3.2 illustrates the histograms of absolute transform coefficient levels for the used transform sizes. The data used to generate the histograms were acquired by decoding existing bitstreams, and only scanning positions with coded b_{sig} flags were considered. That avoids a bias of the histograms towards zero-valued transform coefficient levels, which should be avoided, because the design already considers the effect of insignificant areas at the high-frequency locations by coding b_{last} flags. Furthermore, $\hat{H}(X)$ denotes the empirical entropy for the corresponding histograms. Figure 3.2 indicates that the relative frequency (or empirical probability) $\hat{p}(\cdot)$ for insignificant scanning positions is higher for larger transform sizes, resulting in a significantly lower $\hat{H}(X)$ value. This observation indicates that larger transforms tend to be sparser, i.e., there are more insignificant scanning positions along the scanning path. Note that the histograms do not reflect the probabilities of real coding conditions (e.g., for the b_{sig} flag), but they show that the probabilities are not independent of the transform size.

Additional encoding experiments were performed where all transform block sizes employ the same context models, i.e., $n = 0$ for the notation in equation (3.3). That implementation showed an inferior coding efficiency than the anchor implementation, where each transform block size employs distinctive context models. The measured BD-rate, averaged over the entire test set, are 1.32% in the *All-Intra* and 0.68% in the *Random-Access* configurations.

3.3 | Alternative Design with 4 × 4 Sub-Blocks

The two coding phases of the generalized AVC design are universally applicable, implying that supporting additional transform sizes requires only the definition of the corresponding zigzag scanning patterns. However, the amount of additional context models directly depends on the number of additional transform

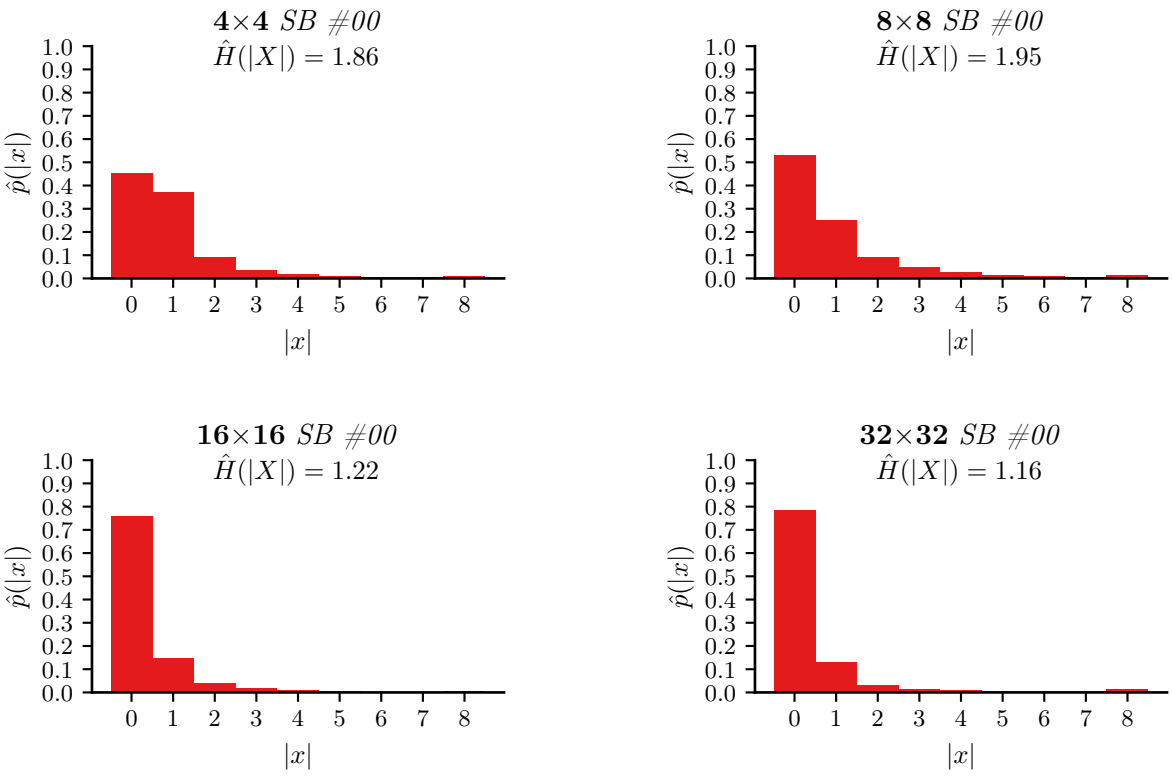

Figure 3.3
Histograms of absolute transform coefficient levels for the sub-block containing the DC frequency position of different transform sizes, acquired by coding the first frame of the *BQTerrace* sequence using HM-16.22. Absolute transform coefficient levels greater than eight were grouped into the last bin of the histogram, and the QP was set to 27.

sizes. That dependency should be broken to achieve a universal context modeling variant, which leads to a completely straightforward design, i.e., no further modifications are necessary when introducing additional transform sizes except for the definition of the scanning patterns.

A concept that would resolve the dependency mentioned above is partitioning the transform blocks into different regions and using the same context models for regions with similar statistical properties. The number of context models is then fixed and independent of the number of supported transform sizes. Such an alternative approach could even improve coding efficiency when those regions exist among the different transform sizes. A possible realization is the partitioning of transform blocks larger than 4×4 samples into 4×4 sub-blocks. Within each 4×4 sub-block, the same coding order and context modeling as for 4×4 transform blocks could be used to keep the existing design for 4×4 transform blocks.

3.3.1 | Properties of 4 × 4 Sub-Blocks

The motivation behind the alternative design with 4×4 sub-blocks is to share context models among different transform sizes. However, it can only work when 4×4 sub-blocks with similar statistical properties among different transform sizes exist. It is not evident that they exist, but it should be supposed that 4×4 sub-blocks with similar statistical properties do not always emerge at the same locations and that their appearance depends on the input signal, the transform size, the transform type, the quantization step size, and other parameters.

Figure 3.3 illustrates the histograms of 4×4 sub-blocks covering the DC frequency position (denoted as sub-block #00 with absolute offsets $(x = 0, y = 0)$ relative to the top-left origin of the transform block) for the used transform sizes. An observation is that the sparsities for the larger transform sizes (8×8, 16×16, and 32×32) are less pronounced than in the histograms for the entire transform blocks in figure 3.2. Another

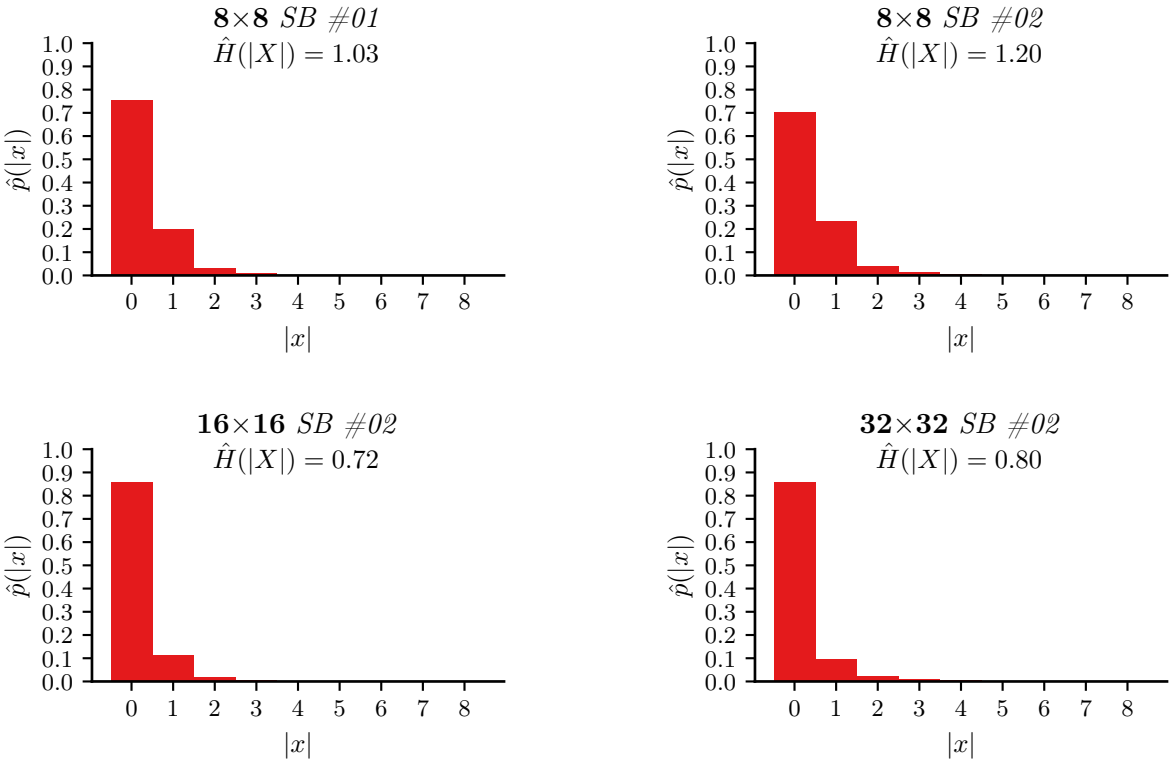

Figure 3.4
Histograms of absolute transform coefficient levels for various 4×4 sub-blocks of different transform sizes, acquired by coding the first frame of the *BQTerrace* sequence using HM-16.22.

observation is that the histogram for 8×8 transform blocks becomes more similar to the histogram of 4×4 transform blocks. The above observations show that the empirical probability distribution can be different than that of the entire transform block when considering 4×4 sub-blocks. Figure 3.4 illustrates the histograms of sub-blocks with the identifier #01 ($x = 4, y = 0$) and #02 ($x = 0, y = 4$) for the different transform sizes. An observation is that the two histograms for 8×8 transform blocks are very similar, and the same observation can be made for the histograms of the sub-blocks for 16×16 and 32×32 transform blocks. Both observations suggest that 4×4 sub-blocks with very similar statistical properties (or empirical probability distributions) can emerge within larger transform sizes. Nevertheless, the histogram for 4×4 transform blocks shows a different empirical distribution than the histograms for the 4×4 sub-blocks of larger transform sizes. With that observation, 4×4 transform blocks should use different context models than for larger transform block sizes.

The above analyses of the histograms indicate that 4×4 sub-blocks with similar statistical properties among different transform sizes may exist. Given these observations, the challenge is to detect 4×4 sub-blocks with similar statistical properties so that the same context models can be used. Note that the histograms do not represent the probabilities for real coding conditions, but the coding efficiency results presented during the discussion in section 3.2.5 demonstrate that they often give first useful indications.

3.3.2 | Coding Phases with 4 × 4 Sub-Blocks

In the alternative approach, 4×4 sub-blocks can be realized by modifying the existing scanning patterns, as illustrated in figure 3.5 for a 16×16 transform block. While the only change to the first coding phase is the scanning pattern, there are two possibilities to implement the coding order for the second coding phase (called **scanning options** in the remainder of this subsection).

For the first scanning option, the 4×4 sub-blocks are processed in forward scanning order, and for each sub-block, the second coding phase (reverse scanning order) directly follows the first coding phase (forward

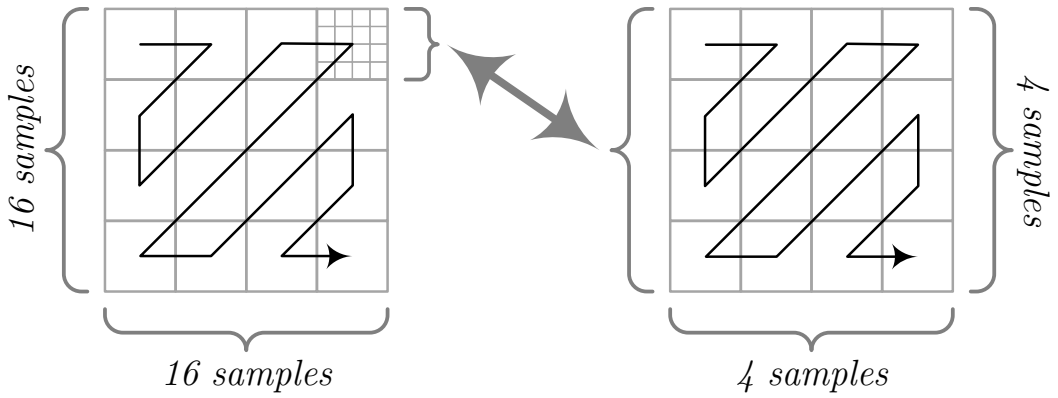

Figure 3.5
Decomposition of a 16×16 transform block into 4×4 sub-blocks and the corresponding scanning for the first coding phase. Each sub-block is processed using the zigzag scanning pattern (on the left), and within each 4×4 sub-block, the same scanning pattern is used (on the right).

scanning order). Hence, the two coding phases are interleaved on a granularity of sub-blocks, as it is summarized in algorithm 3.4. Compared to the AVC approach for 4×4 transform blocks in algorithm 3.1, the scanning pattern is modified, and the coding order switches between the first and the second coding phases.

For the second scanning option, the second coding phase starts when the first coding phase is completed for the whole transform block as summarized in algorithm 3.5. The second coding phase starts at the last significant scanning position and uses the reverse scanning pattern until the signaling for the transform block is completed. Compared to algorithm 3.1, only the scanning pattern is modified.

Both variants have advantages and disadvantages concerning memory requirements and context modeling. The first scanning option requires less local memory than the second scanning option, because data of each 4×4 sub-block are processed only once. More precisely, a local buffer for 16 transform coefficient levels is sufficient, whereas the second scanning option requires either extra storage or memory access to hold information for incompletely reconstructed 4×4 sub-blocks. However, the first scanning option may result in a sub-optimal context modeling for the second coding phase, because the 4×4 sub-blocks are processed from low- to high-frequency positions (whereas the scanning inside a sub-block is from high- to low-frequency locations).

Algorithm 3.4 Pseudo-code of the level coding with 4×4 sub-blocks using the first scanning option for the second coding phase.

```
1: i ← 0, i_last ← −1
2: while i_last < 0 do
3:     m ← 0, i_0 ← i
4:     while m < 16 ∧ i_last < 0 do
5:         i_last ← significanceAndLast (i)
6:         c_1 [i] ← 1, c_2 [i] ← 0
7:         m ← m + 1, i ← i + 1
8:     end while
9:     m ← m − 1
10:    while m ≥ 0 do
11:        absoluteValueAndSign (i_0 + m, c_1, c_2)
12:        m ← m − 1
13:    end while
14: end while
```

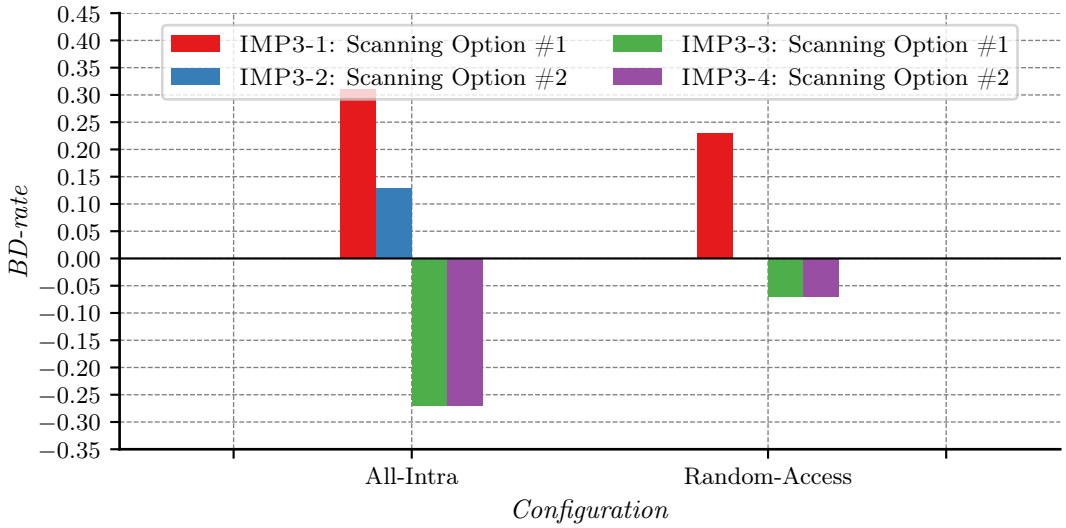

Figure 3.6
Coding efficiency of the two scanning options with and without tracking variables reinitialization (TVR). AI denotes the *All-Intra* configuration and RA the *Random-Access* configuration. IMP3-1 is the implementation for scanning option #1, where the first and second coding phases are interleaved on a sub-block granularity. IMP3-2 is the implementation for scanning option #2, where the second coding phase starts after the first coding phase is finished for the whole transform block. IMP3-3 corresponds to IMP3-1 with TVR enabled and IMP3-4 corresponds to IMP3-2 with with TVR enabled.
Anchor for BD-rate computations: IMP3-0

Coding Efficiency of Scanning Options

Coding experiments were conducted to investigate the coding efficiency for the two scanning options with the implementation of the first scanning option is called IMP3-1, and the second scanning option is called IMP3-2. Experimental results for the conducted coding experiments are summarized by the four left bars in figure 3.6. The scanning order is the only modification compared to the anchor, which is IMP3-0 and represents the generalized AVC design. On the one hand, the first scanning option shows inferior coding efficiency relative to the anchor (the generalized AVC design) in both configurations (0.31% and 0.23%). On the other hand, the second scanning option maintains the same coding efficiency in the *Random-Access* configuration and experiences a smaller loss in coding efficiency than the first scanning option in the *All-Intra* configuration. The inferior coding efficiency of the first scanning option is probably due to the sub-optimal context modeling for the second coding phase.

Algorithm 3.5 Pseudo-code of the level coding with 4×4 sub-blocks using the second scanning option for the second coding phase.

$i \leftarrow 0$, $i_{last} \leftarrow -1$
while $i_{last} < 0$ **do**
    $i_{last} \leftarrow significanceAndLast\,(i)$
    $m \leftarrow m + 1$, $i \leftarrow i + 1$
end while
$m \leftarrow m - 1$, $i \leftarrow i - 1$
$c_1\,[i] \leftarrow 1$, $c_2\,[i] \leftarrow 0$
while $i \geq 0$ **do**
    while $m \geq 0$ **do**
        $absoluteValueAndSign\,(i, c_1, c_2)$
        $m \leftarrow m - 1$, $i \leftarrow i - 1$
    end while
    $m \leftarrow 15$
end while

Tracking Variables Reinitialization

A potential problem for the first scanning option occurs when $\delta_{x>1} = 0$ and the probabilities for $b_{|x|>1} = 1$ of the upcoming scanning positions are expected to be high. However, the probabilities for $b_{|x|>1} = 1$ do not increase, because the scanning for the sub-blocks is from low-frequency to high-frequency positions.

A possible solution for this context selection problem in the second coding phase of the first scanning option is to reinitialize the tracking variables c_1 and c_2 to their initial values at the start of each sub-block. Recall that c_1 and c_2 are responsible to count the number of coded $b_{|x|>1}$ bins and their values and are used for the context modeling of the second coding phase as summarized in algorithm 3.3. For the first scanning option, line six in algorithm 3.4 then becomes obsolete and algorithm 3.4 is extended by the statements $c_1[i_0 + m] \leftarrow 1$ and $c_2[i_0 + m] \leftarrow 0$ in between line nine and ten.

The sub-block-based reinitialization of the tracking variables can also be used for the second coding option. In that setting, line seven in algorithm 3.5 becomes obsolete and algorithm 3.5 is extended by the statements $c_1[i] \leftarrow 1$ and $c_2[i] \leftarrow 0$ in between line eight and nine.

In the following investigation, IMP3-3 represents the implementation of scanning option #1 with tracking variables reinitialization and IMP3-4 the implementation of scanning option #2 with tracking variables reinitialization. The experimental results are summarized by figure 3.6 for both scanning options with tracking variables reinitialization. IMP3-0 implementing the generalized AVC design served as the anchor, and the only modification relative to the previous experiments is setting $c_1[i] = 1$ and $c_2[i] = 0$ at the beginning of each 4×4 sub-block. Both scanning options provide coding efficiency improvements relative to their counterparts without tracking variables reinitialization, and both scanning options provide -0.27% in the *All-Intra* and -0.07% in the *Random-Access* configurations relative to the generalized AVC design. Interestingly, the second scanning option also benefits from the tracking variable reinitialization. When using the second scanning option without reinitialization, there are also coding conditions where the assumption that the probability for $b_{|x|>1} = 1$ is high is not fulfilled. An example of such a case is when $\delta_{x>1}$ becomes zero for a certain sub-block, but there are sub-blocks at high horizontal or vertical frequency positions that follow the sub-block in coding order in the second pass. Due to the energy compaction property of the transform, the actual probabilities for $b_{|x|>1} = 1$ inside these sub-blocks are not so large as indicated by $\delta_{x>1} = 0$, which eventually results in a sub-optimal context modeling for the second coding phase. Given that both scanning options provide the same coding efficiency improvements with the tracking variable reinitialization, the first scanning option is the preferred variant as it allows for more efficient implementations. Therefore, the scanning option #1 with tracking variable reinitialization is used as the base version for the following investigations.

3.3.3 | Context Modeling for 4 × 4 Sub-Blocks

Modifying the scanning pattern and reinitializing the tracking variables sets the foundation for the alternative design with 4×4 sub-blocks. However, the goal of a universally applicable context modeling has not been achieved yet, because the number of required context models still depends on the number of supported transform sizes.

Context Quantization for δ_{sig} and δ_{last}

In the generalized AVC design, the context quantizer assigns the same context model to $N^2/16$ successive scanning positions when an N×N transform block is coded. This approach becomes infeasible for larger transform block sizes, because a close spatial relationship between the locations is not always given, i.e., the first and the last scanning positions sharing the same context model can be located in completely different regions of the transform block. A solution to that problem is assigning the same context model to locations with close spatial relationships, where the number of frequency positions sharing the same context model remains the same as before. Particularly, a subsampling of frequency positions is performed and the modification results in four neighboring frequency positions sharing the same context model when the transform block is 8×8. For 16×16 transform blocks, all frequency positions within a sub-block share the same context model, and four neighboring 4×4 sub-blocks share the same context model for 32×32 transform

Class	Luma	C_B	C_R
	All-Intra		
A	-0.25 (-0.06)%	-0.47 (0.06)%	-0.37 (0.20)%
B	-0.39 (-0.18)%	-0.37 (-0.22)%	-0.42 (-0.21)%
C	-0.46 (-0.17)%	-0.41 (-0.20)%	-0.44 (-0.17)%
D	-0.33 (-0.11)%	-0.32 (-0.11)%	-0.34 (-0.07)%
E	-0.98 (-0.47)%	-0.87 (-0.46)%	-0.88 (-0.48)%
Overall	**-0.45 (-0.18)%**	**-0.46 (-0.18)%**	**-0.46 (-0.13)%**
	Random-Access		
A	-0.15 (-0.32)%	-0.89 (-0.28)%	-0.56 (-0.18)%
B	-0.23 (-0.20)%	-0.35 (-0.12)%	-0.38 (-0.34)%
C	-0.34 (-0.14)%	-0.25 (-0.17)%	-0.24 (-0.11)%
D	-0.20 (-0.11)%	-0.47 (0.01)%	-0.07 (-0.10)%
E	-0.66 (-0.40)%	-0.71 (-0.26)%	-0.64 (-0.37)%
Overall	**-0.29 (-0.22)%**	**-0.51 (-0.16)%**	**-0.36 (-0.22)%**

Table 3.2
Coding efficiency of the modified context quantizer for δ_{sig} and δ_{last} of the first coding phase in the 4×4 sub-blocks design (IMP3-5). BD-rates not in brackets were computed using IMP3-0 as the anchor, i.e., the generalized AVC design. BD-rates in brackets were computed using IMP3-3, i.e., the 4×4 sub-block processing with tracking variables reinitialization.
Anchor for BD-rate computations: IMP3-0 (IMP3-3)

blocks. This approach is summarized as follows, where N denotes the transform block size and $(x(i), y(i))$ specifies the location of the i-th scanning position inside the transform block:

$$\begin{aligned} s_x &= x(i) \gg \log_2(\mathrm{N} \gg 2), \\ s_y &= y(i) \gg \log_2(\mathrm{N} \gg 2), \\ \delta_{sig}[i] = \delta_{last}[i] &= s_x + (s_y \ll 2) + 16n. \end{aligned} \tag{3.4}$$

Similar as in equation (3.3), the context models for different transform sizes are distinguished by the index $n = \log_2 \mathrm{N} - 2$.

IMP3-5 represents the implementation that is based on IMP3-3 (scanning option #1 with tracking variables reinitialization) and implements the context quantization denoted in equation (3.4) for δ_{sig} and δ_{last}. Table 3.2 summarizes the coding efficiency provided by IMP3-5. The values in brackets denote BD-rates calculated by using IMP3-3 the anchor, i.e., the 4×4 sub-block processing with the first scanning option and tracking variables reinitialization. For the BD-rates not in brackets, the anchor is IMP3-0, i.e., the generalized AVC design; these values represent the overall coding efficiency improvements. The coding experiment confirms that the AVC context quantizer is sub-optimal, and modifying the context quantizer that assigns the same context model to neighboring locations provides higher coding efficiency.

Adaptive Context Model Sets for 4 × 4 Sub-Blocks

One of the main goals was to design an approach for level coding that can be straightforwardly extended to additional block sizes. Based on the basic concept with 4×4 sub-blocks, this can be achieved by switching between different context model sets depending on the statistical property of a sub-block. The term context model set denotes a subset of context models used for certain sub-blocks. For example, each value of n in equation (3.3) and equation (3.4) indicates a unique context model set. One possibility to introduce context model sets for different classes of sub-blocks is to estimate the statistical properties of a sub-block depending on the number of absolute levels greater than one located in the preceding sub-block in coding order. A reason for investigating this quantity for context model switching is that it represents the already used tracking variable c_2 at the start of a sub-block before reinitialization. With i_0 denoting the first scanning position of a sub-block, the quantity selected for context model set switching is given by:

$$c_2[i_0] = \sum_{k=i_0-1}^{i_0-16} b_{|x|>1}[k]. \tag{3.5}$$

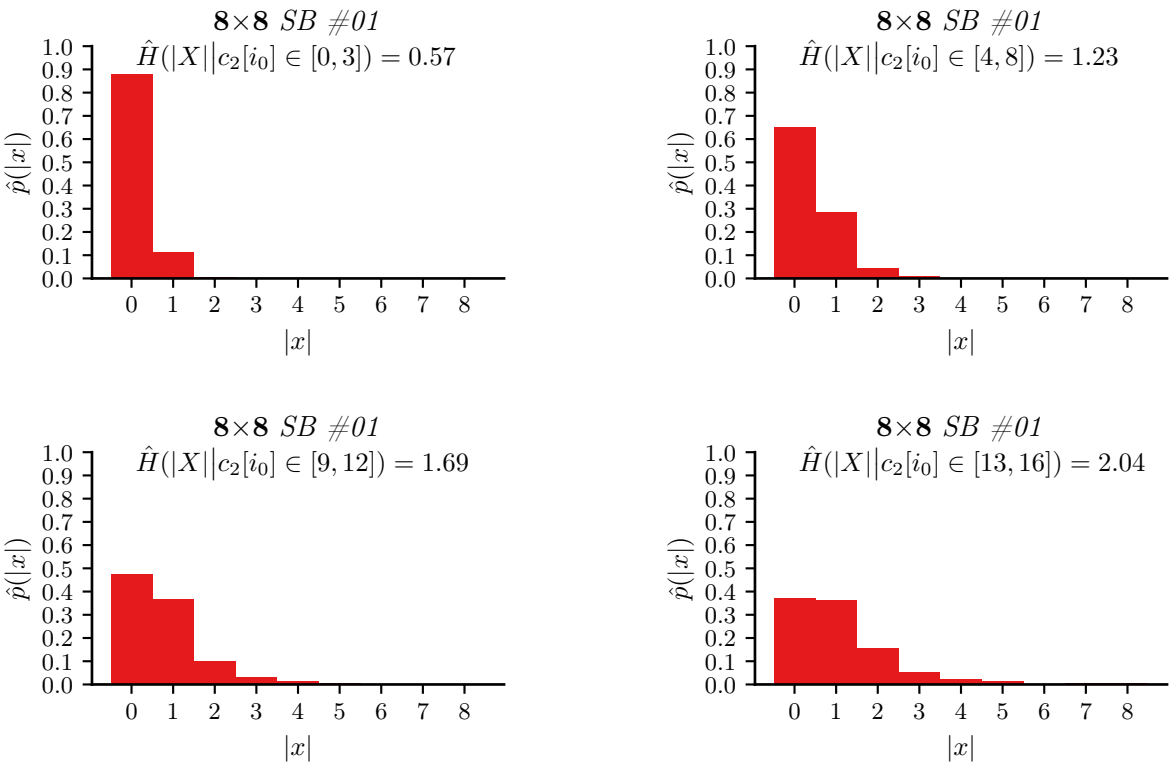

Figure 3.7
Histograms of sub-blocks with the identifier #01 ($x = 4, y = 0$) of 8×8 transform blocks, conditioned on the number of bins $b_{|x|>1} = 1$ located in the preceding sub-block in coding order.

Because the tracking variable c_2 is already used in the second coding phase, the selected procedure was to first investigate the context modeling for the second coding phase. After establishing a suitable algorithm for the context model switching for the second coding phase, similar optimization techniques are used for extending the context model switching to the first coding phase.

Analysis of Conditional Entropies

Before describing the parameters and optimization for the context model switching, a statistical analysis is presented suggesting that using $c_2\,[i_0]$ as an indicator for the statistical properties of a sub-block is indeed feasible. In figure 3.7, conditional histograms are shown for the sub-block with the identifier #01 ($x = 4, y = 0$) of 8×8 transform blocks. Note that the underlying data used in figure 3.7 and in the histogram with the identifier #01 of figure 3.4 are the same. The condition for the four histograms in figure 3.7 varies, where the top-left histogram is conditioned on $c_2\,[i_0] \in [0, 3]$. For the top-right histogram, the condition is $c_2\,[i_0] \in [4, 8]$, and it is $c_2\,[i_0] \in [9, 12]$ and $c_2\,[i_0] \in [13, 16]$ for the bottom-left and the bottom-right histograms, respectively.

Compared to the empirical marginal histogram in figure 3.4, the empirical conditional distributions are different, which is also reflected by the values for the empirical conditional entropies $\hat{H}\,(X|c_2 \in [a, b])$. The empirical conditional entropy for the example is $\hat{H}\,(X|C) = 0.93$, which is 10% less than the empircal marginal entropy of $\hat{H}\,(X) = 1.03$. This result clearly indicates that there exist statistical dependencies between the transform coefficient levels of a sub-block and the variable $c_2\,[i_0]$. Note that the histograms do not reflect the probabilities in real coding conditions (i.e., the binary probabilities for a bin); they are mainly used to derive indications for suitable coding conditions.

Quantization of Context Information

The necessary modifications to enable the selection of different context model sets for each 4×4 sub-block change the definition of n in equation (3.3), while the offset calculations in equation (3.3) remain the same.

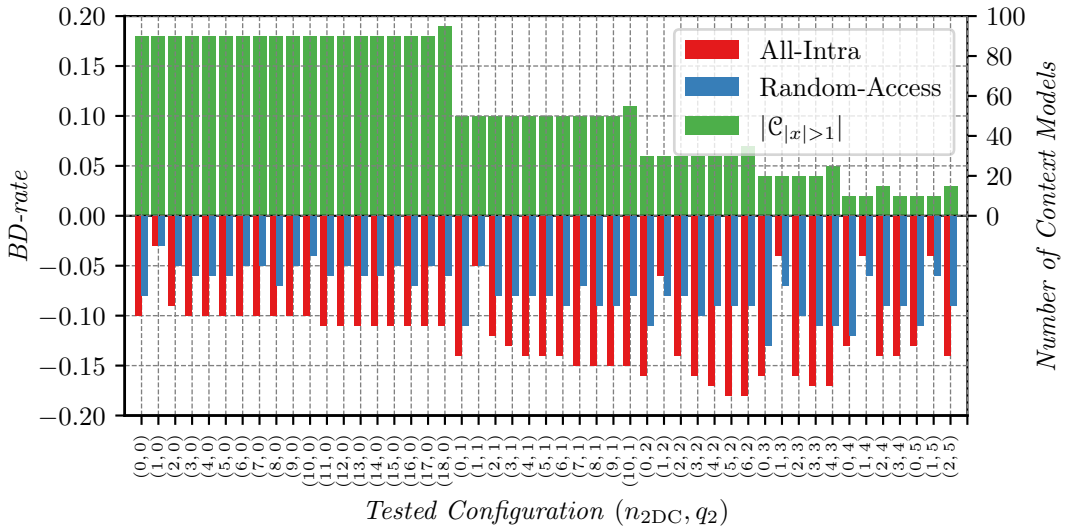

Figure 3.8
Coding efficiency of IMP3-6 with all tested combinations of n_{2DC} (context model offset for the first sub-block) and q_2 (the quantization parameter in equation (3.6)).
Anchor for BD-rate computations: IMP3-5

When operating without context quantization ($n = c_2\,[i_0]$), 17 different values for n are possible, resulting in 90 context models for $\mathcal{C}_{x>1}$ and another 90 context models for $\mathcal{C}_{x>2}$ ($17 \cdot 5 + 5$, because 4×4 transform blocks use dedicated context models). In total, 180 context models are necessary (independent of the number of supported transform sizes), while the generalized AVC design requires only 40 context models when supporting the four different transform sizes of 4×4, 8×8, 16×16, and 32×32.

Coding experiments with a context quantization that only requires binary-shift operations were conducted to reduce the number of required context models. In addition to the bit-shift, which is referred to as quantization parameter q_2, the context model offset n_{2DC} for the first sub-block, which contains the DC frequency position and does not have a preceding sub-block, has to be selected.

With N being the width and height of a transform block (which is only required for detecting 4×4 transform blocks), and q_2 and n_{2DC} being the parameters to be optimized, the sub-block offset is given by:

$$n_2 = \begin{cases} 0, & \text{if } \mathrm{N} = 4, \\ n_{2DC}, & \text{if } \mathrm{N} \neq 4 \wedge i_0 = 0, \\ (c_2\,[i_0] \gg q_2) + 1, & \text{if } \mathrm{N} \neq 4 \wedge i_0 > 0. \end{cases} \tag{3.6}$$

Note that n_2 is used instead of n in the formula above, because it only affects the context model indices of the second coding phase, and to simplify the optimization process, n_2 is used for both context model indices $\delta_{x>1}$ and $\delta_{x>2}$.

Experimental Results for 2^{nd} Coding Phase

A set of coding experiments was conducted using different combinations of the values for n_{2DC} and q_2 in equation (3.6). Meaningful values for q_2 range from zero to five, inclusively, resulting in $m = \{18, 10, 6, 4, 3, 2\}$ distinctive sets of context models for $q_2 \in [0, 5]$, where m is given by:

$$m = (16 \gg q_2) + 2. \tag{3.7}$$

For the parameter n_{2DC}, meaningful values range from zero to m, where $n_{2DC} = 0$ means that the context models of 4×4 transform blocks are used initially, whereas $n_{2DC} = m$ means that a distinctive set of context models is used for the first sub-block. The number of necessary context models for $\mathcal{C}_{x>1}$ and $\mathcal{C}_{x>2}$ depends

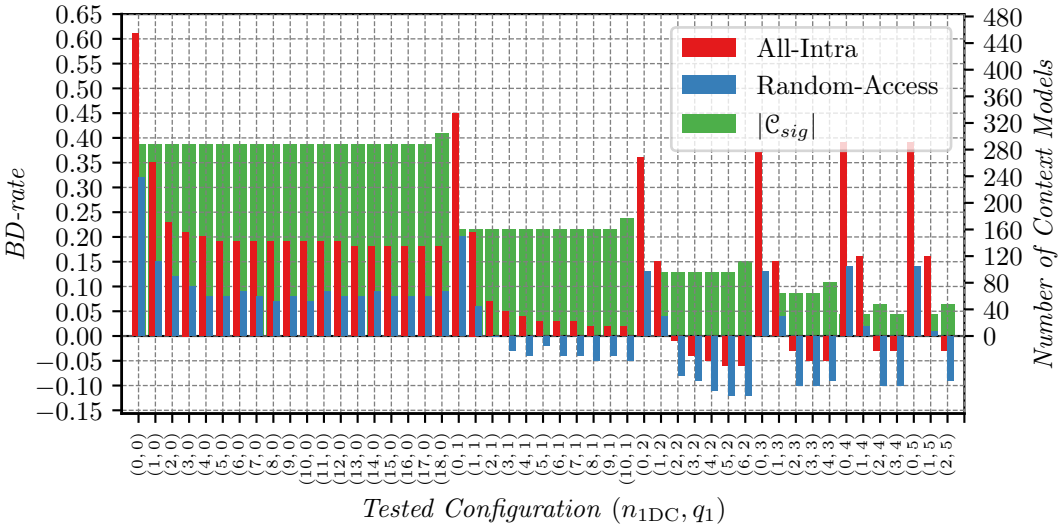

Figure 3.9
Coding efficiency of IMP3-7 with all tested combinations of n_{1DC} (context model offset for the first sub-block) and q_1 (the quantization parameter in equation (3.6)).
Anchor for BD-rate computations: IMP3-6*

on the parameters n_{2DC} and q_2 as follows:

$$|\mathcal{C}_{x>k}| = \begin{cases} m+1, & \text{if } n_{2DC} = m, \\ m, & \text{otherwise.} \end{cases} \tag{3.8}$$

The implementation for this investigation is referred to as IMP3-6 and the anchor to compute the BD-rates is IMP3-5. Figure 3.8 summarizes the experimental results and the number of context models for the tested parameters in IMP3-6. The configuration of q_2 mainly controls the number of context models, because $q_2 = 0$ implies that there is no quantization, whereas $q_2 = 5$ implies that the same context model set is used for all sub-blocks independent of $c_2\,[i_0]$.

The best performing combination in the *All-Intra* configuration is $(n_{2DC} = 5, q_2 = 2)$ with a BD-rate of -0.18%, whereas the best combination in the *Random-Access* configuration is $(n_{2DC} = 0, q_2 = 3)$ with a BD-rate of -0.13% . By analyzing these two configurations, the following observations can be made. In the best *All-Intra* configuration $(n_{2DC} = 5, q_2 = 2)$, the first sub-block of transform blocks larger than 4×4 is coded with the context model set that would otherwise only be used when all transform coefficient levels of the preceding sub-block have absolute values greater than one. It indicates that, for intra slices, the first sub-block is implicitly assumed to contain very high signal energy. In contrast to that, the best *Random-Access* configuration $(n_{2DC} = 0, q_2 = 3)$ reuses the 4×4 context model set for the first sub-block of larger transform blocks, which indicates that the first sub-blocks in inter slices have similar statistical properties as 4×4 transform blocks. Furthermore, a larger quantization parameter q_2 and, thus, a smaller set of context models is preferable for inter slices.

For the following coding experiments, the configuration $(n_{2DC} = 5, q_2 = 2)$ is used for transform blocks in intra slices, whereas the configuration $(n_{2DC} = 0, q_2 = 3)$ is used for all transform blocks in inter slices. This particular configuration is referred to as IMP3-6*.

Experimental Results for 1^{st} Coding Phase

With the aim of completely decoupling the number of context models from the number of supported transform sizes, similar coding experiments as for the second coding phase were also conducted for the context models used in the first coding phase. That means, the context offset n in equation (3.4) is replaced by a context offset n_1, which is derived in the same ways as n_2 in equation (3.6).

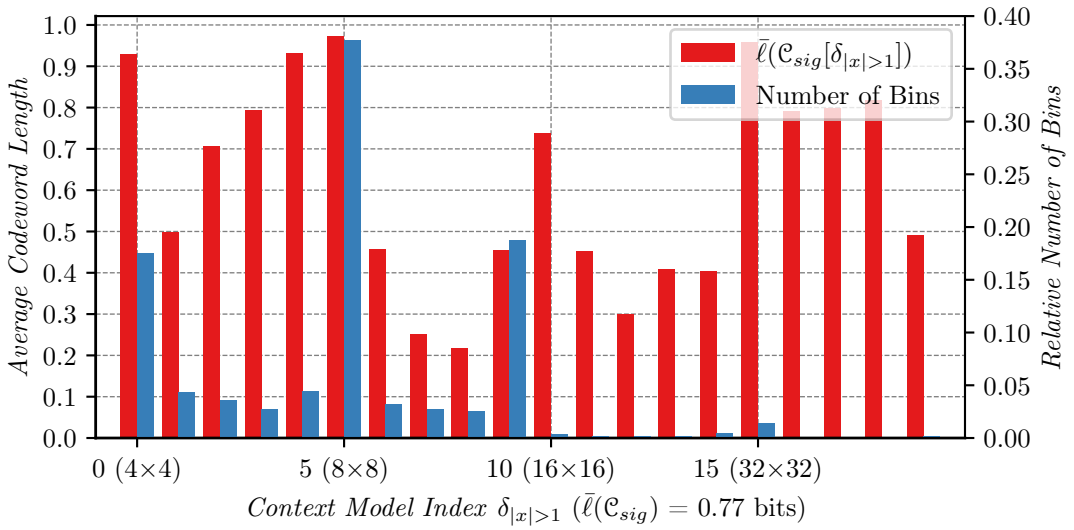

Figure 3.10
Average codeword length $\bar{\ell}(\mathcal{C}_{sig}[\delta_{|x|>1}])$ of context models used for coding $b_{|x|>1}$ in the generalized AVC design, where each transform block size uses a dedicated context model set.

Instead of $n_{2\mathrm{DC}}$ and q_2, the corresponding parameters for the first coding phase are referred to as $n_{1\mathrm{DC}}$ and q_1. Similar as for the second coding phase, the same values of $n_{1\mathrm{DC}}$ and q_1 are used for both context model indices δ_{sig} and δ_{last}.

The implementation for this investigation is referred to as IMP3-7, and the anchor used to compute the BD-rates is IMP3-6*. Figure 3.9 summarizes the experimental results, and most of the tested configurations show inferior coding efficiency relative to the anchor. Interestingly, configurations requiring many context models provide inferior coding efficiency, whereas configurations providing coding efficiency improvements require significantly fewer context models. The best configuration for both the *All-Intra* and the *Random Access* configurations is $(n_{1\mathrm{DC}} = 5, q_1 = 2)$; it is the same setting as the one that was found to be optimal in intra slices for the second coding phase. The adaptive selection of context model sets depending on the preceding 4×4 sub-block for the first coding phase provides slight coding efficiency improvement. Furthermore, by using the adaptive context model set selection for both coding phases, the main objective of decoupling the number of context models from the number of supported transform sizes could be achieved.

Context Modeling Analysis

To demonstrate the impact of the developed context modeling on the coding efficiency of the context-coded bins in the level coding, an example analysis of the average codeword length for the $b_{|x|>1}$ bins is presented. In a first experiment, the generalized AVC design was investigated. Figure 3.10 summarizes the average codeword length of all context models used for coding the $b_{|x|>1}$ flags, i.e., the context models of $\mathcal{C}_{x>1}$. Moreover, the overall average codeword length is given below the x-axis of figure 3.10. The data were acquired by decoding the first frame of the *BQTerrace* sequence, which implies that all $b_{|x|>1}$ flags were coded within an intra slice. Specifically, to obtain the actual presentation, the statistics output of the HM decoder with additional debugging options enabled was evaluated; they represent estimates of the actual average codeword lengths for the individual context models. When neglecting context adaptation and inefficiencies of the arithmetic coding engine, the measured overall average codeword length of 0.77 bits per bin basically represents the empirical conditional entropy $\hat{H}(X|C)$ for the used sample.

The same experiment but for the variant with 4×4 sub-blocks and the improved context modeling is summarized in figure 3.11; for this setting, an overall average codeword length of 0.73 bits per bin was measured. The results indicate that a bit-rate saving of about 4% was achieved for the $b_{|x|>1}$ flags by modifying the context modeling.

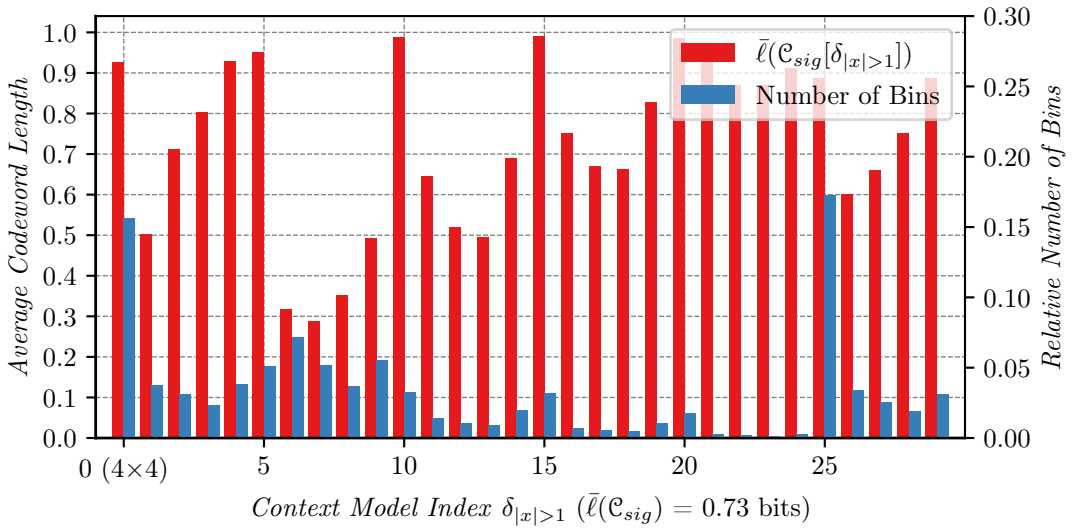

Figure 3.11
Average codeword length of the context models used for coding $b_{|x|>1}$ in the adaptive context model sets configuration, where the first context model set set given by index range $[0, 5)$ is used for 4×4 transform blocks only. The other context model sets are shared across the different transform sizes depending on the preceding sub-block.

For the generalized AVC design, the context model with index $\delta_{x>1} = 5$ (used for 8×8 transform blocks after a level with absolute value greater than one was coded in the reverse zig-zag scan) has the largest average codeword length. This context model is also used the most, as indicated by the relative number of bins in figure 3.10. The comparably large average codeword lengths of the context models used for 32×32 transform blocks ($\delta_{x>1} \geq 15$) is supposedly caused by the insignificant amount of bins coded with these context models. The experimental data for the 4×4 sub-block approach illustrated in figure 3.11 show that the bins are more evenly distributed among the available context models, except for the index range $\delta_{x>1} \in [20, 25)$. In summary, theses experiments indicate that the adaptive selection of context model sets for each 4×4 sub-block achieves a lower conditional entropy $H(X|C)$, which eventually results in a smaller average codeword length.

Class	Luma	C_B	C_R
	All-Intra		
A	-0.69%	-1.75%	-1.72%
B	-0.75%	-1.60%	-1.71%
C	-0.54%	-0.67%	-0.72%
D	-0.42%	-0.83%	-0.86%
E	-1.14%	-1.43%	-1.46%
Overall	**-0.69%**	**-1.26%**	**-1.31%**
	Random-Access		
A	-0.36%	-0.89%	-0.98%
B	-0.55%	-1.20%	-1.31%
C	-0.54%	-0.51%	-0.63%
D	-0.39%	-0.72%	-0.60%
E	-0.91%	-1.34%	-1.55%
Overall	**-0.53%**	**-0.92%**	**-1.00%**

Table 3.3
Coding efficiency of the level coding with 4×4 sub-blocks in its final configuration (IMP3-7*).
Anchor for BD-rate computations: IMP3-0

Algorithm 3.6 IMP3-7*: Level coding with 4×4 sub-blocks in its final configuration.

Require: block size N, slice type s_{type}

$i \leftarrow 0$, $i_{last} \leftarrow -1$, $n_1 \leftarrow 5$
if N $= 4$ **then**
  $n_1 \leftarrow 0$, $n_2 \leftarrow 0$
else if s_{type} is equal to *INTRA* **then**
  $n_2 \leftarrow 5$, $q_2 \leftarrow 2$
else
  $n_2 \leftarrow 0$, $q_2 \leftarrow 3$
end if
while $i_{last} < 0$ **do**
  $m \leftarrow 0$, $i_0 \leftarrow i$
  while $m < 16 \wedge i_{last} < 0$ **do**
    $s_x \leftarrow x\,(i) \gg \log_2(\mathrm{N} \gg 2)$
    $s_y \leftarrow y\,(i) \gg \log_2(\mathrm{N} \gg 2)$
    $z \leftarrow s_x + (s_y \ll 2)$, $\delta_{sig}\,(i) \leftarrow 16n_1 + z$, $\delta_{last}\,(i) \leftarrow 16n_1 + z$
    transmit $b_{sig}\,[i]$ using $\mathcal{C}_{sig}\,[\delta_{sig}\,(i)]$
    if $b_{sig}\,[i] = 1$ **then**
      transmit $b_{last}\,[i]$ using $\mathcal{C}_{last}\,[\delta_{last}\,(i)]$
      if $b_{last}\,[i] = 1$ **then**
        $i_{last} \leftarrow i$
      end if
    end if
    $m \leftarrow m + 1$, $i \leftarrow i + 1$
  end while
  $m \leftarrow m - 1$
  $c_1\,[i_0 + m] \leftarrow 1$, $c_2\,[i_0 + m] \leftarrow 0$
  while $m \geq 0$ **do**
    if $b_{sig}\,[i_0 + m] = 1$ **then**
      $\delta_{x>1}\,(i_0 + m) \leftarrow 5n_2 + \min(c_1\,[i_0 + m], 4)$
      transmit $b_{|x|>1}\,[i_0 + m]$ using $\mathcal{C}_{x>1}\,[\delta_{x>1}\,(i_0 + m)]$
      if $b_{|x|>1}\,[i_0 + m] = 1$ **then**
        $\delta_{x>2}\,(i_0 + m) \leftarrow 5n_2 + \min(c_2\,[i_0 + m], 4)$, $j \leftarrow 1$
        do
          $j \leftarrow j + 1$, $b_j = b_{|x|>j}\,[i_0 + m]$
          transmit b_j using $\mathcal{C}_{x>2}\,[\delta_{x>2}\,(i_0 + m)]$
        while $b_j \wedge j < 14$
        if $b_j \wedge j = 14$ **then**
          transmit $|x_{i_0+m}| - 15$ in bypass mode using $EG0$
        end if
        $c_2\,[i_0 + m - 1] \leftarrow c_2\,[i_0 + m] + 1$, $c_1\,[i_0 + m - 1] \leftarrow 0$
      else if $c_1\,(i_0 + m) \neq 0$ **then**
        $c_1\,[i_0 + m - 1] \leftarrow c_1\,[i_0 + m] + 1$
      end if
      transmit $b_{sign}\,[i_0 + m]$ in bypass mode
    end if
    $m \leftarrow m - 1$
  end while
  $n_1 \leftarrow (c_2\,[i_0] \gg 2) + 1$, $n_2 \leftarrow [c_2\,(i_0) \gg q_2] + 1$
end while

3.3.4 | Final Design with 4 × 4 Sub-Blocks

The final configuration of the alternative level coding with 4×4 sub-blocks is summarized in algorithm 3.6. For the adaptive selection of context model sets, the parameters specifying the initial context model offsets and the quantization parameter for the first coding phase are $n_{1\mathrm{DC}} = 5$ and $q_1 = 2$. The parameters for the second coding phase depend on the slice type s_{type}, and they are $n_{2\mathrm{DC}} = 5$ and $q_2 = 2$ when the current slice is an intra slice. For transform blocks in inter slices, these parameters are $n_{2\mathrm{DC}} = 0$ and $q_2 = 3$, respectively, as denoted in algorithm 3.6. Table 3.3 summarizes the coding performance for the level coding in its final configuration, which is referred to as IMP3-7*, relative to the generalized AVC design (IMP3-0). A coding efficiency improvement is achieved with BD-rates of -0.69% in the *All-Intra* and -0.53% in the *Random-Access* configurations. The total number of context models is equal to 252 ($|\mathcal{C}_{sig}| = |\mathcal{C}_{last}| = 6 \cdot 16$,

and $|\mathcal{C}_{x>1}| = |\mathcal{C}_{x>2}| = 6 \cdot 5$). Even though, for the investigated partitioner, the total number of context models is larger than in the generalized AVC design (168 context models), it does not depend on the number of supported transform sizes and, thus, enables a straightforward extension to additional transform sizes.

3.4 | Findings and Technical Achievements

The presented level coding with 4×4 sub-blocks was developed based on findings that were explored by data analysis and encoding simulations. They can be summarized together with the technical achievements of the level coding with 4×4 sub-blocks as follows:

- Local regions (within a transform block) with similar statistical properties among different transform sizes exist;
- This property allows for the partitioning of transform blocks into 4×4 sub-blocks;
- 4×4 sub-blocks enable hardware implementations to employ a unified logic for all additional block sizes;
- Within 4×4 sub-blocks, the existing context modeling designed for 4×4 transform blocks can be reused to achieve a coding efficiency similar to a design with dedicated context models per transform block size;
- The statistical properties of a 4×4 sub-block can be estimated based on coded data of the preceding 4×4 sub-block;
- That allows the estimation of suitable conditions for context model selection that are independent of the supported transform sizes;
- Such a conditional coding approach requires a fixed amount of context models, since those same context models are used for different transform sizes;
- The decoupling of the context memory from the number of supported transform sizes enables a straightforward design where additional transform sizes and shapes can be introduced by specifying the scanning pattern only;
- A coding efficiency improvement is achieved compared to the generalized AVC design.

The coding concept with 4×4 sub-blocks was proposed in [39, 68] during the development of the HEVC standard. Even though the final standard includes additional improvements, the basic design of the level coding with 4×4 sub-block is included in the HEVC standard and also its successor, the VVC standard.

The HEVC standard introduced variable transform block sizes of 4×4, 8×8, 16×16, and 32×32 samples where only square shapes are permitted for complexity reasons. Even larger transform sizes were considered, but they were not further pursued as they did not significantly improve the coding efficiency. HEVC specifies a 4×4 sub-block processing for both coding phases and a selection of context model sets depending on already coded sub-blocks. As in the presented alternative level coding, the adaptive selection of context model sets is employed for the second coding phase using the number of positions x within the preceding sub-block for which $b_{|x|>1}$ holds [37]. In contrast to the interleaved signaling of the last significant scanning position, HEVC employs a forward signaling of the last significant scanning position [29]. This signaling enables the application of the reverse scanning pattern for both coding phases, where the coding phases are interleaved on a 4×4 sub-block granularity, as described in this chapter. For the first coding phase, HEVC employs a context modeling that relies on the here presented evaluation of coded 4×4 sub-blocks with the following two enhancements. Firstly, two neighboring sub-blocks are evaluated to select one out of three fixed context model assignments for each sub-block [69]. Secondly, HEVC employs a b_{csf} flag (*coded sub-block flag*) that is similar to the b_{cbf} flag, but applied to 4×4 sub-blocks [70]. In summary, all core aspects of the level coding presented in this chapter can be found in the HEVC standard.

As VVC is the successor of HEVC, the level coding built upon that of HEVC and the main concepts already existing in HEVC can also be found in VVC. Here, the processing is realized with 4×4 sub-blocks and other

smaller sub-shapes for block sizes that are not divisible by 4×4 [71]. Employing already coded sub-blocks has been replaced by more advanced context modeling techniques for bins related to the transform coefficient levels. These advanced context modeling techniques are discussed in chapters 5 and 6. Nevertheless, the concept of evaluating already coded sub-blocks still exists in VVC for the context modeling of the b_{csf} flag.

3.5 | Chapter Summary

This chapter has presented an alternative level coding for variable transform block sizes with 4×4 sub-blocks. Its application allows reusing the existing context modeling design of 4×4 transform blocks within each 4×4 sub-block and enables the possibility for feasible hardware implementations. The concept of selecting context model sets depending on the preceding 4×4 sub-block within the same transform block further improves the coding efficiency. In conjunction with sharing the context models across different transform block sizes except for 4×4 transform blocks, a fixed number of context models is achieved. That property means that the number of context models is decoupled from the number of supported transform sizes and shapes, which is even more crucial for the *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC) standard.

The developed level coding impacts real-world applications because of its specification in the HEVC and VVC standards, as the design was successfully submitted to the standardization process. Particularly, the processing of larger transform blocks in smaller sub-blocks, the selection of context model sets depending on the preceding sub-block, and sharing context models among different transform sizes and shapes are integral parts of HEVC and VVC.

Continued efforts are necessary when there is a desire to achieve further coding efficiency, as provided by the presented level coding. The local activity has to be analyzed at a finer granularity than done by the current approach with 4×4 sub-blocks to identify statistical dependencies among the transform coefficient levels. Further throughput optimizations by reducing context memory are desirable for more efficient hardware implementations, since context memory requires surface area. Both aspects are further examined in chapter 5, where a more advanced context modeling design is presented, which is also the basis for the level coding in the VVC standard.

CHAPTER 4

Adaptive Binarization of Transform Coefficient Levels

Contents

Binary arithmetic coding and especially *context-based adaptive binary arithmetic coding* (CABAC) [28] achieves an efficient compressed representation by using adaptive context models. An adaptive context model represents a (conditional) binary probability model and resides in the writeable memory, allowing the model parameters to be updated. In the CABAC framework, a context model is updated by feeding the associated decoded binary symbol instantaneously back to the context model, which creates a direct dependency on the decoded symbol. This dependency causes a processing delay, which affects the overall throughput and parallel processing capability of software and hardware architectures. A reason why context model updates are expensive compared to the operational cost of the engine is due to the physical distance of the context memory in hardware implementations. Typically, context memory is located farther away from the engine logic due to its size and necessity of being writeable, whereas the read-only and limited-sized memory for the arithmetic coding engine can be placed close to the logic. The physical distances lead to longer signal paths, which implies more clock cycles for addressing and performing the reading and writing operations.

A single factor that is used to represent the above-described relationship between context memory and overall throughput is the maximum number of context-coded binary symbols, referred to as **bins**, that can occur within a bitstream. When designing hardware decoders, the engineers need to ensure that the chipset can handle the maximum number of context-coded bins supported by a profile and level combination. The static binarization process specified for transform coefficient levels in *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC) is the main contributor to the overall number of context-coded bins, as described in this chapter.

In this chapter, an alternative binarization scheme for transform coefficient levels that significantly reduces the maximum number of context-coded bins is presented. This alternative binarization achieves virtually the same coding efficiency as the AVC design by introducing adaptive prefix-free codes into the binarization process of transform coefficient levels. All bins of the newly introduced prefix-free codes employ the bypass processing path of CABAC. Both the *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC) [63, 47] standard, and its successor, the *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC) [64, 49] standard include the adaptive binarization for transform coefficient levels developed in this chapter.

4.1 | Problem Statement

Binary arithmetic coding engines process binary symbols; hence, non-binary input values require a mapping to binary sequences, referred to as **bin strings**. Such a mapping for the integer-valued transform coefficient levels can be described by a binarization function $f : z \in \mathbb{Z} \rightarrow (\mathtt{b}_n)_{n\in\mathbb{N}} : \mathtt{b}_n \in \mathcal{B}$. The input parameter

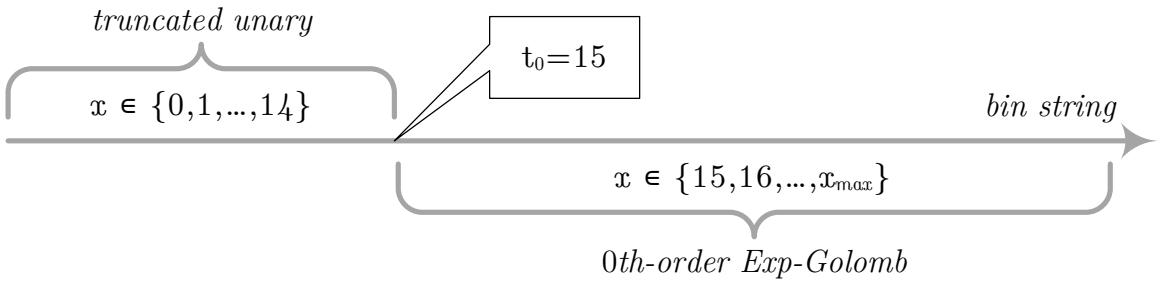

Figure 4.1
Static binarization scheme for transform coefficient levels in AVC, involving the concatenation of TRU and EG0 codes. For $x \in [0, 14] \cap \mathbb{N}_0$, the bin string consists of a TRU code only, whereas for $x \geq 15$, the bin string consists of the TRU code for $x = 15$ and an EG0 code for $x - 15$.

required for the binarization process of transform coefficient levels in AVC is the transform coefficient level itself, and this so-called static binarization allows for the reconstruction of the input value given the bin string only. Such a static binarization requires a certain amount of bins coded with adaptive context models to achieve a specific coding efficiency. However, this relatively high amount of context-coded bins is an issue for efficient hardware implementations, even though static binarization is relatively simple at the logic layer. Besides the general throughput limitation caused by the physical distance of the context memories from the engine, another bottleneck occurs within the processing when using the same context model for n consecutive bins. In such a case, the decoder parses n times the symbol from the bitstream and updates the context model after each bin. Due to the serial processing of bins in the binary coding engine, the decoder is stuck with the single context model and cannot execute further processes in parallel by, e.g., employing speculative processing via pipelining strategies. Further aspects and constraints that need to be considered for hardware implementations in entropy coding of practical video coders are discussed in [72].

4.2 | Static Binarization

The binarization of transform coefficient levels specified in AVC is static and involves two different prefix codes: The *truncated unary* (TRU) and the 0^{th}*-order exponential-golomb* (EG0) codes. The output of the binarization process is a bin string representing the transform coefficient level. Each bin is either coded with an adaptive context model, where the internal state of the context model has to be updated depending on the reconstructed bin value, or it is coded in the bypass mode, where a non-adaptive uniform *probability mass function* (pmf) is used. Bins belonging to the TRU code employ adaptive context models, whereas bins belonging to the EG0 code are coded in the bypass mode. The reason for using the bypass mode for EG0 bins is that the resulting codewords are expected to be close to a minimum-redundancy code and rarely occur for typical operation points. For the sake of more understandable and clearer notations, only absolute transform coefficient levels, denoted by x, with $x \in \mathbb{N}_0$, are considered in the remainder of this chapter.

4.2.1 | Binarization of Transform Coefficient Levels in AVC

For an absolute transform coefficient level $x \leq 14$, the bin string consists of a TRU codeword only. When $x \geq 15$, the bin string consists of the TRU codeword for $x = 15$ (i.e., 15 times $'1'$) followed by the EG0 codeword for $x - 15$. Let $f_{TRU}(\cdot, \text{maxVal})$ denote the TRU code with the maximum representable value maxVal, f_{EG0} denote the EG0 code, and $*$ denote the concatenation operator for bin sequences, then the binarization $f(x)$ in AVC can be expressed by:

$$f(x) = f_{TRU}(x, 15) * g_{EG0}(x - 15), \tag{4.1}$$

where,

$$g_{EG0}(z) = \begin{cases} (\emptyset), & \text{if } z < 0, \\ f_{EG0}(z), & \text{otherwise.} \end{cases} \tag{4.2}$$

Table 3.1 in section 3.2 summarizes the bin strings of the first 20 absolute transform coefficient levels x. Furthermore, the binarization concept is illustrated in figure 4.1, where the axis denotes the bin string

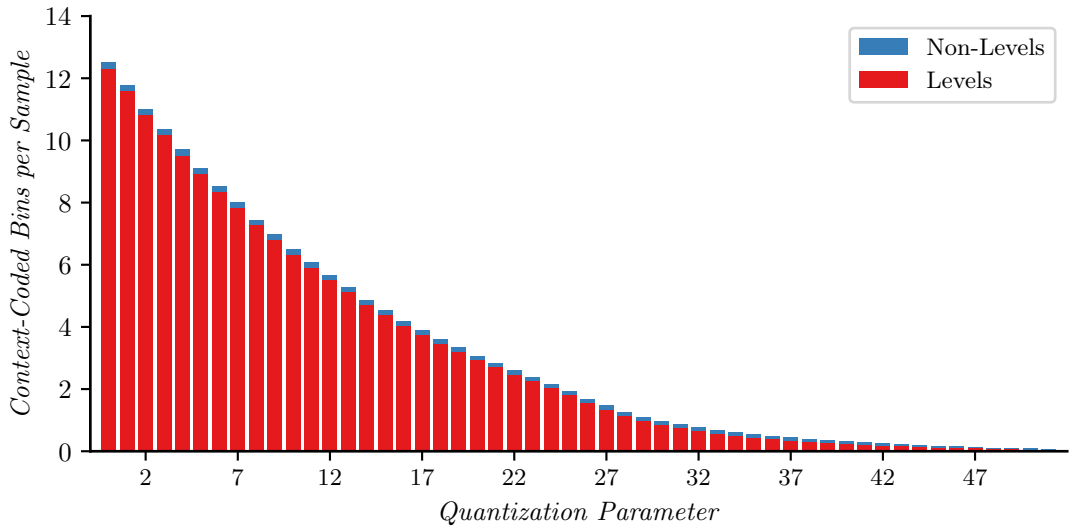

Figure 4.2
The cps vary depending on the operation point, which is selected via the *quantization parameter* (QP). The red portion of the bars denotes the cps for transform coefficient levels, while the blue portion represents the cps for non-level data.

length. For all bin strings that exceed the denoted fixed threshold, the composition of the bin string consists of a EG0 suffix with the same TRU prefix (15 times $'1'$). The bin string then represents an absolute level equal to or greater than 15, whereas all absolute levels less than or equal to 14 do not have a EG0 suffix. The fixed transition t_0 from TRU to EG0 is equal to $t_0 = 15$ for the binarization of transform coefficient levels specified in AVC, i.e., when $x \geq t_0$, the bin string has a EG0 suffix. As all bins belonging to TRU are coded with adaptive context models, the transition boundary t_0 directly dictates the maximum number of context-coded bins for transform coefficient levels.

4.2.2 | Context-Coded Bins per Sample

Besides transform coefficient levels, context-coded bins are being used for coding non-level data, such as the prediction modes, prediction data, and partitioning information. Therefore, the context-coded bins can be split into two categories: The bins for level coding and the bins for signaling the side information. An upper limit for the number of context-coded bins, expressed in *context-coded bins per sample* (cps), can be estimated under the assumption that an encoder always selects the smallest available block size, which is equal to 4×4. In that scenario, the prediction mode and data are signaled for each 4×4 block, whereas the signaling of the partitioning depends on the initial block size. Let us assume that the initial block size is 64×64, which is the maximum initial block size specified in the HEVC standard, and further assume that the partitioning is based on quadtrees only. Then, four level of splits are necessary to signal a partitioning consisting of 4×4 blocks only, which require 85 context-coded bins in total for an area of 64×64 samples (1 bin to 32×32, four bins to 16×16, 16 bins to 8×8, and 64 bins to 4×4 blocks). Each sample has then a share of $^{85}/_{4096} \leq 0.021$ bins for signaling the partitioning. For the prediction, it can be estimated that the mode and associated data require at most ten context-coded bins (1 bin for prediction mode, either intra or inter, and up to nine bins for the prediction information, such as motion vector differences and reference indices) for a 4×4 block. Additional coding tools exist that require signaling with context-coded bins, such as the sample adaptive offset , and it is further assumed that they require up to five context-coded bins. Each sample has a share of less than one context-coded bin for coding the side information, when considering the above estimations and the luma component only.

In summary, for 4:2:0 chroma sub-sampled video content, the approximate worst-case limit is less than 1.5 bins for signaling the side information. That value is significantly smaller than the cps for transform coefficient levels, which is equal to 22.5 for video coded in the $Y'C_BC_R$ 4:2:0 format, when using the AVC

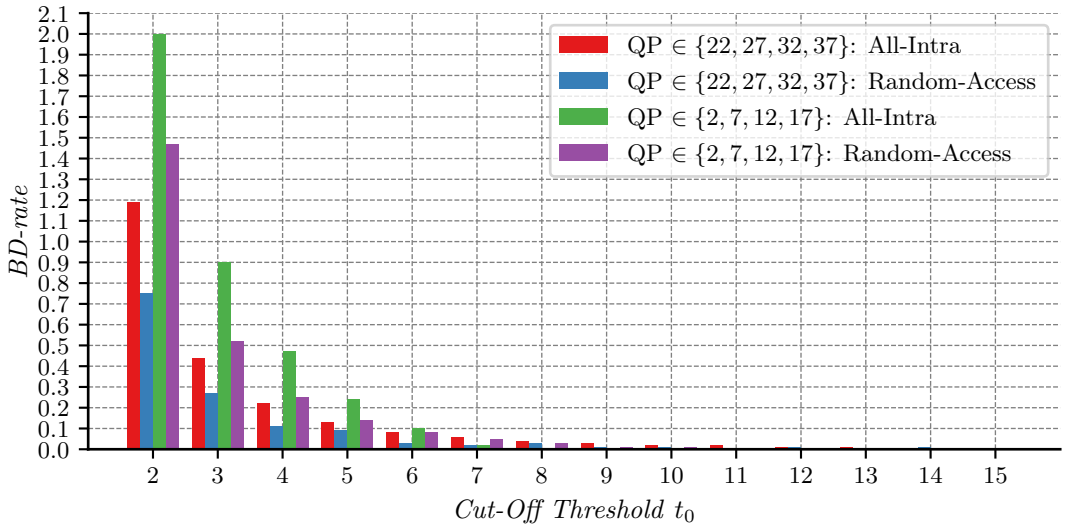

Figure 4.3
Coding efficiency of the investigation on different cut-off thresholds t_0 (IMP4-0).
Anchor for BD-rate computations: IMP3-7*

binarization.

The above estimation of the worst-case cps limit is rather theoretical and may only occur for artificial input signals. An investigation on the actual cps for typical video content, depending on the operation point, was experimentally analyzed, and the results are summarized in figure 4.2. The data used for the analysis were acquired by using existing bitstreams, which were encoded using IMP3-7*, i.e., the level coding in its final implementation presented in chapter 3. In figure 4.2, the cps are separately listed for transform coefficient levels and non-level data, which represents the side information. The cps increases monotonically with the operation point by a factor close to seven from 0.03 for $QP = 51$ to 0.21 for $QP = 0$ for the non-level data. The observation indicates that the encoder chooses a finer partitioning structure with increased bit-rate, i.e., the encoder selects smaller block sizes for prediction and transform coding. Note that the relatively small cps values for the side information compared to the estimated upper bound is because the maximum number of context-coded bins for the side information is usually not required for typical camera-captured content. For transform coefficient levels, the cps increase is steeper, starting from 0.04 for $QP = 51$ and ending at 12.30 for $QP = 0$. In summary, the experimental results in figure 4.2 confirm that the context-coded bins for transform coefficient levels dominate the overall cps. It is, therefore, sufficient to concentrate on the context-coded bins for the transform coefficient levels to reduce the total number of context-coded bins.

4.2.3 | Alternative Fixed Thresholds

Without modifying the basic concept of the AVC binarization, the cps worst-case bound can be reduced by lowering the cut-off threshold t_0. The impact of lowering the t_0 value on the coding efficiency was experimentally evaluated by performing coding experiments with the QPs specified in the *common test conditions* (CTC). The implementation of this investigation is referred to as IMP4-0, and the anchor used for *Bjøntegaard delta bit-rate* (BD-rate) computations is IMP3-7*. Experimental results for this investigation are summarized in figure 4.3. Besides the CTC QP set $\{22, 27, 32, 37\}$, additional BD-rates for the QP set $\{2, 7, 12, 17\}$, representing the high bit-rate operation points, are presented. A cut-off threshold lower than two was not tested as it would disable the adaptive selection of context model sets for each 4×4 sub-block developed in chapter 3, resulting in inferior coding efficiency.

For the first QP configuration, i.e., the operation points according to the CTC, slight losses in coding efficiency can be observed when the cut-off is reduced to $t_0 = 4$, while the inferior coding efficiencies are significant for $t_0 \in \{2, 3\}$. A similar observation can be made for the QP configuration representing the high bit-rate operation points, where the absolute transform coefficient levels have higher magnitudes, and the

losses are higher compared to the first QP set. The experimental results show that setting $t_0 = 6$ can reduce the worst-case cps significantly with little penalty on the coding efficiency. Nevertheless, the reduction is not sufficient, since the cps of transform coefficient levels is still significantly higher than the cps for the side information. Note that a reduction in the number of context models only occurs for $t_0 = 2$, where the context models of $\mathcal{C}_{x>2}$ are no longer required, which would save 90 context models. For the configuration $t_0 = 3$, the same number of context models are necessary as before, but the usage of the same context model for successive bins can be avoided.

4.3 | Adaptive Binarization

When lowering t_0 significantly to achieve a worst-case cps value that is closer to that of the side information, e.g., to $t_0 \in \{2, 3\}$, the coding efficiency losses are significant. The observed inferior coding efficiencies arise because the EG0 code is sub-optimal for the actual distribution of the remainders when t_0 is small. Nevertheless, it is possible to reduce the loss in coding efficiency when the code used for binarizing the remainders is adjusted to the actual probability distribution. For typical camera-captured content, the probability distribution of the remainders depends on various parameters, such as the transform size, the QP, the input signal, and more. An adaptive approach that selects different prefix codes that suit the actual probability distribution of the remainder is a solution for retaining the coding efficiency provided by the current design. Different granularities for the adaptivity are possible, e.g., by applying the same prefix code for a region of a transform block, for the whole transform block, or determining a code for each scanning position. The latter design is aligned with the existing tracking variables used in the second coding phase, where the context modeling evaluates the preceding scanning position. Before presenting further investigations for the adaptive design that uses the preceding scanning position to determine a suitable prefix-free code for the remainder of the current scanning position, a probability model for the remainder of transform coefficient levels is introduced.

4.3.1 | Probability Model

An often-used probability model for the distribution of transform coefficients (see, for example, [73]), which can to a certain degree also be verified experimentally, is the zero-mean Laplacian distribution. According to this model, the absolute transform coefficients c, with $c \in \mathbb{R}^+$, have an exponential *probability density function* (pdf) given by:

$$p_C(c) = \mu \exp(-\mu c). \tag{4.3}$$

The probability masses for the absolute transform coefficient levels are determined by the pdf of the Laplacian model and the encoder algorithm for quantization. However, practical video coding standards specify the inverse quantization (scaling process) only, while the quantization itself is not included in the specifications. That gives practical encoder implementations a certain degree of freedom to conduct the quantization, e.g., by performing an entropy-constrained scalar quantization scheme [74], such as *rate-distortion optimized quantization* (RDOQ) [75] in the HM reference software implementation. Nevertheless, considering those aspects in the following approximations is intractable, and low-complexity encoders often use simple quantization algorithms similar to what is described next. Let Δ be the quantization step size and $a \in [0, 0.5]$ a rounding offset, then an absolute transform coefficient level x results from the following quantization of an absolute transform coefficient c according to:

$$x = \lfloor \frac{c}{\Delta} + a \rfloor. \tag{4.4}$$

To obtain the pmf $p_X(x)$ for the absolute transform coefficient levels from $p_C(c)$, the considered quantization denoted in equation (4.4) can be applied to the pdf. A distinction is necessary between the first and all other intervals, because the interval size is equal to $(1 - a)\Delta$ for the first interval, whereas it is equal to Δ for the other intervals. For the first interval, the probability is:

$$\begin{aligned} p_X(x = 0) &= \int_0^{(1-a)\Delta} \mu \exp(-\mu c)\, dc \\ &= 1 - \exp(-\mu\Delta(1 - a)). \end{aligned} \tag{4.5}$$

For the remaining intervals ($x > 0$), the probabilities are:

$$\begin{aligned} p_X(x) &= \int_{(x-a)\Delta}^{(x+1-a)\Delta} \mu \exp(-\mu c)\, dc \\ &= \exp(-\mu\Delta(x-a)) - \exp(-\mu\Delta(x+1-a)) \\ &= \exp(-\mu\Delta(x-a))(1-\exp(-\mu\Delta)). \end{aligned} \tag{4.6}$$

The remainders $z \in \mathbb{N}_0$ of absolute transform coefficient levels emerge from applying a cut-off t_0 in the binarization process, and they are only transmitted when an absolute level x is greater than or equal to the cut-off value t_0. With y representing $\exp(-\mu\Delta)$, the pmf of the remainders for all cut-off values $t_0 > 0$, which are only transmitted when the absolute level is greater than or equal to t_0, is given by:

$$\begin{aligned} p_Z(z) &= p_X(x = z + t_0 | x \geq t_0) \\ &= \frac{p_X(x = z + t_0)}{\int_{(t_0-a)\Delta}^{\infty} p_C(c)\, dc} \\ &= \frac{y^{z+t_0-a}(1-y)}{\int_{(t_0-a)\Delta}^{\infty} \mu \exp(-\mu c)\, dc} \\ &= \frac{y^z (1-y)\, y^{t_0-a}}{\exp(-\mu\Delta(t_0-a))} \\ &= \frac{y^z (1-y)\, y^{t_0-a}}{y^{t_0-a}} \\ &= y^z (1-y). \end{aligned} \tag{4.7}$$

By equation (4.7), the remainders of absolute transform coefficient levels follow a geometric pmf for all cut-off values $t_0 > 0$, when assuming a zero-mean Laplacian model for transform coefficients and the encoder quantization algorithm in equation (4.4), and the model parameter y of the geometric distribution is independent of t_0. A further conclusion from the analysis is that, when considering context-adaptive binary arithmetic coding, the first bin (b_{sig}) should be coded with a dedicated context, whereas the remainders should be binarized with the unary code and coded with another dedicated context model. Employing a shared context model for the remainders as done by the existing context modeling in AVC is, therefore, a suitable choice given the analysis above. Note that the model parameter y ($0 \leq y \leq 1$) of geometric distributions controls two properties. Firstly, it determines the probability for $p_Z(z = 0) = 1 - y$, and secondly, it controls the skewness of the distribution.

4.3.2 | Empirical Conditional Distribution

The theoretical considerations can be verified by empirical data, as illustrated in figure 4.4 that contains four different histograms using transform coefficient levels of 4×4 transform blocks. The original data were used for the top-left histogram, i.e., without applying a cut-off value, and the histogram represents the empirical marginal pmf of the absolute transform coefficient levels. For the three remaining histograms, a cut-off value was applied, and these are conditional histograms. Hence, they represent the empirical pmfs of the remainder values. From the above analysis, the conditional pmfs should be geometric when applying a cut-off $t_0 > 0$. For the overlayed geometrical distributions, the model parameter y of each pmf is selected according to the relative frequency (or empirical probability) of the first bin. If the marginal pmf is considered, i.e., the case without a cut-off, the overlaid geometric distribution does not fit the empirical data well. However, for the conditional pmfs, the overlaid geometric distributions show more suitable approximations to the empirical conditional histograms. Although the theoretical analysis showed that the distribution is independent of the value of t_0, the histograms show that the distribution parameter depends on the value of t_0, which is supposedly due to the inaccuracy of the zero-mean Laplacian model. Nevertheless, these observations indicate that the geometric distribution, derived by theoretical analysis, is a suitable approximation for the remainders, which is sufficient for selecting an appropriate prefix code.

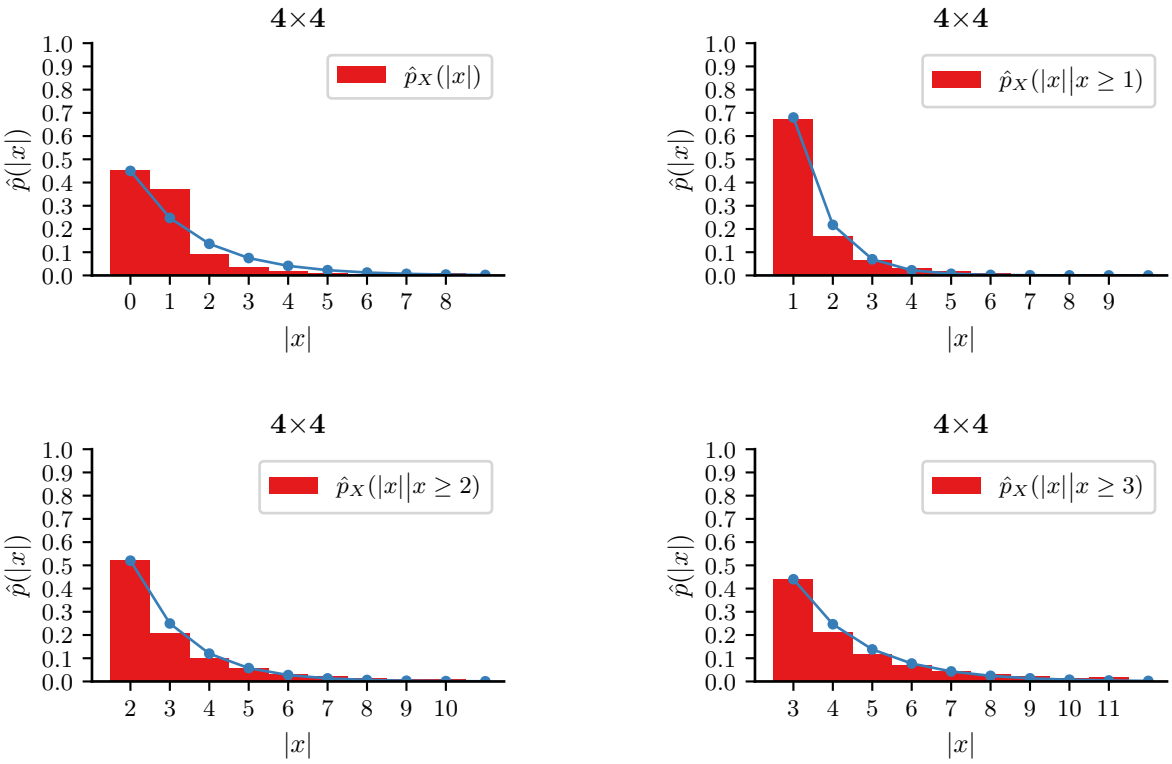

Figure 4.4
Four histograms are shown above using absolute transform coefficient levels of 4×4 transform blocks. The top-left histogram represents the (empirical) marginal pmf, whereas the other histograms represent the (empirical) conditional pmfs considering absolute transform coefficient levels greater than or equal to a specific threshold. For the overlaid geometric distributions, the model parameters y are selected based on the empirical probability for the first histogram entries, $y = 1 - p_Z\ (z = 0)$

4.3.3 | Golomb and Rice Codes

For geometric distributions, Golomb codes [76, 77] were proven to be an optimal class of variable-length codes and can be used for the binarization of the remainders (i.e., using the bypass coding mode for the resulting bins). Golomb codes consist of a prefix and a suffix, where the prefix is a unary code, and the suffix is a truncated fixed-length code. An actual Golomb code that can be used to represent a remainder z is selected by the Golomb parameter b, with $b \geq 0$. The main property of Golomb codes is a constant bucket size, i.e., each prefix represents b codewords, as illustrated in figure 4.5.

Construction of Rice Codes

Rice codes are a subset of Golomb codes, and the corresponding parameter k is used instead of b to specify the actual Rice code, where the relationship between k and b is given by:

$$b = 2^k. \tag{4.8}$$

Because b is now a power-of-two value, the prefix construction and reconstruction can be implemented by binary-shift operators, whereas the suffix can be represented by the fixed-length binary code, where its length is equal to k. The construction and reconstruction of Rice codes are summarized in algorithm 4.1. Using the subset of Rice codes is implementation-friendly, but the optimality provided by Golomb codes for all geometric pmfs is not given anymore, because only Golomb codes with the Golomb parameters $\{1, 2, 4, 8, 16, \ldots\}$ can be used. However, further investigations in the next subsection show that the most-used Rice parameters are $k \in \{0, 1\}$ for the remainders z, which allows for using Rice codes with minor penalties on the coding efficiency.

In table 4.1, the codewords for $z \in [0, 7] \cap \mathbb{N}_0$ with different Rice parameters are listed, where the case

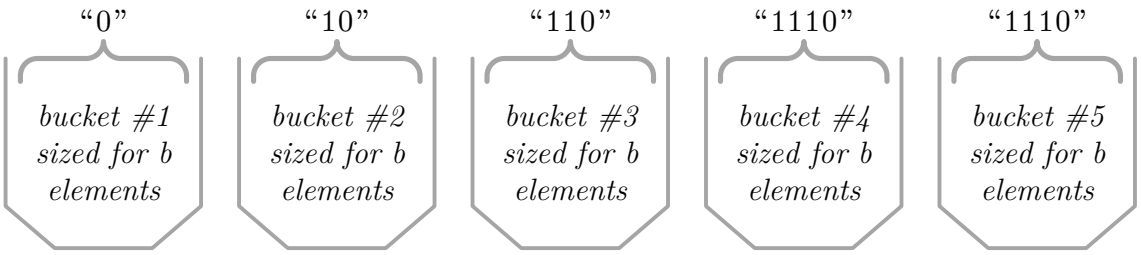

Figure 4.5
Bucket model of Golomb codes where the number of codewords in a bucket (identified by the prefix) is constant and is equal to the Golomb parameter b. This property is beneficial for $b = 2^k$ yielding a fixed-length suffix, allowing the decoder to determine the suffix length as soon as the Rice parameter k is determined.

Algorithm 4.1 Construction and reconstruction of a remainder z with Rice codes, where k is the Rice parameter.

```
1: procedure RICEENCODE(z, k)
2:     q ← z ≫ k
3:     r ← z − q ≪ k
4:     unary(q)
5:     fixedLengthBinary(r, k)
6: end procedure
7: procedure RICEDECODE(k)
8:     q ← unary()
9:     r ← fixedLengthBinary(k)
10:    return (q ≪ k) + r
11: end procedure
```

$k = 0$ was left out, because the codewords are the same as those of the unary code. To understand the code rate (increase in codeword lengths) for the different prefix-codes, figure 4.6 illustrate codeword lengths $\ell(z)$ depending on the remainder z. When the Rice parameter k is large, there is a significant penalty for small remainders, and they should only be used for geometric pmfs with very small probabilities for $p_Z(z = 0)$. Note that the histograms in figure 4.4 indicated that the model parameter y increases with increasing cut-off value t_0. Therefore, the optimal Rice parameter k is expected to increase with an increasing cut-off value t_0.

Optimal Rice Parameter

For a geometric source with a known model parameter y, the optimal Golomb parameter b can be derived according to [58, p.37]. Let $\phi = \left(1+\sqrt{5}\right)/2$ denote the golden ratio, then the optimal Rice parameter k for a given model parameter y can be calculated by [78]:

$$k_{opt} = \max\left(0, 1 + \left\lfloor \log_2\left(\frac{\log(\phi - 1)}{\log y}\right)\right\rfloor\right). \tag{4.9}$$

z	$k=1$		$\ell(z)$	$k=2$		$\ell(z)$	$k=3$		$\ell(z)$	$k=4$		$\ell(z)$
0	0	0	2	0	00	3	0	000	4	0	0000	5
1	0	1	2	0	01	3	0	001	4	0	0010	5
2	10	0	3	0	10	3	0	010	4	0	0010	5
3	10	1	3	0	11	3	0	011	4	0	0011	5
4	110	0	4	10	00	4	0	100	4	0	0100	5
5	110	1	4	10	01	4	0	101	4	0	0101	5
6	1110	0	5	10	10	4	0	110	4	0	0110	5
7	1110	1	5	10	11	4	0	111	4	0	0111	5

Table 4.1
Codewords for the remainders $z \in \{0, 1, \ldots, 7\}$ when using Rice codes with the parameter $k \in \{1, 2, 3, 4\}$.

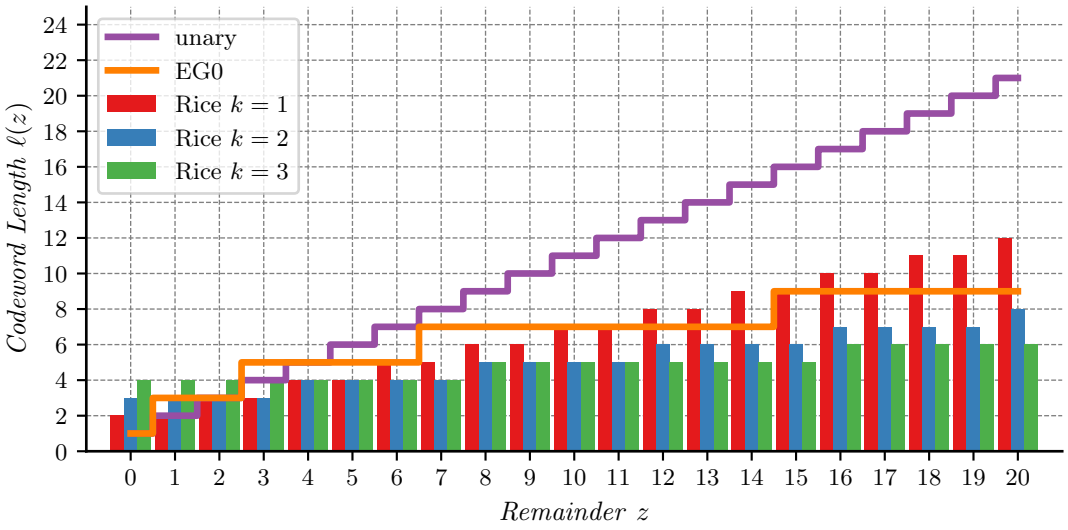

Figure 4.6
Codeword lengths $\ell(z)$ for unary, EG0, and different Rice codes, where the actual Rice parameters are $k \in [1, 4] \cap \mathbb{N}_0$.

The optimal Rice parameter given the model parameter y of geometric distributions according to the above formula is illustrated in figure 4.7. When $y < \phi - 1 \approx 0.62$, the optimal Rice parameter is $k = 0$, i.e., the best Rice code is the unary code. Note, however, that the unary code is only a zero-redundancy code for $y = 0.5$.

Optimum Rice Parameter Given Remainder

The calculation in equation (4.9), which was derived in [78], is based on the assumption that the model parameter y is available for a block of input values, and the same Rice code is then used for those input values yielding the minimum average codeword length. That is different from the concept of selecting a Rice code for the remainder z of each scanning position, which is being pursued here.

When selecting the Rice code for each scanning position, the minimal codeword length for a given remainder, and derived from that, the optimal Rice parameter, is of more interest. Multiple Rice codes can result in the same codeword length, and figure 4.8 summarizes minimum Rice parameter k_{min} and the maximum Rice parameter k_{max} that are optimal for remainders z. To minimize the losses in coding efficiency, a switch to a different Rice code is necessary when z is expected to become greater. Figure 4.8 shows the Rice code that has to be used to achieve the shortest codeword length. In such an ideal environment, the points to switch

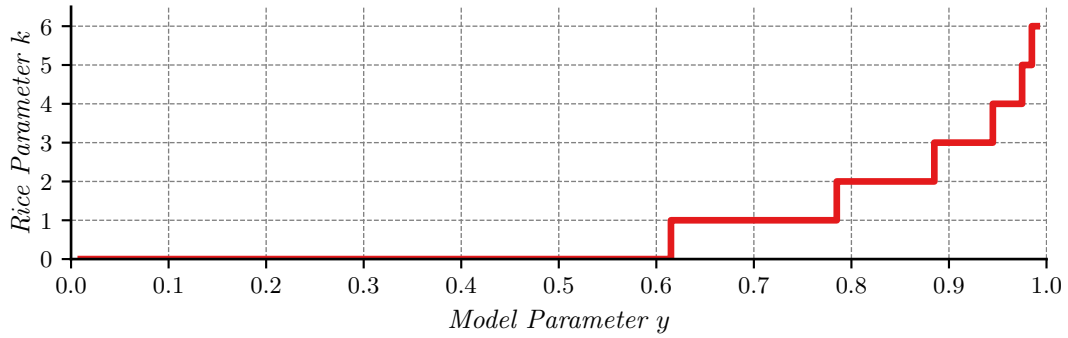

Figure 4.7
Optimal Rice parameter k given the model parameter y of the geometric distribution, derived according to equation (4.9).

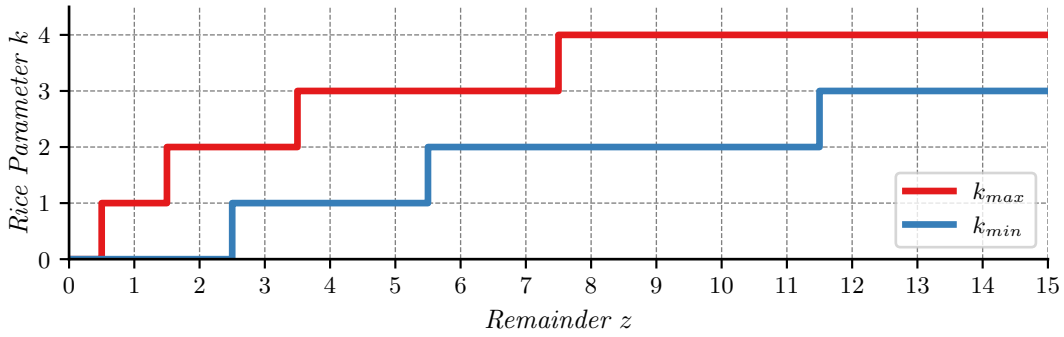

Figure 4.8
Optimal Rice parameters k for a single remainder value z.

between Rice codes can be specified by:

$$\begin{aligned} k = 0 \rightleftharpoons k = 1 &: \text{ for } z \in [1, 2] \cap \mathbb{N}_0, \\ k = 1 \rightleftharpoons k = 2 &: \text{ for } z \in [2, 5] \cap \mathbb{N}_0, \\ k = 2 \rightleftharpoons k = 3 &: \text{ for } z \in [4, 11] \cap \mathbb{N}_0. \end{aligned} \tag{4.10}$$

Fixed Rice Binarization

Two investigations were performed to understand the efficiency of Rice codes for the binarization of transform coefficient levels in practical video coding environments. The first investigation analyzes the frequency of Rice codes that generate the shortest codeword length when the remainder z is known at the decoder side by evaluating existing bitstreams. For the second investigation, additional encoding experiments were performed, and the best Rice parameter for the whole test set was determined, i.e., the same Rice parameter for all test sequences and QPs that provides the best BD-rate. Moreover, the second investigation includes further BD-rate results that were derived by calculating Rice parameters for different operation points.

For the first investigation, the bitstreams generated for the study in section 4.2.3, i.e., IMP4-0 with the $t_0 = 3$ configuration were used, and the evaluation assigned each remainder z to a Rice code that provides the shortest codeword length. When multiple Rice codes generate the same codeword length for a remainder z, the Rice code with the lower Rice parameter is counted as being the optimal Rice code. Figure 4.9 summarizes the investigation results as a histogram that denotes the frequency of optimal Rice codes for the remainders within the bitstreams. The investigation shows that the case $k > 3$ rarely occurs, and most of the remainders can be coded with $k = 0$.

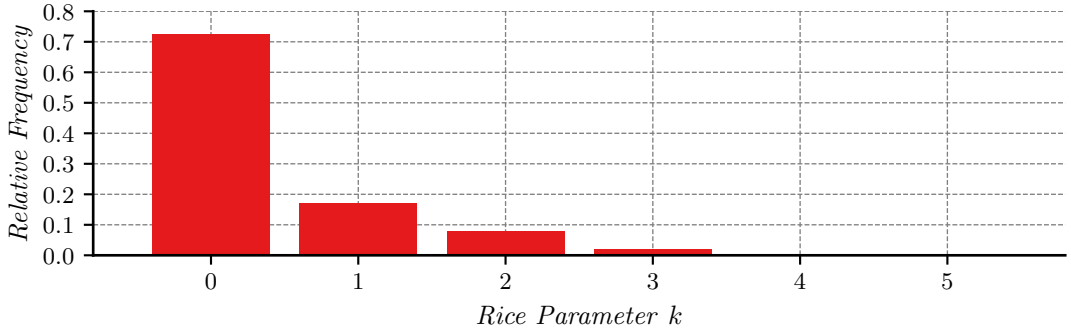

Figure 4.9
Histogram denoting the relative frequency for a Rice code that generates the shortest codeword length given a remainder z, where the remainders were extracted from existing bitstreams.

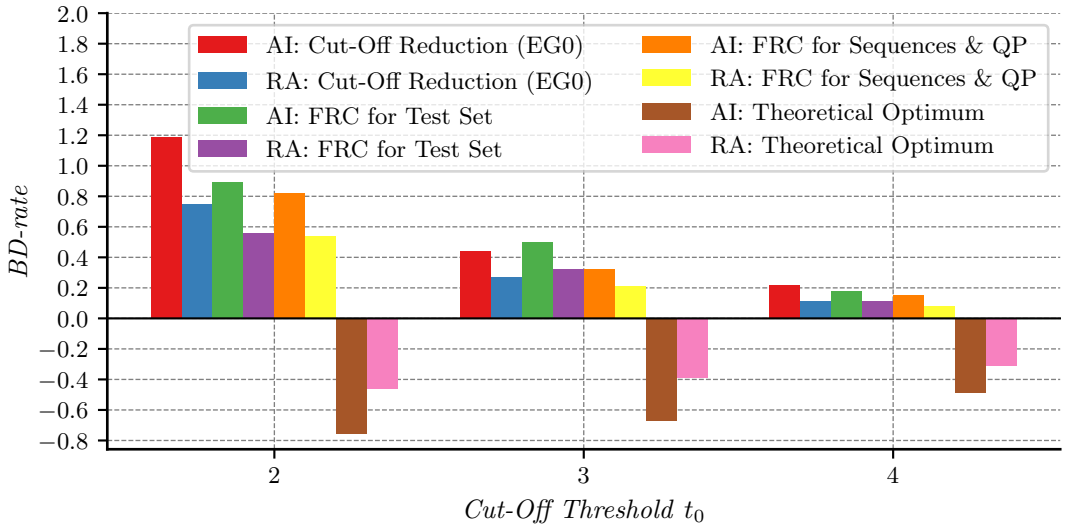

Figure 4.10
Coding efficiency of binarizations with Rice code with $t_0 \in \{2, 3, 4\}$ for the remainders z instead of the EG0 code (IMP4-1). "FRC for Test Set" denotes the best configuration where k is the same for all test sequences and QPs. "FRC for Sequences & QP" employed the results from the same set of experiments, but the BD-rate calculations were performed based on different Rice code combinations for each test sequence and QP, i.e., the Rice parameter k varies among the test sequences and QPs, but is the same for each bitstream. Finally, "Theoretical Optimum" denotes a configuration where the Rice code providing the shortest codeword length for a remainder z is chosen, which is a configuration that is not decodable.
Anchor for BD-rate computations: IMP3-7*

For the second investigation, additional encoding experiments were performed, where the EG0 code was replaced by different Rice codes, and the corresponding implementations are referred to as IMP4-1. Each set of experiments was configured to use a different Rice code, where $k \in [0, 8] \cap \mathbb{N}_0$, for all test sequences and QPs. The configuration that provides the best coding efficiency is selected, and figure 4.10 summarizes the BD-rate for cut-off thresholds $t_0 \in \{2, 3, 4\}$. For calculating the BD-rates, IMP3-7*, which is the same anchor used for the experiments in figure 4.3, was employed. "Cut-Off Reduction (EG0)" for each cut-off threshold t_0 denotes the coding efficiency of the configuration using the EG0 code for the binarization of the remainder, which is equivalent to the corresponding results in figure 4.3. In figure 4.10, "FRC for Test Set" in figure 4.10 denotes the coding efficiency for the best performing Rice code for all test sequences and QPs, which is $k = 0$ for all cut-off thresholds. "FRC for Sequence & QP" denotes the coding efficiency when the best performing Rice code can be different for each test sequence and QP, where Rice parameters $k < 2$ were chosen for all test sequences and operation points. The results of the second investigation are consistent with the results of the first investigation, i.e., most of the remainders z can be coded with $k < 2$. Furthermore, the second investigation shows that switching among different Rice codes can provide improved coding efficiency, as indicated by the difference in coding efficiency between "FRC for Test Set" and "FRC for Sequence & QP" for $t_0 = 3$ in figure 4.10. Finally, "Theoretical Optimum" denotes a configuration that is not decodable, where the Rice code providing the shortest codeword length for a remainder z is selected for encoding. Although this configuration is not decodable, it illustrates the theoretical limit that can be achieved by using Rice codes with adaptive selection of the Rice parameter

4.3.4 | Backward-Adaptive Rice Parameter Estimation

In the previous subsection, the investigations demonstrate that using Rice codes is feasible and that the theoretical findings match the practical coding environment. The distributions of the remainders z can be approximated by geometric pmfs. By using a Rice code with a fixed Rice parameter for the remainder values, the coding efficiency can be improved relative to using the EG0 code. The investigations further reveal that most remainders have a minimal codeword length for Rice parameters $k \in \{0, 1\}$, and remainders, where the

shortest codeword length is obtained for $k > 3$, rarely occur.

A remainder is transmitted after coding $b_{|x|>1} = 1$ or $b_{|x|>2} = 1$, depending on the configuration of t_0. Therefore, the selection of the Rice parameter for the current scanning position should rely on preceding scanning positions, because this approach aligns the adaptive binarization with the existing tracking variables used in the second coding phase. With that consideration, it can be assumed that the remainders for the first scanning positions of the transform block, usually located at high-frequency locations, are small, and they usually become greater towards low-frequency scanning positions. Additional configurations have to be determined besides the backward-adaptive estimation of the Rice parameter when replacing the EG0 code by Rice codes:

- **Initial Rice parameter:** For the first scanning position of the transform block, there is no preceding scanning position with a coded remainder. Given the investigations above, the initial Rice parameter should be $k = 0$, because most of the remainders have the shortest codeword length for $k = 0$. Another reason for the choice is that because the second coding phase usually starts at high-frequency locations, it is expected that the remainders are typically small, for which $k = 0$ provides the shortest codeword length.
- **Maximum Rice parameter:** The studies in the previous subsection indicated that $k > 3$ rarely occurs; therefore, the maximum Rice parameter should be $k = 3$ for the upcoming investigations.
- **Rice parameter reinitialization:** 4×4 sub-blocks are scanned from low-frequency to high-frequency locations within a transform block, whereas the scanning within each sub-block is from high-frequency to low-frequency locations for the second coding phase. Reinitializing the tracking variables used for the context modeling in the second coding phase was feasible, because the context modeling was originally designed using a scan from high-frequency to low-frequency locations. Because the Rice parameter selection is within the second coding phase, a reinitialization of the Rice parameter, e.g., to $k = 0$, should be considered in the upcoming investigations.
- **Rice parameter selection:** The selection of the Rice parameter should depend on the preceding scanning positions with a coded remainder (i.e., with an absolute value $x \geq t_0$), which is aligned with the tracking variables and the context modeling of the second coding phase. Two options have to be investigated for the Rice parameter update: Either allowing an unconstrained update of k or allowing k to increase only. Note that the latter design is comparable to the context modeling of the second coding phase, where the context model index $\delta_{x>2}$ never decreases.

Binarization with Rice Codes

With the analyses presented in this chapter, a starting point for the binarization with Rice codes can be established, where the cut-off threshold is $t_0 \in \{2, 3, 4\}$. Let us assume that there is a way to determine the Rice parameter k for each remainder value z to be coded depending on the preceding scanning position. Then, the static binarization process denoted in equation (4.1) changes as follows, where $f_R(\cdot, k)$ denotes the Rice code with Rice parameter k:

$$f(x) = f_{TRU}(x, t_0) * g_R(x - t_0, k), \tag{4.11}$$

where

$$g_R(z, k) = \begin{cases} (\emptyset), & \text{if } z < 0, \\ f_R(z, k), & \text{otherwise.} \end{cases} \tag{4.12}$$

Rice Parameter Selection

The BD-rates summarized in figure 4.10 also represent the improvements necessary for each t_0 configuration to achieve virtually the same coding efficiency as provided by the static binarization of AVC. Further encoding experiments were conducted to identify the remainder values of the preceding scanning position for which the Rice parameter should be changed, so that the coding efficiency is improved and the losses introduced by reducing the cut-off threshold t_0 are minimized.

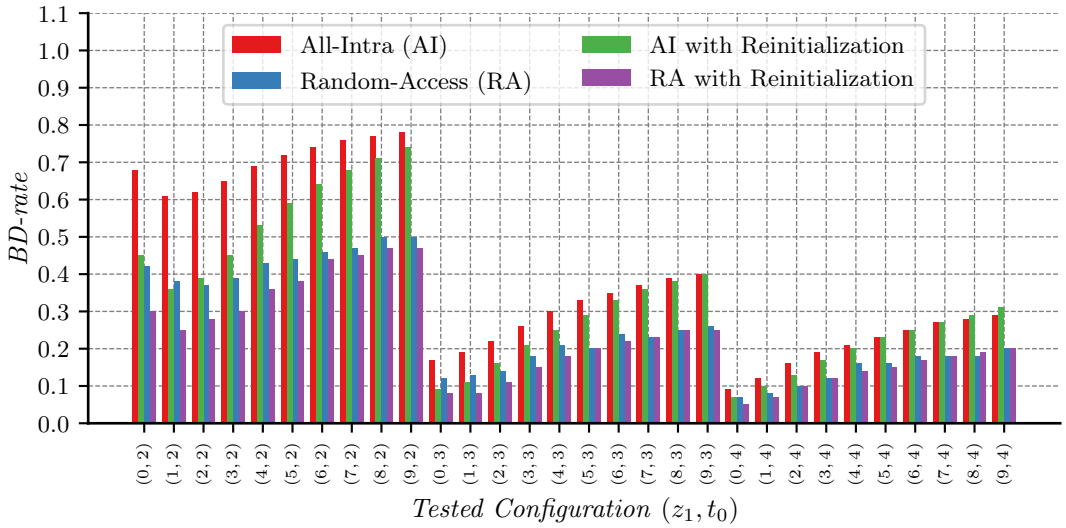

Figure 4.11
Coding efficiency of the investigation on the switch to $k = 1$ for different combinations of z_1 and t_0 for the QP set $\{22, 27, 32, 37\}$ (IMP4-2). The illustration includes the BD-rates for both variants, with and without reinitializing the Rice parameter to $k = 0$ at the beginning of each sub-block.
Anchor for BD-rate computations: IMP3-7*

Let z_{last} denote the remainder of the preceding scanning position, and let $z_{last} = 0$ for a scanning position without coded remainder, i.e., when $x < t_0$. Since it can be expected that, on average, the remainders increase from high-frequency to low-frequency locations, the following approach is investigated: The last remainder z_{last} is compared to a set of thresholds, and the Rice parameter used for coding the current remainder is selected based on the result of these comparisons. Algorithm 4.2 summarizes the implementation used for all investigations, where z_1, z_2, and z_3 denote the thresholds that have to be determined. Furthermore, i denotes the current scanning position, and k_{last} is the Rice parameter of the preceding scanning position. Note that i denotes the forward scanning position, whereas the second coding phase, where the Rice codes are employed, uses the reverse scanning pattern within a sub-block. Therefore, the preceding scanning position within a sub-block is $i + 1$, whereas the preceding scanning position for the first coding position of the second coding phase is $i - (i\&15) - 16$ for sub-blocks not covering the DC frequency position. Let the array containing the Rice parameters for the scanning positions $k[\cdot]$ further be initialized with zeros at the beginning of a transform block. Finally, the two boolean variables *reinit* and *constrained* denoted in algorithm 4.2 control the behavior of the update algorithm. When $reinit = true$, the Rice parameter k is set equal to zero for the first coding position of the second coding phase in a sub-block, whereas it is not reinitialized at the beginning of each sub-block when $reinit = false$. For $constrained = true$, the Rice parameter cannot decrease within a sub-block, whereas $constrained = false$ enables more freedom and k depends solely on z_{last}.

Switch to $k = 1$: The investigations started with the first threshold z_1 where the transition from the initial Rice parameter $k = 0$ to $k = 1$ should occur. The implementation for this investigation is referred to as IMP4-2, and the anchor used for calculating the BD-rates is IMP3-7*. In this investigation, k is set equal to one when the remainder of the preceding scanning position is greater than the threshold z_1, where each set of encoding experiments employed a different value for z_1. Because this investigation step determines z_1 only, the other two thresholds z_2 and z_3 were set to ∞. Both boolean variables denoted in algorithm 4.2 were set to *true* for all experiments of this set, meaning that $k = 0$ for the first coding position of the second coding phase in a sub-block, and k cannot be decreased within a sub-block. This investigation step further analyzed the effect of reinitializing the Rice parameter at the beginning of each sub-block. That was achieved by performing another set of experiments, where each experiment employed a different value for z_1 as before, but the boolean variable *reinit* was set to $false$ for all experiments of this set.

The experimental results for the QP set $\{22, 27, 32, 37\}$ are summarized in figure 4.11, where the x-axis

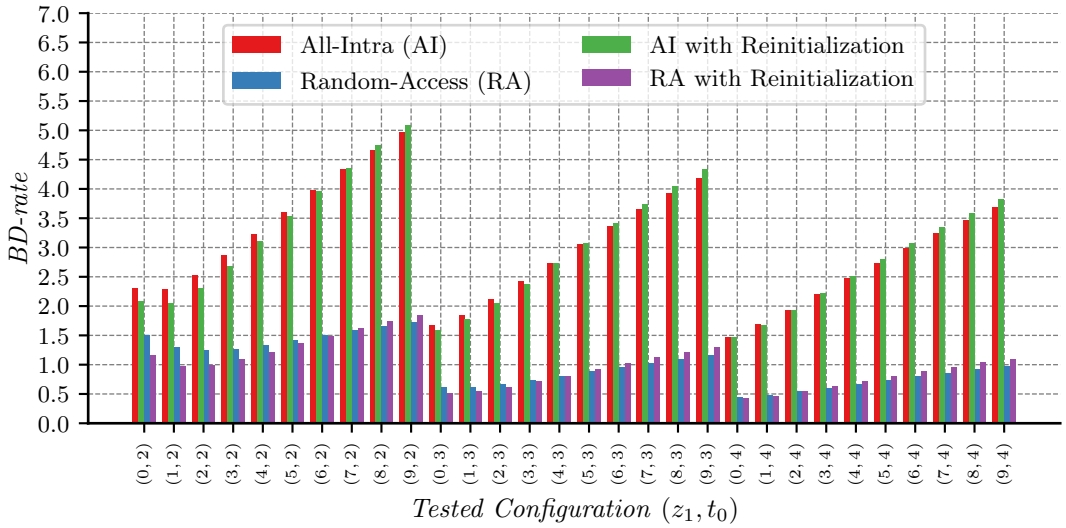

Figure 4.12
Coding efficiency of the investigation on the switch to $k = 1$ for different combinations of z_1 and t_0 for the QP set $\{2, 7, 12, 17\}$ (IMP4-2). The illustration includes the BD-rates for both variants, with and without reinitializing the Rice parameter to $k = 0$ at the beginning of each sub-block.
Anchor for BD-rate computations: IMP3-7*

denotes the configuration of the z_1 and t_0 combination and the y-axis denotes the BD-rate. Note that both variants, i.e., with and without reinitialization at the beginning of each sub-block, are included in figure 4.11. In all three cut-off configurations, the coding efficiency of the variant with Rice parameter reinitialization is superior compared to the corresponding variant without reinitialization. The experimental results show that the optimal remainder value z_1 depends on the configuration of t_0 and is equal to $z_1 = 1$ for $t_0 = 2$ and $z_1 = 0$ for $t_0 \in \{3, 4\}$.

The same set of experiments were performed for the QP set $\{2, 7, 12, 17\}$ representing the high bit-rate operation points, and figure 4.12 summarizes the obtained BD-rates. As observed for the QP set $\{22, 27, 32, 37\}$ in figure 4.11, the variant with Rice parameter reinitialization outperforms the variant without reinitialization. Furthermore, when the threshold z_1 is relatively high, the coding efficiency losses are significant, which was not observed for the QP set $\{22, 27, 32, 37\}$. For this set of experiments, the optimal remainder values z_1 are equal to those derived from the experiments with the QP set $\{22, 27, 32, 37\}$.

Besides the derivation of the optimal z_1, which depends on t_0, this investigation step proves that the Rice parameter reinitialization at the beginning of each sub-block is feasible. For the next investigation steps, the Rice parameter is always reinitialized to $k = 0$ at the first scanning position of each sub-block, i.e., $reinit = true$ for all further investigations. This implementation with $z_1 = 1$ for $t_0 = 2$ and $z_1 = 0$ for $t_0 \in \{3, 4\}$ is referred to as IMP4-2* and serves as the basis for the following implementations.

Switch to $k = 2$: For this investigation, z_3 is set to ∞, while *constrained* and *reinit* are set to *true*. The implementation for this investigation is referred to as IMP4-3 and is based on IMP4-2*. Each set of coding experiments used a different value for z_2 to determine the switch to $k = 2$. Note that the update rule denoted in algorithm 4.2 implies that the Rice parameter k can increase by more than one, i.e., from $k = 0$ to $k = 2$. In this investigation step, the effect for an unconstrained update of k was further analyzed by setting *constrained* to $false$, i.e., the Rice parameter for the current scanning position i can become smaller than the Rice parameter k of the preceding scanning position, implying that the derivation of k does not depend on k_{last}.

The experimental results are summarized in figure 4.13 for the QP set $\{22, 27, 32, 37\}$ Those BD-rate results in figure 4.13 show that the unconstrained update of the Rice parameter k is not feasible as its coding efficiency is inferior to its constrained counterpart. For this investigation step, the optimal thresholds are

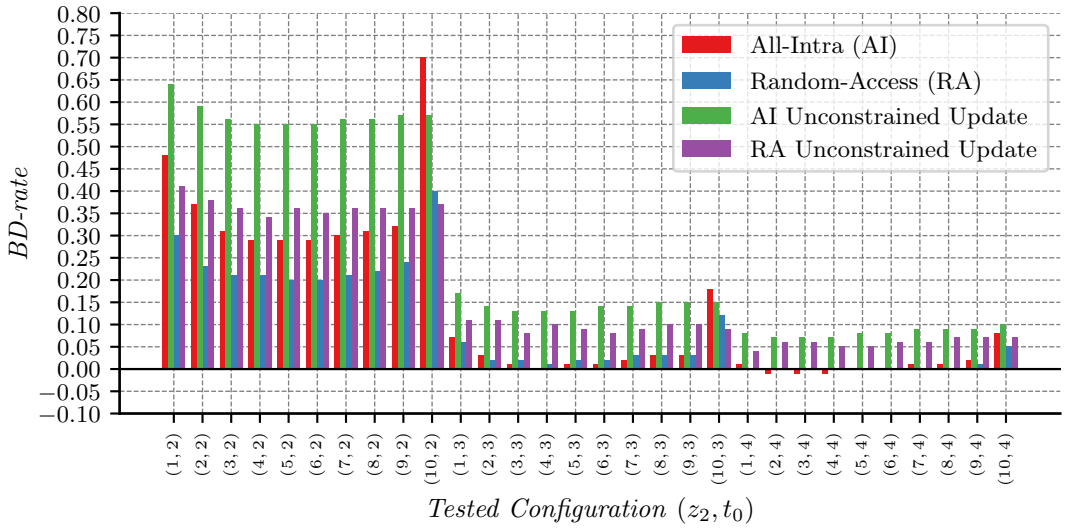

Figure 4.13
Coding efficiency of the investigation on the switch to $k = 2$ for different combinations of z_2 and t_0 for the QP set $\{22, 27, 32, 37\}$ (IMP4-3). The illustration includes the BD-rates for both the variant where the Rice parameter update is unconstrained, i.e., the Rice parameter is independent of k_{last}, and the variant where the Rice parameter is dependent on k_{last}.
Anchor for BD-rate computations: IMP3-7*

$z_2 = 5$ for $t_0 = 2$, $z_2 = 4$ for $t_0 = 3$, and $z_2 = 2$ for $t_0 = 4$. With those thresholds, the coding efficiency is further improved relative to the previous investigation step, where the maximum Rice parameter is $k = 1$. Note that the coding efficiency of the AVC binarization is already achieved for the cut-off configurations $t_0 \in \{3, 4\}$.

In figure 4.14, the experimental results are summarized for the QP set $\{2, 7, 12, 17\}$, where the performance of the unconstrainted update variant is also inferior to the constrained counterpart, as already observed for the QP set $\{22, 27, 32, 37\}$. For the high bit-rate operation points, the optimal thresholds are $z_2 = 4$ for $t_0 = 2$, $z_2 = 3$ for $t_0 = 3$, and $z_2 = 2$ for $t_0 = 4$. Except for $t_0 = 2$ in the *Random-Access* configuration, superior coding efficiency is achieved relative to the AVC binarization.

The experimental results show that the unconstrained update of the Rice parameter is not feasible, and the Rice parameter should never become smaller within a sub-block. This observation corresponds to the usage of the tracking variables c_1 and c_2, which are responsible for the context modeling of the second coding phase. Moreover, the existing coding efficiency provided by the AVC binarization can already be achieved for the CTC QPs when $t_0 \in \{3, 4\}$, whereas superior coding efficiencies are obtained for the high bit-rate operation points. The final configuration of IMP4-3 with z_2 equal to $z_2 = 5$ for $t_0 = 2$, $z_2 = 4$ for $t_0 = 3$, and $z_2 = 2$ for $t_0 = 2$, is referred to as IMP4-3* and serves as the basis for the following implementations.

Switch to $k = 3$: The same investigation was performed for the final switch to $k = 3$. The corresponding implementation is referred to as IMP4-4, and it is based on IMP4-3*. Again, the anchor used to calculate the BD-rates is IMP3-7*. Note that for z_1 and z_2, the values determined by previous investigations were used, and both variables *constrained* and *reinit* are set equal to *true*. Figure 4.15 summarizes the experimental results for both QP sets. For the CTC QPs, the coding efficiencies converge towards a limit, i.e., insignificant improvements relative to the coding efficiency provided by the previous investigation are obtained. This observation also means that for the cut-off configuration $t_0 = 2$, the coding efficiency remains inferior relative to the AVC binarization. That, in turn, excludes the cut-off configuration $t_0 = 2$ from further consideration as a final cut-off configuration.

For the high bit-rate operation points, both candidate cut-off configurations ($t_0 \in \{3, 4\}$) provide superior coding efficiencies relative to the AVC binarization with a further improvement relative to the previous

Algorithm 4.2 Generic implementation for the investigations of the switching points to different Rice parameters, where i denotes the current scanning position, i_{last} the last significant scanning position, z_1, z_2, and z_3 are thresholds for selecting the Rice parameter. The boolean variable $reinit = true$ enables the reinitialization of the Rice parameter at the beginning of each sub-block, whereas the variable $constrained = true$ disables an unconstrained update of the Rice parameter k.

Require: z_1, z_2, z_3, *reinit*, *constrained*
if $i = i_{last} \vee i \bmod 16 = 15$ **then**
  if $reinit \vee i < 16$ **then**
    $k_{last} = 0$
    $z_{last} = 0$
  else
    $k_{last} = k\,[i - (i\&15) - 16]$
    $z_{last} = z\,[i - (i\&15) - 16]$
  end if
else
  $k_{last} \leftarrow k\,[i + 1]$
  $z_{last} \leftarrow z\,[i + 1]$
end if
if $z_{last} > z_3$ **then**
  $k\,[i] \leftarrow 3$
else if $z_{last} > z_2$ **then**
  $k\,[i] \leftarrow 2$
else if $z_{last} > z_1$ **then**
  $k\,[i] \leftarrow 1$
else
  $k\,[i] \leftarrow 0$
end if
if *constrained* **then**
  $k\,[i] = \max\left(k\,[i]\,, k_{last}\right)$
end if

investigation. The best BD-rate in the *All-Intra* configuration is -2.03% obtained by a cut-off threshold of $t_0 = 3$, which is better than a BD-rate of -1.95% for $t_0 = 4$, wheras the best BD-rate in the *Random-Access* configuration in the $t_0 = 4$ configuration is with -0.29% slightly better than for $t_0 = 3$ with -0.25%.

Final update rules of the Rice parameter: As a consequence of the above experimental results, the final cut-off configuration $t_0 = 3$ was selected, because it provides roughly the same coding efficiency for the QP set $\{22, 27, 32, 37\}$ as $t_0 = 4$ and the AVC binarization, but it avoids the usage of the same context model for consecutive bins. Furthermore, the configuration $t_0 = 3$ provides the best coding efficiency for the QP set $\{2, 7, 12, 17\}$ in the *All-Intra* configuration, and almost the same coding efficiency in the *Random-Access* configuration as $t_0 = 4$. The final configuration of IMP4-4 with $t_0 = 3$ and $z_3 = 10$ is referred to as IMP4-4*, and it is summarized as pseudo-code in algorithm 4.3.

4.3.5 | Nested Rice Codes with EG0

Transform coefficient levels are commonly specified to be 16 bits and the range is $[-2^{15}, 2^{15} - 1] \cap \mathbb{N}$, which yields a maximum absolute transform coefficient level of $x_{max} = 2^{15}$. Let $\ell_{EG0}\,(\cdot)$ denote the codeword length when using the EG0 code, and let $\ell_R\,(\cdot, k)$ denote the corresponding codeword length when using the Rice code k. For the AVC binarization ($t_0 = 15$) and the previously established adaptive binarization with Rice codes ($t_0 = 3$), the maximum codeword length, i.e., the maximum number of the bypass-coded bins for x, are:

$$\begin{aligned} \ell_{EG0}\,(x_{max} - 15) &= 29, \\ \ell_R\,(x_{max} - 3, 0) &= 32766, \\ \ell_R\,(x_{max} - 3, 1) &= 16384, \\ \ell_R\,(x_{max} - 3, 2) &= 8194, \\ \ell_R\,(x_{max} - 3, 3) &= 4099. \end{aligned} \tag{4.13}$$

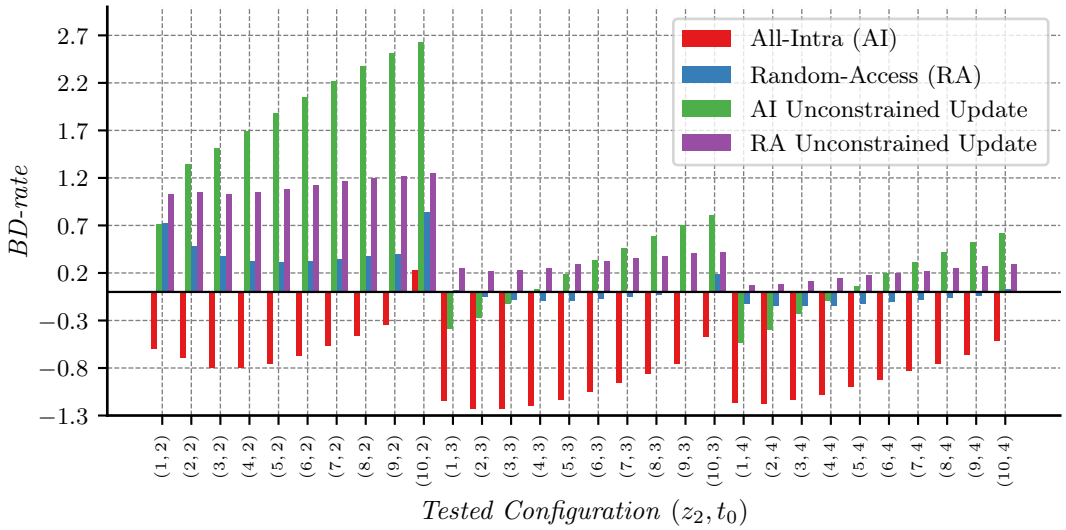

Figure 4.14
Coding efficiency of the investigation on the switch to $k = 2$ for different combinations of z_2 and t_0 for the QP set $\{2, 7, 12, 17\}$ (IMP4-3). The illustration includes the BD-rates for both the variant where the Rice parameter update is unconstrained, i.e., the Rice parameter is independent of k_{last}, and the variant where the Rice parameter is dependent on k_{last}, respectively.
Anchor for BD-rate computations: IMP3-7*

For remainders close to x_{max}, the codeword length of Rice codes is significantly larger than the codeword length of the EG0 code used in the AVC binarization. Although bypass-coded bins require fewer resources for processing, the significant increase in the bin string length relative to the AVC binarization counteracts the achievement of the context-coded bins reduction.

Because the EG0 code provides a significantly shorter codeword length for values close to x_{max}, the pursued solution is to reintroduce the EG0 code into the binarization process. The Rice codes are nested in between the TRU code and the EG0 code to combine the achievements provided by Rice codes, i.e., the coding efficiency improvements for the high bit-rate operation points and the reduction of context-coded bins, with the shorter codeword length for values close to x_{max} of the EG0 code.

Binarization with Rice and EG0 Codes

Figure 4.16 visualizes the concept on the bin string of transform coefficient levels, as already used in figure 4.1 to visualize the bin strings of the static binarization process in AVC. Embedding Rice codes between the TRU and EG0 code introduces a further parameter that has to be determined: the transition point t_1 that specifies the maximum remainder value $x - 3$ to be represented with Rice codes. The static binarization process expressed as a formula in equation (4.1) is changed to the following notation, where $g_R(\cdot, k, \text{maxVal})$ denotes the Rice code with maxVal being the maximum value that can be coded with a Rice code, and $f_{binary}(q, m)$ the m least significant bits of q:

$$f(x) = f_{TRU}(x, 3) * g_R(x - 3, k, t_1) * g_{EG0}(x - 3 - t_1), \tag{4.14}$$

where,

$$g_R(x, k, \text{maxVal}) = \begin{cases} \emptyset, & \text{if } x < 0, \\ f_{TRU}(x \gg k, \text{maxVal} \gg k) * f_{binary}(x, k), & \text{otherwise.} \end{cases} \tag{4.15}$$

Note that a truncation of the codeword for the maximum value of Rice codes is applicable to save unnecessary signaling, where the maximum value for the remainder is equal to t_1. This aspect is not analyzed in detail, because its impact on the coding efficiency is insignificant. A truncation approach is selecting the threshold t_1 equal to $n \cdot 2^k - 1$, with $n > 0$ being an integer, so that the last bucket is complete, and using the TRU

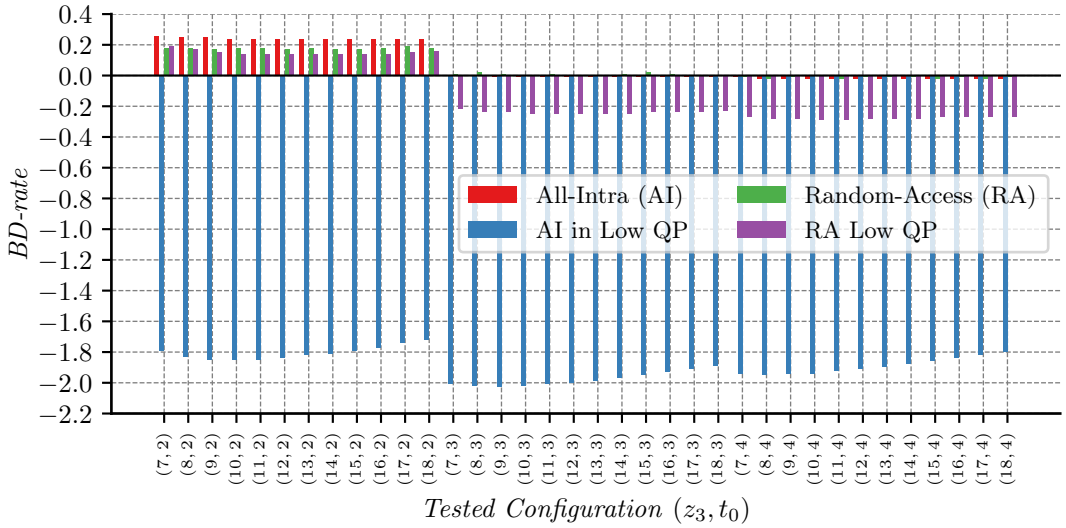

Figure 4.15
Coding efficiency of the investigation on the switch to $k = 3$ for different combinations of z_3 and t_0 for both QP sets (IMP4-4).
Anchor for BD-rate computations: IMP3-7*

code in the prefix instead of the unary code. This truncation is applied later after determining the maximum remainder for the Rice codes.

Maximum Remainder for Rice Codes

The first investigation in this context is to determine whether a single t_1 configuration for all Rice codes is feasible. The implementation for this investigation is based on IMP4-4* and is referred to as IMP4-5. A set of coding experiments using IMP4-5 was conducted, where each experiment employed a different value for t_1. Specifically, only $z_R = \min(z, t_1)$ is binarized using the specified Rice code, whereas $z_{EG0} = z - t_1$ is binarized using the EG0 code. For calculating the BD-rates, again IMP3-7* was used. In contrast to the previously performed experiments, simulations were conducted for the QP set $\{2, 7, 12, 17\}$ only, because it is expected that a change in coding efficiency is only measurable for the higher bit-rate operation points. Figure 4.17 summarizes the BD-rates obtained after performing the encoding experiments. The horizontal solid line denotes the coding efficiency for the variant without the EG0 code (IMP4-4*) in the *All-Intra* configuration, and the dashed horizontal line the corresponding coding efficiency in the *Random-Access* configuration. With an increased value of t_1, the coding efficiency of the nested Rice codes converges towards the corresponding

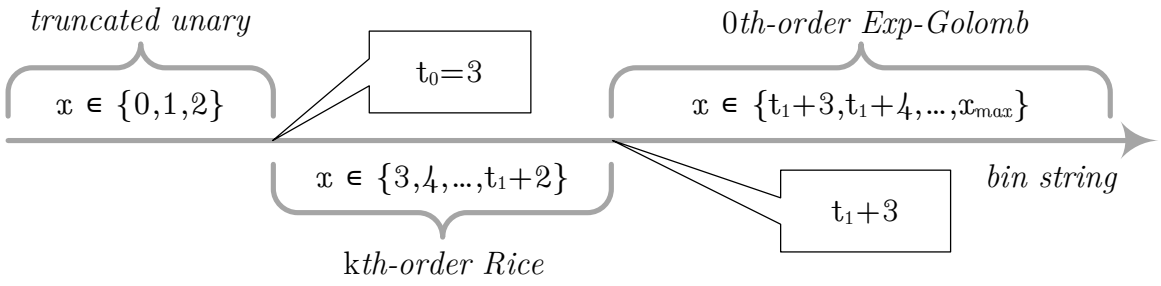

Figure 4.16
Binarization of transform coefficient levels with nested Rice codes and the EG0 code. For $x \in \{0, 1, 2\}$, the bin string consists of a TRU code only, whereas for $x \in \{3, 4, \ldots, t_1 + 2\}$, the bin string consists of the TRU code for $x = 3$ and a Rice code for $x - 3$. When $x \geq t_1 + 3$, the bin string consists of a TRU code for $x = 3$, which is followed by the Rice code for $x - 3$ and the EG0 code for $x - t_1 - 3$.

Algorithm 4.3 IMP4-4*: Implementation of the final Rice parameter selection, where a reinitialization of the Rice parameter to $k = 0$ at the beginning of each sub-block and the constraint that k cannot decrease within a sub-block are always enabled.

```
if i = i_last ∨ i mod 16 = 15 then
    k_last = 0
    z_last = 0
else
    k_last ← k[i + 1]
    z_last ← z[i + 1]
end if
if z_last > 10 then
    k[i] ← 3
else if z_last > 4 then
    k[i] ← 2
else if z_last > 0 then
    k[i] ← 1
else
    k[i] ← 0
end if
k[i] = max(k[i], k_last)
```

variant without the EG0 code. However, encoding experiments beyond a configuration of $t_1 \geq 40$ were not conducted for practical reasons and because the maximum codeword length would become significantly larger for a coding efficiency that is virtually the same as for the variant without the EG0 code.

The experimental results of this investigation showed that a single $t1$ configuration for all Rice codes is not feasible. Instead, the next investigation determined t_1 for each Rice parameter k, because the maximum codeword length highly depends on the Rice code. A set of encoding experiments was conducted to determine the best value for $t_1\,(k = 0)$ by selecting the configuration that provides a coding efficiency close to or equal to the variant without the EG0 code. When multiple values are possible, the lowest value for $t_1\,(k = 0)$ is selected, as the value directly controls the final bin string length. After the determination of the configuration for $t_1\,(k = 0)$, new encoding experiments were conducted to identify the optimal value for $t_1\,(k = 1)$, which is then followed by the determination of the optimal value for $t_1\,(k = 2)$, and finally for $t_1\,(k = 3)$. The corresponding implementation is based on IMP4-4*, and it is referred to as IMP4-6, and the anchor for BD-rate computations is again IMP3-7*.

Figure 4.18 summarizes the experimental results for all steps of this investigation. The main observation is that for $k \in \{0, 1\}$, the value of t_1 can be chosen significantly smaller than the values of t_1 for $k \in \{2, 3\}$. Furthermore, nesting the Rice code can insignificantly improve the coding efficiency for $k \in \{0, 1, 2\}$, but it is measurable and visibile in figure 4.18. For $k = 3$, it is expected that an improved coding efficiency relative to the variant without the EG0 code can be achieved by setting $t_1\,(k = 3)$ to a very large value. The finally determined $t_1\,(k)$ values are:

$$t_1\,(k) = \begin{cases} 9, & \text{if } k = 0, \\ 23, & \text{if } k = 1, \\ 47, & \text{if } k = 2, \\ 95, & \text{if } k = 3. \end{cases} \tag{4.16}$$

Note that the final values for $t_1\,(k)$ were selected according to the condition $t_1\,(k) = n \cdot 2^k - 1$ so that the TRU code can be used for the prefix of the Rice codes. After determining $t_1\,(k)$, which depends on the Rice parameter k, the maximum codeword length for a remainder is:

$$\begin{aligned} &\ell_R\,(x_{max} - 3, 0) + \ell_{EG0}\,(x_{max} - 3 - t_1\,(k = 0)) - 1 = 38, \\ &\ell_R\,(x_{max} - 3, 1) + \ell_{EG0}\,(x_{max} - 3 - t_1\,(k = 1)) - 1 = 41, \\ &\ell_R\,(x_{max} - 3, 2) + \ell_{EG0}\,(x_{max} - 3 - t_1\,(k = 2)) - 1 = 42, \\ &\ell_R\,(x_{max} - 3, 3) + \ell_{EG0}\,(x_{max} - 3 - t_1\,(k = 3)) - 1 = 42. \end{aligned} \tag{4.17}$$

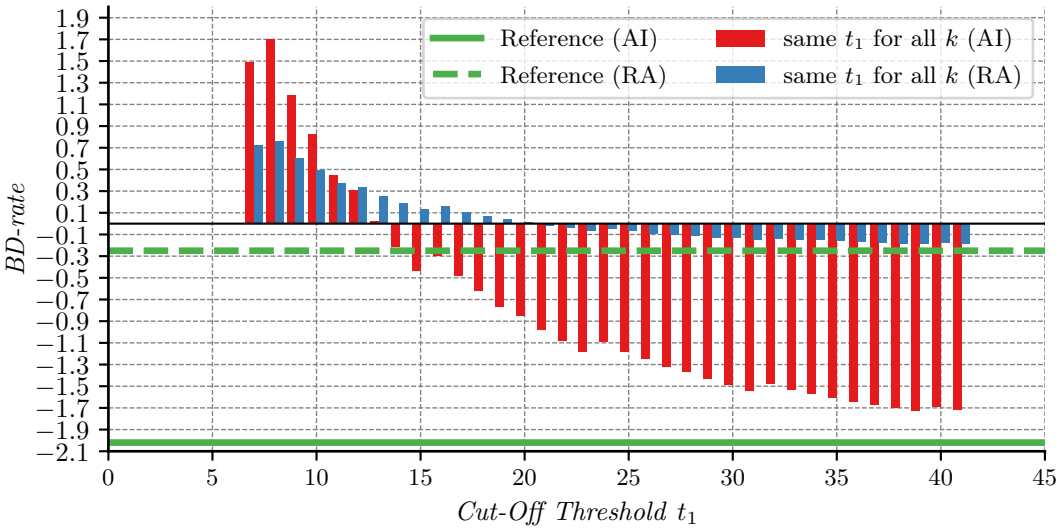

Figure 4.17
Coding efficiency of the investigation on fixed t_1 configurations for the nested Rice codes with the QP set $\{2, 7, 12, 17\}$ (IMP4-5). The solid horizontal line denotes the coding efficiency of the variant without the EG0 code in the *All-Intra* configuration, and the dashed horizontal line denotes the *Random-Access* configuration.
Anchor for BD-rate computations: IMP3-7*

4.3.6 | Final Design with Nested Rice Codes

The final implementation of the adaptive binarization with nested Rice codes provides about the same coding efficiency as achieved by the binarization in AVC for transform coefficient levels. However, it achieves the coding efficiency with a reduction of the cps for transform coefficient levels by 80% in the worst-case scenario. That reduction of the worst-case cps significantly alleviates hardware implementations. Furthermore, the developed binarization achieves a superior coding efficiency relative to the AVC binarization of transform coefficient levels for high and very-high bit-rates.

Table 4.2 summarizes the coding efficiency for the final implementation of the adaptive binarization with nested Rice codes. Its implementation is based on IMP4-6 with the configuration of t_1 denoted in equa-

Class	Luma	C_B	C_R
	All-Intra		
A	-0.01 (-2.57)%	-0.08 (-2.54)%	-0.07 (-2.52)%
B	-0.03 (-0.84)%	-0.05 (-0.82)%	-0.04 (-0.86)%
C	0.02 (-2.62)%	0.01 (-2.62)%	0.01 (-2.70)%
D	-0.03 (-3.87)%	-0.05 (-3.89)%	-0.07 (-4.03)%
E	-0.01 (-0.19)%	-0.06 (-0.23)%	-0.10 (-0.23)%
Overall	**-0.01 (-2.05)%**	**-0.05 (-2.05)%**	**-0.05 (-2.10)%**
	Random-Access		
A	0.03 (-0.41)%	0.12 (-0.40)%	-0.08 (-0.35)%
B	-0.01 (-0.02)%	-0.04 (-0.01)%	-0.17 (0.00)%
C	-0.01 (-0.38)%	-0.01 (-0.36)%	0.10 (-0.36)%
D	-0.04 (-0.54)%	-0.17 (-0.55)%	-0.18 (-0.51)%
E	0.05 (0.06)%	-0.16 (0.04)%	0.18 (0.08)%
Overall	**0.00 (-0.26)%**	**-0.05 (-0.26)%**	**-0.05 (-0.23)%**

Table 4.2
Coding efficiency of the final adaptive binarization with nested Rice codes with the implementation is referred to as IMP4-6*. BD-rates not in brackets denote the coding efficiency for the QP set $\{22, 27, 32, 37\}$. BD-rates in brackets denote the coding efficiency for the QP set $\{2, 7, 12, 17\}$.
Anchor for BD-rate computations: IMP3-7* (IMP3-7*)

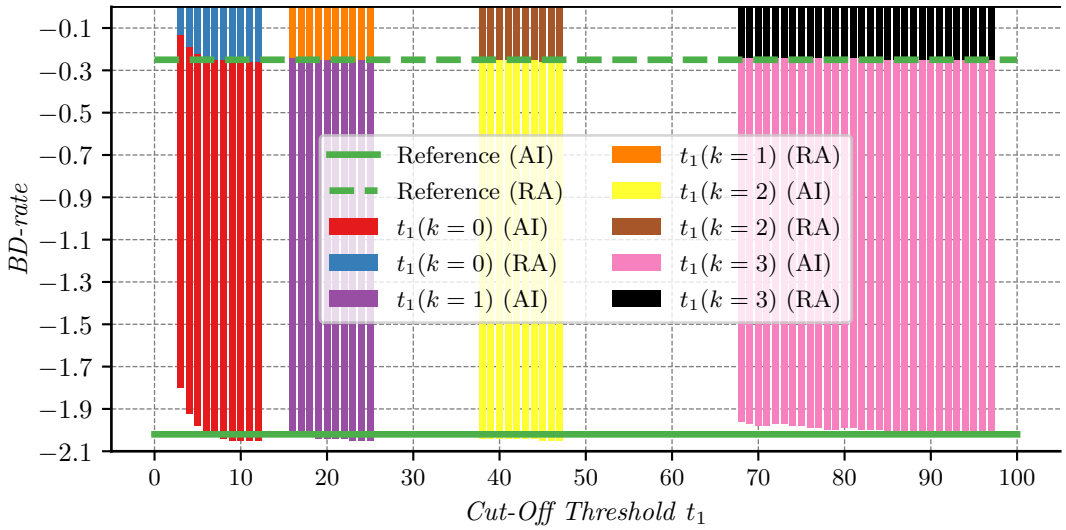

Figure 4.18
Coding efficiency of the investigation on variable $t_1\,(k)$ configurations for the nested Rice codes with the QP set $\{2, 7, 12, 17\}$ (IMP4-6). The solid horizontal line denotes the coding efficiency of the variant without the EG0 code in the *All-Intra* configuration, and the dashed horizontal line denotes the *Random-Access* configuration, respectively.
Anchor for BD-rate computations: IMP3-7*

tion (4.16), and this implementation is referred to as IMP4-6*. Values not in brackets denote the BD-rates for the CTC QPs, whereas values in brackets denote the BD-rates for the QP set $\{2, 7, 12, 17\}$, which represent high bit-rate operation points. The anchor for the BD-rates calculation is the 4×4 sub-block processing developed in the chapter chapter 3, i.e., IMP3-7*. Finally, algorithm 4.4 summarizes IMP4-6* as pseudo-code, which is an extended version of algorithm 3.6.

4.4 | Findings and Technical Achievements

The findings and technical achievements of the adaptive binarization for transform coefficient levels with nested Rice codes are:

- The *context-coded bins per sample* (cps) of transform coefficient levels dominate the overall cps when using the AVC binarization. It is, therefore, sufficient to reduce the number of context-coded bins for transform coefficient levels to reduce the overall cps.
- Remainders of absolute transform coefficient levels can be modeled as being geometrically distributed. This observation can be exploited by Golomb and Rice codes, which are optimal for geometrical probability distributions.
- A reduction of the cut-off threshold that defines the transition from context-coded bins to bypass-coded bins can be achieved without sacrificing the coding efficiency for the QPs specified in the CTC, when using adaptive Rice codes.
- The developed adaptive Rice code selection provided coding efficiency improvements for high and very high bit-rate operation points.
- The cut-off threshold could be reduced to three, for which the usage of the same context model for successive bins of a bin string could be avoided.

The developed adaptive binarization for transform coefficient levels with Rice codes was successfully proposed [79] to be included in the HEVC standard. The VVC standard, which is the successor of HEVC, also specifies an adaptive binarization with Rice codes for absolute transform coefficient levels. Consequently, the adaptive binarization for transform coefficient levels plays an important role in practical applications, because it made

the CABAC entropy coding scheme more implementation friendly in both the HEVC and the VVC standard. Further refinements were included during the development of HEVC, which further eased the implementation efforts, but the basic design presented in this chapter and first proposed by the author in [79] remained untouched. In both video coding standards, Rice codes are embedded between the TRU code and the EG0 code as proposed in [79]. The backward-adaptive determination of the Rice parameter, the maximum Rice parameter equal to three, and the transition to EG0 depending on the current Rice parameter can be found in both video coding standards.

In contrast to its predecessor AVC, the HEVC standard specifies CABAC as the only entropy coding scheme, which in turn requires an essential reduction of the number of context-coded bins to make HEVC feasible for hardware implementations. That reduction was mainly achieved by adopting the adaptive binarization of transform coefficient levels with Rice codes, as proposed in [79]. Further modifications were applied to the parameters of the design, such as the values of the thresholds z_1, z_2 and z_3, and the transition cut-off t_1. A more detailed description of the design specified in the first version of the HEVC standard can be found in [62].

The successor of the HEVC standard is VVC. It initially employed the same binarization for transform coefficient levels as used in HEVC at the beginning of its development. Further adjustments to the Rice parameter selection were applied to comply with the modified coding order and context modeling. Instead of determining the Rice parameter based on the preceding scanning position only, VVC employs a local template that evaluates the already reconstructed neighboring scanning positions [71]. That adjustment is an integral part of the modified context modeling to achieve a more efficient coding efficiency, which is the topic of chapter 5.

4.5 | Chapter Summary

This chapter described the problems that exist for the static binarization of transform coefficient levels in AVC in hardware implementations, where the cps of transform coefficient levels dominate the overall cps. An investigation presented in this chapter showed that the cut-off threshold t_0 of the AVC binarization, which defines the transition from context-coded bins to bypass-coded bins, could be lowered with a little penalty on the coding efficiency. However, the reduction of $t_0 = 15$ to $t_0 = 6$ is not sufficient, because the cps of transform coefficient levels still dominates the overall cps, and the usage of the same context model for successive context-coded bins is still necessary. When using the zero-mean Laplacian distribution as a probability model for transform coefficients and the quantization described in this chapter, the conditional probability model for the remainders is geometric. With this knowledge, the remainders can be binarized with Rice codes, a subset of Golomb codes, where further investigation results presented in this chapter proved the feasibility of the approach. Particularly, a Rice code is determined backward-adaptively for each scanning position by evaluating the remainder of the preceding scanning position. The final Rice parameter update rule achieves virtually the same coding efficiency as provided by the static binarization scheme in AVC with a maximum cps equal to 4.5 instead of 22.5. Finally, the Rice codes were nested between the TRU code and the EG0 code to limit the bin string length without negatively impacting the coding efficiency.

Algorithm 4.4 Pseudo-code of the final implementation for the adaptive binarization with nested Rice codes, which is implemented on top of the level coding with 4×4 sub-blocks.

```
...
while i_last < 0 do
    ...
    k[i_0 + m + 1] ← 0
    while m ≥ 0 do
        k[i_0 + m] ← k[i_0 + m + 1]
        if b_sig[i_0 + m] = 1 then
            δ_{x>1}(i_0 + m) ← 5n_2 + min(c_1[i_0 + m], 4)
            transmit b_{|x|>1}[i_0 + m] using C_{x>1}[δ_{x>1}(i_0 + m)]
            if b_{|x|>1}[i_0 + m] = 1 then
                δ_{x>2}(i_0 + m) ← 5n_2 + min(c_2[i_0 + m], 4), j ← 1
                transmit b_{|x|>2}[i_0 + m] using C_{x>2}[δ_{x>2}(i_0 + m)]
                if b_{|x|>2}[i_0 + m] = 1 then
                    z ← |x[i_0 + m]| − 3
                    if k[i_0 + m] = 0 then
                        z_max ← 9
                    else if k[i_0 + m] = 1 then
                        z_max ← 23
                    else if k[i_0 + m] = 2 then
                        z_max ← 47
                    else
                        z_max ← 95
                    end if
                    z_R ← min(z, z_max)
                    transmit z_R ≫ k[i_0 + m] in bypass mode using TRU
                      with maxVal = z_max ≫ k[i_0 + m]
                    transmit k[i_0 + m] least significant bits of z_R in bypass mode
                    if z ≥ z_max then
                        transmit z − z_max in bypass mode using EG0
                    end if
                    if z_R > 10 then
                        k[i_0 + m] ← 3
                    else if z_R > 4 then
                        k[i_0 + m] ← 2
                    else if z_R > 0 then
                        k[i_0 + m] ← 1
                    else
                        k[i_0 + m] ← 0
                    end if
                    k[i_0 + m] ← max(k[i_0 + m], k[i_0 + m + 1])
                end if
                ...
            else if c_1(i_0 + m) ≠ 0 then
                ...
            end if
            transmit b_sign[i_0 + m] in bypass mode
        end if
        ...
    end while
    ...
end while
```

The presented adaptive binarization with nested Rice codes, where the Rice parameter is derived backward-adaptively for each scanning position, can be found in practical video coding standards. Both the HEVC standard and its successor, the VVC standard, employ the adaptive binarization presented in this chapter for transform coefficient levels.

CHAPTER 5

Template-Based Context Modeling

Contents

Variable transform block sizes initiated the development of the 4×4 sub-block processing presented in chapter 3, and implementation feasibility of the Rice codes in chapter 4. Coding efficiency improvement is the motivation for developing the level coding presented in this chapter, which utilizes an advanced mapping of the already coded transform coefficient levels inside the transform block to context model offsets and the Rice parameter.

The final version of the level coding developed in this chapter implements a single coding phase combined with evaluating the already coded transform coefficient levels of neighboring frequency locations for context modeling. This concept is referred to as template-based context modeling. Its development used the design developed in chapter 3 and chapter 4 as the basis and started with a template-based context modeling for the b_{sig} flags. Instead of relying mainly on the frequency location of the current scanning position for the context model offset, the b_{sig} flags located at neighboring frequency locations are evaluated for context model selection in the first development. In the second development, the two coding phases of the level coding are combined into a single coding phase, i.e., the level magnitudes are coded completely for each frequency location, instead of transmitting the level magnitudes within a sub-block partially in multiple iterations. With that single coding phase, the template-based context modeling can be extended in a way that not only b_{sig} flags are utilized for the context modeling, but absolute transform coefficient levels, which yields further improvement in coding efficiency.

From the viewpoint of the context modeling task (see section 2.2), modifying the context modeling of the b_{sig} flags by utilizing a template can be described as evaluating different input parameters of the conditional function $C(\cdot)$. Employing a single coding phase and utilizing level magnitudes for context modeling represents an extension of the set of input parameters of $C(\cdot)$.

The presented level coding using template-based context modeling and a single coding phase achieves a substantial coding efficiency improvement. That, in turn, makes the design the anchor point for further developments, which finally results in the level coding in the *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC) standard [64, 49].

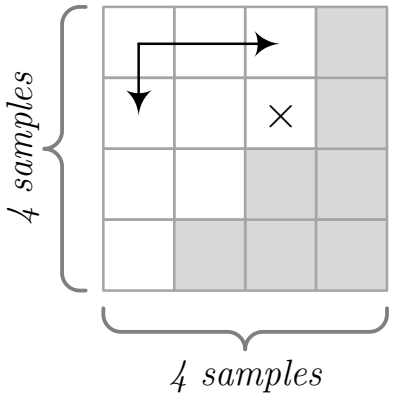

Figure 5.1
Example of the last significant scanning position signaling for a 4×4 transform block using the $x_{last} = 2$ coordinate, which is signaled first, and the $y_{last} = 1$ coordinate. With both coordinates, the last significant scanning position, marked by an ×, is uniquely determined.

5.1 | Problem Statement

In *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC), the level coding, based on the design described in chapter 3 and chapter 4, provided a reasonable trade-off between coding efficiency and implementation feasibility at the given time. With the development of its successor, the VVC standard, more complexity was accepted, provided the achieved coding efficiency improvement would be significant enough. Evaluating partially reconstructed level information of already coded neighboring locations with a template for each scanning position can improve the coding efficiency, because the corresponding conditional $C(\cdot)$ may provide probability estimates that generate a shorter average codeword length (see section 2.2.8). The challenge in this approach is to find a suitable mapping of the available data about the b_{sig} flags or absolute transform coefficient levels (when they can be utilized for context modeling by coding in a single coding phase) located in the vicinity of the current scanning position to context model offsets.

5.2 | Extra Coding Tools Inherited from HEVC

From this chapter onwards, the software basis for coding experiments is the VVC reference software implementation version 17 (VTM-17). The initial VVC development started with the level coding of its predecessor, the HEVC standard. The level coding in HEVC is based on 4×4 sub-block processing, but includes three additional coding tools for implementation feasibility. A brief review of these coding tools is given in the following subsections.

5.2.1 | Last Significant Scanning Position

In *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC), a b_{last} flag, which denotes whether the current non-zero (or significant) scanning position is the last non-zero scanning position, is transmitted immediately after the coding of $b_{sig} = 1$, as described in section 3.2.1. Its signaling is, therefore, interleaved with the signaling of the b_{sig} flags, which allows an efficient representation of the zero-valued (or insignificant) area within a transform block. This concept can become problematic for decoders implementing speculative processing, where all possible context models that could be selected in the next few coding steps are precalculated. It is more feasible for such implementations when only the context model selection of b_{sig} has to be considered.

For this reason, the technique that replaced the interleaved approach in HEVC signals the last non-zero scanning position as absolute offsets relative to the top-left corner of the transform block [29] when using the forward scanning pattern. The last position is coded before transmitting any transform coefficient levels of the transform block. Figure 5.1 illustrates an example for a 4×4 transform block. The x-coordinate is signaled first and is followed by the y-coordinate. This concept achieves virtually the same coding efficiency as the interleaving method. With the introduction of the modified last significant scanning position signaling, the first coding phase, where the b_{sig} flags are transmitted, is modified to employ the reverse scanning pattern, the same as used in the second coding phase (see section 3.2.1). As for the design presented in the previous chapter, transform coefficient levels are transmitted completely for each 4×4 sub-block before moving on to the next sub-block in reverse scanning order.

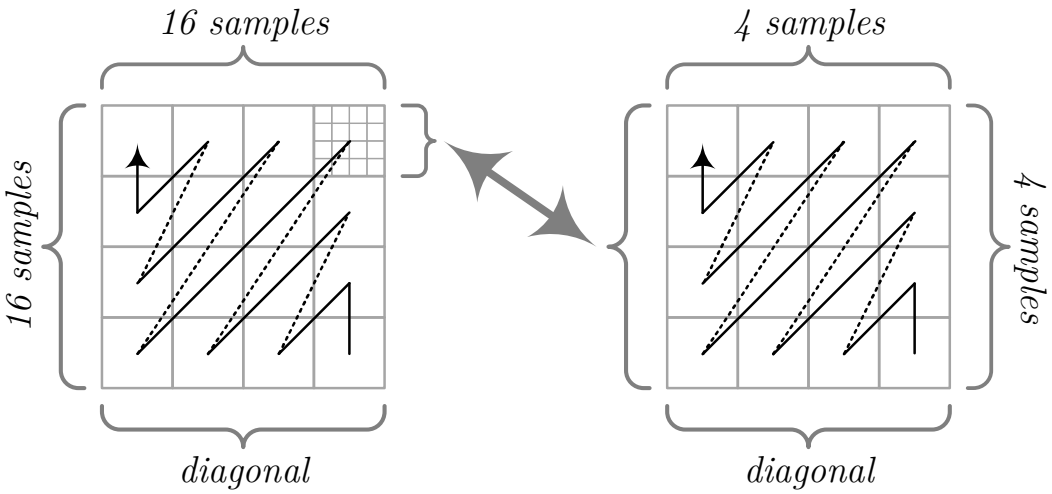

Figure 5.2
Reverse diagonal scanning pattern when applied to the whole transform block, exemplarily for 4×4 on the right, and larger block sizes with 4×4 sub-block processing on the left. The diagonal scanning patterns can be derived from the zigzag scanning pattern by keeping the scanning direction for the diagonals within sub-blocks and of sub-block diagonals constant.

5.2.2 | Diagonal Scanning Pattern

The previously used zigzag scanning pattern within the 4×4 sub-block processing alters the direction for each diagonal within a sub-block, illustrated on the right in figure 3.5, and for each sub-block diagonal, illustrated on the left in figure 3.5. In HEVC, the scan direction for diagonals within a sub-block and of sub-block diagonals is always from bottom-left to top-right [80] (forward scanning pattern). This so-called diagonal scanning pattern is illustrated in figure 5.2 for the 4×4 sub-block processing, which utilizes the reverse variant only. Its advantage is a uniform memory access order, which limits the addressing overhead for hardware implementations.

5.2.3 | Coded Sub-Block Flags

For large transform block sizes and low bit-rate operation points, there may be a vast amount of 4×4 sub-blocks along the scanning path containing only zero-valued frequency locations. The coded sub-block flag (b_{csf}) was introduced to take advantage of such situations and is usually transmitted for each 4×4 sub-block [30]. Its definition is comparable to the coded block flag for a transform block, but applied to 4×4 sub-blocks. For $b_{csf} = 0$, the sub-block contains zero-valued transform coefficient levels only, while for $b_{csf} = 1$, at least one frequency location within the sub-block consists of a non-zero transform coefficient level. In the latter case, the value of the absolute level at the last coding position can be inferred to be non-zero when all previous coding positions within the sub-block have a zero-valued transform coefficient level. There are two cases where b_{csf} is not coded: for the 4×4 sub-block containing the DC frequency position and for the 4×4 sub-block containing the last significant scanning position. For the 4×4 sub-block containing the DC frequency position, all frequency locations are coded as if the b_{csf} syntax element is nonexistent. When the initially mentioned assumption does not hold, i.e., when each sub-block along the scanning path contains at least one non-zero frequency location, the coding of b_{csf} is very efficient due to adaptive context models.

5.2.4 | Reference Implementation and Experimental Setup

The software basis for the investigations conducted in this and the following chapters is the VVC reference implementation version 17.0 (VTM-17). For this chapter, the level coding implementation of VTM-17 is replaced by the design implemented in IMP4-6*, presented in chapter 4. Furthermore, the described extra coding tools inherited from HEVC were integrated into that implementation, and the resulting implementation is referred to as IMP5-0. In the first coding phase of this implementation, the context modeling of b_{sig} relies on the frequency location and adaptive context model sets, introduced in section 3.3.3.

For all conducted coding experiments, the coding performance was evaluated by measuring the *Bjøntegaard delta bit-rate* (BD-rate) [66] for the luma component between two codec versions, where the encoder configurations mainly follow the VVC *common test conditions* (CTC) [81]. In the coding experiments for the analyses, all adaptive context models used for level coding were initialized as *equi-probable* (EP) to avoid any interference from initial probabilities, and only the first second of each test sequence was coded. Furthermore, the coding tools *sign data hiding* (SDH), *rate-distortion optimized quantization* (RDOQ), *transform skip mode* (TSM), and *trellis-coded quantization* (TCQ) were disabled. These coding tools interact with level coding, i.e., a modification in level coding requires adjustments for each of these tools. Past experience has shown that when an approach in level coding provides coding efficiency improvements without these tools, the performance is maintained upon their reactivation with corresponding adjustments.

Disabling the adaptive context model sets for the context modeling of b_{sig} in the first coding phase provides virtually the same coding efficiency relative to IMP5-0. A reason for the ineffectiveness of the adaptive context model sets when used in the first coding phase is the scanning of sub-blocks, which is reversed after incorporating the extra coding tools inherited from HEVC. This inefficiency could presumably be resolved by updating the parameters of the adaptive context model sets. Nevertheless, the pursued and desired design in this chapter should not rely on 4×4 sub-blocks for context modeling anymore, and therefore, the adaptive context model sets were disabled for the context modeling of b_{sig}. This implementation is called IMP5-1 in the following paragraphs and is used as the basis for the investigations described in the next section.

5.3 | Template-Based Context Modeling for Significance

The context modeling of the b_{sig} flags implemented in the level coding presented in the previous chapters, to which the implementations IMP5-0 and IMP5-1 correspond, depends mainly on the frequency location. This concept implicitly assumes that the bin probabilities are determined by the frequencies, but are independent of other transform coefficient levels within the same transform block. Nevertheless, one can observe from actual bitstreams that transform blocks with low and high residual energy coexist within the same picture at the same operating point. An approach that considers such dependencies could be based on analyzing already transmitted data of neighboring frequency locations, which can detect, for example, whether a transform block is one with low or high residual energy.

For each scanning position, a so-called **local template** specifies the already coded frequency locations whose level data should be analyzed regarding their statistical properties. The result of the statistical analysis can then be used to determine context model offsets and the Rice parameter. The shape of a local template is 2-dimensional, which is suitable for the separable transforms used in video coding, because the resulting transform coefficients are arranged in a 2-dimensional structure within the block. Moreover, one can also observe from actual bitstreams that for the separable transforms used in HEVC and VVC, non-zero transform coefficients are often concentrated in local clusters, which vary among the blocks and supposedly depend on the prediction mode, block size, and the original input signal. An example for a local template within a 4×4 transform block is illustrated in figure 5.3, where the ×-marked location represents the current scanning position. The white-shaded locations represent uncoded locations, and the numbered locations represent already coded locations, whereas the orange-shaded locations illustrate an example of a local template geometry.

Evaluating a local activity for the current position is a straightforward and long-established approach to analyze the statistical properties of a location. For example, the lossless image coding standard JPEG-LS [65] uses a local template to predict the current sample. In [37], which the author of this thesis co-authored, the local template was investigated for the context modeling of the b_{sig} flags in the context of HEVC. Note that in [37], the scanning pattern is adaptive, and different context model sets are used for the top and the left edges of the transform block. Both employed techniques indicate that certain statistical dependencies remain after the transform, for example, in the form of a concentration of non-zero transform coefficient levels along the edges of the transform block. Alternatively, such a concentration could be detected and exploited by using alternative scanning patterns, such as in the horizontal or the vertical directions, which, in turn, lead to less coded scanning positions with zero-valued transform coefficient levels [82]. Nevertheless, adaptive scanning patterns cannot exploit statistical dependencies caused by the input signal itself. They require

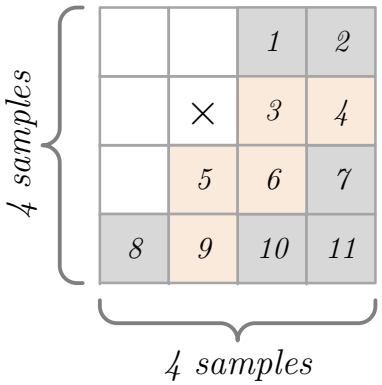

Figure 5.3
Example of a local template geometry within a 4×4 transform block, where each number represents an identification number for already coded neighboring frequency locations. The ×-marked box denotes the current scanning position, and the numbered boxes represent already coded locations with their corresponding identification number. The orange-shaded boxes represent the local template finally chosen based on coding experiments described in this chapter.

some form of existing information that indicates the prevalence of dependencies, such as the direction of the intra predictor. A template-based context modeling is much more flexible; additional zero-valued scanning positions that have to be coded due to a fixed scanning pattern can be represented efficiently using the same adaptive context model.

In the first part of this chapter, the properties of the local template are analyzed by investigating different configurations for the context modeling of the b_{sig} flags. For the first investigation, the anchor implementation IMP5-2 is based on IMP5-1 with a further modification to the context modeling of b_{sig}. More precisely, the level coding of implementation IMP5-2 replaces the frequency-based context modeling for b_{sig} of IMP5-1 with a single context model. Furthermore, a single context model set is used for all transform block sizes. This implementation IMP5-2 provides a better understanding of the experimental results when analyzing the impact of a single neighboring frequency location. Compared to the frequency-based variant, which is IMP5-1, a bit-rate overhead of 0.86% in the *All-Intra* and 0.33% in the *Random-Access* configurations can be observed for IMP5-2.

5.3.1 | Local Template Configuration

A local template is specified by its shape or geometry, and the challenge in this design is to find the best function $C(\cdot)$, i.e., the mapping of already coded level data inside the local template to a context model offset for the current scanning position. However, the direct determination of the mapping includes many degrees of freedom and requires multidimensional optimization, which is impractical. Therefore, the optimization is split into multiple steps to overcome this problem, where the first optimization step is to find a suitable geometry for the local template. Two investigations were performed to determine the geometry of the local template, and in both investigations, the local template was applied to the context modeling of b_{sig} only. The first investigation analyzes the impact of each neighboring frequency location individually. Based on the experimental results of the first investigation, an appropriate geometry for the local template is established in the second investigation.

5.3.2 | Impact of a Single Neighboring Frequency Location

The impact of a neighboring frequency location on the context modeling of b_{sig} mainly depends on its relative spatial distance to the current scanning position. Therefore, coding experiments evaluating a single neighboring location with a fixed spatial offset to the current scanning position were conducted to assess the impact of a single neighboring location. Let x and y denote the spatial coordinates of the current scanning position, and let Δx and Δy denote fixed spatial offsets relative to x and y, respectively. Then, each conducted coding experiment evaluates the b_{sig} flag at a different location $(x + \Delta x, y + \Delta y)$. The value of b_{sig} at the neighboring location $(x + \Delta x, y + \Delta y)$ is 0 when the corresponding transform coefficient level is zero-valued, and it is one when the partially reconstructed level is non-zero. For context modeling, the value of b_{sig} at the neighboring location is mapped directly to the context model offset, i.e., $\delta_{sig} = b_{sig}(x + \Delta x, y + \Delta y)$.

It might happen that the $b_{sig}(x + \Delta x, y + \Delta y)$ flag of the considered neighboring frequency location is unavailable, because it is located outside of the block or has not been coded yet. Therefore, two implementations were tested, denoted as IMP5-3 and IMP5-4, where an unavailable value for $b_{sig}(x + \Delta x, y + \Delta y)$ is treated

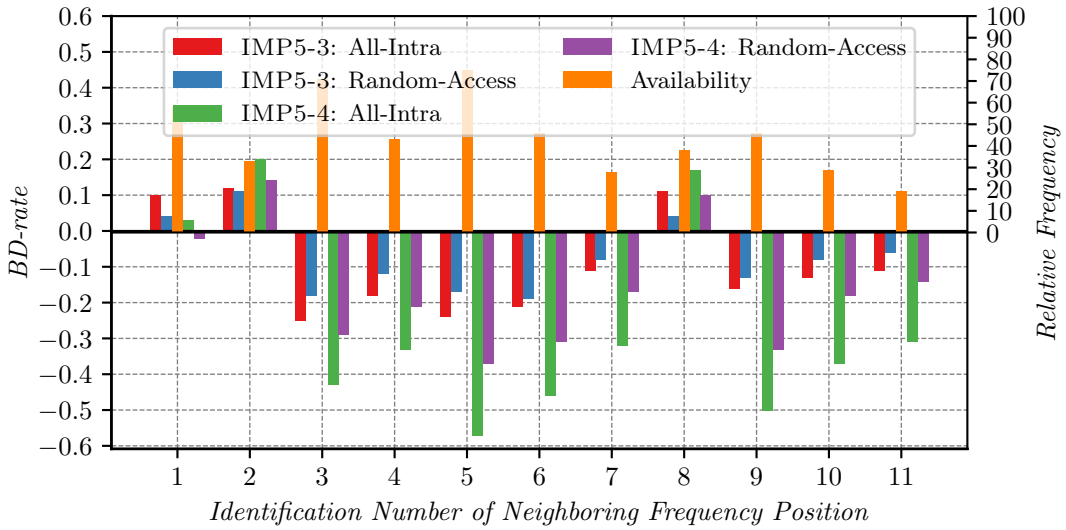

Figure 5.4
Coding efficiency of the investigations on evaluating one fixed neighboring location for the context modeling of b_{sig} (IMP5-3 and IMP5-4). The x-axis denotes the identification number of the neighboring frequency location used in each experiment (compare figure 5.3). An unavailable neighboring location is interpreted as zero-valued in IMP5-3, whereas a dedicated context model is used for it in IMP5-4.
Anchor for BD-rate computations: IMP5-2

differently in these two implementations. In the implementation IMP5-3, an unavailable location is considered zero-valued ($\delta_{sig} = 0$), resulting in $\delta_{sig} \in \{0, 1\}$ and two context models. For the implementation IMP5-4, an additional context model ($\delta_{sig} = 2$) is used for an unavailable location, resulting in $\delta_{sig} \in \{0, 1, 2\}$ and three context models. Note that for the two neighboring frequency locations above the current scanning position (one and two in figure 5.3) and the location in the left column (8 in figure 5.3), an additional check on whether the location was coded before is necessary, because they may be unavailable due to sub-block processing.

Experimental Results

Figure 5.4 summarizes the obtained BD-rates for both tested implementations, where the x-axis denotes the tested neighboring location according to the numbering in figure 5.3. For example, only the neighboring frequency location number three outlined in figure 5.3 was considered in the coding experiment that yields the BD-rates for the entry three of figure 5.4. Besides the BD-rates, figure 5.4 includes the relative frequency for the availability of a neighboring frequency location, denoted by orange bars. The BD-rates for IMP5-3 and IMP5-4 are denoted in figure 5.4 by the corresponding colored bars.

If the neighboring locations are roughly sorted according to the provided BD-rates for IMP5-3, the order is $(3, 5, 6, 9, 4, 7, 10, 11, 1, 2, 8)$, where the last three frequency locations of the list yield a bit-rate overhead. A probable reason for the bit-rate overhead is that when the availability is low, the effectiveness of a location may be biased compared to the case where the location would always be available. However, at least for those three mentioned locations, there is no such relationship, because their availability is between 33% and 58%, whereas location 11, with an availability of 19%, provides coding efficiency improvement. Nevertheless, the assumption that unavailable locations result in biased statistics is supported by the BD-rates obtained for IMP5-4, where a dedicated context model is employed for unavailable locations. The coding efficiency of IMP5-4 is superior to that of IMP5-3 for all neighboring locations providing coding efficiency improvement in IMP5-3.

5.3.3 | Determination of the Local Template Geometry

In this second analysis, the local template size is incrementally increased by one neighboring location to determine its geometry. Two implementations, denoted as IMP5-5 and IMP5-6, were tested for this investigation, and the test procedure was the same for both implementations. In the first experiment of each implementation, the coding efficiency for the local template consisting of the two neighboring locations $\{3, 5\}$, which are the two neighboring locations providing the best coding performance in the previous analysis (section 5.3.2), is evaluated. For each further experiment of each configuration, the template incorporates one additional neighboring location, where neighboring locations were included according to the list $(6, 9, 4, 7, 10, 11, 1, 2, 8)$. Note that this order is chosen according to the obtained coding efficiencies of IMP5-3.

An unavailable neighboring location is considered zero-valued in IMP5-5 of this investigation, and the context model offset is set equal to the number of non-zero neighboring locations within the local template. Let T be the template size or the number of frequency locations covered by the template, and let $b_{sig}\,(i)$ be a b_{sig} flag of a neighboring location inside the template, with $i \in [0, T)$. Then, the context index offset of b_{sig} at the current scanning position is derived by:

$$\delta_{sig} = \sum_{i=0}^{T-1} b_{sig}\,(i)\,. \tag{5.1}$$

In IMP5-5, the number of context models is $T + 1$. Furthermore, this implementation implicitly uses a context quantizer that removes the spatial relationship between the individual neighboring locations inside the template and the current scanning position. The effect of a context quantizer is analyzed with IMP5-6, where a dedicated context model is used for each distinctive combination of the b_{sig} flags inside the template. For the same interpretation of an unavailable location as in IMP5-4, the number of possible distinctive combinations is equal to 2^T, and the context index offset is derived by:

$$\delta_{sig} = \sum_{i=0}^{T-1} b_{sig}\,(i) \cdot 2^i. \tag{5.2}$$

When considering the non-availability of a neighboring location as a third possible value for b_{sig}, the number of possible combinations is 3^T. This implementation was not tested extensively, because preliminary results indicated that the coding performance is worse than IMP5-6, probably due to context dilution.

Experimental Results

The experimental results for IMP5-5 and IMP5-6 are summarized in figure 5.5, both relative to IMP5-2. Local template sizes up to eleven neighboring locations were tested for both implementations. As indicated in figure 5.5 for IMP5-5, an increased template size improves the coding efficiency, except for the last three experiments, where the neighboring locations that provided coding efficiency losses in IMP5-3 of the preceding investigation are included into the template geometry. This observation indicates that only neighboring frequency locations with a positive impact on the coding efficiency (when considered individually) contribute positively to the local template. When increasing the template size from one to two, the bit-rate saving is greater than the sum of the two bit-rate savings of IMP5-3, where a single neighboring location was considered in each experiment only. The sum of BD-rates obtained from IMP5-3 is illustrated in figure 5.5 by orange and yellow bars. A further conclusion from the observation is that the first investigation is suitable for determining the template size but does not directly relate to the final performance. In IMP5-6, a dedicated context model is used for each distinctive combination of the local template, where unavailable neighboring locations are considered zero-valued. The obtained bit-rate savings are slightly smaller than for the same template geometry in IMP5-5, indicating that the employed context quantizer is feasible. Furthermore, the number of context models of IMP5-6, which is 2^T, becomes significantly greater than that of IMP5-5, which is $T + 1$, when $T > 2$.

With the obtained BD-rates, the recommended local template size is equal to 5, because the coding efficiency improvement is relatively small when further increasing the template size. IMP5-5 with the recommended

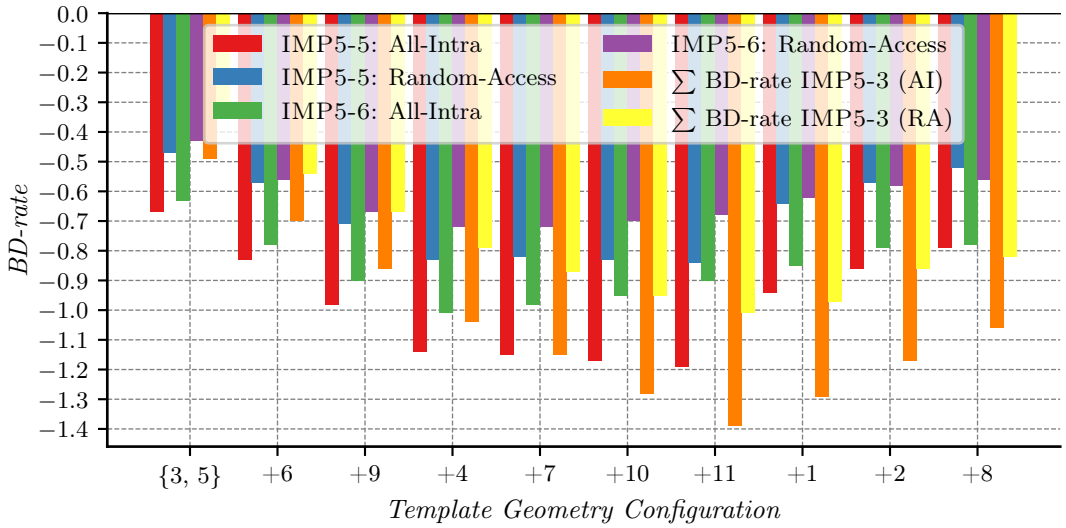

Figure 5.5
Coding efficiency of the investigations on increasing template size incrementally for the context modeling of b_{sig} (IMP5-5 and IMP5-6). In this plot, the x-axis denotes the neighboring frequency locations considered by the template, e.g., the first tick mark represents the first experiment, where the local template consists of the locations $\{3, 5\}$. The second tick mark denotes the second experiment, where the local template consists of the locations $\{3, 5, 6\}$, and each further tick mark denotes the neighboring location further included by the local template.
Anchor for BD-rate computations: IMP5-2

template size configuration is referred to as IMP5-5* in the following text. Compared to the level coding developed in the previous chapter and implemented in VTM-17.0 with the extra coding tools inherited from HEVC, which is the implementation IMP5-0 described in section 5.2, the coding efficiency is improved by -0.30% and -0.53% in the *All-Intra* and *Random-Access* configurations, respectively, for IMP5-5*.

Consideration of Unavailable Locations

Previous coding experiments demonstrated that the coding efficiency could be further improved when considering unavailable neighboring locations separately by using different context models. Such consideration can become complicated for larger templates due to the number of additional context models and context dilution. An alternative implementation IMP5-7 was tested to analyze the aspect of unavailable neighboring locations. This implementation IMP5-7 only requires twice the amount of context models of IMP5-5 and is straightforward to implement. The context model offset is derived as in IMP5-5, denoted by equation (5.1). However, in the case that at least one neighboring location inside the template is unavailable, a second context model set is used. Consequently, the number of context models required in IMP5-7 is doubled compared to IMP5-5, i.e., it is $2T + 2$ instead of $T + 1$.

The coding efficiency for IMP5-7 is summarized in figure 5.6 for template sizes up to eleven neighboring locations relative to IMP5-2; the BD-rate were calculated using the same anchor as for IMP5-3 to IMP5-6. Compared to IMP5-5, which is included in figure 5.6 for reference, the implemented consideration of unavailable locations provides at least -0.1% coding efficiency improvements for the recommended template size equal to 5. For other template sizes, the coding efficiency improvement is up to -0.3%, e.g., for a template size equal to 11. Nevertheless, the case where unavailable frequency locations are treated separately will not be pursued further in this chapter and is a topic for future research.

5.3.4 | Trade-Off Analysis

In the previous design, which is represented by the implementation IMP5-0, the context model offsets of b_{sig} can be determined simultaneously for all frequency locations inside a 4×4 sub-block. The local template seems more complex than the variant implemented in IMP5-0, because the context model offset can only be

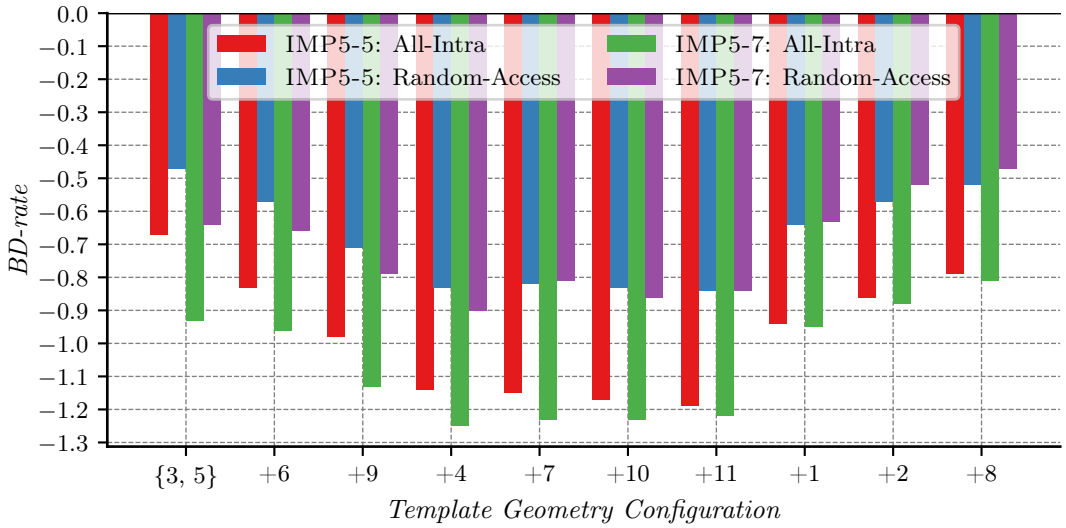

Figure 5.6
Coding efficiency of the investigation on increasing template size incrementally for the context modeling of b_{sig} (IMP5-7). In IMP5-7, the occurrence of an unavailable location within the template results in the usage of a different context model set. The performance of IMP5-5 is included for comparison purposes.
Anchor for BD-rate computations: IMP5-2

determined for a location when all preceding frequency locations are reconstructed. Nonetheless, additional complexity is mainly introduced by the local template with additional memory accesses, i.e., more reconstructed information is necessary. These additional memory accesses can be limited in practical applications. For example, the processing performance can be significantly improved by employing caching mechanism in hardware and stack or local memory in both hardware and software implementations. Especially interesting for hardware implementations is the significantly reduced number of context models, where for the final and recommended local template configuration, only six context models are necessary for luma, whereas the previous design requires 32 context models. In IMP5-0, 4×4 transform blocks require a dedicated context model set, whereas the same context models can be used for all transform block sizes in the template-based variant. The conjecture assumed here is that a different context model set for 4×4 transform blocks should no longer be necessary, because the local template can detect low and high residual energy transform blocks. This aspect will be investigated in the context of additional context model sets in the second part of this chapter, which focuses on the evaluation of absolute transform coefficient levels. In summary, the obtained results prove the feasibility of a template-based context modeling, and indicate that improved coding efficiency can be achieved with fewer context models.

5.4 | Single Coding Phase and Level Magnitudes

Employing the local template in a meaningful configuration for context modeling of the b_{sig} flags provides improved coding efficiency, as demonstrated in the preceding section. This concept can be extended to the second coding phase, described in section 3.2.1, where the remaining level magnitudes are transmitted. It is even reasonable to investigate different template geometries for the context modeling of the $b_{|x|>1}$ and the $b_{|x|>2}$ flags, and for determining the Rice parameter. However, such an approach appears unattractive, because templates with different geometries have to be evaluated multiple times with different level data, which makes the concept impractical. An alternative strategy to incorporate the local template for the coding of remaining level magnitudes is to combine the two coding phases into a single coding phase and analyze the level data inside the template only once.

5.4.1 | Enabling Evaluation of Absolute Transform Coefficient Levels

The level magnitudes were transmitted in two coding phases in all previously considered level coding designs. Combining the two coding phases into a single phase without further modifications to the context modeling became possible after introducing the reverse scanning pattern for the first coding phase. This combination does not alter the coding performance, because the different context modeling techniques rely on the same syntax element type. At the same time, a single coding phase enables the availability of level magnitudes for already transmitted frequency locations, which can be exploited for context modeling. It is reasonable to expect that, for example, the probability for a non-zero location is greater when the reconstructed neighborhood consists of transform coefficient levels with large magnitudes. However, a disadvantage of a single coding phase is that speculative processing for context modeling becomes challenging, because context models for the same type of flags are not determined successively. That can be faced by providing improved coding efficiency, or by a design where all context model offsets and the Rice parameter are derived by evaluating the local template only once. In the first part of this section, the context modeling of b_{sig} is analyzed when evaluating absolute transform coefficient levels located in the vicinity using a local template.

5.4.2 | Non-Zero Locations with Absolute Level Magnitudes

For this first investigation, the starting point is IMP5-5*, but employing a single coding phase instead of two coding phases. This implementation served as the anchor for BD-rate computations and is referred to as IMP5-8. After merging the two coding phases of IMP5-5* into a single coding phase, limited experiments were conducted to investigate the influence of the sign information. The experimental results of this investigation indicated that no statistical dependencies exist between the level of the current scanning position and the signs of neighboring locations. Therefore, the following investigations only consider absolute transform coefficient levels, denoted by x.

Motivated by the simplicity and the coding performance of the context modeling for the b_{sig} flags in IMP5-5*, this investigation reuses the context quantizer approach. But in the single coding phase implementation, $b_{sig}\,(i)$ of equation (5.1) implemented in IMP5-8 is replaced by $x\,(i)$. The absolute sum of the level magnitudes inside the local template is assigned to δ_{sig}, and unavailable frequency locations are considered zero-valued. Unlike the sum in equation (5.1), the sum with absolute level magnitudes can significantly exceed the number of neighboring frequency locations. This fact may not only face a challenge to the number of necessary context models but may also lead to suboptimal coding performance due to context dilution. Therefore, the context quantizer has to be extended to limit the number of context models, and the selected approach is to clip the sum when it exceeds a threshold M_{sig}. Let T be the size of the template, and let $x\,(i)$ be an absolute level flag of a neighboring location inside the template, with $i \in [0, T)$. Then, the context index offset for a b_{sig} flag is derived by:

$$\delta_{sig} = \min\left(M_{sig}, \sum_{i=0}^{T-1} x\,(i)\right). \tag{5.3}$$

In contrast to the context modeling denoted by equation (5.1), the number of context models is $M_{sig} + 1$ instead of $T + 1$ in this implementation. The implementation used for this investigation is referred to as IMP5-9 in the following paragraphs.

Experimental Results

In the coding experiments for this investigation, clipping values $M_{sig} \in [1, 11]$ were tested. The corresponding BD-rates are summarized in figure 5.7, where the anchor for all BD-rate calculations is IMP5-8. The first finding based on the results illustrated in figure 5.7 is that evaluating absolute level magnitudes inside the template improves coding efficiency compared to the variant where only b_{sig} is considered. A second finding is that the coding efficiency is already improved with a clipping value $M_{sig} = 2$, which means that only three context models can provide a higher coding efficiency than IMP5-8 with six context models. For $M_{sig} = 5$ that requires six context models in total, which is equivalent to the number of context models of IMP5-8, the tested implementation IMP5-9 yields BD-rates of -0.14% and -0.12% in the *All-Intra* and *Random-Access* configurations, respectively. Further investigations regarding the impact of specific or unavailable neighboring

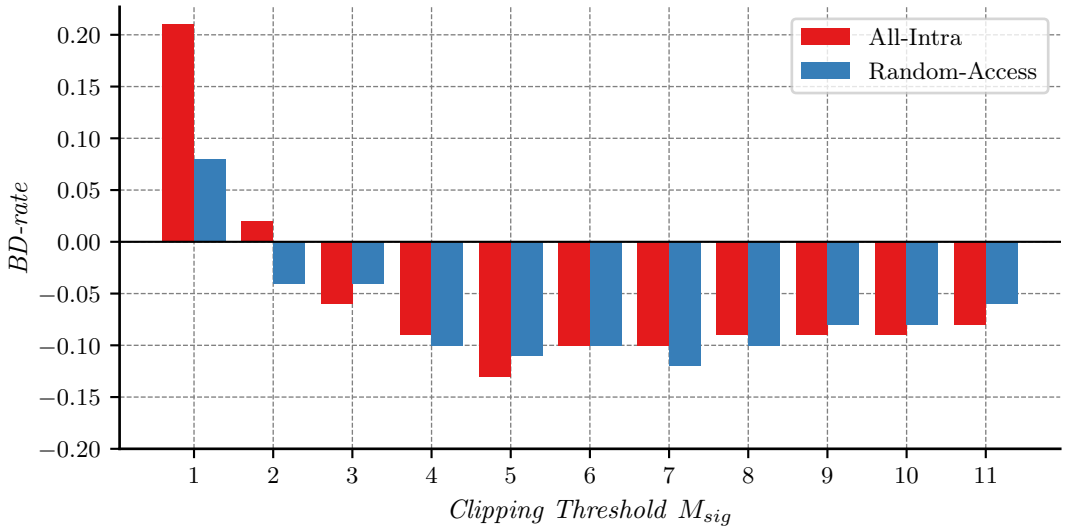

Figure 5.7
Coding efficiency of the investigation on clipping the sum of absolute levels located at the neighboring frequency locations inside the local template (IMP5-9). The sum of the neighboring absolute levels is clipped to M_{sig} and mapped directly to the context model offset of b_{sig}.
Anchor for BD-rate computations: IMP5-8

locations are conceivable. Nevertheless, these considerations are not further pursued in this thesis and are topics for future research, as mentioned during the presentation of the investigations for the local template geometry. IMP5-9 with $M_{sig} = 5$ was chosen as the basis for further investigations and it is referred to as IMP5-9*.

5.4.3 | Coding of Magnitudes with a Local Template

The second coding phase of IMP5-9* implements tracking variables for context modeling of the $b_{|x|>1}$ and $b_{|x|>2}$ flags, which are reinitialized at the beginning of each sub-block, and utilizes adaptive context model sets, both described in section 3.3. Furthermore, $x-3$ is binarized using a combination of Rice and 0^{th}-*order exponential-golomb* (EG0) codes, as described in section 4.3. These approaches derive the probabilities for the coding of $b_{|x|>1}$, $b_{|x|>2}$, and the Rice parameter for $x-3$ by evaluating the same kind of data transmitted for preceding scanning positions within the transform block. Investigations on a suitable template-based replacement for coding the remaining level magnitudes are pursued in the following paragraphs. The optimization is broken down into two investigations. In the first investigation, a suitable context modeling for both the $b_{|x|>1}$ and $b_{|x|>2}$ flags is studied, and the following second investigation pursues a reasonable Rice parameter derivation.

Context Modeling of $b_{|x|>1}$ and $b_{|x|>2}$ with the Local Template

For both flags, the context modeling of IMP5-9* relies on the number of already coded $b_{|x|>1}$ flags within the same sub-block, denoted by equation (3.1) for $b_{|x|>1}$ and equation (3.2) for $b_{|x|>2}$. It can, therefore, be presumed that there are high statistical dependencies between the transmitted $b_{|x|>1}$ flags and the probability of $b_{|x|>1}$ and $b_{|x|>2}$ for the current scanning position. A translation of this concept to the local template-based context modeling is analyzing either the $b_{|x|>1}$ flags or $x-1$ of the frequency locations inside the template. Although it would be possible to consider the $b_{|x|>2}$ flags or $x-2$ for the context modeling of $b_{|x|>2}$, this implementation is not further investigated, because the existing context modeling of IMP5-9* indicates a relationship to $b_{|x|>1}$ rather than $b_{|x|>2}$. Furthermore, it was revealed in the previous investigation that analyzing the level magnitudes for the context modeling of b_{sig} provides a higher coding performance than analyzing the b_{sig} flags. With this background, it is straightforward to consider the level magnitudes instead of the $b_{|x|>1}$ flags for context modeling. As for the previous investigation, the number of context models

becomes extensive when mapping the sum directly to a context index offset. Therefore, the implementation IMP5-10 validates different clipping values M_{gtX}, i.e., the sum is clipped when it exceeds a threshold M_{gtX}. Furthermore, the context model offsets for $b_{|x|>1}$ and $b_{|x|>2}$ are derived in the same manner for a scanning position, but each flag uses its own context model set. Let $\delta_{gtX} = \delta_{x>1} = \delta_{x>2}$, then the context index offset for the $b_{|x|>1}$ and $b_{|x|>2}$ flags of IMP5-10 is derived by:

$$\delta_{gtX} = \min\left(M_{gtX}, \sum_{i=0}^{T-1} \max\left(x\left(i\right) - 1, 0\right)\right). \tag{5.4}$$

The context modeling of $b_{|x|>1}$ and $b_{|x|>2}$ in IMP5-10 is derived from the existing context modeling implemented in IMP5-9*, which implies that $x\left(i\right) = 1$ inside the template has no relationship to the probability. This case is expressed in equation (5.4) by excluding neighboring frequency locations with $x\left(i\right) \leq 1$ while calculating the sum. It can be speculated that levels with $x\left(i\right) = 1$ affect the probability of $b_{|x|>1}$ and $b_{|x|>2}$ positively, i.e., the context modeling can attain a higher coding efficiency when considering $x\left(i\right) = 1$. For example, let the neighboring frequency locations inside the template consist of $x\left(i\right) = 1$ only. It can be presumed that the probability of $b_{|x|>1} = 1$ is higher than for the case where the neighboring frequency locations consist of zero-valued levels. For this hypothesis, absolute levels x are analyzed for context modeling instead of $x - 1$, and this hypothesis is tested with the implementation IMP5-11, where the context index offset for the $b_{|x|>1}$ and $b_{|x|>2}$ flags is derived by:

$$\delta_{gtX} = \min\left(M_{gtX}, \sum_{i=0}^{T-1} x\left(i\right)\right). \tag{5.5}$$

Clipping values $M_{gtX} \in [1, 11]$ were tested for both implementations, with a summary of the BD-rates illustrated in figure 5.8. For both tested implementations, the anchor for BD-rate computations is IMP5-9*. Different than expected, IMP5-10 does not provide coding efficiency improvements relative to IMP5-9* for all tested M_{gtX} thresholds. For comparison, the context modeling of $b_{|x|>1}$ and $b_{|x|>2}$ in IMP5-9* converges to a final setting after the coding of a $b_{|x|>1} = 1$ flag for the context modeling of $b_{|x|>1}$, and after the coding of five $b_{|x|>1} = 1$ flags for the context modeling of $b_{|x|>2}$ (equation (3.1) and equation (3.2)). This results in a setting where fixed context index offsets are used for the remaining scanning positions ($\delta_{x>1} = 0$ and $\delta_{x>2} = 4$), which stabilizes the context modeling. In contrast to IMP5-10, the implementation IMP5-11 provides improved coding performance relative to IMP5-9*. Given the experimental results for IMP5-11, it can be assumed that analyzing x provides a more robust context modeling than considering $x - 1$, when the explanation for the coding performance of IMP5-10 is correct. For clipping values $M_{gtX} > 5$, further coding efficiency improvements are insignificant, which results in the selection of $M_{gtX} = 5$ as the recommendation. This configuration in IMP5-11 is referred to as IMP5-11* in the following paragraphs.

Based on IMP5-11 with $M_{gtX} = 5$, two further investigations were performed to resolve the following questions: Is it feasible to use the same context model set for both $b_{|x|>1}$ and $b_{|x|>2}$, and is it feasible to employ a dedicated context model for the last significant scanning position? The first question addresses the number of context models, where the underlying expectation is that the same context model set can be employed, because the $b_{|x|>2}$ flags rarely occur. For the second question, it is assumed that the last significant scanning position has a higher probability for $b_{|x|>1} = 0$ than other scanning positions, especially for operation points with lower bit-rates. The implementation for the first question is referred to as IMP5-12, and it provides -0.04% in the *All-Intra* and -0.02% in the *Random-Access* configurations, respectively, relative to IMP5-11*. Consequently, sharing the same context model set for $b_{|x|>1}$ and $b_{|x|>2}$ provides virtually the same coding performance while saving context memory. The implementation for the second question is based on IMP5-12 and is referred to as IMP5-13. This IMP5-13 provides the same coding performance in the *All-Intra* and 0.03% in the *Random-Access* configurations, respectively, relative to IMP5-12. These results indicate that a dedicated context model used for coding the $b_{|x|>1}$ and $b_{|x|>2}$ flags at the last significant scanning position provides no additional coding efficiency improvement. Further investigations are based on IMP5-12 that requires six context models in luma for coding the $b_{|x|>1}$ and $b_{|x|>2}$ flags, whereas the context modeling in IMP5-9* and earlier requires more than 30 context models.

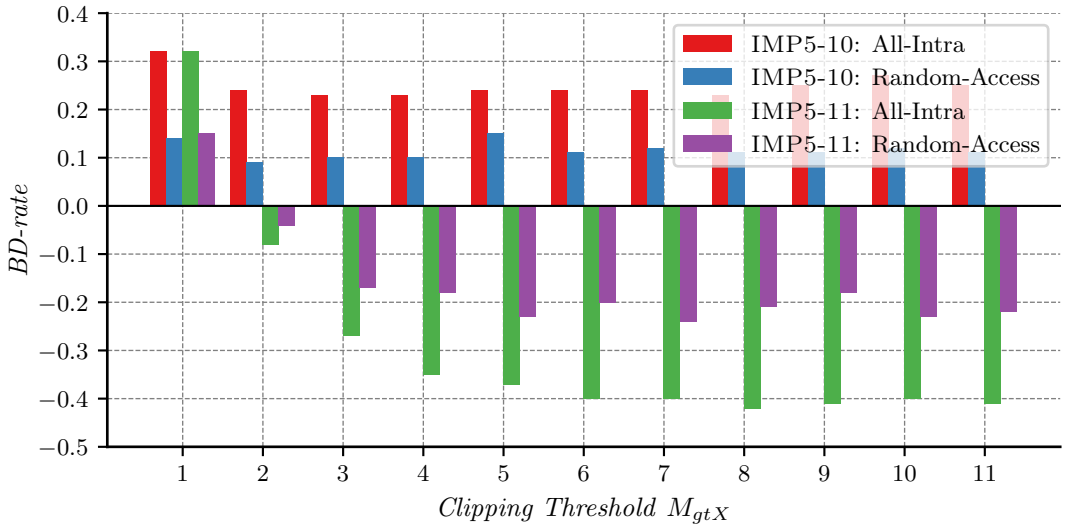

Figure 5.8
Coding efficiency of the investigations on clipping of the sum of absolute levels located at the neighboring frequency locations inside the local template (IMP5-10 and IMP5-11). The sum of the neighboring absolute levels is clipped to M_{gtX} and mapped directly to the context model offset of $b_{|x|>1}$ and $b_{|x|>2}$.
Anchor for BD-rate computations: IMP5-9*

Rice Parameter Selection with the Local Template

The preceding investigations of this section indicated that analyzing the absolute levels inside the template provides higher coding efficiency than analyzing the same syntax element type, e.g., $b_{|x|>1}(i)$ are considered for context modeling of $b_{|x|>1}$ only. Based on this finding, the investigation on selecting the Rice parameter with a local template only analyzes absolute transform coefficient levels. Let the Rice parameter be k, and let $\mathcal{S}$ be the sum of absolute transform coefficient levels for the neighboring frequency locations inside the template. The investigated Rice parameter selection chooses a value for k by comparing $\mathcal{S}$ against the fixed thresholds $M_{Rice}^{k=1}$, $M_{Rice}^{k=2}$, and $M_{Rice}^{k=3}$ (with $0 \leq M_{Rice}^{k=1} < M_{Rice}^{k=2} < M_{Rice}^{k=3}$) for each scanning position. This Rice parameter assessment can be summarized by:

$$k = \begin{cases} 1, & \text{if } \mathcal{S} > M_{Rice}^{k=1} \wedge \mathcal{S} \leq M_{Rice}^{k=2}, \\ 2, & \text{if } \mathcal{S} > M_{Rice}^{k=2} \wedge \mathcal{S} \leq M_{Rice}^{k=3}, \\ 3, & \text{if } \mathcal{S} > M_{Rice}^{k=3}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.6}$$

The implementations of this investigation are called IMP5-14, followed by IMP5-15 based on IMP5-14, and finally IMP5-16 based on IMP5-15. In IMP5-14, different values for $M_{Rice}^{k=1}$ are tested, while the other thresholds are set to be $M_{Rice}^{k=2} = M_{Rice}^{k=3} = \infty$, meaning that the Rice parameter is limited to $k \in \{0, 1\}$. After obtaining the coding performances of IMP5-14, an appropriate value for $M_{Rice}^{k=1}$ is chosen, on which IMP5-15 is based to verify different values for $M_{Rice}^{k=2}$. In the final IMP5-16, the value for $M_{Rice}^{k=3}$ is determined after choosing a suitable value for $M_{Rice}^{k=2}$, which also completes the investigation. For all implementations, the anchor for the BD-rate computation is IMP5-12, i.e., the Rice parameter selection developed in chapter 4. In contrast to the Rice parameter selection of IMP5-12, the Rice parameter k can be selected freely for each scanning position in IMP5-14, IMP5-15, and IMP5-16. Furthermore, this template-based Rice parameter selection does not implement a fixed setting, where the same Rice parameter is used for the remaining scanning positions of a sub-block when a certain condition is met, such as in IMP5-12.

The obtained coding efficiencies for all three implementations are summarized in figure 5.9. The finally chosen thresholds are $M_{Rice}^{k=1} = 5$, $M_{Rice}^{k=2} = 9$, and $M_{Rice}^{k=3} = 18$, and IMP5-16 with the corresponding thresholds is referred to as IMP5-16* for further investigations. Note that the selected thresholds represent a compromise

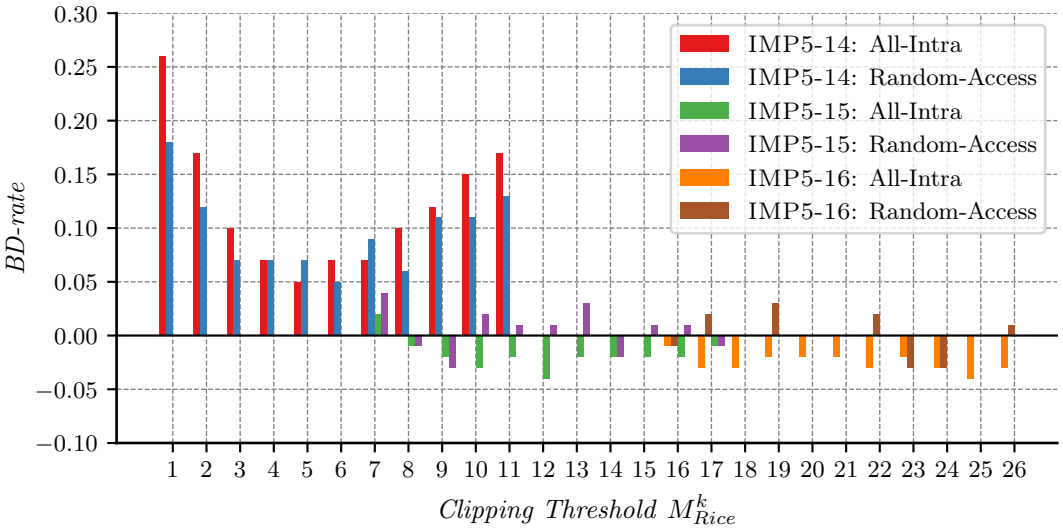

Figure 5.9
Coding efficiency of the investigations on updated thresholds for the Rice parameter selection with the local template evaluation (IMP5-14 to IMP5-16). In IMP5-14, the sum of absolute levels for frequency locations inside the template is compared against different $M^{k=1}_{Rice}$ thresholds, while the other thresholds are set to be $M^{k=2}_{Rice} = M^{k=3}_{Rice} = \infty$. In IMP5-15, different $M^{k=2}_{Rice}$ thresholds are evaluated, while the other thresholds are set to be $M^{k=1}_{Rice} = 5$ and $M^{k=3}_{Rice} = \infty$. In IMP5-16, different $M^{k=3}_{Rice}$ thresholds are evaluated, while the other thresholds are set to be $M^{k=1}_{Rice} = 5$ and $M^{k=2}_{Rice} = 9$.
Anchor for BD-rate computations: IMP5-12

between the coding performance in the *All-Intra* and the *Random-Access* configurations. Compared to the variant developed in chapter 4 and implemented in IMP5-12, the Rice parameter selection with the local template provides the same coding performance.

5.4.4 | Position-Dependent Context Model Sets

The underlying assumption for adaptive context model sets used in IMP5-0 is that absolute levels can be clustered spatially within a transform block. The variance of the absolute levels may differ for each cluster, which the context modeling can exploit to improve the coding efficiency. In IMP5-0, the implementation exploits this assumption by analyzing the number of $b_{|x|>1} = 1$ flags within the preceding sub-block, and this implementation serves two purposes. Firstly, a change in the variance of the absolute level distribution is exploited by using different context model sets, i.e., whether the current sub-block distribution is one with a low or high variance is estimated. Secondly, the context modeling is reinitialized, which enables the adaptation to a possibly changed distribution for the absolute levels of the current sub-block. It is worth mentioning that reinitialization is necessary, because both the context modeling (for the $b_{|x|>0}$ and $b_{|x|>1}$ flags) and the Rice parameter estimation include a state leading to fixed settings, i.e., the same context models and Rice parameter are used for the remaining scanning positions. The second purpose can be resolved using the local template due to its high flexibility and its current configuration without a state using fixed settings. However, the first purpose is not adequately covered by the local template, because for a sudden change in the variance, the context modeling has to code enough data so that the context models can adapt to the updated statistics. This circumstance is investigated by introducing additional context model sets, where the selection of a context model set should depend on a spatial property of the current scanning position.

Context Model Sets for Context-Coded Flags

The chosen spatial property, on which the selected context model set should depend, is the diagonal $D(x, y) = x+y$ of the current scanning position (x, y), because of the 2-dimensional structure that appears for transform coefficients due to the used transforms in VVC. This investigation is limited to two additional context model

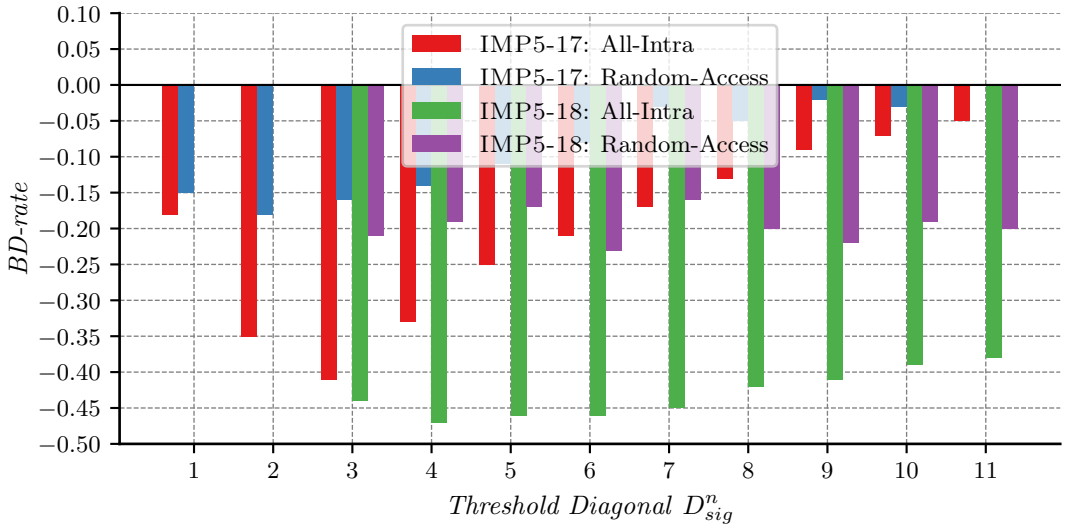

Figure 5.10
Coding efficiency of the investigations on the switch to a different context model sets for b_{sig} depending on the diagonal $D(x,y) = x+y$ (IMP5-17 and IMP5-18). In IMP5-17, a second context model set is selected for coding the b_{sig} flags when $D(x,y) < D^0_{sig}$, while the second threshold is set to be $D^1_{sig} = \infty$. In IMP5-18, a third context model set is selected when $D(x,y) < D^1_{sig}$, while a second context model set is selected when $D^0_{sig} = 2$.
Anchor for BD-rate computations: IMP5-16*

sets for luma. IMP5-17 of this investigation, which is based on IMP5-16*, different context model sets are selected by comparing the diagonal $D(x,y)$ against the thresholds D^0_{sig} and D^1_{sig}. Let δ^*_{sig} denote the context index offset calculated using equation (5.3) with $M_{sig} = 5$, then the final context offset δ_{sig} is derived by:

$$\delta_{sig} = \delta^*_{sig} + \begin{cases} 6, & \text{if } D(x,y) < D^0_{sig}, \\ 12, & \text{if } D(x,y) < D^1_{sig} \wedge D(x,y) \geq D^0_{sig}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

This investigation involves the two implementations IMP5-17 and IMP5-18. In IMP5-17, different values for D^0_{sig} are tested while $D^1_{sig} = \infty$, i.e., only one additional context model set is tested. After choosing an appropriate value for D^0_{sig}, IMP5-18 based on IMP5-17 evaluates different values for D^1_{sig}. A summary of the coding performances for both implementations is illustrated in figure 5.10, where the anchor used to calculate the BD-rates is IMP5-16*. For IMP5-17, the diagonal providing the best coding performance for the *All-Intra* and *Random-Access* configurations differs, and the chosen compromise is $D^0_{sig} = 2$. The compromise for IMP5-18 is $D^1_{sig} = 6$, which provides, in total, a significant coding efficiency improvement relative to IMP5-16* that employs a single context model set. IMP5-18 with $D^0_{sig} = 2$ and $D^1_{sig} = 6$ is referred to as IMP5-18* and serves as the basis for the last investigation in this chapter.

As in the preceding investigation on the context modeling of b_{sig} with additional context model sets, a second context model set for coding of the $b_{|x|>1}$ and $b_{|x|>2}$ flags is selected when $D(x,y) < D_{gtX}$. This time, however, only one configuration is tested, because the obtained improvement in coding efficiency is relatively small compared to the previous investigation. IMP5-19 of this investigation is based on IMP5-18*, which is also used to calculate the BD-rates. Figure 5.11 summarizes the experimental results and demonstrates that further coding efficiency improvement is also achievable by using an additional context model set for the coding of $b_{|x|>1}$ and $b_{|x|>2}$. For the sake of convenience, IMP5-19 with $D_{gtX} = 2$ is referred to as IMP5-19* in the remainder of this chapter.

Summary of the Investigations on Position-Dependent Context Model Sets

Both investigations on additional context model sets for the context-coded flags provide coding efficiency improvements relative to a corresponding variant using a single context model set for each flag. It should

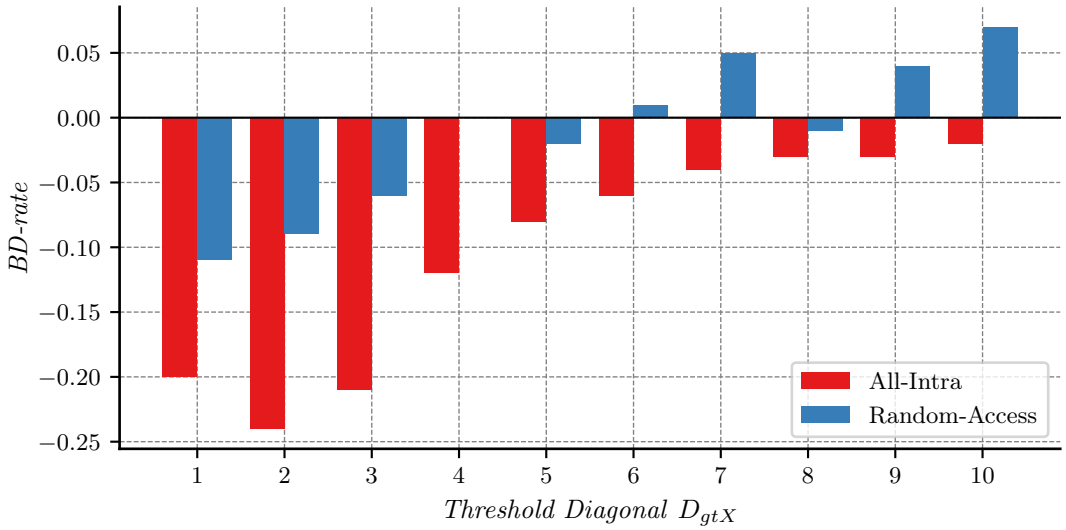

Figure 5.11
Coding efficiency of the investigation on the switch to a different context model sets for $b_{|x|>1}$ and $b_{|x|>2}$ depending on the diagonal $D(x,y) = x + y$ (IMP5-19). A second context model set is selected for coding the $b_{|x|>1}$ and $b_{|x|>2}$ flags when $D(x,y) < D_{gtX}$.
Anchor for BD-rate computations: IMP5-18*

be noted that the investigations were conducted for luma only, and using additional context model sets for chroma might provide further coding efficiency improvements. Moreover, different settings of the thresholds in the *All-Intra* and the *Random-Access* configurations, respectively, could be investigated in future research. It is even conceivable to apply the concept for the Rice parameter estimation, e.g., by using different values for the thresholds $M_{Rice}^{k=1}$, $M_{Rice}^{k=2}$, and $M_{Rice}^{k=3}$ for specific areas of the transform block.

Assessment of Dedicated Context Models for 4 × 4 Transfrom Blocks

While discussing the implementation IMP5-1 in section 5.3, where the impact of a neighboring frequency location was investigated, it was suspected that a dedicated context model set for 4×4 transform blocks is not required when using the local template for context modeling. Two coding experiments were conducted to support this hypothesis. In both coding experiments, dedicated context model sets for b_{sig}, $b_{|x|>1}$, and $b_{|x|>2}$ are employed for transform blocks with a width or a height smaller than or equal to four. IMP5-20 of the first experiment is based on IMP5-16*, which represents the condition before introducing the position-dependent context model sets. IMP5-21 of the second experiment is based on IMP5-19* and represents the condition after introducing the position-dependent context model sets. The coding performance of both implementations is virtually the same compared to the corresponding basis implementation, which indicates that the hypothesis is accurate.

5.4.5 | Reported Implementation and Performance in VVC

A context modeling that evaluates the absolute levels inside the local template was presented to the *Joint Video Experts Team* (JVET) during the development of VVC in [43]. The VTM software version employed for the coding experiments in [43] is version 1.0 (VTM-1), which implements the level coding of HEVC. Table 5.1 summarizes the BD-rates reported to the JVET for the template-based level coding that analyzes absolute transform coefficient levels inside the template in [43], relative to the implementation in VTM-1, and with RDOQ enabled. A comparable investigation using the implementations presented in this chapter is between the configuration IMP5-19* and IMP5-0, but without RDOQ. In the *All-Intra* configuration, a BD-rate of -1.55% was measured, and -1.20% was measured for the *Random-Access* configuration, respectively. Both BD-rates are close to the corresponding numbers reported in [43]. A direct comparison between the level coding implemented in VTM-17 and IMP5-19* with the encoders configured according to the description in

Class	Luma	C_B	C_R
	All-Intra		
A1	-1.32%	-1.19%	-1.68%
A2	-1.67%	-0.63%	-2.22%
B	-1.60%	-1.86%	-1.33%
C	-1.93%	-1.77%	-1.84%
E	-2.25%	-2.71%	-2.46%
Overall	**-1.75%**	**-1.67%**	**-1.84%**
D	-1.98%	-1.47%	-1.36%
	Random-Access		
A1	-1.06%	-0.78%	-0.91%
A2	-0.99%	-0.15%	-1.51%
B	-1.11%	-1.57%	-0.79%
C	-1.42%	-1.29%	-1.10%
Overall	**-1.16%**	**-1.05%**	**-1.04%**
D	-1.39%	-0.77%	-0.66%

Table 5.1
Coding efficiency of a template-based context modeling proposed in [43] that evaluates absolute transform coefficient levels in VTM-1. The anchor for BD-rate computations is the level coding in VTM-1, which corresponds to that specified for HEVC.
Anchor for BD-rate computations: VTM-1

section 5.2.4, i.e., with EP initialized context models and without RDOQ, IMP5-19* achieves BD-rates of -0.18% in the *All-Intra* configuration and 0.19% in the *Random-Access* configuration.

It should be noted that the parameters for binarization and context modeling of IMP5-19* and [43] are different. The discrepancies are supposedly due to coding tools not present in VTM-1 but included in VTM-17, such as multiple transform set or matrix-based intra prediction, and their presence leads to changes in the statistics for the transform coefficient levels. The main differences are:

- Coding of additional context-coded flags $b_{|x|>3}$ and $b_{|x|>4}$ in [43], while the presented implementations do not include these two context-coded flags.
- Different context model sets are used for $b_{|x|>1}$, $b_{|x|>2}$, $b_{|x|>3}$, and $b_{|x|>4}$ in [43], while the presented implementations employ the same context model sets for $b_{|x|>1}$ and $b_{|x|>2}$.
- Context modeling of the $b_{|x|>1}$, $b_{|x|>2}$, $b_{|x|>3}$, and $b_{|x|>4}$ flags in [43] uses $x\,(i) - 1$ instead of $x\,(i)$ as in IMP5-19*.
- Rice parameter estimation in [43] relies on $x\,(i) - 1$ instead of $x\,(i)$ as in IMP5-11*.
- Clipping thresholds have different values, and the implementation of [43] utilizes more position-dependent context model sets in luma and chroma for all context-coded flags.

5.5 | Findings and Technical Achievements

The findings and achievements of the presented context modeling based on the analysis of the absolute transform coefficient levels inside a local template can be summarized as follows:

- A significant coding efficiency improvement (in the context of level coding) is achieved with the template-based context modeling and Rice parameter selection.
- The introduced configuration of the template-based context modeling has a limited complexity impact on practical implementations since the absolute levels inside a template are analyzed only once.
- An improved coding efficiency relative to the initial implementation IMP5-0 is achieved with fewer context models. In the initial variant IMP5-0, the context modeling corresponds to the implementation presented in chapter 3 and the binarization corresponds to the implementation presented in chapter 4.

- Further coding efficiency improvements can be achieved by additional context model sets for the context-coded bins. The chosen context model set depends on the diagonal of the scanning position.

The performed investigations revealed statistical dependencies that were not further pursued in this thesis, but preliminary analyses indicate that they can be exploited to achieve higher coding efficiency. These findings can be analyzed in-depth when developing a level coding for a video coding technology beyond VVC. They can be summarized as follows:

- The impact of a neighboring frequency location is depending on the spatial distance, as indicated by the analysis of the impact of a single neighboring frequency location. This variation may depend on the coding tools used for prediction and transform, the input signal itself, and could be exploited by utilizing a weighted sum. An extensive analysis in conjunction with the configuration of the context quantizer is necessary to find a weighting configuration that could provide further coding efficiency improvement.
- Treating the unavailable neighboring frequency locations as zero-valued can be interpreted as biasing the statistic. Considering the unavailable locations in the context modeling can improve the coding efficiency, as indicated by the experimental results in this chapter.
- The position-dependent context model sets for the context-coded flags, which depend on the diagonal within the transform block of the current scanning position, turned out to be efficient in this chapter. Investigations for an adaptive variant, where alternative context model sets are selected depending on the absolute levels located at the preceding diagonal, may provide further coding efficiency. Furthermore, this concept can be extended to the Rice parameter selection. For example, different thresholds are employed for different areas within a transform block, and the maximum Rice parameter may differ depending on the area.

A template-based context modeling for coding the significance flags of transform blocks larger than 8×8 samples was used for the level coding in the Fraunhofer HHI response [39] to the *Call for Proposals* [40] that initiated the development of HEVC. The coding efficiency for that template-based context modeling was further analyzed in [68]. It was included in the first Test Models, from the initial Test Model under Consideration (TMuC) [41] to HM version 6, inclusively, but was later replaced by a simplified variant that evaluates the number of absolute levels greater than one located in the neighboring sub-blocks right and below the current sub-block [69]. Note that the context modeling of the significance flags in [69] can be regarded as a variation of the concept developed in chapter 3. A template-based context modeling that utilizes absolute levels was later proposed to be included in the HEVC standard in [42]. Even though the proposal successfully demonstrated that coding efficiency improvements could be achieved by exploiting statistical dependencies between transform coefficient levels, the technology was not adopted into the HEVC standard because of complexity concerns. During the exploration phase for video coding technologies beyond HEVC, a software package, referred to as *Joint Exploration Model* (JEM), served as the basis for coding tool experiments. The level coding in JEM was derived from the level coding proposed in [42] with some refinements. An improved template-based level coding utilizing absolute levels was employed in the Fraunhofer HHI response [83] to the *Call for Proposals* [84] that initiated the development of VVC. The level coding in VVC [56] employs a template-based level coding that is based on the basic concept presented in this chapter. Modifications to the level coding described in [83] that led to the level coding design in VVC are presented in the following chapter.

5.6 | Chapter Summary

This chapter presented an approach for level coding that provides improved coding efficiency relative to the initial design developed in the chapter 3 and chapter 4. In the first step, an improved coding efficiency is achieved by a template-based context modeling that analyzes the already coded b_{sig} flags inside the local template. In the next step, the coding order within a sub-block is modified, resulting in a single coding phase, where the level magnitudes are transmitted completely for each scanning position instead of in multiple iterations for each sub-block. This modification, in turn, enables the analysis of the absolute transform

coefficient levels inside the local template, which further improves the coding efficiency when applying it to the context modeling of the context-coded flags and the Rice parameter selection. In the final step, additional context model sets are introduced for the context-coded flags, where the selection of a context model set depends on the diagonal of the current scanning position. The concepts used for the level coding presented in this chapter served as the basis for an implementation proposed initially for the VVC [64] development [43]. Although the final level coding of VVC differs from the level coding regarding some details presented in this chapter, the basic concepts presented in this chapter can be found in the level coding of VVC.

CHAPTER 6

Level Coding Suitable for Trellis-Coded Quantization

Contents

Both the *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC) and the *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC) standards specify *uniform reconstruction quantization* (URQ) for the reconstruction of transform coefficients. In URQ, the reconstruction of a transform coefficient depends on the associated quantization index (or transform coefficient level) and the quantization step size. A transform coefficient can be reconstructed independently of other transform coefficients within the block, and the reconstruction of transform coefficients is independent of the level coding.

The coding efficiency of scalar quantization, however, highly depends on how an encoder selects the quantization indices. A simple quantization algorithm would only minimize the distortion by selecting the closest quantization level to the value obtained after dividing the transform coefficient by the quantization step size. In order to achieve higher coding efficiency, it is beneficial to additionally consider the codeword lengths or bit-rates yielded by entropy coding during quantization. More sophisticated scalar quantization algorithms that consider both the distortion and the codeword lengths are referred to as *rate-distortion optimized quantization* (RDOQ) [85, 86, 87]. The quantization levels providing the lowest Lagrangian cost $D + \lambda R$ for a transform block are selected in RDOQ algorithms, with D being the distortion, R the bit-rate, and λ a fixed Lagrangian multiplier, which depends on the selected *quantization parameter* (QP).

Compared to scalar quantization, significantly higher coding efficiency can be achieved with vector quantization [88]. Unfortunately, due to its complexity, unconstrained vector quantization is impractical for real-world applications.

This chapter is based on the concept of *trellis-coded quantization* (TCQ) [89], which provides a significant coding efficiency improvement over scalar quantization with RDOQ. In contrast to other constrained vector quantization algorithms, the existing level coding can be reused in TCQ due to its resemblance to URQs.

Even though TCQ could be implemented in combination with the level coding specified in HEVC, several key elements of that design impede the full potential provided by TCQ. From the review of TCQ in this chapter, it becomes clear why the level coding developed in chapter 5 is more suitable for TCQ.

In the first part of this chapter, a brief review of scalar quantization and RDOQ is presented. This review is followed by a description of TCQ, its implementations for both encoders and decoders, and the variant proposed and implemented in *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC). The works developed in this thesis begin with section 6.3 and start with a modification of the context modeling for the b_{sig} flags for TCQ. In the following section 6.4, a modified level coding for the design presented in chapter 5 is developed, which separates the coding of context-coded and bypass-coded bins to reduce implementation efforts. This separation is achieved via different coding phases while the design preserves its compatibility with TCQ. In a final investigation presented in section 6.5, the limitation on the number of context-coded bins is studied. This investigation leads to a level coding with a reduced worst-case number of context-coded bins.

6.1 | Scalar Quantization

Two classes of quantization algorithms exist: Scalar quantization [90, 74] and vector quantization [88, 91]. Until the VVC standard, scalar quantization has been used in all practical hybrid video coding applications by specifying URQ for the reconstruction of transform coefficients. Encoders can improve the coding performance by implementing more enhanced scalar quantization algorithms, such as considering the bit-rates in RDOQ algorithms.

From the decoder point-of-view, the reconstruction of transform coefficients within a transform block can generally be denoted by a mapping:

$$Q' : \mathsf{J} \to \mathbb{R}^N, \tag{6.1}$$

where J is an index set. For scalar quantization, J can be written as:

$$\mathsf{J} = J_1 \times \cdots \times J_N. \tag{6.2}$$

The reconstruction of transform coefficients in scalar quantization is given by:

$$Q'(x_1, \ldots, x_N) = \left(Q'_1(x_1), \ldots, Q'_N(x_N)\right), \tag{6.3}$$

where

$$Q'_k : J_k \to \mathbb{R}, \quad k \in \{1, \ldots, N\}. \tag{6.4}$$

Equation (6.3) and equation (6.4) signify that the transform coefficient levels of a block can be reconstructed independently of each other. All reconstruction rules that do not correspond to equation (6.3) and equation (6.4) are referred to as vector quantization. An independent reconstruction of transform coefficients according to equation (6.3) is not possible for vector quantization.

6.1.1 | Reconstruction of Transform Coefficients

The only parameter necessary for reconstructing a transform coefficient c' in URQ is the quantization step size Δ. If x is a quantization index or transform coefficient level, then c' is reconstructed as:

$$c' = Q'(x) = x \cdot \Delta. \tag{6.5}$$

This reconstruction rule is specified in both the AVC and HEVC standards.

6.1.2 | Simple Quantization Algorithm

Let $\operatorname{sgn}(\cdot)$ be the signum function. An example of a simple quantization for a transform coefficient c is given by:

$$x = \lfloor \frac{|c|}{\Delta} + a \rfloor \cdot \operatorname{sgn}(c). \tag{6.6}$$

In this example, a quantization level is selected by adding a rounding offset $a \in [0, 0.5]$ to $|c|/\Delta$ and rounding down the result to the next integer.

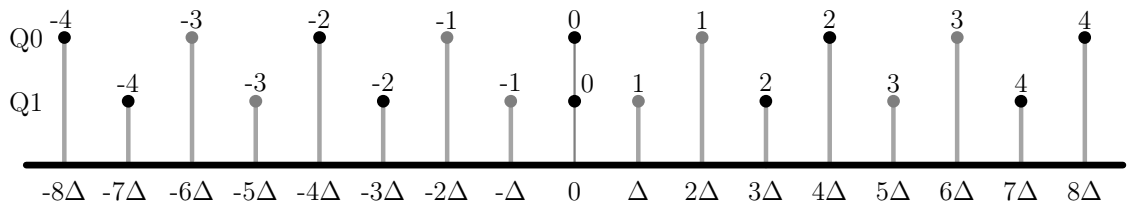

Figure 6.1
Quantizer configuration in TCQ with the two scalar quantizers Q_0 and Q_1. Both quantizers include the reconstruction value equal to zero, and even multiples of the quantization step size Δ are assigned to Q_0, whereas odd multiples of Δ are assigned to Q_1. The numbers above each dot denote the quantization indices associated with the corresponding reconstruction values.

6.1.3 | Rate-Distortion Optimized Quantization

The idea of RDOQ is to select a set of quantization levels for a transform block that minimizes the Lagrangian cost $D + \lambda R$. An actual implementation highly depends on the used entropy coding. The following review is based on the RDOQ algorithm implemented in the VTM reference encoder.

For each scanning position, the cost for two candidate quantization levels $x^- = \operatorname{sgn}(c) \cdot \lfloor |c|/\Delta \rfloor$ and $x^+ = \operatorname{sgn}(c) \cdot \lceil |c|/\Delta \rceil$ should be different. The algorithm selects the candidate with the lower rate-distortion cost. After determining the quantization levels for all scanning positions, further optimizations are performed, such as testing different scanning positions as the last significant scanning position, evaluating the uncoded case, where all quantization levels are zero-valued, and more. Because the codeword lengths for all candidate quantization levels are necessary to calculate the Lagrangian cost, the level coding has to be emulated during quantization.

6.2 | Trellis-Coded Quantization

The structural constraints of transform coding, i.e., an orthogonal transform followed by scalar quantization, cannot achieve the fundamental rate-distortion bound. The cause behind this circumstance is that the set of representable blocks containing N samples forms an orthogonal lattice in the N-dimensional space [91]. This efficiency gap can only be reduced by incorporating a vector quantization algorithm, where unconstrained approaches are too complex for practical implementations. To this end, TCQ represents a low-complex vector quantization that can be embedded into the existing transform coding pipeline with limited modifications.

6.2.1 | Design Overview

Conceptually, TCQ consists of two scalar quantizers, denoted as Q_0 and Q_1 in the following description, and a mechanism for switching between the two quantizers. In the TCQ implementation of VVC, both quantizers include the reconstruction value equal to zero, and Q_0 further includes even multiples of the quantization step size Δ, whereas Q_1 further includes odd multiples of Δ. This quantizer configuration is illustrated in figure 6.1 with the associated reconstruction values for a given quantization step size Δ.

The switching between the two quantizers in TCQ is realized via a *finite-state machine* (FSM) that requires a specification of the number of states and the state transitions. Half of the states defined by the FSM are assigned to the quantizer Q_0 and the other half to the quantizer Q_1. The number of states is determined by the parameter $K \geq 1$, which results in 2^K states. Given the number of states, a definition for the transitions from one state to the other states is finally necessary to complete a TCQ design. On the right of figure 6.2, an example of a four-state FSM is illustrated, where a node represents a state, and the arrows represent state transitions. A further property of TCQ is that the coding efficiency increases with the number of states used in an actual implementation.

$S(i)$	$Q(i)$	$S(i-1)$	
		$u_i = 0$	$u_i = 1$
S_0	Q_0	S_0	S_2
S_1	Q_0	S_2	S_0
S_2	Q_1	S_1	S_3
S_3	Q_1	S_3	S_1

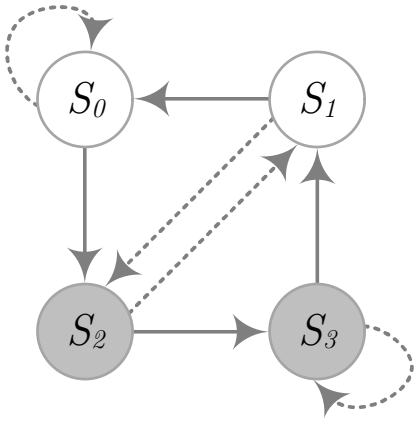

Figure 6.2
On the left: State-transition table summarizing the FSM used in the implemented four-state TCQ. On the right: Graphical representation of the four-state FSM used in TCQ. For each transform block, the initial state is S_0, and the dotted lines represent the state transitions when the parity $u_i = 0$, whereas solid lines denote the state transitions for $u_i = 1$. The assignment of a state to one of the two quantizers is given by the background shading of the state nodes, where white-shaded states represent Q_0 and gray-shaded states represent Q_1.

6.2.2 | TCQ Implementation in VVC

The TCQ implementation in VVC utilizes four states, which represents a reasonable trade-off between coding efficiency and complexity. The state for the first scanning position of the transform block is $S(i) = S_0$. For the remaining scanning positions, the state $S(i)$ is determined by the state $S(i+1)$ and the parity of the quantization level $x(i+1)$ for the preceding scanning position $i+1$. This means that even quantization levels, i.e., for the parity $u_i = x(i) \bmod 2 = 0$, and odd quantization levels, i.e., for $u_i = x(i) \bmod 2 = 1$, result in different transitions. The state transitions for the four-state FSM are illustrated on the right of figure 6.2 by solid and dotted arrows. The solid arrows denote transitions for $u_i = 1$, and dotted arrows denote transitions for $u_i = 0$.

Although conceptually not necessary, the scanning pattern used for reconstructing the transform coefficients is the same as in the level coding. This design choice enables the derivation of the quantization states already during entropy coding. Furthermore, the bit-rate estimation necessary for the quantization at the encoder side is more practicable when using the same scanning pattern for the reconstruction of transform coefficients and the entropy coding.

Reconstruction of Transform Coefficients

To reconstruct a transform coefficient $c'(i)$ at the scanning position i, the quantization step size Δ, the quantization level $x(i)$, and the state $S(i)$ are necessary. Instead of using a quantization level x as the input for the reconstruction of c', an intermediate quantization level x' is derived first according to:

$$x' = \left(2 \cdot |x| - \left\lfloor \frac{S(i)}{2} \right\rfloor\right) \cdot \operatorname{sgn}(x). \tag{6.7}$$

The existing reconstruction process of c' can be reused with x' instead of x as the input, i.e., $c' = x' \cdot \Delta$. When TCQ is disabled, $S(i) = S_0$ for all scanning positions i of the transform block, resulting in the usage of Q_0 only. In this case, the reconstruction of c' is analogous to the URQ case with the quantization step size 2Δ. In summary, a TCQ support on the decoder side requires the derivation of x' and a state-dependent context model selection, which will be discussed after reviewing the quantization with TCQ on the encoder side.

Quantization of Transform Coefficients

The assignment of states to quantizers in VVC is summarized by the table on the left in figure 6.2. In this table, it is denoted that Q_0 is used when $S(i) \in \{S_0, S_1\}$ and Q_1 is used when $S(i) \in \{S_2, S_3\}$. This assignment is coupled to the reconstruction of the transform coefficients denoted in equation (6.7).

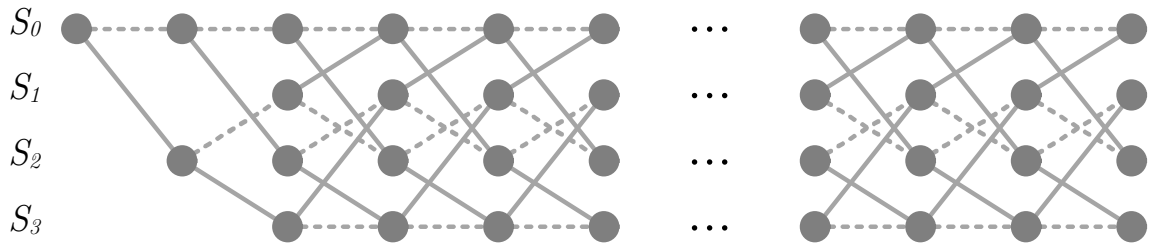

Figure 6.3
Example of a four-state trellis, where each transition (dotted for $u_i = 0$ and solid for $u_i = 1$) is associated with a rate-distortion cost $C = D + \lambda R$. An encoder implementing TCQ selects the path with the lowest total rate-distortion cost, which is the sum of the costs for each edge along the path.

For the quantization process, the level coding has to be emulated to estimate the codeword lengths. Furthermore, the correct quantization states are necessary for calculating the distortion. Therefore, using the same scanning pattern for reconstructing transform coefficients and entropy coding is reasonable, because it simplifies the whole design. All potential quantization options for a transform block can be represented as paths through a trellis. Figure 6.3 exemplarily illustrates the trellis for the implemented four-state TCQ. The state transitions from one scanning position to the next scanning position correspond to those specified by the FSM and are associated with a rate-distortion cost. The set of quantization levels that provides the lowest rate-distortion cost corresponds to a certain path through the trellis. This path can be found by utilizing the Viterbi algorithm [92] when ignoring some minor dependencies in the entropy coding.

Context Modeling in TCQ

The distance between two quantization levels is always 2Δ for Q_0. For Q_1, it is 2Δ except for the quantization interval around zero, where it is Δ, as illustrated in figure 6.1. This quantization configuration results in a higher probability for non-zero transform coefficient levels when Q_1 is used compared to the case that Q_0 is used. Therefore, it is reasonable to determine the context models for the b_{sig} flags depending on the quantizer used. Because the context modeling now depends on the used quantizer, which can be derived by $S(i)$, the state derivation has to be performed already during the level coding to select the correct context models for b_{sig}. From this relationship, it becomes clear that the single coding phase variant developed in chapter 5 is suitable for TCQ, because the parity of the level at the preceding scanning position is available before coding the b_{sig} flag of the current scanning position. This parity information is not available in the HEVC level coding that uses multiple coding phases to transmit the level magnitudes.

Let δ^*_{sig} denote the context model index derived after analyzing the neighboring frequency locations inside the template and the diagonal of the current scanning position. Furthermore, let $|\mathcal{C}^*_{sig}|$ be the number of context models used for coding the b_{sig} flags without TCQ. Then, the context model index for the b_{sig} flags is calculated by:

$$\delta_{sig} = \delta^*_{sig} + \begin{cases} 0, & \text{if } S(i) \in \{S_0, S_1\}, \\ |\mathcal{C}^*_{sig}|, & \text{otherwise.} \end{cases} \tag{6.8}$$

Statistical investigations indeed showed that the probability for a non-zero valued scanning position is significantly higher when the current state is $S(i) \in \{2, 3\}$ compared to $S(i) \in \{0, 1\}$. Actual coding experiments confirm that context modeling with different context model sets for the b_{sig} flags, as denoted in equation (6.8), provide higher coding efficiency. In VTM-17, the variant with two context model sets provides -1.36% in *All-Intra* and -1.27% in *Random-Access*, compared to an implementation without state-dependent context model sets. Further investigations analyzing the same approach for other context-coded flags indicated that a slight improvement in coding efficiency is achieved for the $b_{|x|>1}$ flags, while the coding efficiency remains virtually the same for $b_{|x|>2}$.

Class	Luma	C_B	C_R
	All-Intra		
A1	-3.16%	-2.48%	-1.63%
A2	-3.88%	-1.63%	-0.91%
B	-3.30%	-0.05%	0.06%
C	-3.16%	0.99%	0.89%
E	-2.66%	0.92%	0.61%
Overall	**-3.31%**	**-0.32%**	**-0.11%**
D	-3.16%	1.46%	1.40%
	Random-Access		
A1	-2.25%	-1.67%	-0.56%
A2	-1.98%	-1.09%	-1.28%
B	-2.62%	-0.32%	-0.27%
C	-2.09%	-0.58%	-0.86%
Overall	**-2.27%**	**-0.81%**	**-0.69%**
D	-2.09%	0.67%	-0.75%

Table 6.1
Coding efficiency of TCQ on top of the template-based context modeling proposed in [43] in VTM-1. The anchor for BD-rates computations is the template-based level coding also proposed in [43] with RDOQ enabled.
Anchor for BD-rate computations: VTM-1

6.2.3 | Coding Performance of TCQ in VVC

The presented four-state TCQ was implemented on top of the template-based level coding and was presented to the *Joint Video Experts Team* (JVET) during the development of VVC in the same input document [43]. Table 6.1 summarizes the *Bjøntegaard delta bit-rate*s (BD-rates) reported to the JVET for the four-state TCQ relative to the template-based level coding in [43], i.e., the improvement is on top of the level coding. Note that in this comparison, the anchor implementation used for BD-rate computations has RDOQ enabled.

In a late development phase of VVC, an eight-state TCQ implementation was presented in [93], which yields a coding efficiency improvement of -0.42% in the *All-Intra* and -0.34% in the *Random-Access* configurations relative to the four-state TCQ implementation.

6.3 | Extended Context Modeling for TCQ

Given the improved coding efficiency provided by the state-dependent context model sets, the question arises whether additional context model sets can achieve further improvements. In-depth statistical analyses were carried out with the level data extracted from bitstreams coded with TCQ enabled. It was discovered that the probability distribution of b_{sig} for S_2 is different than for S_3. This discovery suggests that the variant with two context model sets denoted in equation (6.8) could be improved by adding a third context model set. Let $|\mathcal{C}^*_{sig}|$ be again the number of context models used for coding the b_{sig} flags without TCQ, and δ^*_{sig} denote the context model index derived by evaluating the template. Then, the context model index for the b_{sig} flags for the additional context model set is calculated by:

$$\delta_{sig} = \delta^*_{sig} + \begin{cases} 0, & \text{if } S(i) \in \{S_0, S_1\}, \\ |\mathcal{C}^*_{sig}|, & \text{if } S(i) = S_2, \\ 2 \cdot |\mathcal{C}^*_{sig}|, & \text{otherwise.} \end{cases} \tag{6.9}$$

The first conducted coding experiment is based on VTM-17 and investigates the coding efficiency of the three state-dependent context model sets denoted in equation (6.9) relative to two state-dependent sets according to equation (6.8). All context models related to level coding in both the anchor and the tested candidate were initialized as *equi-probable* (EP). This implementation with two state-dependent context model sets for the context modeling of b_{sig} is referred to as IMP6-0, while the implementation utilizing three state-dependent context model sets is referred to as IMP6-1. For the *All-Intra* configuration, using three context model sets according to equation (6.9) results in 0.09% bit-rate overhead, and for the *Random-Access* configuration, the bit-rate overhead is 0.11%. These experimental results do not match the statistical findings.

Class	Luma	C_B	C_R
	All-Intra		
A1	-0.25%	-0.20%	-0.01%
A2	-0.32%	-0.28%	-0.10%
B	-0.10%	-0.03%	-0.09%
C	-0.06%	-0.14%	0.16%
E	0.00%	-0.33%	-0.24%
Overall	**-0.14%**	**-0.17%**	**-0.05%**
D	-0.08%	0.04%	-0.68%
	Random-Access		
A1	-0.09%	0.09%	-0.17%
A2	-0.09%	-0.44%	-0.12%
B	-0.08%	-0.13%	0.29%
C	-0.07%	-0.37%	-0.38%
Overall	**-0.09%**	**-0.21%**	**-0.06%**
D	0.05%	-0.90%	-0.34%

Table 6.2
Coding efficiency of the investigation on an additional state-dependent context model set for the coding of the b_{sig} flags (IMP6-1).
Anchor for BD-rate computations: IMP6-0

A detailed analysis of the BD-rates revealed coding efficiency improvements for test sequences with relatively high bit-rates, such as ultra-high-definition content coded in the *All-Intra* configuration. This observation indicates that the initialization of the context models as EP negatively impacts the coding performance. Different methods for determining initial values of context models were investigated in [94, 95, 96, 97]. The initialization values were determined according to the first method described in [95] for both the anchor and the variant with three state-dependent context model sets. This time, the introduction of the third state-dependent context model set for the b_{sig} flags provides improved coding efficiency, which is summarized in table 6.2. These experimental results demonstrate that any context modeling optimization with TCQ should derive updated initialization values for the context models, because the performance highly depends on the initialization of the context models.

6.4 | Separation of Context- and Bypass-Coded Bins

For high bit-rate operation points, the throughput of the entropy coding engine can be improved significantly when a high amount of bypass-coded bins are processed successively [72]. The level coding plays an important role in the throughput at high bit-rate operation points, because most of the bins belong to transform coefficient levels, as shown in section 4.2.2.

A possibility to improve the throughput in this context is to separate the coding of context-coded and bypass-coded bins, i.e., to split the single coding phase into at least two coding phases [72]. In the case of two coding phases, the first coding phase contains context-coded bins, whereas the second coding phase transmits the remaining level magnitudes in bypass mode. These two coding phases differ from those initially used in the level coding for AVC, as described in section 3.2. In AVC, the b_{sig} flags are coded in the first coding phase with context models, whereas the second coding phase transmits the remaining level magnitudes. This second coding phase involves an interleaving of context-coded and bypass-coded bins, which should be avoided for higher throughput.

Although a transmission of coefficient levels in two coding phases would increase the number of bypass-coded bins that can be coded successively, the compatibility with TCQ would become broken due to the unavailability of the parity information in the first coding phase. Particularly, the level magnitude has to be reconstructed to determine the parity. However, an absolute level may become available only after reconstructing the remaining magnitude, which is coded in the second phase. Therefore, the determination of the context model set for the coding of the b_{sig} flag with TCQ enabled, as denoted in equation (6.9), becomes impossible.

6.4.1 | Solution and Constraints

The chosen solution presented in this thesis is to code the parity of the levels as a dedicated flag, denoted by b_{par} in the following description. This b_{par} flag has to be coded in the first coding phase so that the parity becomes available for the context modeling of the b_{sig} flag. Furthermore, the chosen level coding design should provide a competitive coding efficiency relative to the existing design when TCQ is disabled. Because the intention is to have phases with context-coded or bypass-coded bins only, the b_{par} flags have to be context-coded. Let $|x|_{max}^{par}$ be the maximum value that can be represented by the flags coded before transmitting the b_{par} flag. Then, a consequence of the b_{par} flag is that the remaining absolute magnitude is halved, i.e., only $\lfloor \max(0, (|x| - |x|_{max}^{par})/2) \rfloor$ has to be transmitted after coding the b_{par} flag.

Even though the chosen solution seems straightforward, aspects regarding hardware implementations should already be considered during the development. Firstly, a binarization with b_{par} flag increases the *maximum number of context-coded bins per sample* (mcps) of the levels by one bin from three to four bins. Secondly, a direct dependency between successively context-coded bins should be avoided by design, i.e., the context modeling of a bin should not directly depend on the value of the preceding bin. Both aspects, when considered already during the development, could ease practical hardware implementations [72]. Finally, the coding efficiency provided by the binarization with b_{par} flags should be similar to that of the existing level coding. Unfortunately, there is no optimal solution for the chosen design, as indicated by the results provided at the end of this section, and all available options only form a compromise among the mentioned aspects.

6.4.2 | Level Coding with Parity Flag

The b_{par} flag can be transmitted before coding the b_{sig} flag, after coding the b_{sig} flag, after the $b_{|x|>1}$ flag, or after the $b_{|x|>2}$ flag. When coding the b_{par} flag before the b_{sig} flag, $b_{par} = 1$ indicates that the level is non-zero, while $b_{par} = 0$ allows no statement whether the level is zero-valued or not. Although all four options have the same mcps, this option should result in the highest *context-coded bins per sample* (cps) on average, because it requires two context-coded bins to signal a zero-valued transform coefficient level. Therefore, this binarization option is not considered for further investigations.

Binarization Options

Each of the three remaining binarizations for the b_{par} flag results in different bin strings, which is summarized in table 6.3 for the context-coded bins. Each of the options for placing the b_{par} flag is referred to as *binarization* $\#n$ in the following description, with $n \in \{1, 2, 3\}$. From table 6.3, it can be observed that a binarization $\#n$ increases the number of necessary context-coded bins for the absolute level $|x| = n$ by one bin relative to a straightforward unary binarization without b_{par} and three context-coded bins. This property also explains the significantly higher cps when coding b_{par} before b_{sig}, because most of the coded levels are zero-valued, as demonstrated by the histograms illustrated in figure 3.2 of chapter 3. To avoid a significant increase in the cps, the optimal binarization should be $\#3$, i.e., coding the b_{par} flag after the signaling of $b_{|x|>2} = 1$. However, this option has the disadvantage that the context modeling of the b_{sig} flags relies on the value of the directly preceding bin, which is the b_{par} flag of the preceding scanning position. As already mentioned, such a dependency should be avoided, and in most cases, such a dependency does not exist for the template-based context modeling.

Let $|x|_{max}^{1}$ denote the absolute values that can be represented if the remainder, which is signaled after the first coding phase, is equal to zero. Its actual value depends on the value of the b_{par} flag and represents the maximum magnitudes that can be reconstructed for a scanning position after the context-coded phase. Without changing the binarization, i.e., without the b_{par} flags, one has $|x|_{max}^{1} = 3$ when separating the single coding phase into one context-coded phase and one bypass-coded phase. When switching over to a binarization that includes the b_{par} flag, the value of $|x|_{max}^{1}$ depends on the selected binarization. The values of $|x|_{max}^{1}$ for the different binarization with the b_{par} flags are delineated in the rightmost column of table 6.3. As can be seen from table 6.3, the values of $|x|_{max}^{1}$ increase the further forward the b_{par} flag is transmitted in the first coding phase. The optimal binarization that does not affect the template-based context modeling, described in section 5.4, is $\#1$. In this case, $|x|_{max}^{1}$ is not smaller than the clipping value applied to the sum of absolute levels inside the template for context modeling (see section 5.4.2). This insight represents an

Binarization #1					
$\|x\|$	b_{sig}	b_{par}	$b_{\|x\|>2}$	$b_{\|x\|>4}$	$\|x\|^1_{max}$
0	0				
1	1	1	0		
2	1	0	0		
3	1	1	1	0	
4	1	0	1	0	
5	1	1	1	1	*5*
6	1	0	1	1	*6*

Binarization #2					
$\|x\|$	b_{sig}	$b_{\|x\|>1}$	b_{par}	$b_{\|x\|>3}$	$\|x\|^1_{max}$
0	0				
1	1	0			
2	1	1	0	0	
3	1	1	1	0	
4	1	1	0	1	*4*
5	1	1	1	1	*5*

Binarization #3					
$\|x\|$	b_{sig}	$b_{\|x\|>1}$	$b_{\|x\|>2}$	b_{par}	$\|x\|^1_{max}$
0	0				
1	1	0			
2	1	1	0		
3	1	1	1	1	*3*
4	1	1	1	0	*4*

Table 6.3
Context-coded bins of the bin string and for the different binarizations with b_{par} and the corresponding $|x|^1_{max}$ values. The actual value of $|x|^1_{max}$ itself depends on the value of the corresponding b_{par} flag.

opposite effect to the earlier statement that the b_{par} flag should be coded as the last context-coded flag of the first coding phase to reduce the cps. Therefore, a first investigation was performed to analyze the impact of the limitation on $|x|^1_{max}$ for the template-based context modeling.

Limitation to Non-Fully Reconstructed Levels for Context Modeling

All coding experiments in this investigation were performed with TCQ disabled to analyze and encapsulate the effects of the modified binarization and context modeling. A further reason why TCQ is disabled is that supporting two different level coding paths, one for TCQ enabled and another for TCQ disabled, would be undesirable for practical implementations. For this reason and because of supporting encoder implementations that do not implement TCQ, the level coding with TCQ disabled has to provide a suitable coding performance. Furthermore, there is no reason to assume that a level coding providing an inferior coding efficiency for conventional quantization with URQs could achieve a reasonable performance when TCQ is enabled.

In IMP6-2 of this investigation, absolute transform coefficient levels of neighboring frequency locations inside the template were clipped to different $|x|^1_{max}$ values. As the implementation basis, as well as the anchor for BD-rates computations, the final implementation of the previous chapter, denoted as IMP5-19* in chapter 5, was used. In addition to the sum $\mathcal{S}$ formed by the absolute transform coefficient levels of the neighboring frequency locations inside the template (see equation (5.3)), the clipped sum $\mathcal{S}_c$ is calculated by:

$$\mathcal{S}_c = \sum_{i=0}^{T-1} \min\left(|x\,(i)|, |x|^1_{max}\right). \tag{6.10}$$

The sum $\mathcal{S}_c$ is then used as the input for context model selection without further modifications to the existing context modeling in IMP5-19* (see equation (5.4) and subsequent). For the Rice parameter selection, $\mathcal{S}$ is used instead, because the fully reconstructed absolute transform coefficient levels would be available in the second coding phase. Because the impact of the non-fully reconstructed absolute levels for context modeling

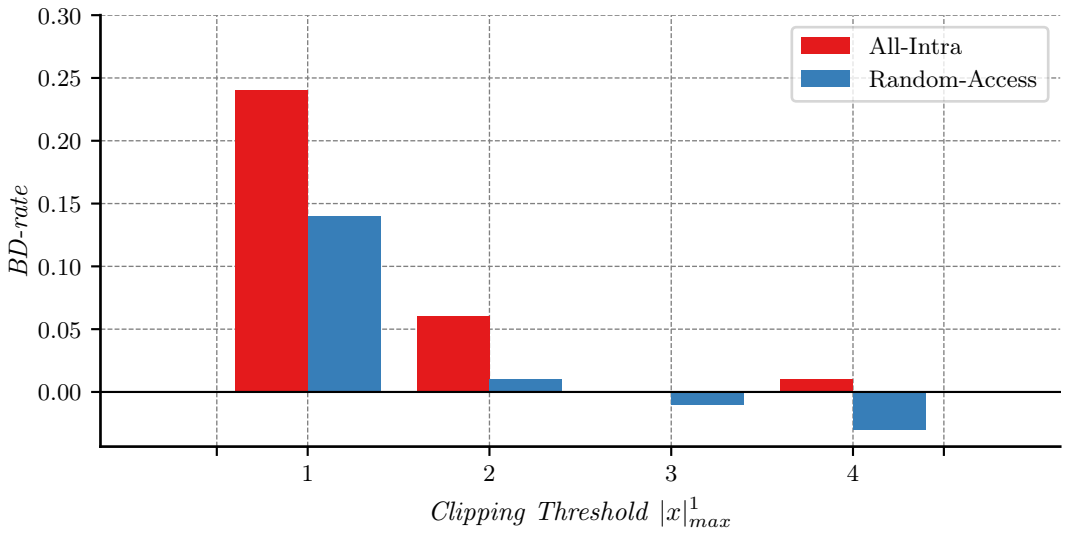

Figure 6.4
Coding efficiency of the investigation on clipping the template sum $\mathcal{S}$ to $|x|^1_{max}$ for the context modeling of all context-coded flags without changing the context modeling and the Rice parameter selection (IMP6-2). In IMP6-2, each neighboring absolute level inside the template is clipped to $|x|^1_{max}$.
Anchor for BD-rate computations: IMP5-19*

is investigated only, the level coding still employs a single coding phase, and the b_{par} flag is nonexistent, i.e., $|x|^1_{max}$ consists of a single value and is independent of b_{par}.

The results of this investigation are summarized in figure 6.4 for $|x|^1_{max} \in \{1, 2, 3, 4\}$. For $|x|^1_{max} > 4$, the coding efficiency is the same as for the reference IMP5-19*, because of the clipping of $\mathcal{S}$ to five before selecting a context model. The observed losses in coding efficiency are negligible for $|x|^1_{max} > 2$, which is unexpected compared to the improvements obtained by the investigations in section 5.4.3 on M_{sig} and M_{gtX}. The position-dependent context model sets described in section 5.4.4, introduced after the investigations in section 5.4.3, could be considered as the reason for the negligible losses in coding efficiency. Presumably, the introduction of additional context model sets allows the reduction of the clipping values M_{sig} and M_{gtX} without negatively impacting the coding efficiency. Given the experimental results, all binarizations with b_{par} under consideration are acceptable, and no loss in coding efficiency is achieved with binarization #1, where $|x|^1_{max} \in \{5, 6\}$.

This investigation shows that the impact on the existing context modeling is marginal for all candidate binarizations of b_{par}. However, more aspects that come with the b_{par} flags may negatively affect the coding performance, which is investigated next.

Coding Performance of Binarizations with b_{par}

In IMP6-3, IMP6-4, and IMP6-5 of this investigation, where IMP6-3 corresponds to binarization #1, IMP6-4 corresponds to binarization #2, and IMP6-5 corresponds to binarization #3, further aspects are analyzed by coding the b_{par} flags in the existing single coding phase. Again, as the implementation basis, as well as the anchor for BD-rate computations, IMP5-19* of chapter 5 was used. No modifications were made to the context modeling of the existing context-coded flags. Furthermore, the Rice parameter selection was kept unmodified, and all performed coding experiments had TCQ disabled. For the b_{par} flags, the same context modeling as implemented for $b_{|x|>1}$ and $b_{|x|>2}$ is used, but with dedicated context model sets for b_{par}. Due to the presence of b_{par}, the remaining absolute value after coding b_{par} is halved.

Figure 6.5 summarizes the results of the coding experiments conducted for this investigation. For all tested binarizations, the coding efficiency is worse than the anchor. However, the bit-rate overhead varies significantly depending on the binarization. The lowest bit-rate overhead in the *All-Intra* configuration is achieved

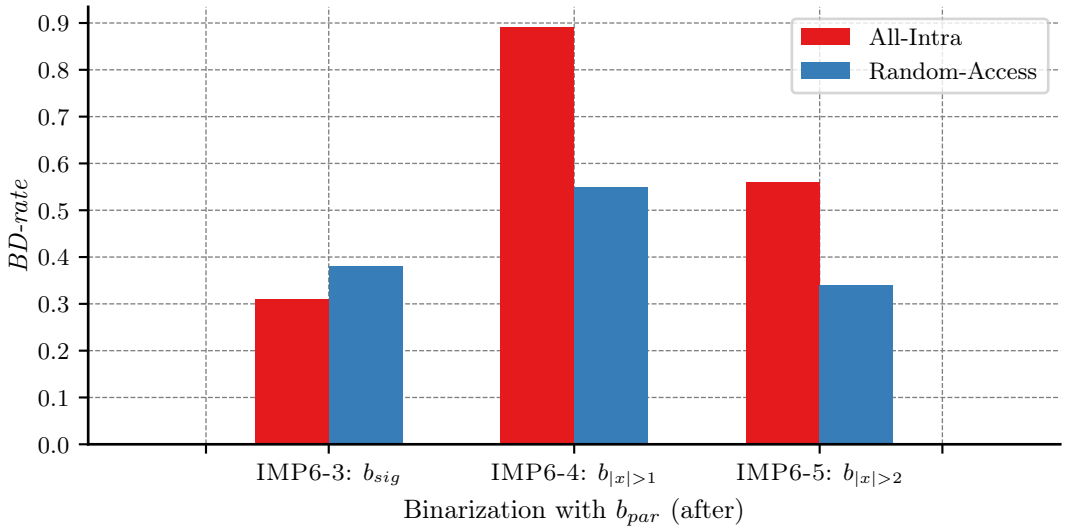

Figure 6.5
Coding efficiency of the investigations on different binarizations with the b_{par} flags, based on the level coding implementation with a single coding phase (IMP6-3 to IMP6-5).
Anchor for BD-rate computations: IMP5-19*

by IMP6-3 (coding b_{par} after b_{sig}), and the lowest bit-rate overhead in the *Random-Access* configuration is achieved by IMP6-5 (coding b_{par} after $b_{|x|>2}$). For both configurations, IMP6-4 (coding b_{par} after $b_{|x|>1}$) provides significantly higher bit-rate overheads than IMP6-3 and IMP6-5. Given the coding performance in both configurations, binarization #1 should be used to minimize the loss in coding efficiency. Note that these bit-rate overheads are not due to the context modeling, because all implementations use a single coding phase, i.e., the template-based context modeling is not altered.

Besides losses in coding efficiency, the cps has to be evaluated in the actual coding environment to assess the impact of the different binarizations with b_{par}. A statistical investigation is presented next to analyze the cps and to understand why binarization #2 provides the highest loss in coding efficiency. Furthermore, the investigation helps to study whether updating the Rice parameter selection is feasible.

Statistical Analysis of Data Samples from Bitstreams

Bitstreams generated by the coding experiments of the previous investigations were analyzed by extracting the number of bins and bits and comparing those numbers. It should be noted that the bitstreams were generated with different encoder decisions, because of the different binarizations, and different statistics are needed to interpret the data. In the following paragraphs, three different tables are presented. They provide different point-of-views on the data, which should then lead to some indicators for the effect of each binarization on the encoder decisions.

For the following presentation, the number of bins and bits are mainly compared to those of the anchor for each bitstream to extract the different ratios, i.e., for the same configuration, test sequence, and QP. An average ratio is then calculated by averaging the calculated ratios of all bitstreams.

Relative change in bins and bits due to binarization: The first analysis compares the change in the number of coded bins and the corresponding compression ratio for absolute levels. Let $|x|_h$ (head) and $|x|_t$ (tail) denote sections of the bin string containing successive bins, such that the bin string can be reconstructed by concatenating $|x|_h$ and $|x|_t$. Their exact definition depends on the binarizations that should be compared. For the tail $|x|_t$, bins signaled after the b_{par} flag and the b_{par} flag itself are counted, and for the head $|x|_h$, only bins signaled before the b_{par} flag are counted. For the comparison between the anchor and binarization #1, $|x|_h = \{b_{sig}\}$ for both the anchor and binarization #1, while the remaining bins form $|x|_t$, i.e., $|x|_t = \{b_{|x|>1}, b_{|x|>2}, \text{bypass bins}\}$ for the anchor and $|x|_t = \{b_{par}, b_{|x|>2}, b_{|x|>4}, \text{bypass bins}\}$ for binarization

	All-Intra		Random-Access	
	Bins	**Bits**	**Bins**	**Bits**
Binarization	**Bins of $\|x\|_h$**			
#1	-1.51%	-1.26%	-2.60%	-2.48%
#2	-0.40%	0.50%	-0.41%	0.63%
#3	-0.84%	-0.61%	-0.64%	-0.59%
	Bins of $\|x\|_t$			
#1	37.64%	2.41%	36.98%	4.47%
#2	9.50%	2.99%	11.32%	4.37%
#3	8.44%	6.69%	9.00%	7.50%

Table 6.4
Differences of bins and bits for different binarizations with b_{par} relative to anchor implementation without b_{par} flag and the three context-coded bins b_{sig}, $b_{|x|>1}$, and $b_{|x|>2}$. For the tail $|x|_t$, bins signaled after the b_{par} flag and the b_{par} flag itself are counted, and for the head $|x|_h$, only bins signaled before the b_{par} flag are counted. In the anchor without the b_{par} flag, the counting of $|x|_h$ and $|x|_t$ is split at the position of the bin string where b_{par} of the corresponding binarization would be coded.

#1. Correspondingly, $|x|_h = \{b_{sig}, b_{|x|>1}\}$ for the comparison between the anchor and binarization #2, while $|x|_t = \{b_{|x|>2}, \text{bypass bins}\}$ for the anchor and $|x|_t = \{b_{par}, b_{|x|>3}, \text{bypass bins}\}$ for binarization #2. For the comparison between the anchor and binarization #3, $|x|_h = \{b_{sig}, b_{|x|>1}, b_{|x|>2}\}$ and $|x|_t = \{\text{bypass bins}\}$ for the anchor, while $|x|_t = \{b_{par}, \text{bypass bins}\}$ for binarization #3.

Condensed simulation results are listed in table 6.4 for the three considered binarizations, where all numbers are relative to the data extracted from bitstreams coded without the b_{par} flag, i.e., from the anchor used for the BD-rate computations in the previous investigation (IMP5-19*). All three binarizations result in bitstreams with fewer bins than in the anchor for the head $|x|_h$. While the corresponding compression ratio or efficiency of the context-coded bins is higher than in the anchor (negative percentage) for the binarization #1 and #3, the binarization #2 is less efficient. The circumstance for $|x|_t$ is completely different, where the number of bins is significantly higher for all binarizations than for the anchor. A notable increase can be observed for binarization #1, while the increase for binarizations #2 and #3 is significantly lower and relatively similar. Nonetheless, the efficiency loss of $|x|_t$ provided by binarization #1 is less than for the other binarizations, meaning that binarization #1 achieves a particularly high compression ratio for those bins. Given these results, it is interesting to know whether the observed increase in the number of bins for $|x|_t$, especially for binarization #1, also affects the number of bins for the whole bitstream.

	All-Intra		Random-Access	
	Bins	**Bits**	**Bins**	**Bits**
Binarization	**Bins of $\|x\|$**			
#1	16.22%	12.92%	15.19%	11.56%
#2	1.12%	1.26%	1.43%	1.45%
#3	0.01%	0.39%	0.17%	0.42%
	All Bins			
#1	6.84%	0.19%	4.37%	0.11%
#2	1.13%	0.85%	0.89%	0.62%
#3	0.47%	0.56%	0.39%	0.41%

Table 6.5
Differences of bins and bits for different binarizations with b_{par} relative to anchor implementation without b_{par} for coded absolute levels and all bitstream data.

Relative change in bins and bits for absolute levels and all syntax elements: The same comparison of bins and bits was performed for absolute levels and the whole bitstream, and the numbers are summarized by table 6.5. While the increase in the number of bins is marginal when considering absolute levels and all bins of the bitstream for binarizations #2 and #3, the increase for binarization #1 remains significantly higher. These results show that binarization #1 provides a disproportional increase in the cps compared to the other binarization options, which impacts the complexity of encoder and decoder implementations. A

potential cause for the increase in cps is the inefficiency of the existing Rice parameter selection when using a binarization with b_{par}, which could be addressed by updating the thresholds.

Relative change in bit-to-bin ratio for absolute levels and all syntax elements: The efficiency of the entropy coding, the efficiency of context-coded bins for the absolute levels, and the number of bypass-coded bins of the absolute levels relative to the anchor are analyzed in a final investigation. A summary for these analyzes is summarized in table 6.6, where the efficiency is in terms of the bit-to-bin ratio, but denoted as percentages, e.g., 72.25% means that one bin requires 0.7225 bits on average. The values for all bins and bits of the bitstreams also include the bypass-coded bins, while the values for absolute levels only consider the context-coded bins. The numbers of bypass-coded bins of the absolute levels relative to the anchor are denoted in the two rightmost columns of table 6.6. Further, the bit-to-bin ratios of the b_{par} flag for the different binarization options were measured (not listed in the tables). Although all three binarizations use the same context modeling, the efficiency of b_{par} varies significantly with 67% for binarization #1, 87% for #2, and 96% for #3 (combined for *All-Intra* and *Random-Access*).

For binarization #3, these numbers indicate that the loss in coding efficiency is introduced by the inefficiency of the b_{par} flag and, supposedly, by the inefficiency of the Rice parameter selection. For binarization #2, the overall inefficiency may be further increased by the inefficiency of the $b_{|x|>3}$ flags that are coded after the b_{par} flags. The binarization #1 has a position of its own in this context, because of the significant increase in the cps and the substantial decrease in the number of bypass-coded bins.

Conclusions of the Investigations

Coding efficiency losses are more significant for the change in binarization than for the change in the template-based context modeling. Given the available options, the analyses reveal that there is no optimal solution for a binarization with b_{par}. The optimal binarization in terms of coding efficiency is #1, but this binarization significantly increases the number of context-coded bins. The optimal binarization in terms of the number of context-coded bins is binarization #3, but this binarization results in a direct dependency between two context-coded flags, i.e., the context modeling of b_{sig} depends on the b_{par} flag directly coded before. The optimal binarization that would form a trade-off between the other two binarizations is #2, but this binarization results in the highest overhead in coding efficiency.

Nevertheless, the analyses provide different aspects that are useful for further investigations. Firstly, the encoding using binarization #1 results in a disproportionate increase in the cps compared to binarizations #2 and #3, which negatively impacts the complexity of encoder and decoder implementations. Secondly, the number of bypass-coded bins decreases, and the reduction depends on the selected binarization with b_{par}. However, they do not completely vanish, indicating that a sufficient number of remainders are present that are binarized with Rice codes. Consequently, updating the thresholds of the Rice parameter selection could reduce the loss in coding efficiency.

	Relative Bit-to-Bin Ratio					
	All Bins		$\|x\|$		**Level Bypass**	
Binarization	**AI**	**RA**	**AI**	**RA**	**AI**	**RA**
#1	77.22%	77.25%	67.43%	68.17%	29.65%	29.89%
#2	82.07%	80.31%	77.21%	77.03%	46.23%	46.07%
#3	82.38%	80.54%	77.63%	77.51%	70.38%	70.56%
IMP5-19*	82.31%	80.52%	76.76%	76.78%		

Table 6.6
Efficiency of the entropy coding for different binarizations with b_{par} and that of the anchor. The numbers for all bins denote the overall efficiency of the entropy coding in percentage, meaning that one bin requires 0.8231 bits on average when the percentage value is equal to 82.31%. For the numbers of $|x|$, only the context-coded bins are considered, i.e., without an offset that would be introduced when including the bypass-coded bins. Finally, the number of bypass-coded bins for absolute transform coefficient levels relative to the anchor is denoted on the two rightmost columns for each binarization.

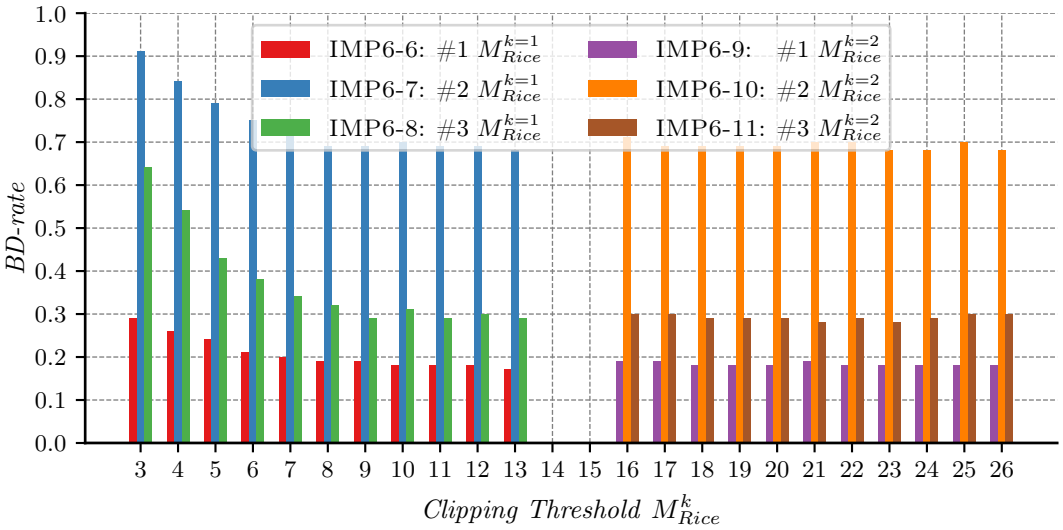

Figure 6.6
Coding efficiency of the investigations on updated thresholds for the Rice parameter selection for binarizations with b_{par} (IMP6-6 to IMP6-11). In IMP6-6 to IMP6-8, $M_{Rice}^{k=1}$ is determined for binarization #1 (IMP6-6), #2 (IMP6-7), and #3 (IMP6-8), while $M_{Rice}^{k=2}$ is set equal to ∞. In IMP6-9 to IMP6-11, $M_{Rice}^{k=2}$ is determined for binarization #1 (IMP6-9 with $M_{Rice}^{k=1} = 8$), #2 (IMP6-10 with $M_{Rice}^{k=1} = 9$), and #3 (IMP6-11 with $M_{Rice}^{k=1} = 10$).
Anchor for BD-rate computations: IMP5-19*

6.4.3 | Rice Parameter Selection

Because b_{par} is coded in the first coding phase, the probability distribution of the remainder z, binarized with Rice codes, is different from that of the unmodified variant. Depending on the selected binarization with b_{par}, the remaining level coded in the second coding phase with Rice binarization is $z = \lfloor (|x| - |x|^1_{max})/2 \rfloor$. Consequently, it can be expected that the bit-rate overhead introduced by the binarizations with b_{par} can be reduced by determining new thresholds for the Rice parameter selection, as denoted in equation (5.6). However, this time the threshold for $k = 3$ should be unnecessary, because the geometric distributions of the remainder z should have a significantly larger model parameter. Let $z = \lfloor z'/2 \rfloor$ denote the remainder for the binarization with the b_{par} flag, where z' represents the remainder without coding a parity flag. Then, the probability of the remainder z for the binarization with b_{par} flag is equal to $p(z') + p(z' + 1)$, which results in a higher value for the model parameter of the geometric distribution. Another consequence is that more remainders are coded with $k = 0$ than before, which should decrease the overall efficiency of the Rice codes.

Determination of Thresholds

Coding experiments were performed to determine updated thresholds for the Rice parameter selection, as described in section 5.4.3. In IMP6-6 to IMP6-8, $M_{Rice}^{k=1}$ is determined for the different binarizations with b_{par}, while $M_{Rice}^{k=2} = \infty$ for all implementations. After performing the coding experiments of IMP6-6 to IMP6-8, $M_{Rice}^{k=2}$ is determined in IMP6-9 to IMP6-11. In IMP6-9, the threshold $M_{Rice}^{k=1} = 8$ was selected based on the obtained BD-rates for IMP6-6. Accordingly, the selected thresholds are $M_{Rice}^{k=1} = 9$ for IMP6-10, and $M_{Rice}^{k=1} = 10$ for IMP6-11.

The measured BD-rates are summarized in figure 6.6 for the *All-Intra* configuration only for the sake of a compact representation. For the tested operation points, the determination of $M_{Rice}^{k=2}$ does not provide a significant coding efficiency improvement, an assumption that was made for $M_{Rice}^{k=3}$, but not already for $M_{Rice}^{k=2}$ in all binarization options. This observation is due to the higher model parameter of the geometric distribution (compare figure 4.7) when using a binarization with b_{par}, for which $k = 0$ is often the optimal Rice code. For the representation of the final BD-rates, the thresholds used in IMP6-9 (binarization #1) are $M_{Rice}^{k=1} = 8$ and $M_{Rice}^{k=2} = 23$, and the implementation is denoted as IMP6-9*. Correspondingly, the thresholds are $M_{Rice}^{k=1} = 9$ and $M_{Rice}^{k=2} = 26$ for IMP6-10* (binarization #2), and are $M_{Rice}^{k=1} = 10$ and $M_{Rice}^{k=2} = 26$ for

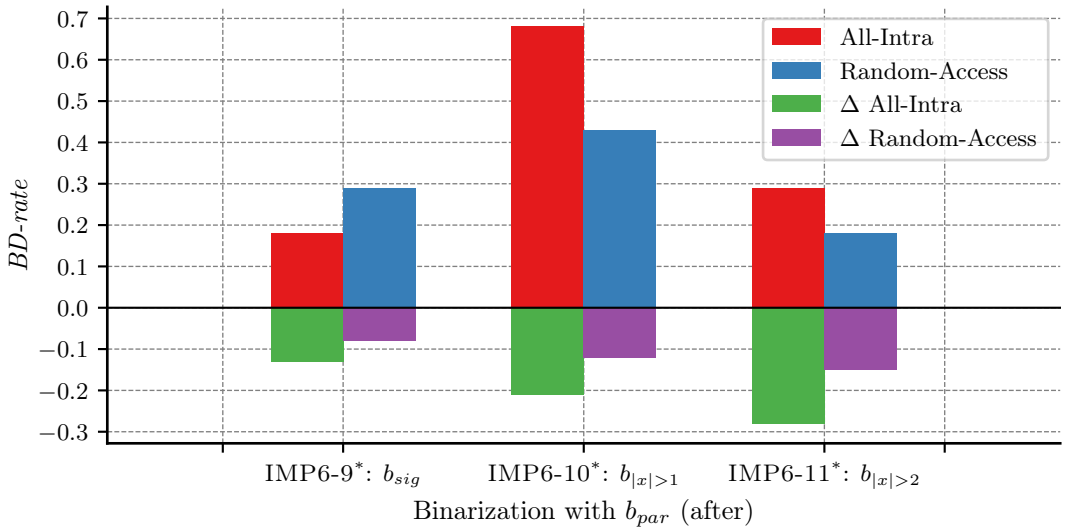

Figure 6.7
Coding efficiency of updated thresholds for the Rice parameter selection (IMP6-9* to IMP6-11*). The bars labeled by Δ denote the achieved coding efficiency improvement of each binarization, i.e., the corresponding BD-rate computations are between IMP6-9* and IMP6-6, IMP6-10* and IMP6-7, and between IMP6-11* and IMP6-7.
Anchor for BD-rate computations: IMP5-19*

IMP6-11* (binarization #3).

Coding Efficiency of Updated Thresholds

The coding efficiency achieved by updating the thresholds for the Rice parameter selection is summarized in figure 6.7. The loss in coding efficiency is reduced for all three binarizations compared to the configuration IMP6-3 to IMP6-5, where the thresholds of the Rice parameter selection are unmodified. In figure 6.7, the changes in the BD-rates due to the updated thresholds for Rice parameter selection for the different binarizations are denoted by the corresponding bars labeled with Δ. Although the updated Rice parameter selection alleviates the overall loss in coding efficiency, the binarization with b_{par} remains less efficient than the binarization without b_{par}.

6.4.4 | Reported Coding Performance in VVC

A level coding implementing the presented separation for the context-coded and bypass-coded bins was presented to JVET during the development of VVC in the input document JVET-K0072 [98]. The chosen binarization in [98] is #1, i.e., the b_{par} flag is coded after the b_{sig} flag. However, the first coding phase implemented in [98] includes the context-coded flags b_{sig}, b_{par}, and $b_{|x|>2}$ only. Instead of separating the single coding phase into two coding phases, the chosen design consists of three coding phases, two context-coded phases, and one bypass-coded phase. In the second coding phase, the context-coded flags $b_{|x|>4}$ are transmitted exclusively, while the third coding phase consists of the remainders coded in the bypass mode (compare binarizations with b_{par} in table 6.3). This design does not retain the coding efficiency provided by the template-based context modeling with absolute levels, as described in section 5.4, because of $|x|^1_{max} \in \{3, 4\}$ instead of $|x|^1_{max} \in \{5, 6\}$. However, the introduced losses in coding efficiency are negligible, as indicated by the investigation presented in section 6.4.2 with IMP6-2. For the presentation here, the different experiments and associated results reported in [98] are denoted as follows:

72[1] In this experiment, the coding efficiency of the presented level coding using three coding phases is evaluated for the case that TCQ is not used, i.e., TCQ is disabled, while RDOQ is enabled (short for JVET-K0072).

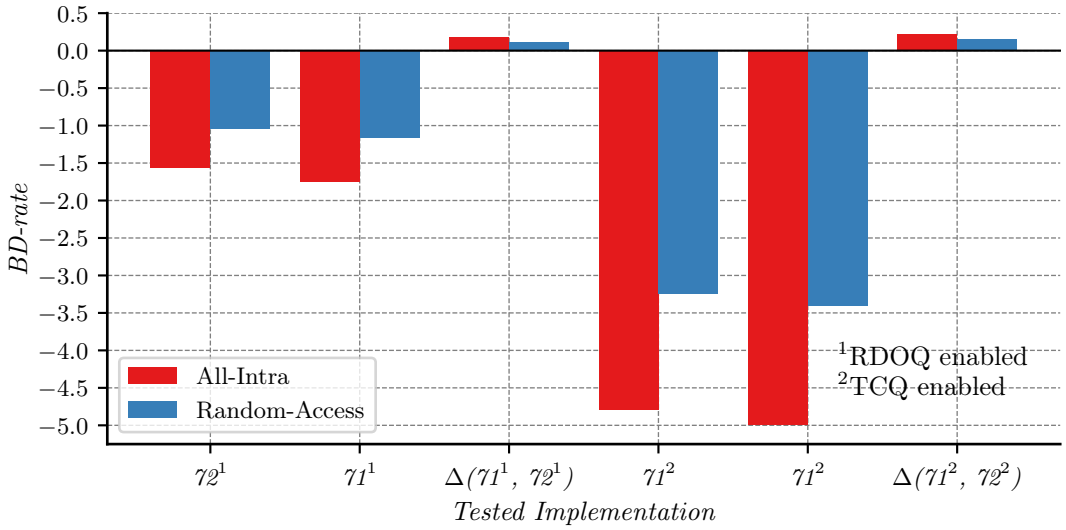

Figure 6.8
Coding efficiency of 71^1, 72^1, 71^2, and 72^2, where 71^1 and 71^2 correspond to the level coding of JVET-K0071. 72^1 and 72^2 correspond to the level coding of JVET-K0072 [98]. In 71^1 and 72^1, as well as in the corresponding anchor for BD-rate the computations, RDOQ is enabled, and TCQ is disabled. In 71^2, 71^2, and the corresponding anchor, TCQ is enabled in the coding experiments.
Anchor for BD-rate computations: VTM-1

71^1 This experiment summarizes the coding efficiency of the template-based context modeling with a single coding phase (see chapter 5) to show the change in coding efficiency caused by the introduction of the b_{par} flag in IMP6-13 (short for JVET-K0071). The experimental results correspond to those summarized in table 5.1 in section 5.4.5, i.e., TCQ is disabled, while RDOQ is enabled.

72^2 This experiment corresponds to 72^1 with TCQ enabled.

71^2 This experiment corresponds to 71^1 with TCQ enabled.

For all experiments, the anchor for BD-rate computations is the level coding implemented in VTM-2 [99]. A summary of the reported experimental results is illustrated in figure 6.8, which includes the relative change in BD-rate between 71^1 and 72^1, as well as between 71^2 and 72^2. A slight loss in coding efficiency can be observed when TCQ is disabled, and that loss remains relatively similar when TCQ is enabled. Compared to the overall improvement in coding efficiency provided by the level coding developed in this thesis for both configurations, i.e., with and without TCQ enabled, the losses introduced by the binarization with the b_{par} flag are comparably small.

6.5 | Reduction of Context-Coded Bins

The parameter t_0 of the binarization for absolute levels illustrated in figure 4.1 (section 4.2.1) regulates the mcps for the transform coefficient levels. This mcps for absolute levels is equal to the value of t_0, which is 15 in AVC and was reduced significantly to three with the introduction of the adaptive binarization with Rice codes presented in section 4.3. Such a reduction was necessary to allow for feasible hardware implementations of *context-based adaptive binary arithmetic coding* (CABAC), i.e., the mcps indicates the achievable throughput of the overall design and should be as low as possible. In the final HEVC level coding, the parameter t_0 is equal to three and is reduced backward-adaptively, which leads to a further reduction of the mcps without negatively impacting the coding efficiency [100]. While the mcps for the adaptive binarization used in HEVC is equal to 1.5625 only, the mcps of the level coding in VVC increases from three to four with the introduction of the binarization with b_{par} flags. This possible discrepancy between HEVC and VVC is undesirable, and a similar number of mcps as in HEVC is pursued during the development of VVC. This section starts with a brief review of the technique specified in HEVC and is followed by a description of the concept developed

in this thesis and implemented in VVC.

6.5.1 | Adaptive Binarization Bound in HEVC

Instead of reducing the number of context-coded bins by decreasing the maximum value binarized with the *truncated unary* (TRU) code, which leads to a significant loss in coding efficiency (as indicated by the investigations in chapter 4), the parameter t_0 is selected backward-adaptively during the level coding [100]. This approach reduces the mcps from three, when using a fixed $t_0 = 3$, to 1.5625. Its implementation is as follows.

Implicit Restriction of Context-Coded Bins

The concept behind the adaptive reduction of context-coded bins implemented in HEVC is that for transform coefficient levels with high magnitudes, fewer context-coded bins are necessary to achieve a similar coding efficiency. This condition can be achieved because the bypass-coded bins dominate the codeword length in such a case. This aspect can be exploited by evaluating the coded $b_{|x|>1}$ and $b_{|x|>2}$ flags to estimate the beginning of an area consisting of levels with high magnitudes. Conceptually, the reduction is based on limiting the number of coded $b_{|x|>1}$ and $b_{|x|>2}$ flags within each 4×4 sub-block. At the beginning of a 4×4 sub-block, the binarization bound t_0 is set equal to three, i.e., the context-coded flags coded for this configuration are b_{sig}, $b_{|x|>1}$, and $b_{|x|>2}$ (see figure 4.1 and figure 4.16 of chapter 4). After coding one $b_{|x|>2}$ flag within the sub-block, t_0 is set equal to two, i.e., no more $b_{|x|>2}$ flags are coded for the current sub-block. When eight $b_{|x|>1}$ flags have been coded within the sub-block, the bound is further decreased to $t_0 = 1$, i.e., the only context-coded flag left for the remaining levels in the sub-block is b_{sig}. A direct transition from $t_0 = 3$ to $t_0 = 1$ occurs when eight $b_{|x|>1}$ flags with a value equal to zero have been coded for the sub-block. Besides reducing the mcps, this approach also saves context memory, because only one context model is necessary for coding the $b_{|x|>2}$ flag (compare context modeling of $b_{|x|>2}$ in AVC and HEVC denoted in equation (3.2) in section 3.2.1).

Reported Coding Efficiency and Number of Context-Coded Bins

The technique above reduces the mcps to roughly 50% compared to the fixed configuration $t_0 = 3$. Virtually the same coding efficiency relative to the fixed configuration $t_0 = 3$ was reported in [100] for both the *All-Intra* and *Random-Access* configurations. Furthermore, the amount of context-coded bins is reported to be up to 15% lower than for the fixed configuration $t_0 = 3$ for operation points targeting consumer applications, as reported in [100]. These relative savings in context-coded bins are higher for high bit-rate operation points.

6.5.2 | Adaptation of the Concept to VVC

A similar reduction of context-coded bins as in HEVC was targeted during the development of VVC, where the mcps was equal to four due to the binarization with b_{par} at the time. However, the approach implemented in HEVC cannot be applied directly on top of the level coding in VVC, because the level coding in HEVC employs a dedicated coding phase for each context-coded flag. This coding in multiple phases for context-coded flags does not have direct dependencies among the different context-coded flags, e.g., between b_{sig} and $b_{|x|>1}$. However, when applying this approach to the context-coded phase implemented in VVC, direct dependencies emerge, where the context modeling of a flag depends on the value of the previously coded flag. For example, when implementing the HEVC scheme straightforwardly, only the b_{sig} and the b_{par} flags are coded (binarization #1) after transmitting eight $b_{|x|>2}$ flags. In this state, the context modeling of b_{sig} depends on the b_{par} flag coded directly before the b_{sig} flag. Another problem arises in combination with template-based context modeling, where the maximum available level magnitude of the neighboring frequency locations inside the template frequently changes.

Explicit Restriction of Context-Coded Bins

Instead of limiting the mcps implicitly, as in HEVC, the concept of the approach developed in this thesis is explicit by specifying a maximum number of context-coded bins for a sub-block, which is tracked by a variable $\mathcal{A}$. This variable $\mathcal{A}$ is initialized to a specific value at the beginning of each sub-block, and no

more context-coded flags are transmitted for the remaining scanning positions of the sub-block when $\mathcal{A} < n$, with n being the maximum number of context-coded flags transmitted for each scanning position within the coding phase. That implies that all remaining absolute levels of the sub-block are binarized with Rice codes only, and the resulting bins are coded in the bypass mode, which corresponds to the setting $t_0 = 0$. For the absolute levels coded completely in bypass mode, the Rice parameter selection is updated to accommodate the change in the probability distribution parameter. Instead of evaluating the local template, the Rice parameter is selected according to algorithm 4.3, described in section 4.3.4, but using different thresholds than described in section 4.3.4.

The developed approach, proposed in [101], is implemented on top of the level coding presented in [98], which consists of two coding phases for the context-coded flags to avoid a direct dependency between $b_{|x|>2}$ and $b_{|x|>4}$. Because of the two coding phases, two variables tracking the number of context-coded bins are necessary to control and limit the context-coded flags of the level coding. These two variables, denoted as $\mathcal{A}_1$ and $\mathcal{A}_2$, are maintained for each sub-block and are initially set equal to $\mathcal{A}_1 = 28$ and $\mathcal{A}_2 = 4$ for a 4×4 sub-block. This initialization of $\mathcal{A}_1$ and $\mathcal{A}_2$ corresponds to a limitation to a mcps equal to two. For a given absolute level, the flags b_{sig}, b_{par} and $b_{|x|>2}$ are only coded in the first coding phase if $\mathcal{A}_1\,(i) \geq 3$. After coding a context-coded flag in the first coding phase, the number of remaining context-coded bins stored in $\mathcal{A}_1$ is decremented by one. Correspondingly, $\mathcal{A}_2$ is decremented by one after the coding a $b_{|x|>4}$ flag in the second coding phase, and further $b_{|x|>4}$ flags are only coded when $\mathcal{A}_2 \geq 1$. This configuration with two variables tracking the number of context-coded bins and their initial values is a trade-off between coding efficiency and the number of context-coded bins. Note that the setting $t_0 = 0$ can occur without coding a $b_{|x|>4}$ flag, i.e., when $\mathcal{A}_1 < 3$ and all coded $b_{|x|>2}$ flags have a value equal to zero. To further reduce the number of context-coded bins in actual coding conditions, the coding order of b_{par} and $b_{|x|>1}$ is exchanged, which corresponds to binarization #2 as described in section 6.4.2. This change significantly reduces the actual number of context-coded bins, as the investigation presented in section 6.4.2 demonstrated. However, the modification introduces a dependency between the context modeling of b_{sig} and the previously coded b_{par} flag. Nonetheless, the reduction in the number of context-coded bins (on average) was deemed more significant. A summary of the implementation is provided in algorithm 6.1.

Reported Coding Efficiency

The above restriction of context-coded bins was implemented on top of the level coding in VTM-2, and with TCQ enabled, it provides BD-rates of -0.20% in the *All-Intra* and -0.14% in the *Random-Access* configurations, relative to the unmodified level coding of VTM-2. For the configuration with TCQ disabled, the coding efficiency is virtually the same as provided by the level coding of VTM-2. A reason for the improved coding efficiency with TCQ enabled, which cannot be expected due to the analysis in section 6.4.2, is the disappearance of CABAC zero words [102]. [1]

Further Refinements Towards Final Version of VVC

The explicit restriction of context-coded bins developed in this chapter was further refined in the finalized version of VVC. In VTM-17, which implements the first published VVC specification, the level coding consists of a single context-coded phase, i.e., the second context-coded phase is integrated into the first coding phase [104]. Because of this change, a single variable $\mathcal{A}$ tracking the number of context-coded bins remained, and the first coding phase is only executed when $\mathcal{A} \geq 4$. A second refinement changes the granularity of the restriction from the sub-block to the transform block level, i.e., the tracking variable $\mathcal{A}$ is initialized once at the beginning of a transform block [105], and the resulting mcps was reduced to 1.75. For absolute levels coded completely in the bypass mode, the Rice parameter selection evaluates the template, as for the case where the levels are not coded completely in the bypass mode, but using different thresholds. Finally, the binarization of the absolute levels was further modified by swapping the codeword of the remainder $z = 0$ and a predicted value z_p, which depends on the TCQ state and the sum $\mathcal{S}$ [101].

[1] Without the presented restriction of the number of context-coded bins, it was observed that the generated bitstreams were padded with CABAC zero words [103]. These CABAC cabac zero words are necessary to lower the bin-to-bit ratio, i.e., the number of bins that can be extracted from a single bit [72], which is important for practical hardware implementations.

Algorithm 6.1 Pseudo-code of the context-coded bins reduction scheme with two budgets that limits the number of context-coded bins to a maximum of two bins per sample.

Require: first scanning position of sub-block i_0
last coded scanning position of first pass $i_1 \leftarrow i_0$
$i \leftarrow i_0 + 15$, $\mathcal{A}_1(i) \leftarrow 28$
while $i \geq i_0 \wedge \mathcal{A}_1(i) \geq 3$ **do**
  if $b_{sig}(i)$ cannot be inferred to be equal to one **then**
    transmit $b_{sig}(i)$ using a context model
    $\mathcal{A}_1(i) \leftarrow \mathcal{A}_1(i) - 1$
  end if
  if $b_{sig}(i) = 1$ **then**
    transmit $b_{|x|>1}(i)$ using a context model
    $\mathcal{A}_1(i) \leftarrow \mathcal{A}_1(i) - 1$
    if $b_{|x|>1}(i) = 1$ **then**
      transmit $b_{par}(i)$ using a context model
      $\mathcal{A}_1(i) \leftarrow \mathcal{A}_1(i) - 1$
    end if
  end if
  if $\mathcal{A}(i) < 3$ **then**
    $i_1 \leftarrow i$
  end if
  $\mathcal{A}_1(i-1) \leftarrow \mathcal{A}_1(i)$
  $i \leftarrow i - 1$
end while
last coded scanning position of second pass $i_2 \leftarrow i_1$
$i \leftarrow i_0 + 15$, $\mathcal{A}_2(i) \leftarrow 4$
while $i \geq i_1 \wedge \mathcal{A}_2(i) \geq 1$ **do**
  if $b_{|x|>1}(i) = 1$ **then**
    transmit $b_{|x|>3}(i)$ using a context model
    $\mathcal{A}_2(i) \leftarrow \mathcal{A}_2(i) - 1$
  end if
  if $\mathcal{A}_2(i) = 0$ **then**
    $i_2 \leftarrow i$
  end if
  $\mathcal{A}_2(i-1) \leftarrow \mathcal{A}_2(i)$
  $i \leftarrow i - 1$
end while
$i \leftarrow i_0 + 15$
while $i \geq i_2$ **do**
  if $b_{|x|>3}(i) = 1$ **then**
    transmit $z(i) = \lfloor (|x(i)| - 4)/2 \rfloor$ in bypass mode
  end if
  $i \leftarrow i - 1$
end while
$i \leftarrow i_2 - 1$
while $i \geq i_1$ **do**
  if $b_{|x|>1}(i) = 1$ **then**
    transmit $z(i) = \lfloor (|x(i)| - 2)/2 \rfloor$ in bypass mode
  end if
  $i \leftarrow i - 1$
end while
$i \leftarrow i_1 - 1$
while $i \geq i_0$ **do**
  transmit $|x(i)|$ in bypass mode
end while

6.6 | Using Intermediate Levels for Context Modeling in TCQ

The dequantization process with TCQ, as outlined in section 6.2.2, necessitates the calculation of an intermediate level x', as summarized in equation (6.7). This intermediate level x' is derived from x, which is the level reconstructed based on the binarization table, and the TCQ state $S(i)$, with i being the scanning position. After the computation of x', the intermediate level is used as the input for the subsequent scaling process instead of x.

In contrast to the transmitted quantization indices x, the intermediate levels x' are proportional to the reconstructed transform coefficients c', which are calculated by multiplying the intermediate levels x' with the quantization step size Δ (when rounding operations of the actual scaling process are neglected). Since the intermediate levels x' incorporate the TCQ state $S(i)$, they offer a greater precision than the transmitted quantization indices x. This additional precision, attributed to the TCQ state, could be exploited in the context modeling to improve the coding efficiency. In some respect, such an approach resembles the context modeling of b_{sig}, where the TCQ state is employed to select a context model set, as outlined in section 6.2.2. Since the intermediate levels could be already computed during the parsing of the bitstream, the idea of using them for context modeling is feasible. In the subsequent sections, the potential of incorporating these intermediate levels to improve the coding efficiency is investigated.

6.6.1 | Impact of Intermediate Levels on Context Modeling

The computation of x' during the parsing of the bitstream was first proposed in [106] in an early phase of the VVC development. The authors of [106] observed that computing x' during the parsing of the bitstream offers the advantage that only one pass of TCQ state transitions is required instead of two. As a consequence, the scaling process can be conducted in parallel since all intermediate levels of a block are available before the scaling process begins. Adjustments to context modeling are important to maintain similar coding efficiency, as the amplitudes of x' are roughly double those of x (see equation (6.7)). The authors of [106] focused on implementation feasibility, and the modifications described in [106] ensured the coding efficiency achieved was comparable to the unaltered version. In contrast, the potential of coding efficiency improvement by using intermediate levels is investigated in this section.

6.6.2 | Context Modeling Adjustments for Intermediate Levels

In the first investigation, the adjustments to the context modeling described in [106] are examined using VTM-17.0. This version corresponds to the final VVC standard, whereas the investigations in [106] were based on a software implementation used during the development phase of VVC. Both the software implementation used in [106] and the VVC standard utilize the template sums $\mathcal{S}_c$ and $\mathcal{S}$ for context modeling. Specifically, the template sum $\mathcal{S}_c$, calculated according to equation (6.10) in VVC, is utilized to derive the context models for all context-coded bins. In contrast, the template sum $\mathcal{S} = \sum_{i=0}^{T-1} |x(i)|$, with T denoting the template size, is used to derive the Rice parameter. Considering that the amplitude of x' is approximately twice that of x, the adjustments in [106] involve halving the template sums. This halving is accomplished by right-shifting the template sums by one, which preserves the range of the template sums and the output range of the final context model indices.

Note that the clipping value $|x|^1_{max}$ used in equation (6.10) for the computation of $\mathcal{S}_c$ alters when using intermediate levels. This clipping value denotes the maximum amplitude that can be signaled by the first coding phase that only contains context-coded bins, and it is required for encoder implementations only. In VVC, $|x|^1_{max} = 4 + (|x(i)|\&1)$ with $|x(i)|\&1$ being the value of the corresponding b_{par} flag at the scanning position i. When using the intermediate levels, this clipping value has to be changed to $|x'|^1_{max} = 7 + (|x'(i)|\&3)$.

Based on the description of [106], three adjustments were made in VTM-17.0 for the initial investigation. The first adjustment concerns the calculation of the context index offset δ_{sig}, which specifies the context model for coding a b_{sig} flag. In VVC, the δ_{sig} is calculated according to:

$$\delta_{sig} = \min\left((\mathcal{S}_c + 1) \gg 1, 3\right). \tag{6.11}$$

Let $\mathcal{S}'_c$ and $\mathcal{S}'$ denote the template sums using the intermediate levels, which are given by:

$$\mathcal{S}'_c = \sum_{i=0}^{T-1} \min\left(|x'(i)|, |x'|^1_{max}\right), \tag{6.12}$$

and

$$\mathcal{S}' = \sum_{i=0}^{T-1} |x'(i)|. \tag{6.13}$$

According to [106], the derivation of δ_{sig} with intermediate levels becomes:

$$\delta_{sig} = \min\left(\left(\left(\mathcal{S}'_c \gg 1\right) + 1\right) \gg 1, 3\right). \tag{6.14}$$

The second adjustment affects the derivation of the context index offset δ_{gtX} for the context-coded flags $b_{|x|>1}$, $b_{|x|>3}$, and b_{par}. For the last significant scanning position, the context index offset is $\delta_{gtX} = 0$, which remains unchanged when using intermediate levels. For all other scanning positions along the scanning path, δ_{gtX} is computed in VVC as:

$$\delta_{gtX} = \min\left(\mathcal{S}_c - \sum b_{sig}, 4\right) + 1, \tag{6.15}$$

where $\sum b_{sig}$ represents the number of occurrences of $b_{sig} = 1$ inside the local template. When using the intermediate levels, the computation of δ_{gtX}, according to [106], updates to:

$$\delta_{gtX} = \min\left(\left(\mathcal{S}'_c \gg 1\right) - \sum b_{sig}, 4\right) + 1. \tag{6.16}$$

However, it was observed that δ_{gtX} can become negative when modifying the context modeling as in the above equation, an issue not mentioned in [106]. To avoid a negative δ_{gtX}, the following calculation is used instead:

$$\delta_{gtX} = \max\left(0, \min\left(\left(\mathcal{S}'_c - 2\sum b_{sig}\right) \gg 1, 4\right)\right) + 1. \tag{6.17}$$

For the Rice parameter k, the template sum $\mathcal{S} = \sum_{i=0}^{T-1} |x\,(i)|$, with T denoting the template size, is used in VVC. After determining $\mathcal{S}$, an intermediate value s is calculated according to $s = \max\left(\mathcal{S} - 5\beta, 0\right)$, where β is the so-called base level. This base level β is equal to 4 during the coding phase where remainders are encoded, i.e., partial values have already been coded by context-coded flags for the respective scanning positions. When only bypass coding is employed, i.e., when the budget for context-coded bins is exhausted (see section 6.5), $\beta = 0$. This intermediate value s is then utilized to determine the Rice parameter k as follows:

$$k = \begin{cases} 0, & \text{if } s < 7, \\ 1, & \text{if } s \geq 7 \wedge s < 14, \\ 2, & \text{if } s \geq 14 \wedge s < 28, \\ 3, & \text{otherwise.} \end{cases} \tag{6.18}$$

Following the concept of [106], the derivation of s is updated to $s = \max\left(\left(\mathcal{S}' \gg 1\right) - 5\beta, 0\right)$.

For the first and all subsequent investigations in this subsection, the anchor used for BD-rate computations is the unaltered version of VTM-17.0. The experimental results of the first investigation, i.e., using intermediate levels and applying the concept of [106] for context modeling, showed inferior coding efficiency. For the *All-Intra* configuration, a bit-rate overhead of 0.49% was observed. In the *Random-Access* configuration, the overhead was 0.25%. The coding efficiency achieved with the modified VTM-17.0 does not align with the coding performance reported in [106]. A possible explanation is that, in comparison to the VTM version used in [106], the VVC standard includes a modified binarization and a reduction of context models. It is still surprising that straightforward context modeling adjustments, specifically halving the template sums as outlined in the equations above, yielded suboptimal results. Note that to match the coding efficiency of VVC, the only required modification pertains to the computation of the template sums, since the absolute values of the quantization indices $|x|$ can be recovered from the absolute intermediate levels $|x'|$ according to $|x| = \left(|x'| + 1\right) \gg 1$. In such a setting, the context index offsets and the Rice parameter should be calculated as in VVC, but with the template sums given by $\mathcal{S}_c = \sum_{i=0}^{T-1} \min\left(\left(|x'\,(i)| + 1\right) \gg 1, 4 + \left(\left(\left(|x'\,(i)| + 1\right) \gg 1\right) \,\&1\right)\right)$ and $\mathcal{S} = \sum_{i=0}^{T-1} \left(|x'\,(i)| + 1\right) \gg 1$. Compared to the adjustments introduced above, the differences appear to be minimal. In the following investigations, each adjustment is evaluated individually to better understand its impact on the coding performance.

6.6.3 | Refinements for Context Modeling of $b_{|x|>1}$, $b_{|x|>3}$, and b_{par}

In the second investigation, the adjustment to the context modeling of the context-coded flags $b_{|x|>1}$, $b_{|x|>3}$, and b_{par} is examined. The reason that δ_{gtX} can become negative when using equation (6.16) is that the

intermediate value $|x'|$ for $|x| = 1$ can be either 1 or 2, depending on the TCQ state. But equation (6.16) implicitly assumes that when $b_{sig} = 1$, the corresponding intermediate level is $|x'| \geq 2$. This assumption can lead to a negative value of δ_{gtX} under specific circumstances. Furthermore, assuming that $|x'| \geq 2$ may overestimate the influence of $b_{sig} = 1$. To address these issues, one could assume that $|x'| \geq 1$ when $b_{sig} = 1$. As a consequence, in the second investigation, the calculation of δ_{gtX} is revised from equation (6.17) to:

$$\delta_{gtX} = \min\left(\left(\mathcal{S}'_c - \sum b_{sig}\right) \gg 1, 4\right) + 1. \tag{6.19}$$

The experimental results for this second investigation led to noticeable coding efficiency improvements. With just this one adjustment, the average BD rate for the *All-Intra* configuration was observed to be 0.03%, improving from 0.49%. For the *Random-Access* configuration, an average BD rate of -0.03% was recorded, improving from 0.24%. These experimental findings suggest that equation (6.17) does overestimate the effect of $b_{sig} = 1$ and that assuming $|x'| \geq 1$ for $b_{sig} = 1$ gives a more accurate estimation of the conditional probabilities.

6.6.4 | Refinements for Context Modeling of b_{sig}

The third investigation examines the context modeling of b_{sig} with the corresponding computation of δ_{sig}. Its derivation according to the concept of [106] is denoted in equation (6.14) and consists of two right shift operations. Generally, when a right shift operation is involved in the computation, the precision of the computation is enhanced by performing the right shift operation at the very end. This is because the right shift operation not only represents a division by an positive integer power of two but also rounds off the result due to integer precision. Furthermore, when incrementing $\mathcal{S}'_c$ by one, the output after the right shift operation changes so that the information whether a significant neighboring location is inside the template remains. This can be seen from the following example. The output of the input values $n = \{0, 1, 2, 3, 4\}$ using the operation $n \gg 1$ are $\{0, 0, 1, 1, 2\}$, while the output for the variant $(n + 1) \gg 1$ are $\{0, 1, 1, 2, 2\}$. Considering both aspects, the calculation for δ_{sig} can be reformulated as:

$$\left(\left(\mathcal{S}' + 1\right)\frac{1}{2} + 1\right)\frac{1}{2} = \frac{\mathcal{S}' + 3}{4}. \tag{6.20}$$

Based on the above considerations, the computation of δ_{sig} is modified according to:

$$\delta_{sig} = \min\left(\left(\mathcal{S}'_c + 3\right) \gg 2, 3\right). \tag{6.21}$$

The experimental results for this third investigation, where the adjustment in the context modeling of δ_{sig} was additionally incorporated, show further improvement in coding efficiency. For the *All-Intra* configuration, an average BD-rate of -0.09% was observed, improving from a previous 0.03% for the second investigation. For the *Random-Access* configuration, an average BD rate of -0.13% was measured, which is an improvement from a previous -0.03% for the second investigation. These experimental results demonstrate that the precision of calculations is important when dealing with integer-based implementations.

6.6.5 | Refinements to the Rice Parameter Derivation

In the final investigation, the last adjustment for the intermediate levels is examined, which is the modification to $\mathcal{S}'$ before selecting the Rice parameter. As for the third investigation, $\mathcal{S}'$ is incremented by one before applying the right shift operation. The calculation of $s = \max\left((\mathcal{S}' \gg 1) - 5\beta, 0\right)$ is updated to $s = \max\left(((\mathcal{S}' + 1) \gg 1) - 5\beta, 0\right)$. The experimental results of this final coding experiment showed the same coding efficiency for the *All-Intra* configuration as for the third investigation. For the *Random-Access* configuration, a marginal improvement is observed with an average BD-rate of -0.15%.

6.6.6 | Conclusion on Intermediate Levels

The detailed experimental results for the final coding experiments are summarized in table 6.7. Compared to the initial version using the context modeling described in [106], the adjustments to the context modeling are relatively minor. These experiments indicate that incorporating intermediate levels can enhance coding

Class	Luma	C_B	C_R
	All-Intra		
A1	-0.11%	0.27%	-0.06%
A2	-0.15%	0.22%	0.07%
B	-0.10%	0.18%	0.33%
C	0.02%	0.33%	-0.05%
E	-0.11%	0.31%	0.08%
Overall	**-0.09%**	**0.26%**	**0.10%**
D	-0.06%	0.07%	0.83%
	Random-Access		
A1	-0.29%	0.67%	0.18%
A2	-0.05%	0.29%	0.12%
B	-0.14%	0.21%	0.00%
C	-0.12%	0.08%	0.12%
Overall	**-0.15%**	**0.28%**	**0.09%**
D	-0.29%	0.64%	-0.67%

Table 6.7
Coding efficiency of the investigation on intermediate levels and adjusted context modeling.
Anchor for BD-rate computations: VTM-17.0

efficiency for TCQ. However, the primary focus of the investigations was on directly modifying the existing context modeling. Analyzing the statistics of intermediate levels and developing a dedicated context modeling may provide additional benefits.

6.7 | Findings and Technical Achievements

The modified level coding presented in this chapter, based on the design of chapter 5, represents a suitable trade-off among several aspects. Its development leads to the following technical achievements:

- An additional context model set for coding the b_{sig} flags improves the coding efficiency when TCQ is enabled.
- It is known that the level coding impacts the throughput significantly. The developed level coding achieves the desired throughput increase while maintaining its compatibility with TCQ.
- The number of context-coded bins is further reduced by decreasing the first binarization threshold backward-adaptively.
- When using TCQ, intermediate levels are calculated before the actual scaling process, i.e., the multiplication with the quantization step size. These intermediate levels can already be computed during the parsing of the bitstream and exploited for context modeling, which provides coding efficiency improvements.

The presented developments directly impact practical video coding applications, because their design elements are specified in the VVC standard. The following list summarizes the key elements:

- The state-dependent context modeling of the b_{sig} flags in VVC utilizes three context model sets with the same mapping of state to context model set as presented in this chapter.
- The level coding in VVC implements the developed separation of context-coded and bypass-coded bins using multiple coding phases, as described in this chapter.
- The developed support for TCQ when separating the context-coded and bypass-coded bins with a modified binarization that includes the b_{par} flag is specified for VVC.
- The basic concept of reducing the (worst-case) number of context-coded bins developed and adjusted to the level coding with separated context-coded and bypass-coded phases is used by the level coding of VVC.

The VVC standard specifies a level coding based on the design developed in this chapter, with the final design inheriting further refinements. Instead of three coding phases, as initially proposed in [98], the level coding in VVC employs two coding phases as described later in this chapter. Furthermore, the binarization with the b_{par} flag corresponds to the binarization #2 (IMP6-4) described in this chapter to reduce context-coded bins, as proposed in [101]. The only adjustment to the context-coded bins reduction with a single tracking variable $\mathcal{A}$ is extending the concept to the whole transform block with the initialization $\mathcal{A} = 1.75 \cdot WH$, with W being the width and H being the height of the block [101]. An in-depth description for four coding tools improving the residual coding, i.e., TCQ, joint chroma residual coding, the arithmetic core coding engine, and the level coding specified in the final version of VVC, are presented in [56]. Among these coding tools, level coding was reported to have a share of 25% of the total coding efficiency improvement.

6.8 | Chapter Summary

A brief review of scalar quantization and TCQ was given at the beginning of this chapter, which included an explanation for the practicality of the level coding developed in chapter 5 for TCQ. In the first section that describes the work developed in this chapter, the state-dependent context model sets, used to code the b_{sig} flags with TCQ enabled, were extended. This extension exploits differences in the probability distribution of b_{sig} for each state by using different context model sets for S_2 and S_3. Next, the throughput for practical implementations was increased by separating the coding of context-coded and bypass-coded bins. The reason for the throughput improvement is that this separation increases the number of bypass-coded bins that can be transmitted successively. Because the parity of the level located at the preceding scanning position becomes unavailable in that design, which is necessary to determine to state for TCQ, the parity is transmitted as a dedicated b_{par} flag in the first context-coded phase. Based on the results of different coding experiments, a level coding design was developed that achieves a suitable coding performance with both TCQ enabled and disabled. Next, a concept for reducing the maximum number of context-coded bins is presented. As discussed in chapter 4, the second binarization bound could be reduced to achieve fewer context-coded bins. This strategy would introduce undesirable coding efficiency losses. The concept developed for and finally specified in VVC varies the first binarization bound of the adaptive binarization backward-adaptively by limiting the number of context-coded bins via tracking variables. Whenever the remaining number of context-coded bins, stored in tracking variables, could be insufficient for coding the context-coded bins of the next scanning position, the first binarization bound is reduced so that the absolute levels of the remaining scanning positions are binarized with Rice and EG0 only. The resulting bins are then coded in bypass mode. In the last part of this chapter, it was investigated whether the additional information contained in intermediate levels can be exploited for the context modeling of transform coefficient levels. The experimental results indicate that by carefully adjusting the derivation of context index offsets, the coding efficiency of TCQ can be improved by using intermediate levels instead of quantization indices for the context modeling.

CHAPTER 7

Transform Skip Residual Coding

Contents

Video coding applications were mainly dominated by camera-captured content up to the time of the development of *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC). However, during the development of *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC) and *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC), non-camera-captured content became steadily more popular. Typical signals representing non-camera-captured content are computer screen desktop recordings or animations created by computer rendering. Such signals are commonly referred to as **screen content**. In screen content applications, non-camera-captured signals often do not cover the whole video sequence, but rather some specific regions of the frames. Examples of signals in such applications are graphics and character content rendered, mixed or overlaid with traditional camera-generated content. Due to the different signal characteristics of screen content, such as sharp edges, repeated patterns, or the non-existence of sensor noise, not all coding technologies suitable for camera-captured content are efficient for screen content. Furthermore, the subjective quality perception for screen content is different than for camera-captured signals. Coding tools optimized for camera-captured content tend to cause blurriness, ringing, and "mosquito" artifacts with computer-generated content in a way that is commonly more disturbing to viewers than for camera-captured content. As a consequence, dedicated screen content coding tools have been developed, such as *intra block copy* (IBC) [107] or the *palette mode* (PLT) [108]. A relatively simple approach to improve the coding efficiency of screen content, which is specified in HEVC, is to bypass the transform, referred to as *transform skip mode* (TSM) [109], where the residual samples are quantized directly.

This chapter presents an alternative level coding for the quantization levels of blocks where the transform is bypassed, and these levels are referred to as TSM levels. While the existing level coding, used when the transform is not bypassed, is referred to as conventional level coding, an alternative second level coding specifically designed for transform skip blocks is referred to as *transform skip residual coding* (TSRC) in this chapter. For TSM levels, TSRC provides a higher coding efficiency than the conventional level coding. This improvement in coding efficiency can be achieved because the conventional level coding is not designed for the statistics of directly quantized residual samples. The development of TSRC started with the design of the existing conventional level coding and was then refined with respect to both increasing the coding efficiency and simplifying implementations. This chapter starts with a brief review of the TSM and its coding efficiency compared to other dedicated screen content coding tools. TSRC is then presented with an in-depth analysis

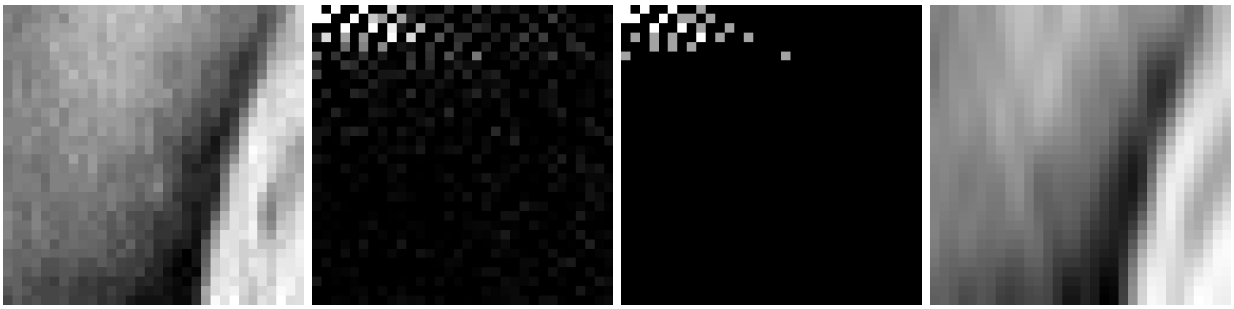

Figure 7.1
Example of a 32×32 DCT-II transform, quantization, and inverse transform for a camera-captured image. The leftmost illustration shows the original block samples, and the next illustration shows the transform coefficients after applying a DCT-II transform. The following illustration shows the transform coefficients after quantization, and the rightmost illustration shows the reconstructed block of samples.

of its components.

7.1 | Problem Statement

The introduction of dedicated screen content coding tools in HEVC was a major leap towards supporting screen content applications [110]. This step also shows that dedicated screen content coding tools are necessary to assist video coding architectures in achieving higher coding efficiency, because coding tools for camera-captured content are less effective for screen content due to different signal characteristics. Bypassing the transform improves coding efficiency for screen content, because the residual samples of screen content show no or negligible energy compaction after applying a transform, e.g., an integer approximation of the *discrete cosine transform* (DCT) or *discrete sine transform* (DST). An example of this behavior is illustrated in figure 7.1 and figure 7.2. In figure 7.1, the leftmost illustration shows a 32×32 block of samples of a camera-captured image, and the following illustration shows the transform coefficients after applying a 32×32 DCT-II transform. The following illustration in figure 7.1 shows the transform coefficients after quantization, while the rightmost illustration shows the reconstructed block of samples. This example demonstrates the energy compaction property of the transform and its effectiveness in combination with quantization for camera-captured content. On the contrary, for screen content, the transform typically yields a much smaller energy compaction, as it is illustrated for a 32×32 block of a desktop screenshot in figure 7.2. Furthermore, for screen content, the inverse transformed quantization noise appearing in the reconstructed block of samples is more disturbing, demonstrating that applying a transform is ineffective. While camera-captured content commonly consists of textures that are distributed smoothly over an area, screen content does not consist of such textures. Instead, sharp edges can typically be found in screen content (compare original block samples of figure 7.1 and figure 7.2). This difference is the reason for the transform being less effective for screen content than for camera-captured content.

An important reason that prevents even higher coding efficiency for screen content when bypassing the transform is that, in the approach typically used, the directly quantized residual samples are coded with the same entropy coding as the quantized transform coefficients in conventionally coded blocks. Although it is known that the coding efficiency of TSM can be further improved by adjusting the level coding, as done by modifications to the context modeling in HEVC *Range Extensions* (REext) [45], the potential coding efficiency improvement achievable by a dedicated TSRC remains unclear.

7.2 | Transform Skip Mode in HEVC

TSM is an additional coding mode for which the residual samples of a block obtained after prediction are quantized directly, and the resulting TSM levels are coded with the conventional level coding in HEVC. A dedicated flag called b_{tsm} specifying whether the transform is bypassed or not is transmitted after the signaling of $b_{cbf} = 1$ (*coded block flag*) for each block. This signaling order, i.e., b_{tsm} after $b_{cbf} = 1$, ensures that the signaling overhead is minimized, because the inverse transform also yields zero residual samples for a block with only zero levels. The coding of b_{tsm} employs a single context model in HEVC, which makes the

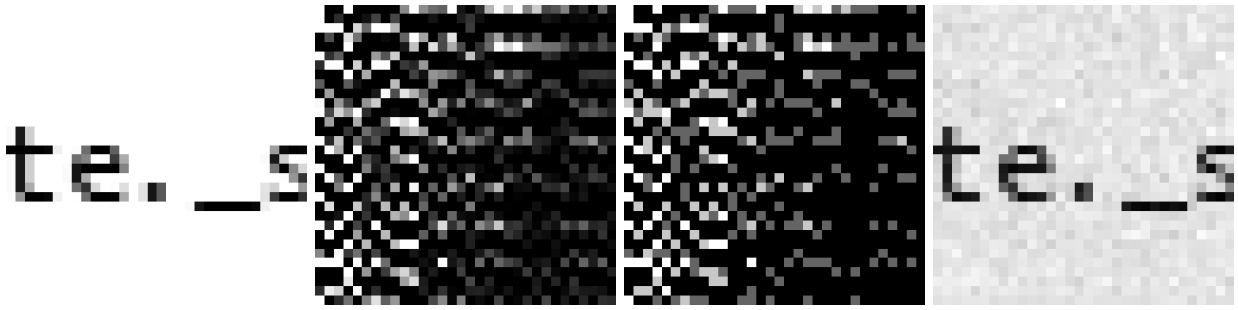

Figure 7.2
Example of a 32×32 DCT-II transform, quantization, and inverse transform for a desktop screenshot. The leftmost illustration shows the original block samples, and the next illustration shows the transform coefficients after applying a DCT-II transform. The following illustration shows the transform coefficients after quantization, and the rightmost illustration shows the reconstructed block of samples.

signaling efficient also for camera-captured content, where blocks with $b_{tsm} = 0$ are primarily coded. TSM is restricted to 4×4 blocks in HEVC [111, 112, 113], i.e., the b_{tsm} flag is only signaled when the transform block has a size of 4×4 samples. Although allowing TSM for larger transform block sizes improves the coding efficiency significantly, the block size restriction was only lifted in HEVC REext. Nonetheless, the default configuration of the REext reference software encoder still restricts the block size for TSM to 4×4 blocks to limit the encoding time. The concept of HEVC was initially maintained for the VVC development, i.e., TSM was restricted to 4×4 blocks, because of the availability of coding tools dedicatedly developed for screen content, such as IBC or PLT. These coding tools provide significant coding efficiency improvements for screen content, which cannot be achieved by TSM restricted to 4×4 blocks.

7.2.1 | Modifications for TSM in HEVC Range Extensions

Some refinements to the conventional level coding for TSM levels were introduced in HEVC REext to provide higher coding efficiency for screen content. Two components of the level coding for TSM levels differ from the conventional level coding in HEVC REext. Firstly, a dedicated context model is used to code the b_{sig} flags of TSM levels to avoid interference with the context models used for transform coefficient levels [114]. This modification is effective because, in contrast to transform coefficient levels, the probability of non-zero TSM levels tends to be rather constant across the block. Secondly, the block of TSM levels is rotated by 180° prior to the level coding, which is equivalent to reversing the scanning pattern [114]. It was observed that for intra-predicted blocks, the prediction errors usually become greater with increasing spatial distance from the reference samples located at the top and left border of the block. This observation implies that the reverse scanning pattern used in the conventional level coding leads to a processing of TSM levels from high to low magnitudes. However, the employed reverse scanning pattern in the conventional level coding expects that the processing of the levels is from low to high magnitudes, which is certainly true for conventional transform coefficient levels due to the energy compaction in the transform domain. Therefore, the rotation adjusts the processing of the levels to be more suitable for the existing context modeling and binarization. Coding efficiency improvements of -2.1% in the *All-Intra* configuration and of -1.4% in the *Random-Access* configuration for class F were reported in [114] when both modifications are utilized, compared to the case without these modifications.

7.2.2 | Reference Implementation and Experimental Setup

In the first investigation of this chapter, the impact of the TSM block size restriction is examined. Therefore, a brief review of the reference software implementation and the experimental setup used for the investigations in this chapter is given first.

The VVC reference software implementation in version 17.0 (VTM-17) is employed as the basis for the investigations conducted in this chapter. VTM-17 implements a conventional level coding and a TSRC, with the latter being a refinement of the variant developed in this thesis. Although the TSRC presented in this chapter was originally developed using older VTM versions, VTM-17 is selected as the software basis,

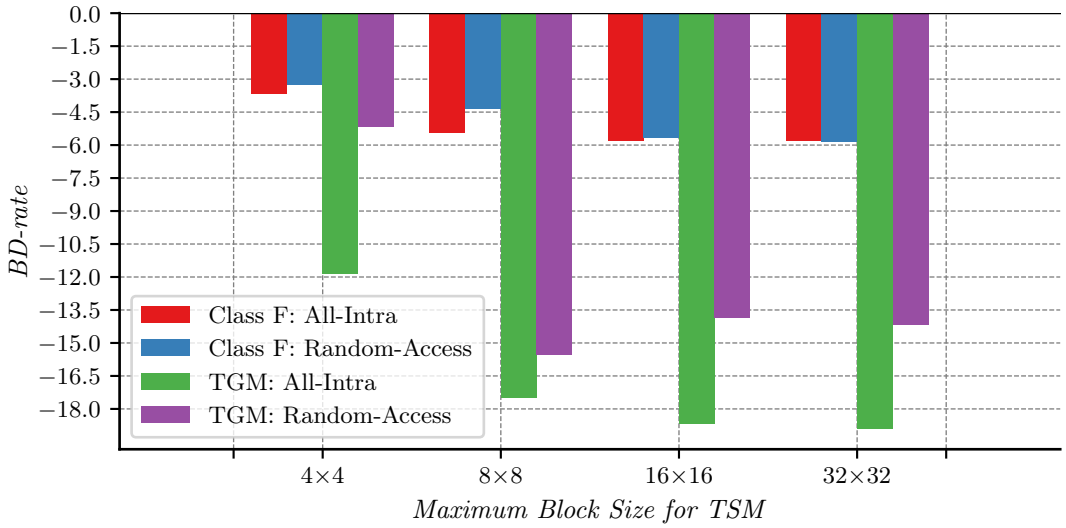

Figure 7.3
Coding efficiency of the investigation on the TSM block size restriction (IMP7-0). The permitted setting for the maximum allowed TSM block size is 4×4, 8×8, 16×16, or 32×32. Let N_{max} denote the edge length of the maximum allowed TSM block size, then TSM is supported for blocks with both the width and height less than or equal to N_{max}.
Anchor for BD-rate computations: IMP7-0 with TSM disabled

because it includes all coding tools supported in the VVC specification. In the initial implementation IMP7-0 of this chapter, the TSRC of VTM-17 is replaced by the conventional level coding of VTM-17, but without the restriction on the number of context-coded bins described in section 6.5 for TSRC. Furthermore, all context models used for the transform coefficient level coding are initialized as *equi-probable* (EP). This implementation is comparable to the development state prior to the development of the TSRC presented in this chapter. For all conducted coding experiments, the coding performance was evaluated by measuring the *Bjøntegaard delta bit-rate* (BD-rate) [66] for the luma component between two codec versions, where the encoder configurations mainly follow the VVC *common test conditions* (CTC) [81]. Besides disabling the coding tools *sign data hiding* (SDH), *rate-distortion optimized quantization* (RDOQ), and *trellis-coded quantization* (TCQ), as in the preceding chapters, for the investigation in this chapter, IBC, PLT, and block-based differential pulse-code modulation were additionally disabled to exclude the impact of other screen content coding tools on TSRC. Finally, BD-rates are reported only for the test sequences of class F and *text and graphics with motion* (TGM) [115], which are screen content test sequences, and only the first second of each test sequence was coded.

7.2.3 | Block Size Restriction and Coding Efficiency

In HEVC, luma transform blocks have a size of 4×4, 8×8, 16×16, or 32×32 samples, while non square-blocks of size $W \times H$, with $W, H \in \{4, 8, 16, 32, 64\}$ are additionally permitted in VVC. The restriction of TSM to 4×4 blocks reintroduced at the beginning of the VVC development is implemented in a way that if the width or the height is greater than the allowed edge size of the square block, TSM is not supported. This restriction poses the question of the achievable coding performance when allowing TSM to be applied to larger block sizes in VVC. A set of coding experiments using IMP7-0 with different settings for the maximum allowed TSM block size was conducted, and the experimental results are summarized in figure 7.3. Four different settings for the maximum allowed TSM block size (4×4, 8×8, 16×16, and 32×32) are investigated. Let N_{max} denote the edge length of the maximum allowed TSM block size, then TSM is permitted for a block when both its width W and height H are less than or equal to N_{max}, i.e., $W \leq N_{max}$ and $H \leq N_{max}$. Significant coding efficiency improvements can be achieved for screen content sequences when increasing the maximum allowed TSM block size. A certain saturation in coding efficiency can be observed when increasing the maximum allowed TSM block size from 16×16 to 32×32. Consequently, a further increase in the maximum allowed

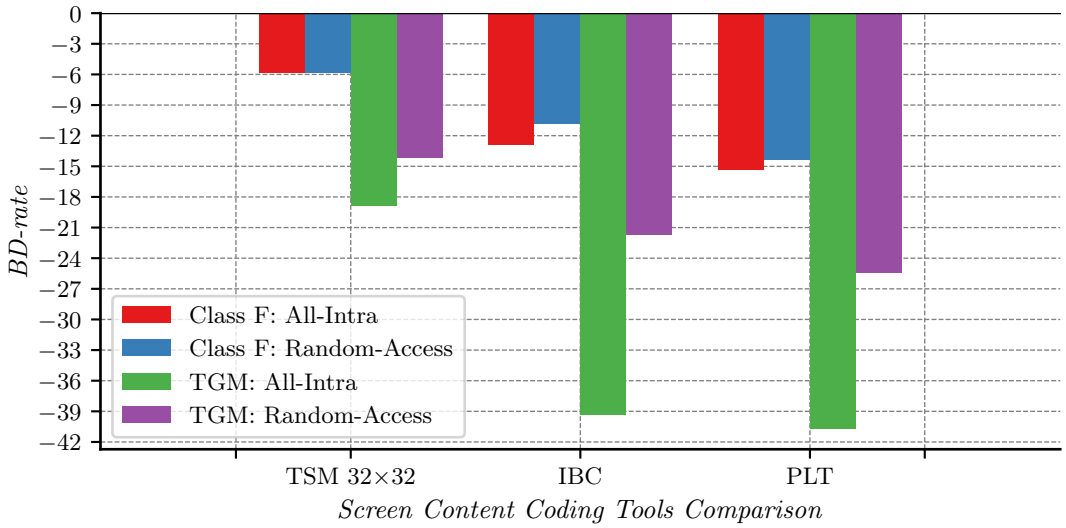

Figure 7.4
Coding efficiency of the investigations on the impact of different screen content coding tools, including TSM with the conventional level coding, but without restricting the number of context-coded bins (IMP7-0).
Anchor for BD-rate computations: IMP7-0 with all screen content tools disabled

TSM block size does not improve the coding efficiency significantly, as analyzed in [116]. The encoding time with TSM enabled increases by about 30% when setting the maximum allowed TSM block size to 32×32 [117] relative to an encoder configuration with TSM disabled.

Because TSRC would provide significantly improved coding efficiency only for the case that TSM can be utilized for larger blocks, a fast encoder control was developed by the author during the development of TSRC presented in this chapter. This fast encoder control for TSM makes a maximum allowed TSM block size equal to 32×32 feasible [116, 46]. For camera-captured content, the developed fast encoder control results in virtually the same coding efficiency and the same encoding time with TSM enabled and a maximum allowed TSM block size equal to 32×32 as for the configuration with TSM disabled. For screen content, this fast encoder control results in the same coding efficiency as the configuration without the encoder speed-up, while the encoding time provided by this fast encoder control is almost the same as for the case where TSM is disabled. In VTM-17, the default setting for the maximum allowed TSM block size is 32×32, because the aforementioned fast encoder control is implemented in the encoder. This fast encoder control is also used for the coding experiments conducted in this chapter, including the results in figure 7.3.

7.2.4 | Comparision to Other Screen Content Tools in VVC

Another set of coding experiments using IMP7-0 was conducted to assess the coding efficiency improvements for the individual screen content coding tools supported in VVC. In figure 7.4, the coding efficiencies are summarized for TSM configured with a maximum allowed block size equal to 32×32 samples (IBC and PLT disabled), for IBC enabled (TSM and PLT disabled), and for PLT enabled (IBC and TSM disabled), relative to IMP7-0 with TSM, IBC, and PLT disabled. The highest coding efficiency is provided by PLT, closely followed by IBC with a slightly lower coding efficiency, while TSM provides roughly half the coding efficiency provided by IBC or PLT. Given these results, the TSM path as it is, i.e., quantizing and coding the residual samples directly, is not sufficient to achieve a coding efficiency comparable with that provided by IBC or PLT. Note that TSM can be combined with IBC, because IBC is implemented as a prediction mode, while a combination with PLT is not possible, because PLT replaces the transform coding completely. The decoding times for all three coding tools are similar to that of the anchor, whereas the encoding times vary significantly. While the encoding time of TSM is similar to that of the anchor due to the fast encoder control of [46], the encoding time of IBC is twice that of the anchor. For PLT, the encoding time is about 10% higher than that of the anchor.

7.2.5 | Impact of Level Coding Components on Coding Efficiency

The modifications introduced in HEVC RExt are based on observations about the signal characteristics of screen content, which can be classified as follows. Firstly, non-zero TSM levels are typically concentrated in a specific region of a given block, where the location of the region may vary across the blocks. Secondly, when non-zero TSM levels exist within a block, their absolute magnitudes tend to increase slightly from the top-left to the bottom-right of the block. This property is different from the typical behavior of absolute transform coefficient levels, where a significant decrease in absolute magnitudes can be observed from the top-left to the bottom-right of the block. A varying local concentration of non-zero TSM levels would also mean that signaling the last significant scanning position might not be beneficial. This assumption is investigated in IMP7-1, which is based on IMP7-0. In IMP7-1, the coding of the last significant scanning position described in section 5.2.1 is disabled for the level coding of transform skip blocks. When no last significant scanning position is transmitted, the b_{sig} flags are coded for all scanning positions. Two exceptions exist: For sub-blocks with $b_{csf} = 0$, where the TSM levels of all scanning positions within the sub-block are inferred to be zero-valued, and for the last scanning position of a sub-block when all previously coded scanning positions of the sub-block are zero-valued. HEVC RExt employs a single dedicated context model for coding the b_{sig} flags, which raises the general question of whether separate context models for TSM levels should be used due to different statistics. An investigation is performed with IMP7-2, which is based on IMP7-1. The template-based context modeling and the position-dependent context model sets are disabled in IMP7-2, and a single context model is used for each context-coded flag. Because the conventional level coding, from which TSRC is derived, utilizes binarization #2 denoted in table 6.3, the context-coded flags are b_{sig}, $b_{|x|>1}$, b_{par}, and $b_{|x|>3}$. A single context model is used for b_{sig}, another context model is used for b_{par}, and a further context model is employed for both $b_{|x|>1}$ and $b_{|x|>3}$, where all context models are initialized as EP. For luma and chroma, different context model sets are used. The investigation on the efficiency of the level coding is further extended in IMP7-3, which is based on IMP7-2. In IMP7-3, the Rice parameter selection is disabled, i.e., the Rice parameter $k = 0$ is always used independently of the sum of absolute levels inside the template. Finally, in IMP7-4 that is based on IMP7-3, the scanning pattern is reversed. Instead of a sub-block-wise scanning from the bottom-right to the top-left of the block as illustrated in figure 5.2, the scanning is sub-block-wise from the top-left to the bottom-right of the block. This modification evaluates the rotation by 180° implemented in HEVC RExt for TSM levels described in section 7.2.1.

The BD-rates for the investigations are summarized in figure 7.5 relative to IMP7-0 with the maximum TSM block size set equal to 32×32. Without signaling the last significant scanning position, the coding efficiency is virtually the same as provided by the anchor for class F, while a slight loss in coding efficiency can be observed for TGM. Disabling the template-based context modeling further improves the coding efficiency for the class F test sequences in the *All-Intra* configuration, while the losses in coding efficiency are significant for TGM. For all test sequences, disabling the Rice parameter selection introduces a significant loss in coding efficiency, while reversing the scanning pattern improves the coding efficiency for all test sequences, even in the *Random-Access* configuration. These investigations provide a starting point and reveal which components of the level coding improve and which components harm the coding efficiency. Firstly, coding the last significant scanning position is unnecessary for transform skip residual samples. Secondly, the template-based context modeling and the Rice parameter selection can provide coding efficiency improvements, and the approach implemented in HEVC RExt is not optimal for all test sequences. Thirdly, reversing the scanning pattern improves coding efficiency for all test sequences, and even for a configuration with inter-predicted blocks. IMP7-4 serves as the basis for further investigations, which are mainly focused on binarization, context modeling, and Rice parameter selection. An advantage of using IMP7-4 is that the context modeling and the Rice parameter selection are extremely simplified, facilitating further investigations for improving the context modeling and Rice parameter selection for transform skip blocks.

7.3 | Binarization and Context Modeling of TSM Levels

The investigations IMP7-1 to IMP7-4 demonstrate that the context modeling and the Rice parameter selection of the conventional level coding provide improved coding efficiency also for the TSM levels. This section focuses on further investigating both mentioned aspects and starts with a brief analysis of the statistics of

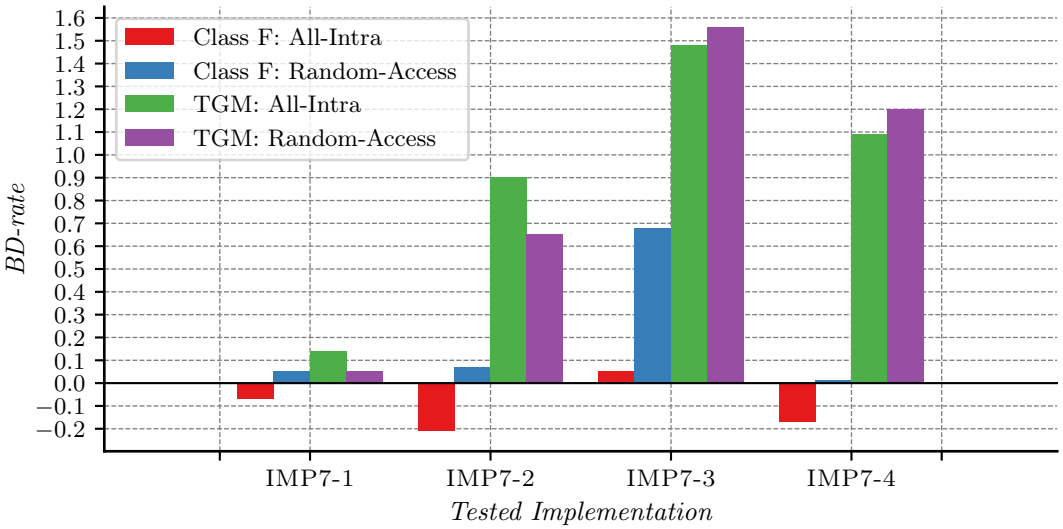

Figure 7.5
Coding efficiency of the investigations on the impact of different level coding components on the coding efficiency (IMP7-1 to IMP7-4). In IMP7-1, the coding of the last significant scanning position is removed, and in IMP7-2 that is based on IMP7-1, each context-coded flag uses a single dedicated context model. In IMP7-3 that is based on IMP7-2, the Rice parameter $k = 0$ is always used, and in IMP7-4 that is based on IMP7-3, the scanning pattern is reversed.
Anchor for BD-rate computations: IMP7-0 with maximum TSM block size equal to 32×32

TSM levels. Based on the findings, investigations on the number of additional context-coded flags and the context modeling are presented.

7.3.1 | Statistics of TSM Levels

The conventional level coding has been designed with the statistics of the absolute levels that appear for camera-captured content in mind, as exemplarily illustrated by the histograms in figure 3.2. These empirical distributions mainly arise from the transform and its energy compaction property for the residual signals of camera-captured content. Without the transform, it is expected that the distributions are completely different. The diagram on the left-hand side of figure 7.6 illustrates the histogram of absolute TSM levels for 8×8 blocks, which is dominated by zero-valued levels. On the right-hand side in figure 7.6, the histogram is illustrated for the same data, but only considering non-zero absolute TSM levels. When ignoring the b_{sig} flags,

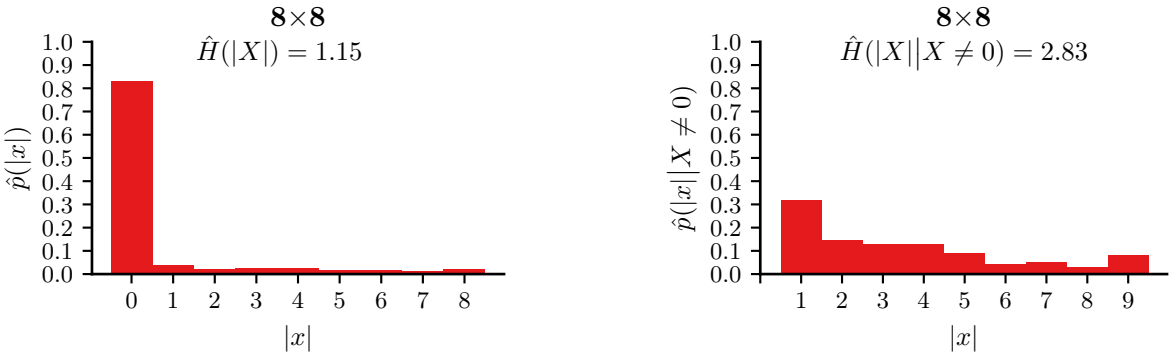

Figure 7.6
Histograms of absolute and non-zero absolute TSM levels for 8×8 blocks, acquired by coding the first frame of the *SlideShow* sequence using VTM-17. Quantized residual samples greater than eight were grouped into the last bin of the histogram, and the QP was set to 32.

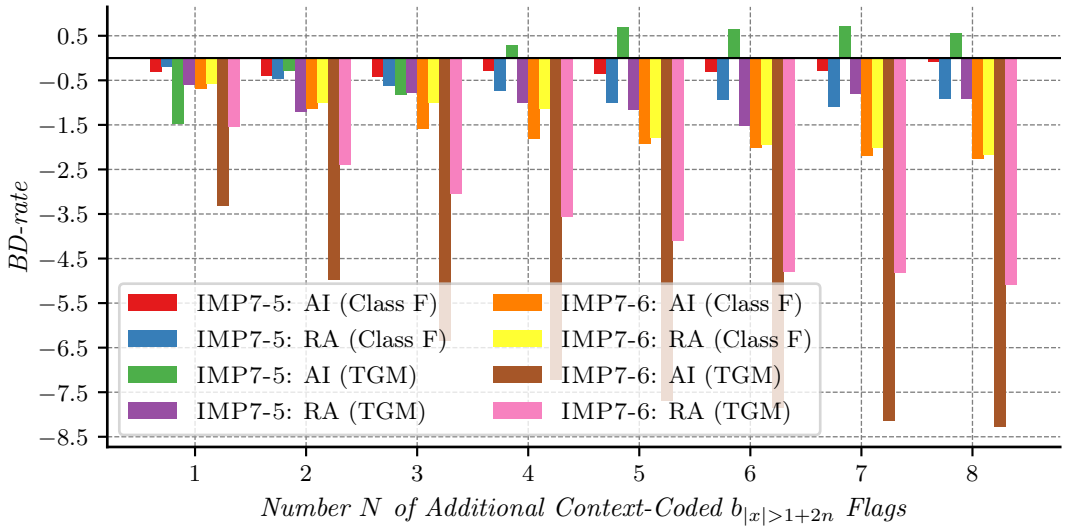

Figure 7.7
Coding efficiency of IMP7-5 and IMP7-6 implementing additional context-coded $b_{|x|>1+2n}$ flags. In IMP7-5, a single context model is used for all $b_{|x|>1+2n}$ flags, a concept that is inherited from the conventional level coding. In IMP7-6, each $b_{|x|>1+2n}$ flag employs a single dedicated context model.
Anchor for BD-rate computations: IMP7-4

the distribution of the absolute TSM levels is relatively flat, e.g., the probabilities for successive values may be very similar, as illustrated on the right-hand side in figure 7.6 for $|x| \in \{2, 3, 4\}$. A further observation is that for some specific values of $|x|$, the probability is smaller than for both the next smaller and the next greater values. In the example illustrated in figure 7.6, a value with that property is $|x| = 6$. Both observations, i.e., a relatively flat distribution and specific values with higher or lower probabilities than their neighboring values, can also be observed in the histograms of absolute TSM levels of other block sizes. Consequently, the geometric distribution used to model the empirical distribution of absolute transform coefficient levels does not fit the observed data for absolute TSM levels. Based on this conclusion, the representation of absolute transform coefficient levels, optimized for a geometric distribution (compare section 4.3), will be modified for TSM levels by increasing the number of context-coded flags to improve the coding efficiency. Note that increasing the number of context-coded flags for each level does not pose a problem to the throughput, because the number of context-coded bins is subsequently limited by a budget, as described in section 6.5.

7.3.2 | Additional Context-Coded $b_{|x|>1+2n}$ Flags

The binarization of TSM levels in IMP7-4 is the same as for absolute transform coefficient levels, which corresponds to binarization #2 denoted in table 6.3 and specifies the context-coded flags b_{sig}, $b_{|x|>1}$, b_{par}, and $b_{|x|>3}$. In the following description, the $b_{|x|>1}$ and the $b_{|x|>3}$ flags are represented as $b_{|x|>1+2n}$ flags, with $n \in \{0, 1\}$. Because TCQ is not used in combination with TSM, the b_{par} flag is no longer required and could therefore be removed from the binarization. However, to keep the syntax and semantics of TSRC aligned with that of the conventional level coding, the binarization of the conventional level coding is only extended by additional context-coded flags for TSRC. In the following investigations, the context modeling of b_{par} remains unchanged compared to IMP7-2, i.e., a single dedicated context model is used, because a modification to its context modeling revealed a negligible impact.

Let N denote the total number of additional context-coded $b_{|x|>1+2n}$ flags which are coded after $b_{|x|>3}$. An example of such a binarization is summarized in table 7.1 for the configuration with four additional context-coded $b_{|x|>1+2n}$ flags. In the first investigation implemented in IMP7-5, the context modeling is untouched, and each additional $b_{|x|>1+2n}$ flag with $n \in \{2, \ldots, N+1\}$ uses the same context model as $b_{|x|>1}$ and $b_{|x|>3}$. This concept of using the same context models for $b_{|x|>1}$ and $b_{|x|>3}$ was inherited from the conventional level coding, where the original investigation, presented in section 5.4.3, showed that using the same set of context

models for $b_{|x|>1}$ and $b_{|x|>2}$ is feasible for transform coefficient levels that are geometrically distributed, as shown in section 4.3. However, the context modeling for $b_{|x|>1+2n}$ in IMP7-5 inherited from the conventional level coding may be unsuitable for TSM levels that are not geometrically distributed. Therefore, a second investigation is performed to address the context modeling, where each $b_{|x|>1+2n}$ flag with $n \in \{0, \ldots, N+1\}$ employs a single dedicated context model. This investigation is implemented in IMP7-6, and the total number of context models for coding the $b_{|x|>1+2n}$ flags is equal to $N+2$.

The experimental results for both investigations on additional context-coded $b_{|x|>1+2n}$ flags are summarized in figure 7.7. While coding additional context-coded $b_{|x|>1+2n}$ flags generally provides coding efficiency improvements, except in the *All-Intra* configuration for the TGM test sequences with IMP7-5, a suitable context modeling has a much greater impact, as indicated by the coding efficiencies provided by IMP7-6. The inappropriate context modeling implemented in IMP7-5 results in varying coding efficiency when increasing the number of additional $b_{|x|>1+2n}$ flags. In contrast, for IMP7-6, the coding efficiency increases steadily with the number of additional $b_{|x|>1+2n}$ flags. Given these results, the context modeling of IMP7-6 with four additional context-coded $b_{|x|>1+2n}$ flags is selected as a compromise between coding efficiency and the *maximum number of context-coded bins per sample* (mcps) for the TSM levels. This configuration is referred to as IMP7-6* and is used as the basis for the following investigations.

7.3.3 | Template-Based Context Modeling of b_{sig}

In IMP7-6*, a single context model is utilized for coding the b_{sig} flags, and the following investigations analyze the impact of a template-based context modeling for b_{sig}. In the conventional level coding, five neighboring locations are evaluated, as depicted by the leftmost illustration in figure 7.8. This local template is mirrored as depicted by the following illustration in figure 7.8, because the forward scanning is used in TSRC, which was introduced in IMP7-4. In the first investigation implemented in IMP7-7, the efficiency of the context modeling design for the conventional level coding is analyzed, where the template consists of five neighboring locations, i.e., the shape of the template is as in the second illustration in figure 7.8. A further investigation analyzes the impact of reducing the template to three neighboring locations as depicted by the third illustration in figure 7.8. This template configuration is investigated in an additional coding experiment using the implementation IMP7-8. In the final investigation implemented in IMP7-9, the template is further reduced to two neighboring locations as depicted in the rightmost illustration of figure 7.8, because screen content signals often consist of horizontal and vertical edges.

$\lvert x\rvert$	b_{sig}	$b_{\lvert x\rvert>1}$	b_{par}	$b_{\lvert x\rvert>3}$	$b_{\lvert x\rvert>5}$	$b_{\lvert x\rvert>7}$	$b_{\lvert x\rvert>9}$	$b_{\lvert x\rvert>11}$	z
0	0								
1	1	0							
2	1	1	0	0					
3	1	1	1	0					
4	1	1	0	1	0				
5	1	1	1	1	0				
6	1	1	0	1	1	0			
7	1	1	1	1	1	0			
8	1	1	0	1	1	1	0		
9	1	1	1	1	1	1	0		
10	1	1	0	1	1	1	1	0	
11	1	1	1	1	1	1	1	0	
12	1	1	0	1	1	1	1	1	0
13	1	1	1	1	1	1	1	1	0
14	1	1	0	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1

Table 7.1
Binarization table when coding four additional context-coded $b_{|x|>1+2n}$ flags in TSRC. The additional context-coded $b_{|x|>1+2n}$ flags are $b_{|x|>5}$, $b_{|x|>7}$, $b_{|x|>9}$, and $b_{|x|>11}$ in this case.

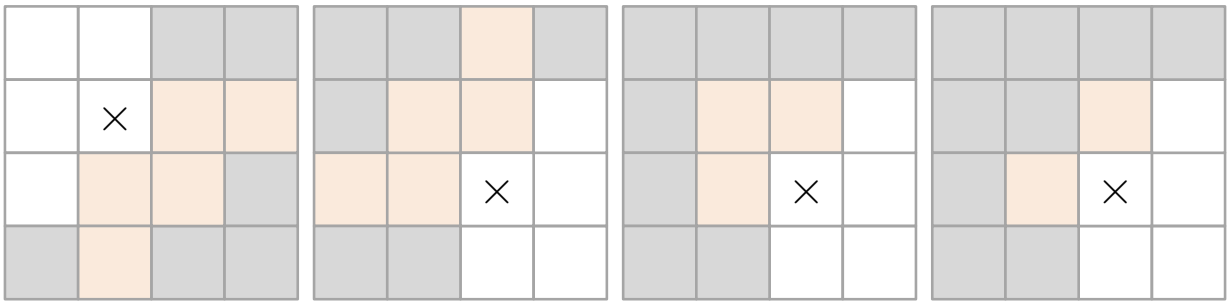

Figure 7.8
Template configuration for IMP7-7 to IMP7-9, where the template-based context modeling of b_{sig} in TSRC is investigated. The leftmost template configuration corresponds to IMP7-4, i.e., the template configuration of the conventional level coding. The following template configurations are implemented for IMP7-7, IMP7-8, and IMP7-9.

Experimental Results for Different Template Shapes

For all tested implementations, the context modeling for b_{sig} inherited from the conventional level coding is used. Let T be the size of the template, then, in the first step, the clipped sum $\mathcal{S}_c$ is derived by:

$$\mathcal{S}_c = \sum_{i=0}^{T-1} \min\left(|x\left(i\right)|, 12 + b_{par}\right). \tag{7.1}$$

Recall that two coding phases are used for the level coding, as in the case of conventional level coding, and only the partially reconstructed values available after the first coding phase can be used as input for the template-based context modeling. Therefore, the maximum absolute level magnitude that can be utilized for the template-based context modeling depends on the number of coded $b_{|x|>1+2n}$ flags, which is equal to six in all tested implementations. For six context-coded $b_{|x|>1+2n}$ flags, the absolute maximum level magnitudes are 12 or 13, depending on the value of b_{par}, as illustrated in table 7.1. In the following second step, $\mathcal{S}_c$ is mapped to the context model offset δ_{sig} according to

$$\delta_{sig} = \min\left(5, \mathcal{S}_c\right). \tag{7.2}$$

Compared to IMP7-6*, the only modification is the context modeling of the b_{sig} flags and the number of context models for b_{sig} is equal to six for all tested implementations.

The experimental results of the investigations are summarized by figure 7.9 using IMP7-6* as the anchor for BD-rate computations. Compared to the experimental results of IMP7-2, where the template-based context modeling for all context-coded flags was disabled, two aspects can be observed. Firstly, the coding efficiency is improved for all tested configurations and implementations, which could not be expected since disabling the template-based context modeling in IMP7-2 resulted in a coding efficiency improvement for class F in the *All-Intra* configuration, but coding efficiency losses in all other configurations. Secondly, all tested template configurations provide higher absolute BD-rate values than those of IMP7-2, meaning that the template-based context modeling of IMP7-7 to IMP7-9 provides higher coding efficiency than the variant of the conventional level coding implemented in IMP7-1 or IMP7-0. The differences between IMP7-7 and IMP7-1 are the sub-block-wise scanning from the top-left to the bottom-right corner of the block, the fixed Rice parameter selection with $k = 0$, and a single context model for all other context-coded flags. None of these changes explain that using the template-based context modeling for b_{sig}, as in IMP7-7 to IMP7-9, has a different effect on the compression efficiency compared to the investigation in IMP7-2, where the template-based context modeling was disabled. Consequently, the main effect can be traced back to separating the context models used for coding the TSM levels and absolute transform coefficient levels. Furthermore, reducing the template size from five to two neighboring locations improves the coding efficiency for TSM levels.

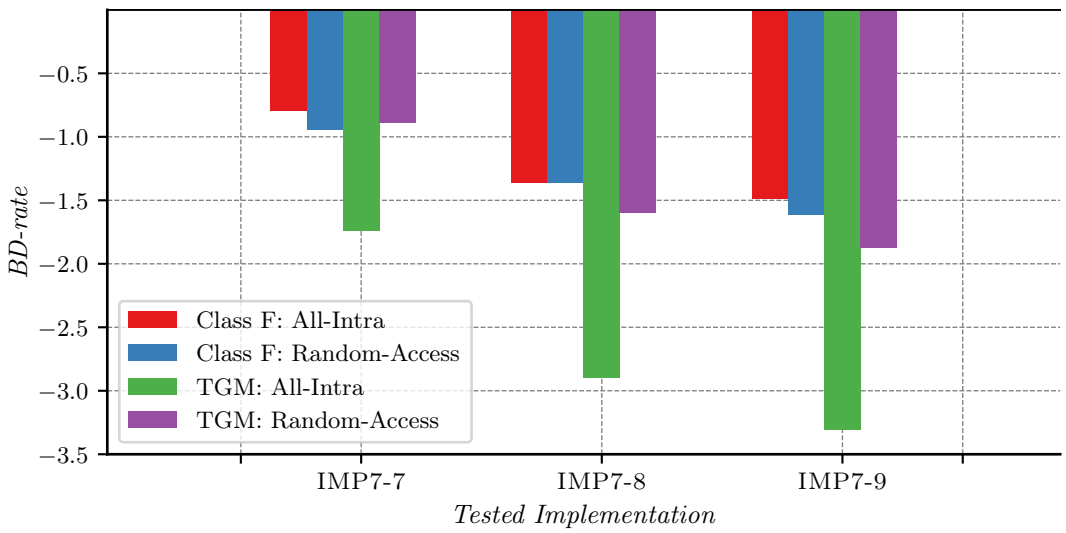

Figure 7.9
Coding efficiency of the investigations on different local template configurations for the context modeling of b_{sig}. In IMP7-7, the local template consists of five neighboring locations similar to the conventional level coding. In IMP7-8, the local template is reduced to the three next neighboring locations, while in IMP7-9, only the top and the left neighboring locations are considered.
Anchor for BD-rate computations: IMP7-6*

Investigation on the Clipping Value M_{sig}

In IMP7-7 to IMP7-9, the shape of the template was experimentally investigated, while the mapping of $\mathcal{S}_c$ to the context model offset δ_{sig} was inherited from the conventional level coding by clipping $\mathcal{S}_c$ to five. A further investigation implemented in IMP7-10, based on the template configuration consisting of two neighboring locations (IMP7-9), was conducted to justify the choice of the selected clipping value. This investigation corresponds to the investigation IMP5-9 performed in section 5.4.2, but for TSM levels and a template configuration with two neighboring locations. Let M_{sig} denote the clipping value, then equation (7.2) is changed in IMP7-10 to:

$$\delta_{sig} = \min\left(M_{sig}, \mathcal{S}_c\right). \tag{7.3}$$

For IMP7-10, the number of context models for the significance flag is equal to M_{sig}+1. The coding efficiencies provided by different values of M_{sig} are summarized in figure 7.10. While increasing the clipping value provides further coding efficiency improvements, the achievements are insignificant compared to the coding efficiency improvement provided by, for example, the introduction of additional context-coded $b_{|x|>1+2n}$ flags. For this reason, the clipping value is kept equal to five, as in the conventional level coding, for further investigations.

Alternative Context Modeling Evaluating b_{sig}

For absolute transform coefficient levels, neighboring frequency locations consisting of a high absolute magnitude increase the probability that the current scanning position is non-zero, as indicated by the investigations in section 5.4.2. For TSM levels, this property is less pronounced, meaning that the estimation of the conditional probabilities for b_{sig} are not improved significantly by evaluating absolute TSM levels greater than one. Consequently, evaluating the value of the neighboring b_{sig} flags could be sufficient. This aspect is briefly analyzed in a further investigation, which is based on the template-based context modeling with two neighboring locations (IMP7-9). Instead of deriving the sum $\mathcal{S}_c$ as in equation (7.1), the sum $\mathcal{S}_c$ is calculated according to

$$\mathcal{S}_c = \sum_{i=0}^{T-1} \min\left(|x|\,(i), 1\right). \tag{7.4}$$

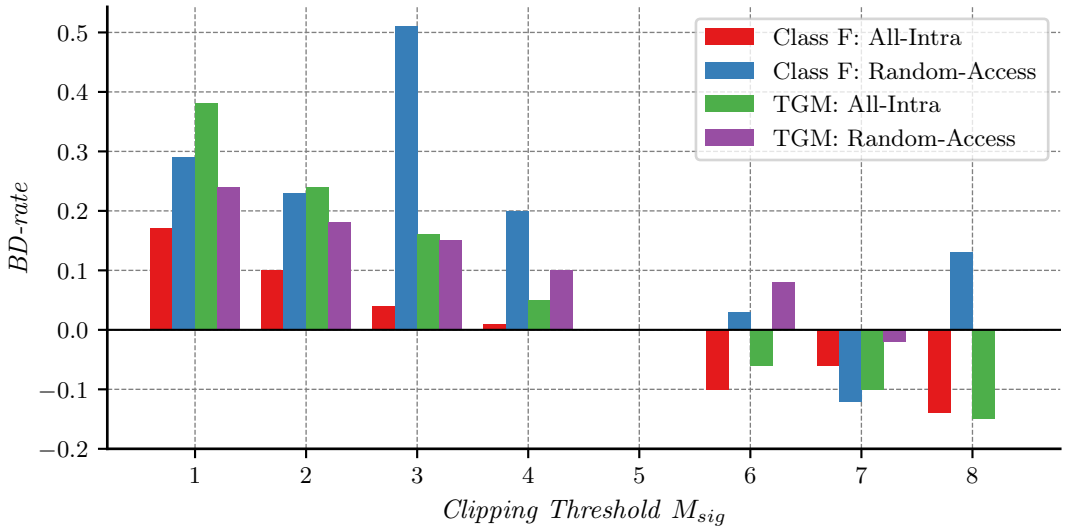

Figure 7.10
Coding efficiency of IMP7-10 that implements different configurations for the clipping value M_{sig}, which is used to limit $\mathcal{S}_c$ prior to the mapping to the context model offset δ_{sig}.
Anchor for BD-rate computations: IMP7-9

The number of context models for coding b_{sig} is equal to three in this investigation. Compared to IMP7-9 that uses equation (7.1) to derive a context model for b_{sig} and serves as the anchor for BD-rate computations, using equation (7.4) results in BD-rates of 0.04% (*All-Intra* Class F), 0.19% (*Random-Access* Class F), -0.02% (*All-Intra* TGM), and 0.09% (*Random-Access* TGM). This alternative context modeling offers slight implementation advantages. The context modeling of b_{sig} for the next scanning positions can be determined already in parallel after parsing a b_{sig} flag, instead of after parsing the last $b_{|x|>1+2n}$ flag for the current scanning position.

7.3.4 | Template-Based Context Modeling of $b_{|x|>1+2n}$

The investigations on the number of additional context-coded $b_{|x|>1+2n}$ flags (section 7.3.2) showed that for $b_{|x|>1+2n}$ flags with different values of n, different context model sets should be used., because the geometric distribution does not accurately represent the TSM levels. This aspect has to be considered for the following investigations on the template-based context modeling of $b_{|x|>1+2n}$ flags. In the conventional level coding implemented in IMP5-19*, which represents the final configuration of the template-based context modeling presented in chapter 5, the sum $\mathcal{S}_c$ is used for deriving a context model for $b_{|x|>1}$ and $b_{|x|>2}$, while both flags share the same context model set. This approach would result in a significant loss in coding efficiency due to sharing the context model sets, as indicated by the experimental results for IMP7-5 and IMP7-6. An adjustment is necessary to comply with the fact that different context model sets should be used for $b_{|x|>1+2n}$ flags with different values of n. In the first investigation implemented in IMP7-11, the sum $\mathcal{S}_c$ is derived as for b_{sig} denoted in equation (7.1), and it is mapped to the context model offset for $b_{|x|>1+2n}$ according to

$$\delta_{|x|>1+2n} = \min(5, \mathcal{S}_c) + 6n. \tag{7.5}$$

The selected clipping value of five should not impact the coding efficiency significantly, as indicated by the investigations on the clipping value M_{gtX} in IMP5-10 and IMP5-11, presented in section 5.4.3. In IMP7-11, the number of context models for each set is extended by five additional context models, which results in $6(N+2)$ context models for the $b_{|x|>1+2n}$ flags in total.

For typical TSM levels, it can be assumed that the neighboring horizontal or vertical location has a similar magnitude, as shown by the leftmost illustration in figure 7.2. Therefore, for the currently coded $b_{|x|>1+2n}$ flag, evaluating whether the neighboring locations inside the template exceed the threshold $1+2n$ might provide more accurate probability estimates. This aspect is examined in implementation IMP7-12, where $\mathcal{S}_c$

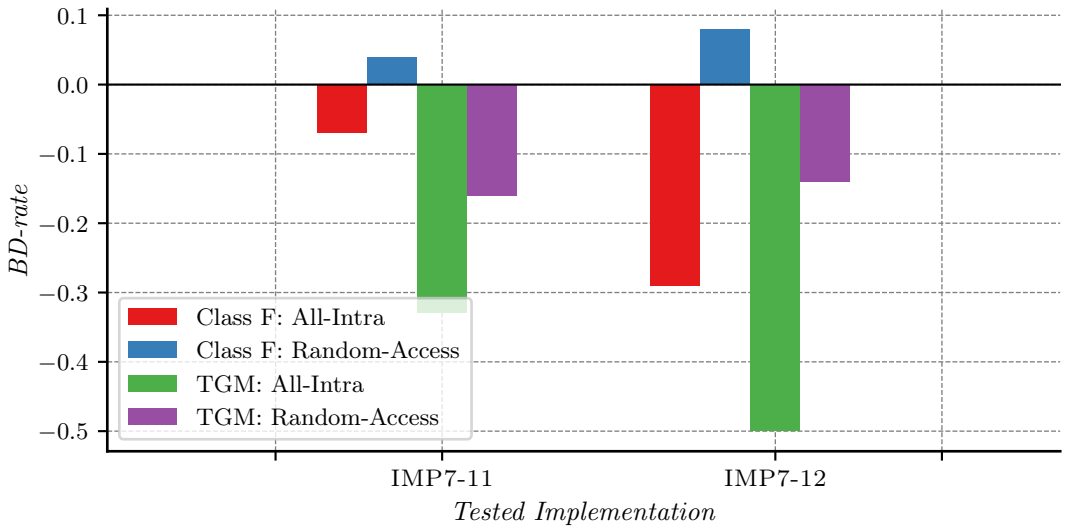

Figure 7.11
Coding efficiency of the investigations on different template-based context modeling for the $b_{|x|>1+2n}$ flags. In IMP7-11, $\mathcal{S}_c$ is clipped to five and mapped to $\delta_{|x|>1+2n}$. In IMP7-12, $\mathcal{S}_{|x|>1+2n}$ is used instead of $\mathcal{S}_c$ to derive the context model offsets $\delta_{|x|>1+2n}$.
Anchor for BD-rate computations: IMP7-9

is replaced by $\mathcal{S}_{|x|>1+2n}$, which is derived by evaluating the value of the neighboring $b_{|x|>1+2n}$ flags with the same value for n as for the $b_{|x|>1+2n}$ flag to be coded. Thus, $\mathcal{S}_{|x|>1+2n}$ represents the number of locations inside the template for which the absolute levels have a value strictly greater than $1+2n$. This evaluation can be summarized by:

$$\mathcal{S}_{|x|>1+2n} = \sum_{i=0}^{T-1} \min\left(\max\left(|x\left(i\right)| - (1+2n), 0\right), 1\right). \tag{7.6}$$

The number of context models for this investigation is equal to $3\left(N+2\right)$, and a clipping of $\mathcal{S}_{|x|>1+2n}$, as for $\mathcal{S}_c$ in IMP7-11, is not required for the derivation of a context model offset. The coding efficiencies obtained for both investigations are summarized in figure 7.11. When using $\mathcal{S}_c$ to derive the context model as in the conventional level coding, the template-based approach provides slight coding efficiency improvements (IMP7-11), especially in the *All-Intra* configuration. For TSM levels, considering only the neighboring $b_{|x|>1+2n}$ flags with the same value of n improves the coding efficiency further, as the BD-rates achieved by IMP7-12 demonstrate. However, these coding efficiency improvements are relatively small compared to the coding efficiency achieved by using a single dedicated context model for each $b_{|x|>1+2n}$. The position-dependent context model sets used in the conventional level coding are not investigated for TSRC, because this approach implicitly assumes that the level magnitudes decrease from the top-left to the bottom-right of a block. Such a concentration of the absolute levels is not given for TSM levels, and therefore, the concept of the position-dependent context model sets is unsuitable for the context modeling of the context-coded flags in TSRC.

7.3.5 | Rice Parameter Selection

For the investigation implemented in IMP7-3 and presented in section 7.2.5, the template-based Rice parameter selection of the conventional level coding was replaced by a fixed Rice parameter of $k = 0$. This investigation indicated that disabling the template-based Rice parameter selection of the conventional level coding harms the coding efficiency. The fixed Rice parameter of $k = 0$ was used throughout all coding experiments from IMP7-3 to IMP7-12. With the introduction of additional $b_{|x|>1+2n}$ flags in section 7.3.2, the probability distribution of the remainders is changed significantly. Due to the additional $b_{|x|>1+2n}$ flags, the number of coded remainders should be less, and the corresponding magnitudes of the remainders should be significantly smaller compared to a configuration without additional $b_{|x|>1+2n}$ flags. Consequently, the

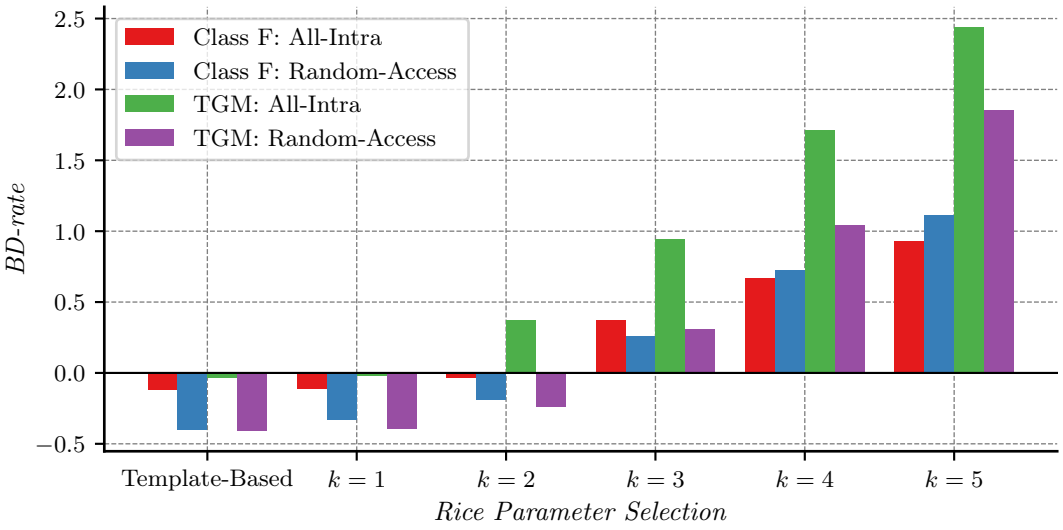

Figure 7.12
Coding efficiency of IMP7-13 and IMP7-14 implementing different techniques for selecting the Rice parameter in TSRC. Different fixed Rice parameters k are investigated in IMP7-13, while in IMP7-14, the template-based Rice parameter selection is utilized using updated thresholds. The anchor uses a Rice parameter of $k = 0$.
Anchor for BD-rate computations: IMP7-9

template-based Rice parameter selection would become less efficient compared to a configuration without additional $b_{|x|>1+2n}$ flags. Based on this indication, the first investigation implemented in IMP7-13 analyzes different fixed Rice parameters. The coding efficiency of the template-based Rice parameter selection with updated thresholds is verified in the second investigation implemented in IMP7-14. The thresholds were determined according to the description in section 5.4.3, and similar to the determination in section 6.4.3 for the binarizations with b_{par}, the maximum useful Rice parameter is equal to two. In IMP7-14, the updated thresholds for the Rice parameter selection are $M^1_{Rice} = 11$ and $M^2_{Rice} = 24$.

In figure 7.12, experimental results for both investigations are summarized, where the bars of the first tick mark on the x-axis denote the coding efficiencies obtained for the template-based Rice parameter selection with updated thresholds implemented in IMP7-14. The coding efficiency provided by fixed Rice parameters implemented in IMP7-13 is summarized in figure 7.12 by the bars of tick marks denoting the used Rice parameter k on the x-axis. Compared to IMP7-9, which serves as the anchor for BD-rate computations and employs a fixed Rice parameter of $k = 0$, updating the thresholds for the template-based Rice parameter selection results in a slight improvement in coding efficiency (IMP7-14). This improvement is, however, almost identical to the improvement provided by using a fixed Rice parameter of $k = 1$ (IMP7-13). Consequently, the assumption that the impact of the Rice parameter selection is marginal after introducing additional $b_{|x|>1+2n}$ flags is reasonable. Based on the experimental results, a fixed Rice parameter of $k = 1$ is chosen for further investigations, and this configuration is referred to as IMP7-13*.

7.3.6 | Context-Coded Sign Information

Up to this point, only absolute TSM levels or absolute transform coefficient levels were considered, because the probability for a positive or negative sign is approximately equal. During the TSRC development, it was discovered that the TSM levels are often asymmetrically distributed around zero, and histograms of non-zero TSM levels are illustrated in figure 7.13 to demonstrate the observation. A probable reason for the observed property is that the predictions supported in the underlying VVC are inefficient on sharp edges commonly found in screen content, resulting in a locally constant signal. Because of this local concentration of constant samples, the predicted samples are either constantly below or above the original samples. Based on this observation, coding b_{sign} with adaptive context models could improve coding efficiency, and the following investigations were performed on the context modeling of b_{sign}:

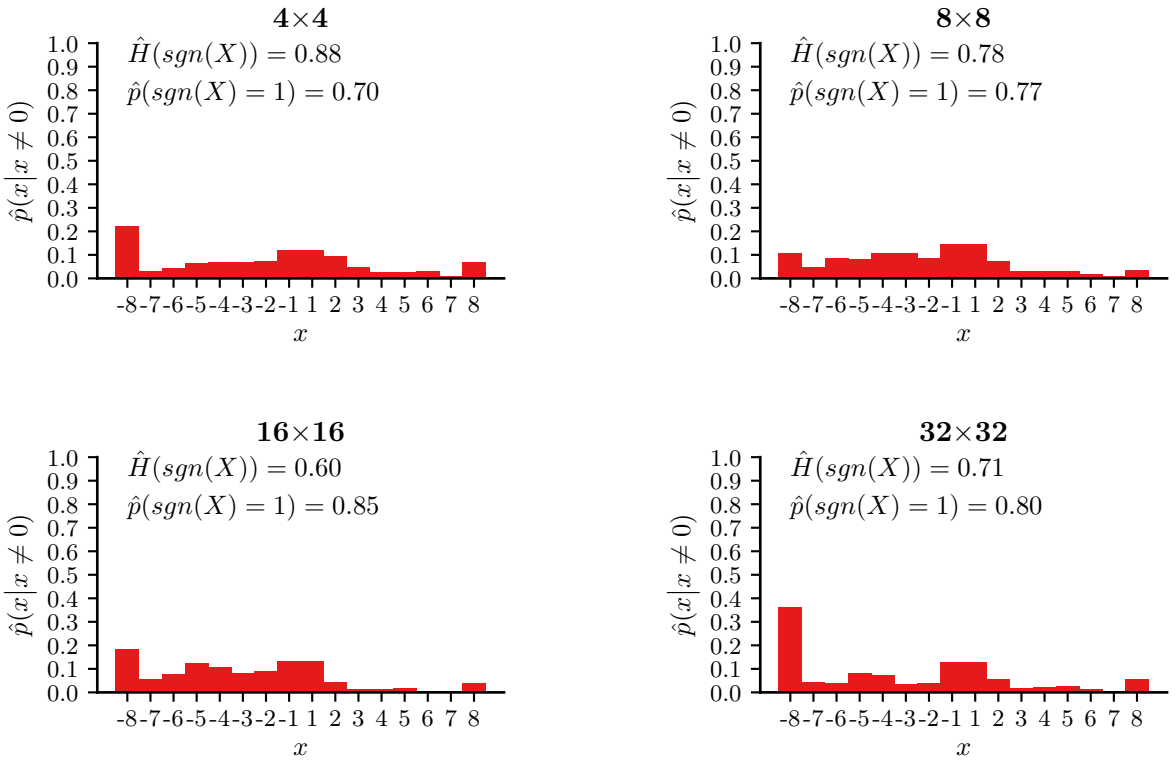

Figure 7.13
Histograms of TSM levels for different block sizes, acquired by coding the first frame of the *SlideShow* sequence using VTM-17. Quantized residual samples with an absolute value greater than eight were grouped into the first or last bin of the histogram, and the QP was set to 32. The empirical probability for a negative sign and the empirical entropy for the sign are denoted below the block size notation on the top of each histogram.

IMP7-15 This investigation aims to verify the effectiveness of an adaptive context model for coding b_{sign}, and the corresponding implementation employs a single dedicated context model for coding the b_{sign} flags.

IMP7-16 In this investigation, it is verified whether a template-based context modeling for the b_{sign} flags can provide additional coding efficiency improvements. The same template as for coding the b_{sig} flags is employed (rightmost illustration in figure 7.8), and the total number of neighboring locations with a negative sign is used as context model offset, resulting in three context models for the b_{sign} flags.

IMP7-17 This implementation further extends the template-based context modeling, which is implemented in IMP7-16, by mapping each unique combination of negative signs inside the template to a context model offset, which results in four context models in total. The goal of this study is to find out whether a more advanced context modeling can provide further coding efficiency improvements.

IMP7-18 An observation made during the development is that the difference in the probability of positive and negative signs becomes more significant with increasing absolute level magnitude. This property can also be observed for the empirical probabilities shown in the example histograms of figure 7.13. This implementation is based on IMP7-16, and it investigates the mentioned observation by using a dedicated context model for $|x| = 1$, which occurs most frequently after zero.

In contrast to the other context-coded flags, where different context model sets are employed in luma and chroma, it was discovered that the distributions of the b_{sign} flags in luma and chroma are comparable.

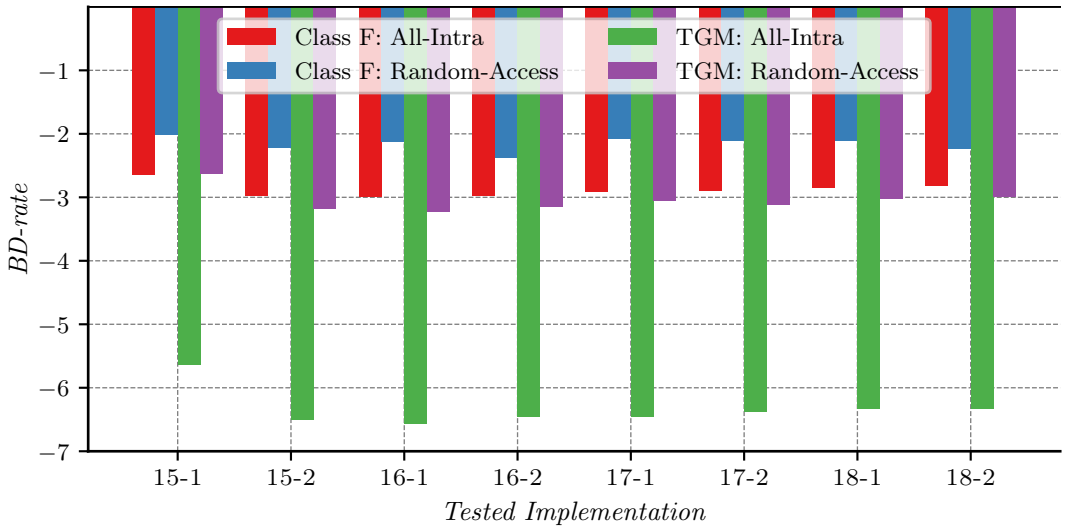

Figure 7.14
Coding efficiency of IMP7-15 to IMP7-18 implementing different context modeling for coding the b_{sign} flags in TSRC, abbreviated as 15 to 18 on the x-axis. The number following the hyphen for each label denotes whether different context models are used for luma and chroma for each implementation. "1" denotes that the same context models are used for luma and chroma, while "2" denotes that different context models are used for luma and chroma.
Anchor for BD-rate computations: IMP7-13*

Therefore, additional coding experiments were conducted for all investigations, where the context models are shared between luma and chroma.

The experimental results of the investigations on coding b_{sign} with context models are summarized in figure 7.14, where the label of each tick mark is the abbreviation of the tested implementation, e.g., 15 is the abbreviation for IMP7-15. The number that follows the hyphen for each label denotes whether different context models are used for luma and chroma for each implementation. "1" denotes that the same context models are used for luma and chroma, while "2" denotes that different context models are used for luma and chroma. Coding the b_{sign} flags with a single context model improves coding efficiency significantly (IMP7-15 as 15-1 in figure 7.14), and using different context models for luma and chroma further increases the improvements (IMP7-15 as 15-2 in figure 7.14). This improvement achieved by different context models for luma and chroma disappeared when using a template-based context modeling, as indicated, for example, by the BD-rates obtained for IMP7-16 in the third and fourth tick marks. Both investigations on using more context models in IMP7-17 and IMP7-18 do not offer a higher coding efficiency than the template-based variant in IMP7-16 that is based on counting the number of negative signs. Based on these experimental results, the chosen context modeling for the b_{sign} is the template-based variant implemented in IMP7-16 with the same context models used for luma and chroma.

7.3.7 | Coding Efficiency Provided by TSRC, IBC, and PLT Enabled

The development of TSRC with its final design implemented in IMP7-16 was presented without considering other screen content tools. Different coding experiments were conducted to investigate the coding efficiencies of TSM and TSRC in combination with the other screen content coding tools, including an analysis of the interference between TSRC and IBC and between TSRC and PLT.

Coding Efficiency of TSM with TSRC

The combined coding efficiency provided by TSM and TSRC is added to the summary presented in figure 7.4 to be able to make a comparison, and the resulting summary is illustrated figure 7.15. For all coding experiments, the anchor used for BD-rate computations is VTM-17 without screen content coding tools

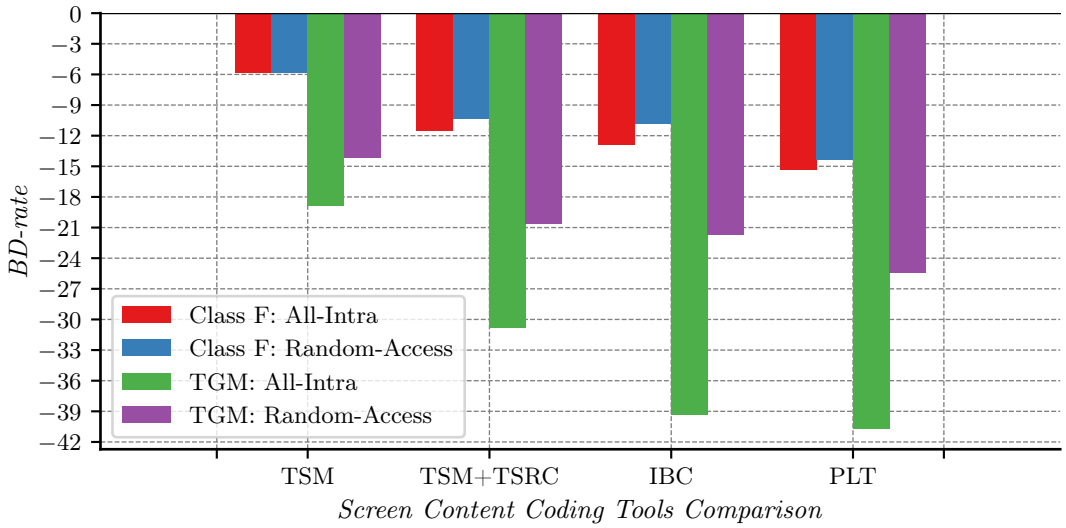

Figure 7.15
Coding efficiency provided by TSM (maximum allowed block size equal to 32×32 samples) with TSRC, IBC, and PLT.
Anchor for BD-rate computations: IMP7-0 without screen content coding tools

(IMP7-0) and the corresponding coding tool is enabled in the tested configuration. The BD-rates achieved by enabling TSM change from -5.82% to -11.54% (*All-Intra*) and from -5.86% to -10.30% (*Random-Access*) for test sequences of Class F when using TSRC. For the test sequences of TGM, the bit-rate savings are increased by about 12% in the *All-Intra* configuration and 6% in the *Random-Access* configuration. Note that TSM with TSRC achieves coding efficiencies close to that of IBC for test sequences of class F. For the test sequences of TGM, virtually the same coding efficiency as for IBC is achieved in the *Random-Access* configuration, while IBC provides a higher coding efficiency in the *All-Intra* configuration. Although the coding efficiency achieved by the combination TSM+TSRC is slightly less than that for IBC or PLT in a configuration using one screen content coding tool, the concept of TSM+TSRC is comparably simple. It requires only small modifications at the encoder side, which is particularly suitable for camera-captured content augmented by screen content overlays, as for some of the test sequences of class F.

Coding Efficiency of IBC and PLT with and without TSRC

Different coding experiments were performed to investigate the interference between TSRC and IBC and between TSRC and PLT. In figure 7.16, the coding efficiency provided by enabling IBC and PLT is reported separately in two different configurations. For the first and second tick marks in figure 7.16, TSM is enabled without the dedicated TSRC in both the anchor and the test candidate, which corresponds to IMP7-0. The configuration used for the BD-rates of the third and fourth tick marks in figure 7.16 has additionally TSRC enabled in both the anchor and the test candidate, which corresponds to IMP7-16. The BD-rates of these coding experiments indicate that the coding efficiency improvements achieved by IBC are higher with a dedicated TSRC, meaning that IBC benefits from TSRC, and both coding tools can be combined effectively, even though TSRC was developed with IBC disabled (compare the BD-rates denoted in the first and third tick marks). Since PLT uses a dedicated coding path, meaning TSRC is not active when a block utilizes PLT, improving the coding efficiency for TSM with TSRC reduces the observed improvement for PLT when TSRC is enabled (compare the BD-rates denoted in the second and the fourth tick marks). However, future research could address this incompatibility because the signaling in PLT is not entirely different from that of TSRC [118].

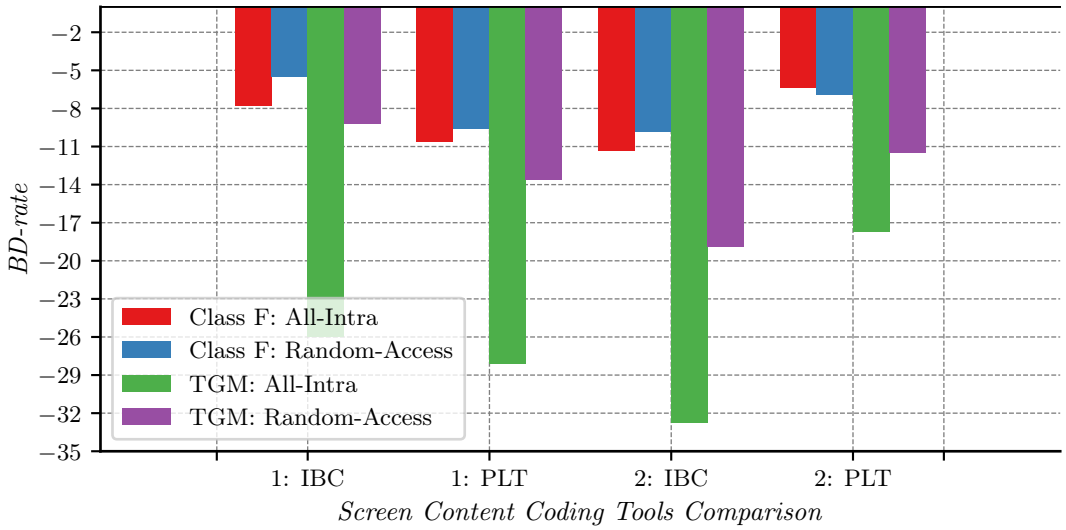

Figure 7.16
Coding efficiency provided IBC and PLT with and without TSRC enabled. The first and second tick marks denote the BD-rates for enabling IBC and PLT, where TSM without TSRC is enabled in both the anchor and the tested candidate. In the third and fourth tick marks, the BD-rates for enabling IBC and PLT are summarized, where TSM with TSRC is enabled in both the anchor and the tested candidate.
Anchor for BD-rate computations: 1: IMP7-0 with TSM enabled
Anchor for BD-rate computations: 2: IMP7-16 with TSM and TSRC enabled

Coding Efficiency of Individual Coding Tools

Figure 7.17 summarizes the coding efficiency achieved by individual screen content coding tools in a configuration with TSM+TSRC, IBC, and PLT enabled. The tested candidate is IMP7-16 with TSM, TSRC, IBC, and PLT enabled, whereas the anchor for the coding tool under investigation has the corresponding coding tool disabled. TSRC proves its effectiveness in this configuration by achieving slight coding efficiency improvement. The combination of TSM+TSRC, for which TSRC is mainly developed, achieves a coding efficiency improvement similar to that for PLT, while IBC achieves the highest improvement in this configuration. That IBC provides the highest coding efficiency in this investigation further supports the indication that TSRC is beneficial in combination with IBC, since it also improves the coding of the IBC residual signal.

Encoding and Decoding Times

A benefit of TSRC is that the encoding and decoding times in the VTM implementation are about 5% faster than an anchor using conventional residual coding for TSM levels. IBC requires about 200% encoding time for the test sequences of class F and 110% for test sequences of TGM, while the decoding time is similar to that of the anchor. These encoding times of IBC suggest that search efforts implemented in the reference encoder are less the more screen content can be found within the video sequence. For PLT, the encoding times are about 105% for test sequences of class F and 100% for the test sequences of TGM. Among the investigated screen content coding tools, the combination of TSM and TSRC has the smallest encoding overhead to achieve a high coding efficiency for screen content. For further improving the coding efficiency and for coding screen content video sequences, the combination of TSM+TSRC with IBC and PLT is suitable, because the required encoder search effort for IBC and PLT is significantly reduced when the video sequence purely consists of screen content.

7.3.8 | Binarization Without the b_{par} Flags

In the investigations on additional context-coded $b_{|x|>1+2n}$ flags in section 7.3.2, it was noted that the b_{par} flag is not required, because TCQ is not supported for transform skip blocks. Nevertheless, the binarization of the conventional level coding (with the parity flag b_{par}) was maintained to keep the syntax and semantics

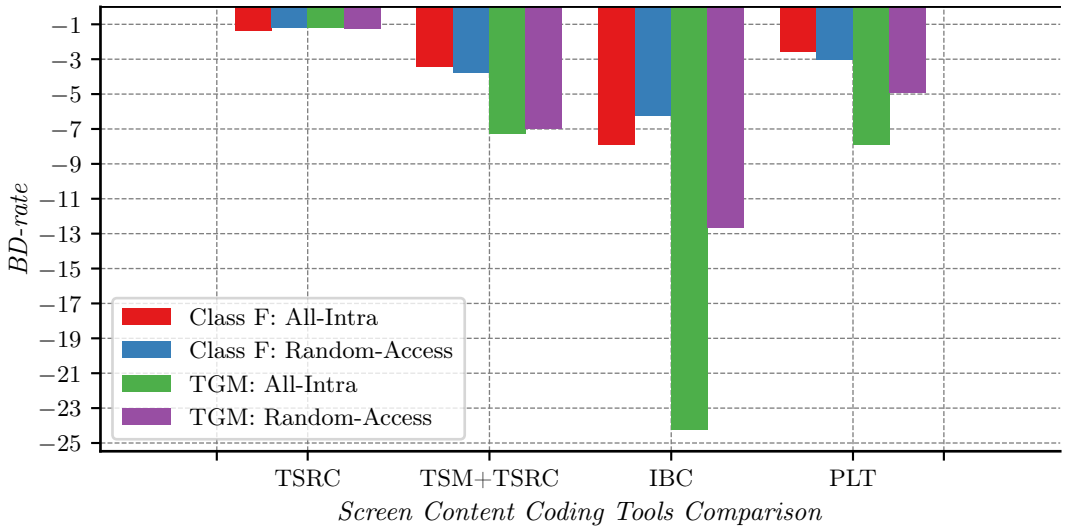

Figure 7.17
Coding efficiency provided by individual screen content coding tools. The tested candidate is IMP7-16 with TSM, TSRC, IBC, and PLT enabled, whereas the anchor is IMP7-16 configured with the coding tool under investigation disabled, but the other screen content coding tools are enabled.
Anchor for BD-rate computations: IMP7-16 with TSRC, TSM+TSRC, IBC, or PLT disabled

of TSRC aligned with that of the conventional level coding. The binarization without b_{par} is briefly studied in the following investigations.

Coding of Additional Context-Coded $b_{|x|>m}$ Flags

In the following investigation, the binarization without the b_{par} flag is implemented by removing the b_{par} flag, and the impact of the additional context-coded $b_{|x|>m}$ flags is analyzed. The corresponding implementation is referred to as IMP7-19 and it is based on IMP7-16. If no additional context-coded $b_{|x|>m}$ flags are coded, the first coding phase of IMP7-19 consists of the context-coded flags b_{sig}, b_{sign}, $b_{|x|>1}$, and $b_{|x|>2}$. Because the meanings of b_{sig}, $b_{|x|>1}$, and b_{sign} are the same as for the binarization with b_{par}, the corresponding context modeling is unmodified. Each context-coded flag transmitted after $b_{|x|>1}$ uses a dedicated context model, as used for the $b_{|x|>1+2n}$ flags in the binarization with b_{par}. Let N be the number of additional context-coded $b_{|x|>m}$ flags. Then, the derivation of $\mathcal{S}_c$ used for the template-based context modeling of b_{sig} is changed to

$$\mathcal{S}_c = \sum_{i=0}^{T-1} \min\left(|x\left(i\right)|, 3+N\right). \tag{7.7}$$

The coding efficiencies of IMP7-19 for different numbers of additional context-coded $b_{|x|>m}$ flags are summarized in figure 7.18. The anchor used for BD-rate computations is IMP7-16, i.e., the final design of TSRC using the binarization with b_{par}. Compared to the binarization with b_{par}, a similar coding efficiency is achieved by coding nine additional context-coded $b_{|x|>m}$ flags. For this configuration, the maximum level magnitude that can be signaled by the context-coded flags for a scanning position is equal to 12, which corresponds to the value for the binarization with b_{par} and four additional context-coded $b_{|x|>1+2n}$ flags. This relationship indicates that the coding efficiency is mainly impacted by the maximum level magnitude that can be signaled by the context-coded flags only, while the impact of the actual binarization (with or without a parity flag) is comparably small.

Comparison Between the Binarization With and Without b_{par}

The previous investigation showed that the cut-off value between the context-coded flags and bypass-coded remainder mainly impacts the coding efficiency, because the absolute TSM levels are not geometrically distributed. To directly compare the binarizations with and without the parity flag, the same coding experiment

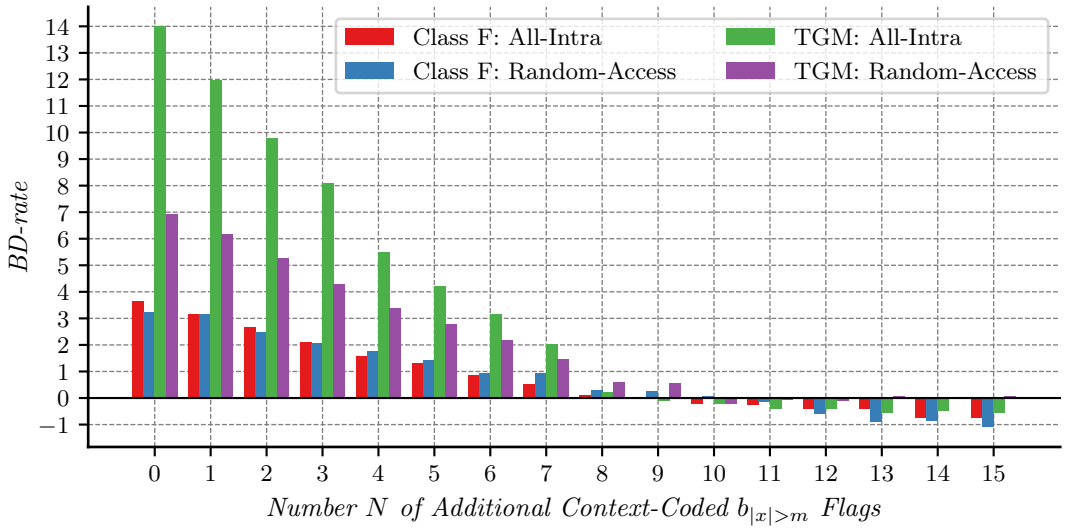

Figure 7.18
Coding efficiency of IMP7-19 implementing additional context-coded $b_{|x|>m}$ flags for a binarization without the b_{par} flag in TSRC.
Anchor for BD-rate computations: IMP7-16

as for the investigation using IMP7-5 was conducted, i.e., the impact of additional context-coded $b_{|x|>1+2n}$ flags was analyzed, but with the final TSRC design implemented in IMP7-16. This implementation is referred to as IMP7-20, and the coding efficiency in comparison to the binarization without b_{par} is summarized in figure 7.19. Each tick mark in the x-axis denotes the total number of context-coded flags, starting at six. For the binarization with b_{par}, the six context-coded flags are b_{sig}, b_{par}, b_{sign}, and three $b_{|x|>1+2n}$ flags, while for the binarization without b_{par}, the flags are b_{sig}, b_{sign}, and four $b_{|x|>m}$ flags. The anchor for BD-rate computations is IMP7-16, which consists of nine context-coded flags. From figure 7.19, it can be concluded that the binarization with b_{par} is more effective, because the coding efficiency is higher than for the binarization without b_{par} for the same number of context-coded flags.

Conclusion on Binarization Without b_{par}

Because the geometric distribution does not represent a suitable model for the distribution of the absolute TSM levels, the critical factor for the coding efficiency is the cut-off value between the context-coded flags and the bypass-coded remainder. Therefore, for achieving the same coding efficiency, the binarization without parity flag requires three to four more context-coded bins than the binarization with parity flag.

7.3.9 | Implementation of TSRC in VVC

The initial TSRC design integrated into VVC originated from the development presented in this chapter [116, 119]. In the TSRC finally specified for VVC, additional aspects contributed by other authors were included. The TSRC in VVC uses eight instead of nine context-coded bins, and the context-coded bins in VVC are coded in two coding phases, where the first coding phase includes the b_{sig}, b_{sign}, b_{par}, and $b_{|x|>1}$ flags and the second coding phase includes the $b_{|x|>3}$, $b_{|x|>5}$, $b_{|x|>7}$, and $b_{|x|>9}$ flags. These two aspects were not addressed explicitly in this thesis, because they mainly affect the interaction with the concept for reducing the maximum number of context-coded bins, which is not covered in this chapter. The basic concept used for limiting the maximum number of context-coded bins in TSRC is similar to the concept used for conventional level coding, which effectively limits the mcps to 1.75 [119, 120]. In addition, the template-based context modeling for b_{sig} evaluates whether the neighboring locations are zero-valued, as presented in the investigations in section 7.3.3 [116]. The context modeling of the $b_{|x|>1+2n}$ flags is a mixture of a single dedicated context model and the template-based approach. For the $b_{|x|>1}$ flag coded in the first coding phase, the template-based context modeling, as investigated in IMP7-11, is employed, while for the

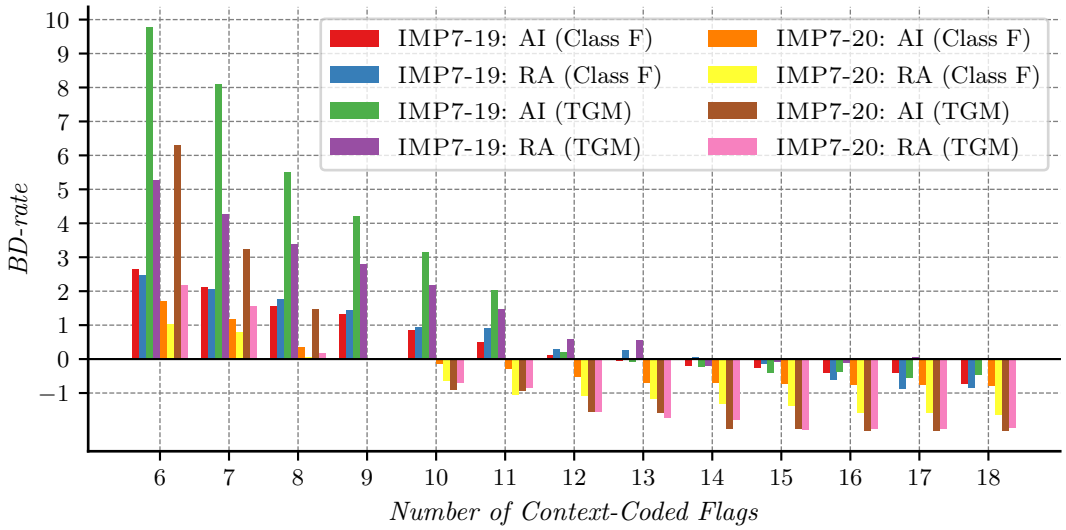

Figure 7.19
Impact of the total number of context-coded bins on the coding efficiency for a binarization with a parity flag (IMP7-20) and a binarization without the parity flag (IMP7-19).
Anchor for BD-rate computations: IMP7-16

$b_{|x|>1+2n}$ flags coded in the second coding phase, a single dedicated context model is used for each flag. The derivation of the context model offset for the template-based context modeling of b_{sign} was modified by comparing the signs of the neighboring locations instead of counting the number of negative signs [121]. Let $\mathrm{sgn}\,(x_l)$ denote the sign of the left neighboring location and $\mathrm{sgn}\,(x_t)$ that of the neighboring location above. Then, the context model offset δ_{sign} is derived according to

$$\delta_{sign} = \begin{cases} 0, & \text{if } (\mathrm{sgn}\,(x_l) = 0 \wedge \mathrm{sgn}\,(x_t) = 0) \vee (\mathrm{sgn}\,(x_l) \cdot \mathrm{sgn}\,(x_t) < 0)\,, \\ 1, & \text{if } (\mathrm{sgn}\,(x_l) \geq 0 \wedge \mathrm{sgn}\,(x_t) \geq 0) \wedge (\mathrm{sgn}\,(x_l) \neq 0 \vee \mathrm{sgn}\,(x_t) \neq 0)\,, \\ 2, & \text{otherwise.} \end{cases} \tag{7.8}$$

Finally, the binarization of the absolute TSM levels is altered by swapping the codewords for $|x| = 1$ and the maximum absolute value of the levels located above and to the left [121], similar to the concept implemented in the conventional level coding for completely bypass-coded levels [101]. Let $|x|_l$ denote the absolute TSM level of the left neighbor, and $|x|_t$ the absolute TSM level of the above neighbor. Furthermore, let $|x|_r$ denote the absolute TSM level reconstructed from the bitstream and $|x|_p = \max(|x|_l, |x|_t)$. Then, the final absolute TSM level $|x|$ is derived according to

$$|x| = \begin{cases} |x|_p, & \text{if } |x|_r = 1 \wedge |x|_p > 0, \\ |x|_r - 1, & \text{if } 1 < |x|_r < |x|_p, \\ |x|_r, & \text{otherwise.} \end{cases} \tag{7.9}$$

7.4 | Findings and Technical Achievements

The development of TSRC, a level coding designed for TSM levels presented in this chapter, is based on statistical analyses of TSM levels and subsequent findings, which can be summarized as follows:

- The signaling of the last significant scanning position, beneficial for coding transform coefficient levels, has no advantage for the coding of TSM levels.
- Scanning the TSM levels sub-block-wise from the top-left to the bottom-right corner of the block is more suitable for coding TSM levels than the sub-block-wise scanning from the bottom-right to the top-left corner of the block used in the conventional level coding.

- Coding additional context-coded $b_{|x|>1+2n}$ flags in combination with a dedicated context model set for each value of n improves the coding efficiency significantly, because the distribution of absolute TSM levels significantly deviates from a geometrical distribution.
- The concept of template-based context modeling of the conventional level coding (introduced in chapter 5) remains suitable for coding the b_{sig} flags in TSRC, but fewer neighboring locations and context models provide higher coding efficiency when coding TSM levels.
- A fixed Rice parameter of $k = 1$ can be used due to the introduction of additional context-coded $b_{|x|>1+2n}$ flags, which provides virtually the same coding efficiency as a template-based Rice parameter selection.
- The distributions of TSM levels are not symmetric around zero, which can be exploited by coding the b_{sign} flags with adaptive context models.

The presented TSRC impacts practical video coding applications, because it is a component of the VVC specification. The following list summarizes the key elements:

- VVC specifies a level coding for TSM levels that provides a similar coding efficiency as the variant presented in this chapter.
- The integration of a dedicated TSRC into VVC originated from the development presented in this chapter [116, 119].
- No last significant scanning position is coded in TSRC of VVC, as presented in this chapter.
- The sub-block-wise scanning from the top-left to the bottom-right corner of the block is specified for TSRC of VVC, as presented in this chapter.
- The template-based context modeling of b_{sig} together with the template configuration consisting of the top and the left neighboring locations, as presented in this chapter, is specified for TSRC of VVC.
- The coding of additional context-coded $b_{|x|>1+2n}$ flags, as developed and presented in this chapter, is also specified for TSRC of VVC.
- The coding of the b_{sign} flags in TSRC with a template-based context modeling, as presented in this chapter and later extended, can also be found in the specification of TSRC in VVC.

7.5 | Chapter Summary

This chapter started with a review of the coding efficiency that can be achieved with TSM and compared it to that of other dedicated screen content coding tools. In the first investigations presented in this chapter, different modifications to the conventional level coding were investigated, including the impact of coding the last significant scanning position, the effectiveness of the template-based context modeling and the Rice parameter selection, as well as reversing the scanning pattern, i.e., using the sub-block-wise scanning from the top-left to the bottom-right corner instead of sub-block-wise scanning from the bottom-right to the top-left corner. Further investigations were focused on context modeling and the coding of additional context-coded $b_{|x|>1+2n}$ flags. With an appropriate context modeling for the $b_{|x|>1+2n}$ flags, the coding of four additional context-coded $b_{|x|>1+2n}$ flags provides significant coding efficiency improvements. The following studies investigated the efficiency of the template-based context modeling for b_{sig} with the conclusion that the template-based context modeling achieves improved coding efficiency. This improvement in coding efficiency is higher for a template shape that consists of only two neighboring locations and requires fewer context models compared to the context modeling of the conventional level coding. However, the effect of the template-based context modeling on the $b_{|x|>1+2n}$ flags is only marginal. After increasing the number of context-coded $b_{|x|>1+2n}$ flags, a fixed Rice parameter of $k = 1$ turned out to provide virtually the same coding efficiency as a template-based variant. Finally, the coding of the b_{sign} flags with adaptive context models, based on the observation that the TSM levels are not symmetrically distributed around zero, further improves the coding efficiency of TSRC significantly. With the achieved coding efficiency improvement delivered by TSRC, the combination

of TSM and TSRC provides an attractive screen content coding tool, especially suitable for camera-captured video sequences with some screen content overlays. It was also shown that IBC benefits from the presented TSRC, because the IBC residuals can often be represented more efficiently by the combination of TSM and TSRC than by transform coding or transform skip combined with conventional level coding.

CHAPTER 8

Summary and Conclusion

This thesis presented various binarization and context modeling techniques for coding transform coefficient levels and directly quantized residual samples in video coding. The works are based on *context-based adaptive binary arithmetic coding* (CABAC), the entropy coding framework widely used in practical video coding standards, such as *Advanced Video Coding (H.264/MPEG-4 Part 10)* (AVC), *High Efficiency Video Coding (H.265/MPEG-H Part 2)* (HEVC), and *Versatile Video Coding (H.266/MPEG-I Part 3)* (VVC). In CABAC, which was reviewed in chapter 2, the estimation of conditional probabilities for entropy coding is separated from the binary arithmetic coding engine. Instead of deriving a conditional probability for coding a binary symbol (bin) directly, an adaptive context model, which implements an estimator for conditional binary probabilities, is selected. The set of rules defining which context model should be selected for a given bin is referred to as context modeling. For non-binary inputs, such as integer-valued transform coefficient levels, a mapping of the input symbols into a sequence of bins, which is referred to as binarization, is required for coding with a binary arithmetic coding engine. The developed context modeling and binarization techniques presented in this thesis focus on a higher coding efficiency or a complexity reduction for the level coding by analyzing statistical properties of the transform coefficient levels. The complexity can be reduced by limiting the computational complexity or the number of used adaptive context models, which is crucial for practical video coding applications.

The introduction of variable transform block sizes in HEVC required the development of a transform coefficient level coding that is suitable for multiple transform block sizes. A potential candidate was a generalized variant of the transform coefficient level coding specified for 4×4 and 8×8 transform blocks in AVC. However, for this approach, the number of context models scales linearly with the number of supported transform sizes. In order to overcome that problem, a level coding based on the coding of 4×4 sub-blocks was developed and presented in chapter 3. The main idea of this concept is based on the partitioning of transform blocks larger than 4×4 samples into 4×4 sub-blocks and the individual coding of each 4×4 sub-block. For the context modeling, a similar set of rules as for the 4×4 transform blocks in AVC is used for the coding of each 4×4 sub-block. Instead of using a dedicated context model set for each transform size, the context models are shared across different transform block sizes except for 4×4 transform blocks. This sharing of context models decouples the number of context models from the number of supported transform sizes. Further coding efficiency improvements were achieved by selecting a context model set for a 4×4 sub-block depending on the number of coded absolute levels greater than one in the preceding sub-block. The concept of sub-block-wise processing is established by its specification for the transform coefficient level coding in HEVC and VVC.

The binarization of absolute transform coefficient levels in AVC is a concatenation of the *truncated unary* (TRU) code followed by the 0^{th}*-order exponential-golomb* (EG0) code, where the maximum absolute value that can be represented by the TRU code is equal to 15. Because all bins of the TRU code are coded with adaptive context models, up to 15 context-coded flags are coded for each transform coefficient. In order to increase the throughput for hardware implementations, this number can be reduced by lowering the maximum value representable by the TRU code. A first investigation presented in chapter 4 showed that reducing that number results in coding efficiency penalties. Based on a study verifying that the geometric distribution is a suitable model for the remainders of absolute transform coefficient levels, Rice codes were introduced. Rice codes form a subset of Golomb codes, which are optimal scalar variable-length codes for geometrically distributed sources. Compared to Golomb codes, the Rice codes have the advantage that the construction of the codewords and the reconstruction of the values can be performed without multiplications. Rice codes replaced the EG0 code in the binarization, meaning that all bins of the Rice codes are coded in the low-complex bypass mode. Rice codes are parameterized by a so-called Rice parameter, which specifies the actual Rice code that is used for the binarization of an absolute transform coefficient level. For choosing the Rice parameter, a backward-adaptive concept was developed that selects the Rice parameter for a scanning position by evaluating the remainder coded in the preceding scanning position. The combination of Rice codes with a reduction of the maximum number of context-coded bins to three resulted in virtually the same coding

efficiency as provided by the binarization used in AVC for operation points targeting consumer applications. For high bit-rate operation points, the developed adaptive binarization with Rice codes significantly improves coding efficiency. However, using Rice codes only as a replacement for the EG0 code would significantly increase the maximum bin string length, which is impractical for hardware implementations. The solution was to reintroduce the EG0 code and place the EG0 code at the end of the binarization, meaning that the maximum value representable by Rice codes was limited. This combination of the TRU code followed by a Rice code and the EG0 code limits the maximum bin string length and provides a similar coding efficiency as provided by the binarization with TRU and Rice codes only. The developed and presented adaptive binarization of absolute transform coefficient levels with Rice codes achieved a similar or improved coding efficiency with a reduced implementation complexity. Given these benefits, both the HEVC and VVC standards specify adaptive binarization schemes based on the presented approach for the binarization of absolute transform coefficient levels.

A higher coding efficiency for entropy coding can be achieved by taking advantage of statistical dependencies between transform coefficient levels within a transform block. These statistical dependencies can be used to select a more appropriate conditional probability (or context) model. A straightforward way of exploiting statistical dependencies between neighboring transform coefficient levels is to evaluate the reconstructed level information inside a local template around the level to be coded. Different investigations on a suitable shape for the local template and the context modeling for coding the significance flags were presented first in chapter 5. Since the experimental results for this investigation showed the feasibility of a template-based context modeling, more conditionals for the context modeling were enabled by maximizing the level information inside the template. That is achieved when the neighboring frequency locations inside the local template consist of completely reconstructed levels. The initially employed level coding did not provide this property because the level magnitudes were gradually transmitted in multiple coding phases for a sub-block. These coding phases were merged into a single coding phase, which requires the instantaneous reconstruction of the level magnitude for each scanning position. A template-based context modeling for the remaining context-coded flags as well as a template-based Rice parameter selection were then developed. The adaptive context model set selection used for the 4×4 sub-blocks was replaced by position-dependent context model sets, where the selection of a context model set depends on the diagonal of the current scanning position. This context model set selection scheme further improves the coding efficiency of the template-based context modeling. With the template-based context modeling, the sharing of context model sets was extended to include 4×4 transform blocks. In the VVC standard, the level coding utilizes a template-based context modeling based on the presented development in this thesis.

VVC supports an optional vector quantizer, commonly referred to as *trellis-coded quantization* (TCQ), which improves coding efficiency. A brief review of TCQ was given at the beginning of chapter 6. The reconstruction process for TCQ employs a finite state machine, where the TCQ variant specified in VVC uses four states. It derives the state for a current scanning position from the state and the parity of the level at the preceding scanning position. Furthermore, the context model set for coding the significance flags depends on the current state when TCQ is enabled, where the first two and the last two states specify the usage of different context model sets. Because of the state-dependent context model sets, the level coding requires that the parities of preceding levels in coding order are available at the decoder. This aspect is the reason why the template-based context modeling with a single coding phase developed in chapter 5 is suitable for TCQ. A modification of the state-dependent context model sets for TCQ was developed after discovering that the probability distributions of the significance flag differ for the third and fourth states. In the second part of chapter 6, the throughput for practical implementations was increased by separating the coding of context-coded and bypass-coded bins. This separation increases the number of bypass-coded bins that can be transmitted successively. However, since the parity, which is required for the context modeling of the significance flag, becomes unavailable after the separation, a dedicated parity flag was introduced. Based on the results of different coding experiments, a binarization with the parity flag was chosen that achieves a suitable coding performance with and without TCQ enabled. Next, an adaptive scheme for reducing the maximum number of context-coded bins for transform coefficient levels was presented. This technique reduces the maximum number of context-coded bins for transform coefficient levels to 1.75 without negatively affecting the coding efficiency. In the final part of chapter 6, the usage of intermediate levels for context modeling to improve the

coding efficiency of TCQ was investigated. The investigations showed that employing intermediate levels, in combination with minor adjustments to the template sums, which maintain the basic design of the context modeling in VVC, can provide coding efficiency improvements.

The statistical properties of screen content differ from that of camera-captured content, requiring dedicated screen content coding tools to achieve reasonable coding efficiency. A straightforward technique that reuses the block-based transform coding architecture and improves coding efficiency is to skip the transform and quantize the residual samples directly. However, the statistical properties of the directly quantized residual samples of screen content are entirely different from that of transform coefficient levels. This difference in the statistical properties is exploited by developing a transform skip residual coding used for entropy coding the directly quantized residual samples in chapter 7. In the first part of chapter 7, different aspects regarding the design of such a level coding for transform skip residuals were investigated. These investigations resulted in an initial design without signaling the last significant scanning position, since the non-zero levels are typically spread across a block. Furthermore, the scanning was modified to proceed sub-block-wise from the top-left to the bottom-right corner instead of from the bottom-right to the top-left corner of the block. This change in the scanning direction improves the coding efficiency, because the level magnitudes of transform skip screen content blocks tend to increase in the forward scanning direction. A statistical analysis of the quantized residual samples for transform skip blocks was presented in the next step, showing that the geometric distribution cannot sufficiently model the distribution of absolute levels for transform skip blocks. Based on this finding, further investigations focused on the coding of additional context-coded flags and the associated context modeling. These additional context-coded flags specify whether the absolute level is greater than a specific threshold (greater flags). With an appropriate context modeling for these greater flags, the coding of four additional context-coded greater flags provides significant coding efficiency improvements. The template-based context modeling used in the conventional level coding was then optimized for the coding of transform skip levels to further improve the coding efficiency. This investigation revealed that a reduced local template consisting of only two neighboring locations achieves the best coding efficiency. For the Rice parameter selection, a fixed Rice parameter provided virtually the same coding efficiency as a template-based variant due to coding more context-coded greater flags, which reduces the number and the magnitudes of the coded remainders. It was further found that, in contrast to transform coefficient levels, transform skip levels are not symmetrically distributed around zero, meaning that the probability of a positive or negative level is not equal to $1/2$. Based on this observation, the sign information is coded with adaptive context models, which further improves the coding efficiency of the developed level coding for transform skip residuals.

The developed binarization and context modeling techniques presented in this thesis demonstrated that, compared to the state-of-the-art, more accurate estimations of conditional probabilities are possible and lead to coding efficiency improvements. Besides coding efficiency improvements or complexity reduction, the presented context modeling and binarization techniques also have a practical impact, because the level codings of HEVC and VVC are based on the developments for this thesis.

List of Implementations

The software implementations discussed in chapter 3 and chapter 4 are based on the HEVC reference software implementation, version 16.22 (HM-16.22). For the remaining chapters, the VVC reference software implementation, version 17.0 (VTM-17), serves as the basis for the implementations. The implementations often allow for the setting of parameters or thresholds to various values for experimental purposes. When an implementation is denoted by a star in the text, for example, IMP6-9*, it indicates that the specific implementation was used with a designated value for the corresponding parameter.

Chapter 3 – Transform Coefficient Level Coding for Variable Block Sizes

IMP3-0 Transform coefficient level coding of AVC generalized for larger transform blocks.

IMP3-1 First and second coding phases are interleaved on a sub-block granularity.

IMP3-2 Second coding phase starts after the first coding phase is completed for the entire transform block.

IMP3-3 Identical to IMP3-1, but with reinitialization of tracking variables.

IMP3-4 Identical to IMP3-2, but with reinitialization of tracking variables.

IMP3-5 Modified context quantizer for δ_{sig} and δ_{last}.

IMP3-6 Allows for the evaluation of different $n_{2\mathrm{DC}}$ and q_2 combinations.

IMP3-7 Allows for the evaluation of different $n_{1\mathrm{DC}}$ and q_1 combinations.

Chapter 4 – Adaptive Binarization of Transform Coefficient Levels

IMP4-0 Allows for the evaluation of different cut-off thresholds t_0.

IMP4-1 Allows for the evaluation of binarization with Rice code for $t_0 \in \{2, 3, 4\}$ for the remainders z.

IMP4-2 Allows for the evaluation of different combinations of z_1 and t_0 to switch to $k = 1$.

IMP4-3 Allows for the evaluation of different combinations of z_1 and t_0 to switch to $k = 2$.

IMP4-4 Allows for the evaluation of different combinations z_1 and t_0 to switch to $k = 3$.

IMP4-5 Fixed t_1 configurations for the nested Rice codes.

IMP4-6 Variable $t_1\,(k)$ configurations for the nested Rice codes.

Chapter 5 – Template-Based Context Modeling

IMP5-0 Implementing transform coefficient level coding developed in chapter 4 in VTM-17.

IMP5-1 Identical to IMP5-0, but without the adaptive context model sets.

IMP5-2 Single context model for coding b_{sig}, serving as the anchor for subsequent investigations.

IMP5-3 Evaluating the impact of using one fixed neighboring frequency location for coding b_{sig}.

IMP5-4 Employing a third context model when the neighboring location is unavailable.

IMP5-5 Evaluating template size, considering an unavailable location as zero-valued.

IMP5-6 Employing a dedicated context model for each template combination.

IMP5-7 Employing a different context model set when a neighboring location is unavailable.

IMP5-8 Using a single coding phase instead of two, without modifications to the context modeling.

IMP5-9 Clipping the sum of neighboring absolute levels inside the template for context modeling of b_{sig}.

IMP5-10 Clipping of the sum of neighboring absolute levels inside the template for coding $b_{|x|>1}$ and $b_{|x|>2}$.

IMP5-11 Alternative context modeling for coding $b_{|x|>1}$ and $b_{|x|>2}$.

IMP5-12 Using the same context model set for coding $b_{|x|>1}$ and $b_{|x|>2}$.

IMP5-13 Using a dedicated context model for coding $b_{|x|>1}$ and $b_{|x|>2}$ of the last significant scanning position.

IMP5-14 Evaluating different $M_{Rice}^{k=1}$ thresholds.

IMP5-15 Evaluating different $M_{Rice}^{k=2}$ thresholds.

IMP5-16 Evaluating different $M_{Rice}^{k=3}$ thresholds.

IMP5-17 Using a second context model set depending on the diagonal $D(x, y)$ for coding b_{sig}.

IMP5-18 Using a third context model set depending on the diagonal $D(x, y)$ for coding b_{sig}.

IMP5-19 Using a second context model set for coding $b_{|x|>1}$ and $b_{|x|>2}$ depending on the diagonal $D(x, y)$.

IMP5-20 Identical to IMP5-16, but using separate context models for 4×4 transform blocks.

IMP5-21 Identical to IMP5-18, but using separate context models for 4×4 transform blocks.

Chapter 6 – Level Coding Suitable for Trellis-Coded Quantization

IMP6-0 TCQ context modeling using two context model sets for coding b_{sig}.

IMP6-1 TCQ context modeling using three context model sets for coding b_{sig}.

IMP6-2 Evaluating the impact of clipping each neighboring absolute level inside the template to $|x|_{max}^{1}$.

IMP6-3 Alternative binarization #1 implementing the b_{par} flag.

IMP6-4 Alternative binarization #2 implementing the b_{par} flag.

IMP6-5 Alternative binarization #3 implementing the b_{par} flag.

IMP6-6 Evaluating different $M_{Rice}^{k=1}$ thresholds for binarization #1.

IMP6-7 Evaluating different $M_{Rice}^{k=1}$ thresholds for binarization #2.

IMP6-8 Evaluating different $M_{Rice}^{k=1}$ thresholds for binarization #3.

IMP6-9 Evaluating different $M_{Rice}^{k=2}$ thresholds for binarization #1.

IMP6-10 Evaluating different $M_{Rice}^{k=2}$ thresholds for binarization #2.

IMP6-11 Evaluating different $M_{Rice}^{k=2}$ thresholds for binarization #3.

Chapter 7 – Transform Skip Residual Coding

IMP7-0 Conventional level coding for transform skip mode.

IMP7-1 Alternative level coding for TSM without coding the last significant scanning position.

IMP7-2 Each context-coded flag employs a dedicated, single context model.

IMP7-3 The Rice parameter $k = 0$ is consistently employed.

IMP7-4 The scanning pattern is reversed.

IMP7-5 A single context model is used for all $b_{|x|>1+2n}$ flags.

IMP7-6 Each $b_{|x|>1+2n}$ flag employs a dedicated, single context model.

IMP7-7 Template as used in the conventional level coding for coding b_{sig}.

IMP7-8 Reduced template containing three neighboring locations for coding b_{sig}.

IMP7-9 Reduced template containing two neighboring locations for coding b_{sig}.

IMP7-10 Clipping M_{sig} to limit $\mathcal{S}_c$.

IMP7-11 $\mathfrak{S}_c$ is clipped to five and mapped to $\delta_{|x|>1+2n}$.

IMP7-12 $\mathfrak{S}_{|x|>1+2n}$ is used instead of $\mathfrak{S}_c$ to derive the context model offsets $\delta_{|x|>1+2n}$.

IMP7-13 Employing different fixed Rice parameters k.

IMP7-14 Template-based Rice parameter selection is utilized, using updated thresholds.

IMP7-15 Using a single context model for coding the b_{sign} flags.

IMP7-16 Template-based context modeling for coding the b_{sign} flags.

IMP7-17 Extended template-based context modeling for coding the b_{sign} flags.

IMP7-18 Dedicated context model for coding the b_{sign} flags when $|x| = 1$.

IMP7-19 Additional context-coded $b_{|x|>m}$ flags for a binarization without the b_{par} flag.

IMP7-20 Additional context-coded $b_{|x|>1+2n}$ flags flags for a binarization with the b_{par} flag.

List of Publications

Multiple aspects of the present thesis have been published in journal papers and conference proceedings. In addition, the techniques developed in the thesis have been proposed to the standardization bodies responsible for the specification of the video coding standards HEVC and VVC. These published papers and the standardization documents are summarized in the following. Besides papers and standardization documents closely related to the presented work on entropy coding, the list additionally includes published documents on other areas of video coding.

Journal Papers on the Topic of Level Coding

1. **T. Nguyen**, X. Xu, F. Henry, R.-L. Liao, M. G. Sarwer, M. Karczewicz, Y.-H. Chao, J. Xu, S. Liu, D. Marpe, and G. J. Sullivan. "Overview of the Screen Content Support in VVC: Applications, Coding Tools, and Performance". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.10 (2021), pp. 3801–3817. DOI: 10.1109/TCSVT.2021.3074312

2. J. Pfaff, H. Schwarz, D. Marpe, B. Bross, S. De-Luxán-Hernández, P. Helle, C. R. Helmrich, T. Hinz, W.-Q. Lim, J. Ma, **T. Nguyen**, J. Rasch, M. Schäfer, M. Siekmann, G. Venugopal, A. Wieckowski, M. Winken, and T. Wiegand. "Video Compression Using Generalized Binary Partitioning and Advanced Techniques for Prediction and Transform Coding". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.5 (2020), pp. 1281–1295. DOI: 10.1109/TCSVT.2019.2945918

3. **T. Nguyen**, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand. "Transform Coding Techniques in HEVC". in: *Journal of Selected Topics in Signal Processing* 7.6 (2013), pp. 978–989. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2013.2278071. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6578061

4. D. Marpe, H. Schwarz, S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, **T. Nguyen**, S. Oudin, M. Siekmann, K. Sühring, M. Winken, and T. Wiegand. "Video Compression Using Nested Quadtree Structures, Leaf Merging, and Improved Techniques for Motion Representation and Entropy Coding". In: *Transaction on Circuits and Systems for Video Technology* 20.12 (2010), pp. 1676–1687. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2010.2092615. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5638132

Conference Papers on the Topic of Level Coding

1. **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand. "Residual Coding for Transform Skip Mode in Versatile Video Coding". In: *2020 Data Compression Conference (DCC)*. 2020, pp. 83–92. DOI: 10.1109/DCC47342.2020.00016

2. **T. Nguyen**, B. Bross, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand. "Extended Transform Skip Mode and Fast Multiple Transform Set Selection in VVC". in: *2019 Picture Coding Symposium (PCS)*. 2019, pp. 1–5. DOI: 10.1109/PCS48520.2019.8954540

3. B. Bross, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. "Transform Skip Residual Coding for the Versatile Video Coding Standard". In: *Proc. SPIE*. vol. 11137. 2019

4. H. Schwarz, **T. Nguyen**, D. Marpe, T. Wiegand, M. Karczewicz, M. Coban, and J. Dong. "Improved Quantization and Transform Coefficient Coding for the Emerging Versatile Video Coding (VVC) Standard". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 1183–1187. DOI: 10.1109/ICIP.2019.8803768

5. H. Schwarz, **T. Nguyen**, D. Marpe, and T. Wiegand. "Hybrid Video Coding with Trellis-Coded Quantization". In: *2019 Data Compression Conference (DCC)*. 2019, pp. 182–191. DOI: 10.1109/DCC.2019.00026

6. B. Bross, P. Helle, S. Oudin, **T. Nguyen**, D. Marpe, H. Schwarz, and T. Wiegand. "Quadtree Structures and Improved Techniques for Motion Representation and Entropy Coding in HEVC". in: *Proc. IEEE International Conference on Consumer Electronics*. 2012

7. **T. Nguyen**, P. Helle, M. Winken, D. Marpe, H. Schwarz, and T. Wiegand. "Entropy Coding of Syntax

Elements Related to Block Structures and Transform Coefficient Levels in HEVC". in: *Proc. SPIE*. vol. 8499. 2012

8. **T. Nguyen**, D. Marpe, H. Schwarz, and T. Wiegand. "Reduced-Complexity Entropy Coding of Transform Coefficient Levels Using Truncated Golomb-Rice Codes in Video Compression". In: *2011 18th IEEE International Conference on Image Processing*. 2011, pp. 753–756. DOI: 10.1109/ICIP.2011.6116664

9. D. Marpe, H. Schwarz, T. Wiegand, S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, **T. Nguyen**, S. Oudin, M. Siekmann, K. Sühring, and M. Winken. "Improved Video Compression Technology and the Emerging High Efficiency Video Coding (HEVC) Standard". In: *Proc. IEEE International Conference on Consumer Electronics*. 2011

10. M. Winken, D. Marpe, H. Schwarz, T. Wiegand, S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, **T. Nguyen**, S. Oudin, M. Siekmann, and K. Sühring. "Highly Efficient Video Coding Based on Quadtree Structures, Improved Motion Compensation, and Probability Interval Partitioning Entropy Coding". In: *Proc. 14th ITG Conference on Electronic Media Technology*. 2011

11. **T. Nguyen**, H. Schwarz, H. Kirchhoffer, D. Marpe, and T. Wiegand. "Improved Context Modeling for Coding Quantized Transform Coefficients in Video Compression". In: *28th Picture Coding Symposium*. 2010, pp. 378–381. DOI: 10.1109/PCS.2010.5702513

12. H. Kirchhoffer, D. Marpe, H. Schwarz, T. Wiegand, S. Boße, B. Bross, P. Helle, T. Hinz, H. Lakshman, **T. Nguyen**, S. Oudin, M. Siekmann, K. Sühring, and M. Winken. "Next-Generation Video Coding Technology Using Quadtree-Based Partitioning and Improved Techniques for Motion Compensation and Entropy Coding". In: *Proc. 13. Symposium Maritime Elektrotechnik*. 2010

13. D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, **T. Nguyen**, S. Oudin, M. Siekmann, K. Sühring, M. Winken, and T. Wiegand. "Highly Efficient Video Compression Using Quadtree Structures and Improved Techniques for Motion Representation and Entropy Coding". In: *28th Picture Coding Symposium*. 2010, pp. 206–209. DOI: 10.1109/PCS.2010.5702464

Contributions to Standardizations on the Topic of Level Coding

1. H. Schwarz, P. Haase, **T. Nguyen**, J. Pfaff, D. Marpe, and T. Wiegand. *EE2-4.1: Results for Dependent Quantization With 8 States*. JVET-V0082. Web Conference: ITU-T and ISO/IEC, Apr. 2021

2. H. Schwarz, S. Schmidt, P. Haase, **T. Nguyen**, D. Marpe, and T. Wiegand. *Additional Support of Dependent Quantization with 8 States*. JVET-Q0243. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020

3. **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand. *CE3-Related: Modified Transform Skip Residual Coding for Lossless Coding*. JVET-Q0462. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020

4. B. Bross, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *AHG18: Two Stage Residual Coding for Lossless*. JVET-P0607. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2019

5. **T. Nguyen** and B Bross. *Performance of CE7-1.3b* in Lossless Mode*. JVET-P1025. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2019

6. **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand. *CE7-3.3/4: Context Modelling of Sign for TS Residual Coding*. JVET-O0089. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019

7. G. Clare, F. Henry, B. Bross, **T. Nguyen**, P. Keydel, H. Schwarz, D. Marpe, T. Wiegand, X. Zhao, X. Li, X. Xu, and S. Liu. *CE8: BDPCM with Harmonized Residual Coding and CCB Limitation (CE8-3.1a, CE8-3.1b, CE8-5.1a, CE8-5.1b)*. JVET-N0214. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

8. B. Bross, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *CE8: Residual Coding for Transform Skip Mode (CE8-4.3a, CE8-4.3b, CE8-4.4a, and CE8-4.4b)*. JVET-N0280. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

9. B. Bross, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *CE8-Related: Context Modelling of Sign for TS Residual Coding*. JVET-N0357. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

10. B. Bross, **T. Nguyen**, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE8: Unified Transform Type Signalling and Residual Coding for Transform Skip*. JVET-M0464. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019

11. H. Schwarz, **T. Nguyen**, D. Marpe, T. Wiegand, M. Karczewicz, M. Coban, and J. Dong. *CE7: Transform Coefficient Coding with Reduced Number of Regular-Coded Bins (Tests 7.1.3a, 7.1.3b)*. JVET-L0274. Macao, China: ITU-T and ISO/IEC, Oct. 2018

12. H. Schwarz, **T. Nguyen**, D. Marpe, and T. Wiegand. *CE7: Transform Coefficient Coding and Dependent Quantization (Tests 7.1.2, 7.2.1)*. JVET-K0071. Ljubljana, Slovenia: ITU-T and ISO/IEC, July 2018

13. H. Schwarz, **T. Nguyen**, D. Marpe, and T. Wiegand. *Non-CE7: Alternative Entropy Coding for Dependent Quantization*. JVET-K0072. Ljubljana, Slovenia: ITU-T and ISO/IEC, July 2018

14. M. Albrecht, C. Bartnik, S. Boße, J. Brandenburg, B. Bross, J. Erfurt, V. George, P. Haase, P. Helle, C. Helmrich, A. Henkel, T. Hinz, S. de Luxan Hernandez, S. Kaltenstadler, P. Keydel, H. Kirchhoffer, C. Lehmann, W.-Q. Lim, J. Ma, D. Maniry, D. Marpe, P. Merkle, **T. Nguyen**, J. Pfaff, J. Rasch, R. Rischke, C. Rudat, M. Schäfer, T. Schierl, H. Schwarz, M. Siekmann, R. Skupin, B. Stallenberger, J. Stegemann, K. Sühring, G. Tech, G. Venugopal, S. Walter, A. Wieckowski, T. Wiegand, and M. Winken. *Description of SDR, HDR and 360° Video Coding Technology Proposal by Fraunhofer HHI*. JVET-J0014. San Diego, USA: ITU-T and ISO/IEC, Apr. 2018

15. H. Kirchhoffer, M. Siekmann, C. Bartnik, D. Marpe, **T. Nguyen**, and T. Wiegand. *AHG5: Unified Coefficient Scan for JCTVC-H0228*. JCTVC-I0397. Geneva, Switzerland: ITU-T and ISO/IEC, Apr. 2012

16. **T. Nguyen**, H. Kirchhoffer, C. Bartnik, and D. Marpe. *CE3: Report for CE3 Subtest 3*. JCTVC-I0406. Geneva, Switzerland: ITU-T and ISO/IEC, Apr. 2012

17. **T. Nguyen**, D. Marpe, and T. Wiegand. *Non-CE11: Proposed Cleanup for Transform Coefficient Coding*. JCTVC-H0228. San Jose, USA: ITU-T and ISO/IEC, Jan. 2012

18. **T. Nguyen**, D. Marpe, M. Siekmann, and T. Wiegand. *Non-CE1: High Throughput Coding Scheme with Rice Binarization*. JCTVC-H0458. San Jose, USA: ITU-T and ISO/IEC, Jan. 2012

19. **T. Nguyen**. *CE11: Coding of Transform Coefficient Levels With Golomb-Rice Codes*. JCTVC-E253. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2011

20. **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *CE11: Evaluation of Transform Coding tools in HE Configuration*. JCTVC-D061. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011

21. **T. Nguyen**, M. Winken, D. Marpe, S. Heiko, and T. Wiegand. *Reduced-Complexity Entropy Coding of Transform Coefficient Levels Using a Combination of VLC and PIPE*. JCTVC-D336. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011

Journal Papers on the Topic of Video Coding

1. **T. Nguyen** and D. Marpe. "Compression Efficiency Analysis of AV1, VVC, and HEVC for Random Access Applications". In: *APSIPA Transactions on Signal and Information Processing* 10 (2021), e11. DOI: 10.1017/ATSIP.2021.10

2. S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinč, D. Neumann, **T. Nguyen**, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek. "DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks". In: *IEEE Journal of Selected Topics in Signal Processing* 14.4 (2020), pp. 700–714. DOI: 10.1109/JSTSP.2020.2969554

3. W.-S. Kim, W. Pu, A. Khairat, M. Siekmann, J. Sole, J. Chen, M. Karczewicz, **T. Nguyen**, and D. Marpe. "Cross-Component Prediction in HEVC". in: *IEEE Transactions on Circuits and Systems for Video Technology* 30.6 (2020), pp. 1699–1708. DOI: 10.1109/TCSVT.2015.2496821

4. M. Siekmann, A. Khairat, **T. Nguyen**, D. Marpe, and T. Wiegand. "Extended Cross-Component Prediction in HEVC". in: *APSIPA Transactions on Signal and Information Processing* 6 (2017), e3. DOI: 10.1017/ATSIP.2017.3
5. D. Flynn, D. Marpe, M. Naccari, **T. Nguyen**, C. Rosewarne, K. Sharman, J. Sole, and J. Xu. "Overview of the Range Extensions for the HEVC Standard: Tools, Profiles, and Performance". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (2016), pp. 4–19. DOI: 10.1109/TCSVT.2015.2478707
6. **T. Nguyen** and D. Marpe. "Objective Performance Evaluation of the HEVC Main Still Picture Profile". In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.5 (2015), pp. 790–797. DOI: 10.1109/TCSVT.2014.2358000

Conference Papers on the Topic of Video Coding

1. C. Rudat, C. R. Helmrich, J. Lainema, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. "Inter-Component Transform for Color Video Coding". In: *2019 Picture Coding Symposium (PCS)*. 2019, pp. 1–5. DOI: 10.1109/PCS48520.2019.8954496
2. S. De-Luxán-Hernández, V. George, J. Ma, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. "An Intra Subpartition Coding Mode for VVC". in: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 1203–1207. DOI: 10.1109/ICIP.2019.8803777
3. M. Schäfer, J. Pfaff, J. Rasch, T. Hinz, H. Schwarz, **T. Nguyen**, G. Tech, D. Marpe, and T. Wiegand. "Improved Prediction via Thresholding Transform Coefficients". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 2546–2549. DOI: 10.1109/ICIP.2018.8451076
4. **T. Nguyen** and D. Marpe. "Future Video Coding Technologies: A Performance Evaluation of AV1, JEM, VP9, and HM". in: *2018 Picture Coding Symposium (PCS)*. 2018, pp. 31–35. DOI: 10.1109/PCS.2018.8456289
5. D. Grois, **T. Nguyen**, and D. Marpe. "Performance Comparison of AV1, JEM, VP9, and HEVC Encoders". In: *Proc. SPIE*. vol. 10396. 2017
6. D. Grois, **T. Nguyen**, and D. Marpe. "Coding Efficiency Comparison of AV1/VP9, H.265/MPEG-HEVC, and H.264/MPEG-AVC Encoders". In: *2016 Picture Coding Symposium (PCS)*. 2016, pp. 1–5. DOI: 10.1109/PCS.2016.7906321
7. **T. Nguyen**, A. Khairat, D. Marpe, M. Siekmann, and T. Wiegand. "Extended Cross-Component Prediction in HEVC". in: *2015 Picture Coding Symposium (PCS)*. 2015, pp. 164–168. DOI: 10.1109/PCS.2015.7170068
8. A. Khairat, **T. Nguyen**, M. Siekmann, D. Marpe, and T. Wiegand. "Adaptive Cross-Component Prediction for 4:4:4 High Efficiency Video Coding". In: *2014 IEEE International Conference on Image Processing (ICIP)*. 2014, pp. 3734–3738. DOI: 10.1109/ICIP.2014.7025758
9. D. Grois, **T. Nguyen**, D. Marpe, and O. Hadar. "Comparative Assessment of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders for Low-Delay Video Applications". In: *Proc. SPIE*. vol. 9217. 2014
10. M. Preiß, D. Marpe, B. Bross, V. George, H. Kirchhoffer, **T. Nguyen**, M. Siekmann, J. Stegemann, and T. Wiegand. "A Unified and Complexity Scalable Entropy Coding Scheme for Video Compression". In: *2012 19th IEEE International Conference on Image Processing*. 2012, pp. 729–732. DOI: 10.1109/ICIP.2012.6466963
11. **T. Nguyen**, D. Marpe, B. Bross, V. George, H. Kirchhoffer, M. Preiß, M. Siekmann, J. Stegemann, and T. Wiegand. "A Complexity Scalable Entropy Coding Scheme for Video Compression". In: *Picture Coding Symposium*. IEEE, May 2012, pp. 421–424. ISBN: 978-1-4577-2049-9. DOI: 10.1109/PCS.2012.6213377. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6213377
12. **T. Nguyen** and D. Marpe. "Performance Analysis of HEVC-Based Intra Coding for Still Image Compression". In: *2012 Picture Coding Symposium*. 2012, pp. 233–236. DOI: 10.1109/PCS.2012.6213335

Contributions to Standardizations on the Topic of Video Coding

1. H. Kirchhoffer, B. Bross, **T. Nguyen**, D. Marpe, H. Schwarz, and T. Wiegand. *AHG14: Throughput and Coding Efficiency Report of JVET-P0300 on VTM-7.0.* JVET-Q0387. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020
2. S. De-Luxán-Hernández, **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand. *MTS Dependent Coefficient Subblock Scanning for Zero-Out.* JVET-Q0448. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020
3. **T. Nguyen**, T.-C. Ma, and A. Nalci. *Suggested Common Test Conditions and Reference Software Configurations for Lossless and Mixed Lossy/Lossless Applications.* JVET-Q0824. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020
4. T.-C. Ma, A. Nalci, and **T. Nguyen**. *JVET Common Test Conditions and Software Reference Configurations for Lossless, Near Lossless, and Mixed Lossy/Lossless Coding.* JVET-Q2014. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020
5. H. Kirchhoffer, B. Bross, **T. Nguyen**, D. Marpe, H. Schwarz, and T. Wiegand. *QP-Independent and Slice Type-Independent Initialization of Context Models for the High Throughput CABAC Mode of JVET-P0300.* JVET-Q0461. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020
6. H. Kirchhoffer, D. Marpe, B. Bross, **T. Nguyen**, C. Rudat, H. Schwarz, and T. Wiegand. *High Throughput CABAC Mode for VVC.* JVET-P0300. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2019
7. B. Bross, **T. Nguyen**, H. Schwarz, D. Marpe, T. Wiegand, M. Karczewicz, Y.-H. Chao, H. Wang, and M. Coban. *AHG18: Enabling Lossless Coding With Minimal Impact on VVC Design.* JVET-P0606. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2019
8. **T. Nguyen**. *AHG18: Non-Lossless Coding Tools in VVC.* JVET-P0612. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2019
9. S. De-Luxán-Hernández, **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, T. Wiegand, T.-C. Ma, X.-Y. Xiu, Y.-W. Chen, and X. Wang. *CE8-3.1: Enable Transform Skip in CUs using ISP.* JVET-O0097. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019
10. C. Helmrich, H. Schwarz, **T. Nguyen**, C. Rudat, D. Marpe, T. Wiegand, B. Ray, G. Van der Auwera, A. K. Ramasubramonian, M. Coban, and M. Karczewicz. *CE7: Joint Chroma Residual Coding With Multiple Modes (Tests CE7-2.1, CE7-2.2).* JVET-O0105. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019
11. A. K. Ramasubramonian, G. Van der Auwera, T. Hsieh, V. Seregin, L. Pham Van, M. Karczewicz, S. De-Luxán-Hernández, B. Bross, **T. Nguyen**, V. George, B. Stabernack, H. Schwarz, D. Marpe, and T. Wiegand. *CE3-1.6: On 1xN and 2xN Subblocks of ISP.* JVET-O0106. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019
12. **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE8: Minimum Allowed QP for Transform Skip Mode.* JVET-O0405. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019
13. S. De-Luxán-Hernández, V. George, G. Venugopal, J. Brandenburg, B. Bross, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE3: Proposed ISP Cleanup.* JVET-O0502. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019
14. C. Helmrich, H. Schwarz, **T. Nguyen**, C. Rudat, D. Marpe, and T. Wiegand. *CE7-Related: Alternative Configuration for Joint Chroma Residual Coding.* JVET-O0543. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019
15. C. Helmrich, H. Schwarz, **T. Nguyen**, C. Rudat, D. Marpe, T. Wiegand, B. Ray, G. Van der Auwera, A. K. Ramasubramonian, M. Coban, and M. Karczewicz. *CE7-Related: Alternative Configuration of CE7-2.2 Joint Chroma Residual Coding.* JVET-O0935. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019

16. C. Helmrich, C. Rudat, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *CE7-Related: Joint Chroma Residual Coding With Multiple Modes.* JVET-N0282. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

17. S. De-Luxán-Hernández, B. Bross, **T. Nguyen**, V. George, B. Stabernack, H. Schwarz, D. Marpe, and T. Wiegand. *AHG16/Non-CE3: Restriction of the Maximum CU Size for ISP to 64x64.* JVET-N0308. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

18. S. De-Luxán-Hernández, B. Bross, **T. Nguyen**, V. George, B. Stabernack, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE3: ISP With Independent Sub-Partitions for Certain Block Sizes).* JVET-N0372. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

19. S. De-Luxán-Hernández, **T. Nguyen**, B. Bross, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE3/Non-CE8: Enable Transform Skip in CUs using ISP.* JVET-N0401. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

20. A. Wieckowski, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *Availability Based Context Modelling for mtt_split_cu_vertical_flag.* JVET-N0696. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019

21. S. De-Luxán-Hernández, V. George, J. Ma, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *CE3: Intra Sub-Partitions Coding Mode (Tests 1.1.1 and 1.1.2).* JVET-M0102. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019

22. A. Wieckowski, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE1: Split-First Signalling for Partitioning.* JVET-M0421. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019

23. S. De-Luxán-Hernández, V. George, J. Ma, **T. Nguyen**, H. Schwarz, D. Marpe, and T. Wiegand. *CE3-Related: Improvement on the Intra Sub-Partitions Coding Mode.* JVET-M0426. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019

24. H. Schwarz and **T. Nguyen**. *CE7-Related: Analysis of Padding Bytes for VTM-2.* JVET-L0276. Macao, China: ITU-T and ISO/IEC, Oct. 2018

25. A. Wieckowski, T. Hinz, B. Bross, **T. Nguyen**, J. Ma, K. Sühring, H. Schwarz, D. Marpe, and T. Wiegand. *NextSoftware as Test Software.* JVET-J0095. San Diego, USA: ITU-T and ISO/IEC, Apr. 2018

26. A. Wieckowski, T. Hinz, B. Bross, **T. Nguyen**, J. Ma, K. Sühring, H. Schwarz, D. Marpe, and T. Wiegand. *Benchmark Set Results.* JVET-J0100. San Diego, USA: ITU-T and ISO/IEC, Apr. 2018

27. K. Sühring and **T. Nguyen**. *Bit Rate Measurement in CTC.* JCTVC-Z0038. Geneva, Switzerland: ITU-T and ISO/IEC, Jan. 2017

28. A. Khairat, **T. Nguyen**, and D. Marpe. *AHG5 and AHG8: Extended Cross-Component Decorrelation for Animated Screen Content.* JCTVC-P0097. San Jose, USA: ITU-T and ISO/IEC, Jan. 2014

29. W.-S. Kim and **T. Nguyen**. *RCE1: Summary Report of HEVC Range Extensions Core Experiment 1 on Inter-Component Decorrelation Methods.* JCTVC-O0035. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2013

30. A. Khairat, **T. Nguyen**, M. Siekmann, and D. Marpe. *Non-RCE1: Extended Adaptive Inter-Component Prediction.* JCTVC-O0150. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2013

31. **T. Nguyen**. *RCE1: Description and Results for Experiment 3, 5, and 6.* JCTVC-N0170. Vienna, Austria: ITU-T and ISO/IEC, July 2013

32. **T. Nguyen**, A. Khairat, and D. Marpe. *Non-RCE1/Non-RCE2/AHG5/AHG8: Adaptive Inter-Plane Prediction for RGB Content.* JCTVC-M0230. Incheon, South Korea: ITU-T and ISO/IEC, Apr. 2013

33. **T. Nguyen** and D. Marpe. *Performance Comparison of HM 6.0 With Existing Still Image Compression Schemes Using a Test Set of Popular Still Images.* JCTVC-I0595. Geneva, Switzerland: ITU-T and ISO/IEC, Apr. 2012

34. H. Kirchhoffer, B. Bross, P. Helle, D. Marpe, **T. Nguyen**, M. Siekmann, J. Stegemann, and T. Wiegand. *CE1: Report of Test Results Related to Subtests B2 (8-bit-init) and C3 (Alt. PMU).* JCTVC-H0266. San Jose, USA: ITU-T and ISO/IEC, Jan. 2012

35. **T. Nguyen**, D. Marpe, H. Schwarz, and T. Wiegand. *Modified Binarization and Coding of MVD for PIPE/CABAC).* JCTVC-F455. Torino, Italy: ITU-T and ISO/IEC, July 2011

36. H. Kirchhoffer, B. Bross, A. Henkel, D. Marpe, **T. Nguyen**, M. Preiß, M. Siekmann, J. Stegemann, and T. Wiegand. *CE1: Report of Test Results Related to PIPE-based Unified Entropy Coding.* JCTVC-G633. Geneva, Switzerland: ITU-T and ISO/IEC, Nov. 2011

37. D. Marpe, H. Kirchhoffer, B. Bross, V. George, **T. Nguyen**, M. Preiß, M. Siekmann, J. Stegemann, and T. Wiegand. *Unified PIPE-Based Entropy Coding for HEVC.* JCTVC-F268. Torino, Italy: ITU-T and ISO/IEC, July 2011

38. **T. Nguyen** and K. Sühring. *Report on Bugfix for CABAC Sub-Block Coding Reported in JCTVC-D027.* JCTVC-D451. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011

39. **T. Nguyen**. *Improved Intra Smoothing for UDI and new AIS Fast Mode.* JCTVC-C302. Guangzhou, China: ITU-T and ISO/IEC, Oct. 2010

List of Figures

List of Tables

Acronyms

AVC
Advanced Video Coding (H.264/MPEG-4 Part 10) 1, 20, 41, 65, 83, 107, 130

BD-rate
Bjøntegaard delta bit-rate 25, 44, 67, 88, 110

CABAC
context-based adaptive binary arithmetic coding 2, 13, 20, 41, 98, 130

CAVLC
context-based adaptive variable length coding 4

cps
context-coded bins per sample 43, 61, 90

CTC
common test conditions 25, 44, 67, 110

DCT
discrete cosine transform 3, 13, 108

DST
discrete sine transform 13, 108

EBCOT
embedded block coding with optimised truncation 4

EG0
0^{th}*-order exponential-golomb* 9, 15, 21, 42, 62, 74, 130

EP
equi-probable 25, 67, 88, 110

EZW
embedded zerotree wavelet 4

FSM
finite-state machine 17, 85

HEVC
High Efficiency Video Coding (H.265/MPEG-H Part 2) 1, 11, 20, 41, 65, 83, 107, 130

IBC
intra block copy 107

JCT-VC
Joint Collaborative Team on Video Coding 8

JEM
Joint Exploration Model 81

JVET
Joint Video Experts Team 8, 79, 88

LSB
least-significant bit 4

Mbps
megabits per second 1

mcps
maximum number of context-coded bins per sample 90, 115

MPEG-2
H.262/MPEG-2 Part 2 3

MPEG-4 Visual
MPEG-4 Part 2 3

MSB
most-significant bit 4

PACC
partitioning, aggregation, and conditional coding 5

pdf
probability density function 45

PLT
palette mode 107

pmf
probability mass function 15, 42, 45

QP
quantization parameter 13, 25, 43, 83

RDOQ
rate-distortion optimized quantization 25, 45, 67, 83, 110

RExt
Range Extensions 7, 108

SDH
sign data hiding 25, 67, 110

SPIHT
set partitioning in hierarchical trees 4

TCQ
trellis-coded quantization 3, 67, 83, 110, 131

TGM
text and graphics with motion 110

TRU
truncated unary 15, 21, 42, 62, 99, 130

TSM
transform skip mode 25, 67, 107

TSRC
transform skip residual coding 107

URQ
uniform reconstruction quantization 13, 83

VVC
Versatile Video Coding (H.266/MPEG-I Part 3) 1, 11, 40, 41, 64, 84, 107, 130

Bibliography

[1] T. Wiegand and H. Schwarz. *Source Coding: Part I of Fundamentals of Source and Video Coding*. Vol. 4. 2010. 2011, pp. 1–222. DOI: 10.1561/2000000010.

[2] Statistica. *Average Internet Connection Speed (Fixed Network) in Germany from October 2020 to February 2023*. Mar. 2023. URL: https://www.statista.com/statistics/1338657/average-internet-speed-germany/.

[3] *Cisco Annual Internet Report (2018–2023) White Paper*. Cisco, Sept. 2020. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[4] V. K. Goyal. "Theoretical Foundations of Transform Coding". In: *IEEE Signal Processing Magazine* 18.5 (2001), pp. 9–21. DOI: 10.1109/79.952802.

[5] J. W. Cooley and J. W. Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series". In: *Mathematics of Computation* 19.90 (1965), pp. 297–301.

[6] N. Ahmed, T. Natarajan, and K. R. Rao. "Discrete Cosine Transform". In: *IEEE Transactions on Computers* 100.1 (1974), pp. 90–93.

[7] H. C. Andrews and W. K. Pratt. "Fourier Transform Coding of Images". In: *Hawaii International Conference on System Sciences*. 1968, pp. 671–619.

[8] W. K. Pratt, J. Kane, and H. C. Andrews. "Hadamard Transform Image Coding". In: *Proceedings of the IEEE* 57.1 (1969), pp. 58–68. DOI: 10.1109/PROC.1969.6869.

[9] A. H. Robinson and C. Cherry. "Results of a Prototype Television Bandwidth Compression Scheme". In: *Proceedings of the IEEE* 55.3 (1967), pp. 356–364. DOI: 10.1109/PROC.1967.5493.

[10] ISO/IEC. *10918-1: Digital Compression and Coding of Continuous-Tone Still Images: Requirements and Guidelines*. 1992. URL: https://www.iso.org/standard/18902.html.

[11] ITU-T. *Recommendation H.262*. 1995. URL: http://www.itu.int/rec/T-REC-H.262.

[12] ISO/IEC. *13818-2: Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video*. 2000. URL: https://www.iso.org/standard/61152.html.

[13] ITU-T. *Recommendation H.263*. 1998. URL: http://www.itu.int/rec/T-REC-H.263.

[14] ISO/IEC. *14496-2: Coding of Audio-Visual Objects - Part 2: Visual*. 2001. URL: https://www.iso.org/standard/61152.html.

[15] G. K. Wallace. "The JPEG Still Picture Compression Standard". In: *IEEE Transactions on Consumer Electronics* 38.1 (1992), pp. xviii–xxxiv. DOI: 10.1109/30.125072.

[16] P. N. Tudor. "MPEG-2 Video Compression Tutorial". In: *IEE Colloquium on MPEG-2 - What it is and What it isn't*. 1995, pp. 2/1–2/8. DOI: 10.1049/ic:19950036.

[17] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. "Overview of the H.264/AVC Video Coding Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (2003), pp. 560–576. DOI: 10.1109/TCSVT.2003.815165.

[18] I. Richardson. *H.264/AVC Context Adaptive Variable Length Coding*. VCodex. URL: https://www.vcodex.com/h264avc-context-adaptive-variable-length-coding/.

[19] S. G. Mallat. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 674–693. DOI: 10.1109/34.192463.

[20] J. M. Shapiro. "Embedded Image Coding Using Zerotrees of Wavelet Coefficients". In: *IEEE Transactions on Signal Processing* 41.12 (1993), pp. 3445–3462. DOI: 10.1109/78.258085.

[21] A. Graps. "An Introduction to Wavelets". In: *IEEE Computational Science and Engineering* 2.2 (1995), pp. 50–61. DOI: 10.1109/99.388960.

[22] A. Said and W. A. Pearlman. "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees". In: *IEEE Transactions on Circuits and Systems for Video Technology* 6.3 (1996), pp. 243–250. DOI: 10.1109/76.499834.

[23] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. "An Overview of JPEG-2000". In: *Proceedings DCC 2000. Data Compression Conference*. 2000, pp. 523–541. DOI: 10.1109/DCC.2000.838192.

[24] D. Taubman, E. Ordentlich, M. Weinberger, G. Seroussi, I. Ueno, and F. Ono. "Embedded Block Coding in JPEG2000". In: *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*. Vol. 2. 2000, 33–36 vol.2. DOI: 10.1109/ICIP.2000.899218.

[25] X. Wu and J.-H. Chen. "Context Modeling and Entropy Coding of Wavelet Coefficients for Image Compression". In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. 1997, pp. 3097–3100. DOI: 10.1109/ICASSP.1997.595447.

[26] D. Marpe and H. L. Cycon. "Very Low Bit-Rate Video Coding Using Wavelet-Based Techniques". In: *IEEE Transactions on Circuits and Systems for Video Technology* 9.1 (1999), pp. 85–94. DOI: 10.1109/76.744277.

[27] Sullivan, Gary. *Q.15/16 Meeting Report (Geneva, Jan.26 - Feb.6, 1998)*. 9801q15r. Geneva, Switzerland: ITU-T, Jan. 1998.

[28] D. Marpe, H. Schwarz, and T. Wiegand. "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard". In: *IEEE Transaction on Circuits and Systems for Video Technology* 13.7 (July 2003), pp. 620–636. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815173. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1218195.

[29] J. Sole, R. Joshi, I.-S. Chong, M. Coban, and M. Karczewicz. *Parallel Context Processing for the Significance Map in High Coding Efficiency*. JCTVC-D262. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011.

[30] N. Nguyen, T. Ji, D. He, G. Martin-Cocher, and L. Song. *Multi-Level Significance Maps for Large Transform Units*. JCTVC-G644. Geneva, Switzerland: ITU-T and ISO/IEC, Nov. 2011.

[31] Y. Piao, W. Choi, and K. Chanyul. *CE7.1.3: Scan Region-Based Coefficient Coding*. JVET-K0138. Ljubljana, Slovenia: ITU-T and ISO/IEC, July 2018.

[32] C. Auyeung, X. Zhao, X. Li, and S. Liu. *CE7-Related: Context Reduction of Transform Coefficient Significance Flag*. JVET-O0617. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[33] M. Gao, X. Fan, D. Zhao, and W. Gao. "An Enhanced Entropy Coding Scheme for HEVC". In: *Signal Processing: Image Communication* 44 (2016), pp. 108–123. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2016.03.008. URL: https://www.sciencedirect.com/science/article/pii/S0923596516300315.

[34] P. de Rivaz and J. Haughton. *AV1 Bitstream & Decoding Process Specification*. 2018. URL: https://aomediacodec.github.io/av1-spec/av1-spec.pdf.

[35] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, Y. Chen, Y. Wang, P. Wilkins, Y. Xu, and J. Bankoski. "A Technical Overview of AV1". In: *Proceedings of the IEEE* 109.9 (2021), pp. 1435–1462. DOI: 10.1109/JPROC.2021.3058584.

[36] J. Han, C.-H. Chiang, and Y. Xu. "A Level-Map Approach to Transform Coefficient Coding". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 3245–3249. DOI: 10.1109/ICIP.2017.8296882.

[37] T. Nguyen, H. Schwarz, H. Kirchhoffer, D. Marpe, and T. Wiegand. "Improved Context Modeling for Coding Quantized Transform Coefficients in Video Compression". In: *28th Picture Coding Symposium*. 2010, pp. 378–381. DOI: 10.1109/PCS.2010.5702513.

[38] T. Nguyen, D. Marpe, H. Schwarz, and T. Wiegand. "Reduced-Complexity Entropy Coding of Transform Coefficient Levels Using Truncated Golomb-Rice Codes in Video Compression". In: *2011 18th IEEE International Conference on Image Processing*. 2011, pp. 753–756. DOI: 10.1109/ICIP.2011.6116664.

[39] M. Winken, S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, D. Marpe, S. Oudin, M. Preiß, H. Schwarz, M. Siekmann, K. Sühring, and T. Wiegand. *Video Coding Technology Oroposal by Fraunhofer HHI*. JCTVC-A116. Dresden, Germany: ITU-T and ISO/IEC, Apr. 2010.

[40] ITU-T SG16/Q6 and ISO/IEC JTC1/SC29/WG11. *Joint Call for Proposals on Video Compression Technology*. VCEG-AM91 and N11113. Kyoto, Japan: ITU-T and ISO/IEC, Jan. 2010.

[41] JCT-VC. *Test Model under Consideration (TMuC)*. JCTVC-A205. Dresden, Germany: ITU-T and ISO/IEC, Apr. 2010.

[42] T. Nguyen, D. Marpe, and T. Wiegand. *Non-CE11: Proposed Cleanup for Transform Coefficient Coding*. JCTVC-H0228. San Jose, USA: ITU-T and ISO/IEC, Jan. 2012.

[43] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand. *CE7: Transform Coefficient Coding and Dependent Quantization (Tests 7.1.2, 7.2.1)*. JVET-K0071. Ljubljana, Slovenia: ITU-T and ISO/IEC, July 2018.

[44] H. Schwarz, T. Nguyen, D. Marpe, T. Wiegand, M. Karczewicz, M. Coban, and J. Dong. "Improved Quantization and Transform Coefficient Coding for the Emerging Versatile Video Coding (VVC) Standard". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 1183–1187. DOI: 10.1109/ICIP.2019.8803768.

[45] D. Flynn, D. Marpe, M. Naccari, T. Nguyen, C. Rosewarne, K. Sharman, J. Sole, and J. Xu. "Overview of the Range Extensions for the HEVC Standard: Tools, Profiles, and Performance". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (2016), pp. 4–19. DOI: 10.1109/TCSVT.2015.2478707.

[46] T. Nguyen, B. Bross, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand. "Extended Transform Skip Mode and Fast Multiple Transform Set Selection in VVC". In: *2019 Picture Coding Symposium (PCS)*. 2019, pp. 1–5. DOI: 10.1109/PCS48520.2019.8954540.

[47] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. "Overview of the High Efficiency Video Coding (HEVC) Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), pp. 1649–1668. DOI: 10.1109/TCSVT.2012.2221191.

[48] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm. "Overview of the Versatile Video Coding (VVC) Standard and its Applications". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[49] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang. "Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC)". In: *Proceedings of the IEEE* (2021), pp. 1–31. DOI: 10.1109/JPROC.2020.3043399.

[50] Y.-W. Huang, J. An, H. Huang, X. Li, S.-T. Hsiang, K. Zhang, H. Gao, J. Ma, and O. Chubach. "Block Partitioning Structure in the VVC Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021), pp. 1–1. DOI: 10.1109/TCSVT.2021.3088134.

[51] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur. "Intra Coding of the HEVC Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), pp. 1792–1801. DOI: 10.1109/TCSVT.2012.2221525.

[52] J. Pfaff, A. Filippov, S. Liu, X. Zhao, J. Chen, S. De-Luxán-Hernández, T. Wiegand, V. Rufitskiy, A. K. Ramasubramonian, and G. Van der Auwera. "Intra Prediction and Mode Coding in VVC". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021), pp. 1–1. DOI: 10.1109/TCSVT.2021.3072430.

[53] J.-L. Lin, Y.-W. Chen, Y.-W. Huang, and S.-M. Lei. "Motion Vector Coding in the HEVC Standard". In: *IEEE Journal of Selected Topics in Signal Processing* 7.6 (2013), pp. 957–968. DOI: 10.1109/JSTSP.2013.2271975.

[54] W.-J. Chien, L. Zhang, M. Winken, X. Li, R. Liao, H. Gao, C.-W. Hsu, H. Liu, and C.-C. Chen. "Motion Vector Coding and Block Merging in Versatile Video Coding Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[55] X. Zhao, S.-H. Kim, Y. Zhao, H. E. Egilmez, M. Koo, S. Liu, J. Lainema, and M. Karczewicz. "Transform Coding in the VVC Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021), pp. 1–1. DOI: 10.1109/TCSVT.2021.3087706.

[56] H. Schwarz, M. Coban, M. Karczewicz, T.-D. Chuang, F. Bossen, A. Alshin, J. Lainema, C. Helmrich, and T. Wiegand. "Quantization and Entropy Coding in the Versatile Video Coding (VVC) Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021), pp. 1–1. DOI: 10.1109/TCSVT.2021.3072202.

[57] D. Marpe, H. Schwarz, S. Boße, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Sühring, M. Winken, and T. Wiegand. "Video Compression Using Nested Quadtree Structures, Leaf Merging, and Improved Techniques for Motion Representation and Entropy Coding". In: *Transaction on Circuits and Systems for Video Technology* 20.12 (2010), pp. 1676–1687. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2010.2092615. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5638132.

[58] A. Moffat and A. Turpin. *Compression and Coding Algorithms*. Boston, MA: Springer US, 2002. ISBN: 978-1-4613-5312-6. DOI: 10.1007/978-1-4615-0935-6. URL: http://link.springer.com/10.1007/978-1-4615-0935-6.

[59] S. Forchhammer, X. Wu, and J. D. Andersen. "Optimal Context Quantization in Lossless Compression of Image Data Sequences". In: *IEEE Transaction on Image Processing* 13.4 (2004), pp. 509–517. ISSN: 1057-7149. DOI: 10.1109/TIP.2003.822613. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1284387.

[60] M. Xu, X. Wu, and P. Franti. "Context Quantization by Kernel Fisher Discriminant". In: *IEEE Transaction on Image Processing* 15.1 (2006), pp. 169–177. ISSN: 1057-7149. DOI: 10.1109/TIP.2005.860357. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1556635.

[61] ITU-T. *Recommendation H.264*. 2003. URL: http://www.itu.int/rec/T-REC-H.264.

[62] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand. "Transform Coding Techniques in HEVC". In: *Journal of Selected Topics in Signal Processing* 7.6 (2013), pp. 978–989. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2013.2278071. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6578061.

[63] ITU-T. *Recommendation H.265*. 2013. URL: http://www.itu.int/rec/T-REC-H.265.

[64] ITU-T. *Recommendation H.266*. 2021. URL: http://www.itu.int/rec/T-REC-H.266.

[65] M. J. Weinberger, G. Seroussi, and G. Sapiro. "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm". In: *Proceedings of Data Compression Conference - DCC '96*. 1996, pp. 140–149. DOI: 10.1109/DCC.1996.488319.

[66] G. Bjøntegaard. *Calculation of Average PSNR Differences between RD Curves*. VCEG-M33. Austin, USA: ITU-T, Apr. 2001.

[67] K. Sharman and K. Sühring. *Common Test Conditions*. JCTVC-AC1100. Macao, China: ITU-T and ISO/IEC, Oct. 2017.

[68] T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand. *CE11: Evaluation of Transform Coding tools in HE Configuration*. JCTVC-D061. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011.

[69] T. Kumakura and S. Fukushima. *Non-CE3: Simplified Context Derivation for Significance Map*. JCTVC-I0296. Geneva, Switzerland: ITU-T and ISO/IEC, Apr. 2012.

[70] T. Ji, N. Nguyen, D. He, and G. Martin-Cocher. *Sub-Block Based Significance Map Region Classification*. JCTVC-H0432. San Jose, USA: ITU-T and ISO/IEC, Jan. 2012.

[71] H. Schwarz, M. Coban, M. Karczewicz, T.-D. Chuang, F. Bossen, A. Alshin, J. Lainema, C. Helmrich, and T. Wiegand. "Quantization and Entropy Coding in the Versatile Video Coding (VVC) Standard". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021), pp. 1–1. DOI: 10.1109/TCSVT.2021.3072202.

[72] V. Sze and A. P. Chandrakasan. "A Highly Parallel and Scalable CABAC Decoder for Next Generation Video Coding". In: *IEEE Journal of Solid-State Circuits* 47.1 (2012), pp. 8–22. DOI: 10.1109/JSSC.2011.2169310.

[73] E. Y. Lam and J. W. Goodman. "A Mathematical Analysis of the DCT Coefficient Distributions for Images". In: *Transaction on Image Processing* 9.10 (2000), pp. 1661–1666. ISSN: 10577149. DOI: 10.1109/83.869177. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=869177.

[74] A. Gyorgy and T. Linder. "Optimal Entropy-Constrained Scalar Quantization of a Uniform Source". In: *IEEE Transactions on Information Theory* 46.7 (2000), pp. 2704–2711. DOI: 10.1109/18.887885.

[75] J. Stankowski, C. Korzeniewski, M. Domański, and T. Grajek. "Rate-Distortion Optimized Quantization in HEVC: Performance Limitations". In: *2015 Picture Coding Symposium (PCS)*. 2015, pp. 85–89. DOI: 10.1109/PCS.2015.7170052.

[76] S. W. Golomb. "Run-Length Encodings (Corresp.)" In: *IEEE Transaction on Information Theory* 12.3 (July 1966), pp. 399–401. ISSN: 0018-9448. DOI: 10.1109/TIT.1966.1053907. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1053907.

[77] P. G. Howard. *The Design and Analysis of Efficient Lossless Data Compression Systems*. 1993.

[78] A. Kiely. *Selecting the Golomb Parameter in Rice Coding*. IPN Progress Report 42-159, Nov. 2004. URL: https://ipnpr.jpl.nasa.gov/progress_report/42-159/159E.pdf.

[79] T. Nguyen, M. Winken, D. Marpe, S. Heiko, and T. Wiegand. *Reduced-Complexity Entropy Coding of Transform Coefficient Levels Using a Combination of VLC and PIPE*. JCTVC-D336. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011.

[80] V. Sze and M. Budagavi. *CE11: Parallelization of HHI_TRANSFORM_CODING (Fixed Diagonal Scan)*. JCTVC-F129. Torino, Italy: ITU-T and ISO/IEC, July 2011.

[81] F. Bossen, X. Li, V. Seregin, K. Sharman, and K. Sühring. *VTM and HM Common Test Conditions and Software Reference Configurations for SDR 4:2:0 10 bit Video*. JVET-Y2010. ITU-T and ISO/IEC, Jan. 2022.

[82] Y. Zheng, M. Coban, X. Wang, J. Sole, R. Joshi, and M. Karczewicz. *CE11: Mode Dependent Coefficient Scanning*. JCTVC-D393. Daegu, South Korea: ITU-T and ISO/IEC, Jan. 2011.

[83] M. Albrecht, C. Bartnik, S. Boße, J. Brandenburg, B. Bross, J. Erfurt, V. George, P. Haase, P. Helle, C. Helmrich, A. Henkel, T. Hinz, S. de Luxan Hernandez, S. Kaltenstadler, P. Keydel, H. Kirchhoffer, C. Lehmann, W.-Q. Lim, J. Ma, D. Maniry, D. Marpe, P. Merkle, T. Nguyen, J. Pfaff, J. Rasch, R. Rischke, C. Rudat, M. Schäfer, T. Schierl, H. Schwarz, M. Siekmann, R. Skupin, B. Stallenberger, J. Stegemann, K. Sühring, G. Tech, G. Venugopal, S. Walter, A. Wieckowski, T. Wiegand, and M. Winken. *Description of SDR, HDR and 360° Video Coding Technology Proposal by Fraunhofer HHI*. JVET-J0014. San Diego, USA: ITU-T and ISO/IEC, Apr. 2018.

[84] ITU-T SG16/Q6 and ISO/IEC JTC1/SC29/WG11. *Joint Call for Proposals on Video Compression with Capability Beyond HEVC*. JVET-H1002. Macao, China: ITU-T and ISO/IEC, Oct. 2017.

[85] K. Ramchandran and M. Vetterli. "Rate-Distortion Optimal Fast Thresholding with Complete JPEG / MPEG Decoder Compatibility". In: *IEEE Transactions on Image Processing* 3.5 (1994), pp. 700–704. DOI: 10.1109/83.334973.

[86] M. Karczewicz, Y. Ye, and I. Chong. *Rate Distortion Optimized Quantization*. VCEG-AH21. Antalya, Turkey: ITU-T, Jan. 2008.

[87] E.-h. Yang and X. Yu. "Soft Decision Quantization for H.264 with Main Profile Compatibility". In: *IEEE Transactions on Circuits and Systems for Video Technology* 19.1 (2009), pp. 122–127. DOI: 10.1109/TCSVT.2008.2009260.

[88] R. M. Gray. "Vector Quantization". In: *IEEE ASSP Magazine* 1.2 (1984), pp. 4–29. DOI: 10.1109/MASSP.1984.1162229.

[89] M. W. Marcellin and T. R. Fischer. "Trellis Coded Quantization of Memoryless and Gauss-Markov Sources". In: *IEEE Transactions on Communications* 38.1 (1990), pp. 82–93. DOI: 10.1109/26.46532.

[90] G. J. Sullivan. "Efficient Scalar Quantization of Exponential and Laplacian Random Variables". In: *IEEE Transactions on Information Theory* 42.5 (1996), pp. 1365–1374. DOI: 10.1109/18.532878.

[91] T. Lookabaugh and R. M. Gray. "High-Resolution Quantization Theory and the Vector Quantizer Advantage". In: *IEEE Transactions on Information Theory* 35.5 (1989), pp. 1020–1033. DOI: 10.1109/18.42217.

[92] G. D. Forney. "The Viterbi Algorithm". In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278. DOI: 10.1109/PROC.1973.9030.

[93] H. Schwarz, S. Schmidt, P. Haase, T. Nguyen, D. Marpe, and T. Wiegand. *Additional Support of Dependent Quantization with 8 States*. JVET-Q0243. Brussels, Belgium: ITU-T and ISO/IEC, Jan. 2020.

[94] F. Bossen, J. Dong, and H. Kirchhoffer. *CE1: Summary Report on CABAC Initialization*. JVET-O0021. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[95] J. Dong, A. Said, H. Wang, V. Seregin, and M. Karczewicz. *CE1-1.2 and CE1-2.1: Simplification of CABAC Initialization Process*. JVET-O0065. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[96] H. Kirchhoffer, D. Marpe, H. Schwarz, and T. Wiegand. *CE1-1.1: Simplification of the Initialization Process for Context Variables*. JVET-O0085. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[97] F. Bossen. *CE1: CABAC Initialization, All Experiments*. JVET-O0112. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[98] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand. *Non-CE7: Alternative Entropy Coding for Dependent Quantization*. JVET-K0072. Ljubljana, Slovenia: ITU-T and ISO/IEC, July 2018.

[99] J. Boyce, K. Sühring, X. Li, and V. Seregin. *JVET Common Test Conditions and Software Reference Configurations*. JVET-J1010. San Diego, USA: ITU-T and ISO/IEC, Apr. 2018.

[100] J. Chen, W.-J. Chien, R. Joshi, J. Sole, and M. Karczewicz. *Non-CE1: Throughput Improvement on CABAC Coefficients Level Coding*. JCTVC-H0554. San Jose, USA: ITU-T and ISO/IEC, Jan. 2012.

[101] H. Schwarz, T. Nguyen, D. Marpe, T. Wiegand, M. Karczewicz, M. Coban, and J. Dong. *CE7: Transform Coefficient Coding with Reduced Number of Regular-Coded Bins (Tests 7.1.3a, 7.1.3b)*. JVET-L0274. Macao, China: ITU-T and ISO/IEC, Oct. 2018.

[102] F. Bossen. *CABAC Cleanup and Complexity Reduction*. JVT-E086. Geneva, Switzerland: ITU-T and ISO/IEC, Oct. 2002.

[103] M. Zhou. *AHG16/Non-CE7: A Study of Bin to Bit Ratio for VTM-5.0*. JVET-O0068. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[104] T.-D. Chuang, S.-T. Hsiang, Z.-Y. Lin, C.-Y. Chen, H. Yu-Wen, and S.-M. Lei. *CE7 (Tests 7.1, 7.2, 7.3, and 7.4): Constraints on Context-Coded Bins for Coefficient Coding*. JVET-M0173. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019.

[105] T.-D. Chuang, S.-T. Hsiang, Z.-Y. Lin, C.-Y. Chen, Y.-W. Huang, S.-M. Lei, M. Coban, and M. Karczewicz. *CE7-1: TB-Level Constraints on Context-Coded Bins for Coefficient Coding*. JVET-O0052. Gothenburg, Sweden: ITU-T and ISO/IEC, July 2019.

[106] Z.-Y. Lin, T.-D. Chuang, C.-Y. Chen, Y.-W. Huang, and S.-M. Lei. *CE7-Related: Context Modeling Using Quantization Index for Dependent Quantization*. JVET-L0097. Macao, China: ITU-T and ISO/IEC, Oct. 2018.

[107] X. Xu, S. Liu, T.-D. Chuang, Y.-W. Huang, S.-M. Lei, K. Rapaka, C. Pang, V. Seregin, Y.-K. Wang, and M. Karczewicz. "Intra Block Copy in HEVC Screen Content Coding Extensions". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6.4 (2016), pp. 409–419. DOI: 10.1109/JETCAS.2016.2597645.

[108] W. Pu, M. Karczewicz, R. Joshi, V. Seregin, F. Zou, J. Sole, Y.-C. Sun, T.-D. Chuang, P. Lai, S. Liu, S.-T. Hsiang, J. Ye, and Y.-W. Huang. "Palette Mode Coding in HEVC Screen Content Coding Extension". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6.4 (2016), pp. 420–432. DOI: 10.1109/JETCAS.2016.2605661.

[109] M. Mrak and J.-Z. Xu. "Improving Screen Content Coding in HEVC by Transform Skipping". In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. 2012, pp. 1209–1213.

[110] J. Xu, R. Joshi, and R. A. Cohen. "Overview of the Emerging HEVC Screen Content Coding Extension". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.1 (2016), pp. 50–62. DOI: 10.1109/TCSVT.2015.2478706.

[111] M. Mrak, A. Gabriellini, N. Sprljan, and D. Flynn. *Transform Skip Mode*. JCTVC-F077. Torino, Italy: ITU-T and ISO/IEC, July 2011.

[112] X. Peng, C. Lan, J. Xu, and G. J. Sullivan. *Inter Transform Skipping*. JCTVC-J0237. Stockholm, Sweden: ITU-T and ISO/IEC, July 2012.

[113] C. Lan, J. Xu, G. J. Sullivan, and F. Wu. *Intra Transform Skipping*. JCTVC-I0408. Geneva, Switzerland: ITU-T and ISO/IEC, Apr. 2012.

[114] J. Sole, R. Joshi, and M. Karczewicz. *RCE2 Test B.1: Residue Rotation and Significance Map Context*. JCTVC-N0044. Vienna, Austria: ITU-T and ISO/IEC, July 2013.

[115] X. Xu, Y.-H. Chao, Y.-C. Sun, and J. X. Xu. *Description of Core Experiment 8 (CE8): Screen Content Coding Tools*. JVET-M1028. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019.

[116] B. Bross, T. Nguyen, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand. *Non-CE8: Unified Transform Type Signalling and Residual Coding for Transform Skip*. JVET-M0464. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019.

[117] T. Tsukuba, M. Ikeda, and T. Suzuki. *Non-CE6: On Transform Skip for Larger Block*. JVET-M0072. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019.

[118] T. Nguyen, X. Xu, F. Henry, R.-L. Liao, M. G. Sarwer, M. Karczewicz, Y.-H. Chao, J. Xu, S. Liu, D. Marpe, and G. J. Sullivan. "Overview of the Screen Content Support in VVC: Applications, Coding Tools, and Performance". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.10 (2021), pp. 3801–3817. DOI: 10.1109/TCSVT.2021.3074312.

[119] B. Bross, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand. *CE8: Residual Coding for Transform Skip Mode (CE8-4.3a, CE8-4.3b, CE8-4.4a, and CE8-4.4b)*. JVET-N0280. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019.

[120] M. Xu, X. Li, X. Xu, M. Gao, and S. Liu. *CE8-Related: BDPCM Entropy Coding with Reduced Number of Context Coded Bins*. JVET-M0449. Marrakech, Morocco: ITU-T and ISO/IEC, Jan. 2019.

[121] M. Karczewicz and M. Coban. *CE8-Related: Sign Context Modelling and Level Mapping for TS Residual Coding*. JVET-N0405. Geneva, Switzerland: ITU-T and ISO/IEC, Mar. 2019.