Intra Coding Tools for Versatile Video Coding (VVC)

Thesis · June 2019

1 author:

Mohsen Abdoli
IRT B-com
25 PUBLICATIONS **97** CITATIONS

SEE PROFILE

université PARIS-SACLAY

CentraleSupélec

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

NNT : 2017SACLE035

Thèse de doctorat

Intra Coding Tools for Versatile Video Coding (VVC)

Thèse de doctorat de l'Université Paris-Saclay
Préparée à Université Paris-Sud

École doctorale n°580
Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité: Traitement du signal et des images

Thèse présentée et soutenue à Rennes, le 4 juin 2019, par

Mohsen Abdoli

Composition du Jury :

Luce Morin Professeure à l'INSA de Rennes, France	Présidente
Aline Roumy Professeure à l'Inria, Rennes, France	Rapporteuse
Olivier Déforges Professeur à l'INSA de Rennes, France	Rapporteur
François-Xavier Coudoux Professor à l'Université Polytechnique des Hauts-de-France	Examinateur
Marco Cagnazzo Professor à Télécom ParisTech, France	Examinateur
Patrice Brault Enseignant/rechercheur à CentraleSupelec Gif-sur-Yvette, France	Directeur de thèse
Félix Henry Ingénieur de recherche à Orange Labs Rennes, France	Encadrant de thèse
Frédéric Dufaux Professor à CentraleSupelec Gif-sur-Yvette, France	Co-Directeur de thèse
Pierre Duhamel Professor à CentraleSupelec Gif-sur-Yvette, France	Co-Directeur de thèse

Abstract

By 2021, 80% of the world's Internet traffic will be somehow video related. This amount of data is expected to be represented in new formats (e.g. 8K, HDR, HFR, immersive) that are incompatible or poorly compatible with the existing video codecs. As a result, the video coding industry is preparing to face a major challenge for a proper compression of this content. In order to cope with this challenge, new video codecs are under development by different standardization committees. Jointly developed by the MPEG and ISO, Versatile Video Coding (VVC) is the next generation video codec that has gathered the most attention and is expected to be finalized by 2020. This codec will supposedly replace its ancestor, High Efficiency Video Coding (H.265/HEVC), with a more efficient compression performance in terms of rate-distortion. As the performance target, the VVC standardization committee aims at reducing the bitrate of compressed videos by 50%, while retaining the same level of objective and subjective quality.

The initial objective of this thesis was to explore coding capacities beyond HEVC. To this end, the efforts have been aligned with the VVC standardization by taking its coding components and improving them. Some of these improvements were eventually proposed to the VVC standardization committee and adopted in the codec syntax. More specifically, the intra coding module of VVC has been thoroughly studied in order to improve its coding performance on natural and synthetic contents. Contributions in two domains have been presented: intra prediction and residual coding. In each domain, a general framework has been designed and integrated in the VVC reference software. Then, different algorithms have been implemented on top of the framework. More specifically, in the prediction domain, the focus has been put on addressing short distance pixel prediction for complex or irregular image textures. As the short distance problem required special attention for residual representation, it led the second domain to the design of alternative methods for the decorrelation of remaining redundancies in prediction residual. Experiments in both domains show that the use of the proposed frameworks provide significant bitrate saving, with a moderate complexity cost.

Contents

List of Figures

List of Tables

List of acronyms

HDR	High Dynamic Range
HFR	High Frame Rate
MPEG	Motion Picture Expert Group
ITU	International Telecommunication Union
VCEG	Video Coding Expert Group
AVC	Advanced Video Coding
HEVC	High Efficiency Video Coding
VVC	Versatile Video Coding
JVET	Joint Video Exploration Team
MB	Macroblocks
SMB	Sub-MB
HD	High Definition
UHD	Ultra High Definition
CTU	Coding Tree Unit
QT	Quad-Tree
CU	Coding Unit
QTBTTT	Quad-Tree, Binary-Tree and Ternary-Tree
ME	Motion Estimation
MV	Motion Vector
CABAC	Context Adaptive Binary Arithmetic Coding
QP	Quantization Parameter
RDO	Rate-Distortion Optimization
PSNR	Peak-Signal-to-Noise-Rate
BD-R	Bjontegaard Delta Rate
GOP	Group Of Picture
AI	All Intra

RA	Random Access
LD	Low Delay
IPM	Intra Prediction Modes
PDF	Probability Distribution Function
MPM	Most Probable Mode
PDPC	Position Dependent Prediction Combination
JEM	Joint Exploration Model (not defined)
CCLM	Cross-Component Linear Model
DRI	Distant Reference Inaccuracy
JCT-VC	Joint Collaborating Team on Video Coding
SDIP	Short Distance Intra Prediction
CIP	Combined Intra Prediction
DPCM	Diffrential Pulse-Code Module
LIC	Line-based Intra Coding
ILR	In-Loop Residual
OLR	Out-Loop Residual
VQ	Vector Quantization
SQ	Scalar Quantization
LOCO-I	LOw COmplexity LOssless COmpression for Images
LBG	Linde-Buzo-Gray
CSF	Codebook Size Factor
EMT	Explicit Multiple Transforms
CBF	Coded Block Flag
JPEG	Joint Photograpy Expert Group
VTM	VVC Test Model
SSE	Sum of Squared Error
SAD	Sum of Absolute Difference

General introduction

Context

As for industry involvement in video coding research, it appears that areas that depend on video technologies are more active than ever. These areas include any application and technology that requires transmission or storage of video/image signals. Hence, without solid and powerful compression technologies, the future of all these areas is ambiguous. For this very reason, video coding standardization committees have recently been occupied with releasing new standards, with a performance beyond the capacity of previous ones.

A new video standardization is currently under development by the largest group of video technologists. This action of standardization, initiated in late 2015, will supposedly release the future standards in 2020. As a major objective, this standard shall be responsible for compression of a large variety of video signals, apart from the conventional videos. Accordingly, this standard has been decided to be called Versatile Video Coding (VVC) and will be under constant development in different domains, until its final release.

The project of this thesis started in December 2015, when the VVC exploration was in its early stages. Therefore, the main focus was put on participation in its development in terms of compression efficiency improvement. To this end, different ideas were implemented and evaluated to see whether they can address drawbacks of the standard under development. Specifically, intra coding of VVC has been targeted in this research and two main contributions have been carried out: block prediction and residual coding. The current manuscript has been organized as follows:

Part I: Context of Video Coding

Chapter 1 A brief introduction on the key modules of video coding standards is provided. For this purpose, a historical perspective has been used to give information about the evolution of tools in each module.

Chapter 2 Further details regarding the specific domain of our contribution are presented. This includes three parts in both domains of contributions: state-of-the-art, problem definition, its shortcomings and alternative solutions to address them.

Part II: Proposed Intra Prediction Tools

Chapter 3 A general framework is proposed to be used as the baseline of the next two chapters. This framework allows applying fundamental changes to the conventional intra prediction scheme. Such changes are potentially useful in certain conditions of video content, specifically, high-detail texture compression.

Chapter 4 First intra prediction algorithm using a Vector Quantization approach is proposed. This algorithm conforms the framework proposed in Chapter 3 and brings additional compression gain on top of the VVC-based anchor codec.

Chapter 5 The second proposed algorithm, based on Scalar Quantization, is presented. This algorithm uses a similar concept as that of Chapter 5, however, structural changes have been made in its design choices. These changes make the algorithm particularly efficient for intra coding of synthetic content, which has an importance in the VVC standardization.

Chapter 6 A conclusion to the second part is given. To this end, two proposed algorithm are compared in terms of their structure and design choices. Moreover, pros and cons of each algorithm are discussed and their possible future tracks are discussed in detail.

Part III: Proposed Transform Coding Tools

Chapter 7 Similar to the Part II, a framework is proposed in this chapter. This framework can be used as a general approach to encode certain coding elements, however, we particularly focus on using it as a transform coefficient coding infrastructure. In a nutshell, the proposed framework allows further benefiting from redundancies in the signal and increases the compression efficiency.

Chapter 8 One implementation of the proposed coefficient coding is presented. In this implementation, a set of informative features are utilized to reduce the rate of transform coefficient signal. This method, which still has room for improvement, has a potential compression gain, especially on synthetic contents. At the end, review of open aspects of the proposed framework are presented.

Finally, a conclusion of this thesis is presented to wrap up the entire project and provide some possible future tracks. Moreover, additional information in the form of Appendixes are added to help readers in following the topic.

Contributions

The work that has been carried out during the project of this thesis, can be summarized in a few main contributions:

- Deeply studying the standard draft as well as the reference software of VVC, in order to spot its remaining drawbacks.
- Implementing different algorithms within the reference software with or without normative changes.
- Proposing two intra prediction algorithms to improve performance on high-detail textures.
- Proposing a transform coefficient coding algorithm to further exploit the remaining redundancies in signal

Part I
Context of the video coding

CHAPTER 1

Principles of video coding

Contents

1 Introduction

Every second in year 2021, more than a million minutes of video content will cross the network. It would take a person more than 5 million years to watch all videos of one month [1]. This forecast is convincing for video coding experts to think of more efficient compression tools and technologies. These technologies are expected to address various emerging video formats, namely High Dynamic Range (HDR), High Frame Rate (HFR), high resolution videos (e.g. 4K, 8K and beyond), immersive 360 videos, screen content and more.

Video coding standards aim at bringing format compatibility between devices. This enables the playback of any video file conforming the syntax of a given standard, with any device supporting it. From the industrial point of view, such unity facilitates the interaction between all components of the broadcast chain, including consumer electronics manufactures, broadcasters, content providers etc. This convenience in interaction, if achieved, can significantly accelerate the progress of the broadcast industry as a whole.

The Motion Picture Expert Group (MPEG) has been the main entity to provide video coding standard specification in the past 30 years. Most successful standardization acts of the MPEG were accomplished after its collaboration with the International Telecommunication Union (ITU), in the late 90's. This joint collaboration, initially called Joint Video Team (JVT), then Joint Collaboration Team on Video Coding (JCT-VC), resulted in some of the most successful video coding standards in the family of "H.26x", notably H.264 Advanced Video Coding (AVC), and H.265 HEVC.

A similar collaboration between the MPEG and ITU-T, called Joint Video Exploration Team (JVET), is currently in progress. This team, consisting of numerous experts with different backgrounds (e.g. hardware, software, network etc.), aims at investigating the potential need for another video coding standard. Such standard is supposedly responsible for compression of various types of video formats. Hence, the name Versatile Video Coding (VVC) has been chosen for it to reflect this primary goal. According to the time-line, VVC is expected to freeze its specification and become a standard in 2020.

The technical details presented in this thesis are based on the principles of VVC that have been determined so far. However, from many aspects, they are easily applicable to other standards, including those from the same family (i.e. H.26x/xVC) as well other families (e.g. AV1, VPx etc.). This generalizability is due to the fact that almost all modern video coding standards share a similar structure based on block-based hybrid compression. In this chapter, a brief review of this compression structure is presented to provide a background for the rest of this thesis. For this purpose, the "hybrid" and "block-based" aspects of the video coding standards are discussed first and then some essential modules to build a codec with this structure are discussed. In order to emphasize on the similarity of general structure between different video coding standards, some sections will also provide equivalent historical information from HEVC or AVC along with VVC.

2 Hybrid block-based video coding

Information in video signals is growingly complex. The bright side of such complexity is an excessive amount of redundancy in different forms, which makes the signal compression more feasible. Integrating different decorrelation techniques in one codec to decorrelate different types of redundancies, is referred to as the hybrid aspect of video coding. That is to say, various technologies, each designed for decorrelation of a specific type of redundancy, are integrated in a hybrid video coding system to exploit and remove as much as possible amount of correlation in video signals.

Figure 1-1 shows the simplified diagram of a hybrid video coding standard. This structure summarizes the high-level algorithm implemented in all of the three standards of AVC, HEVC

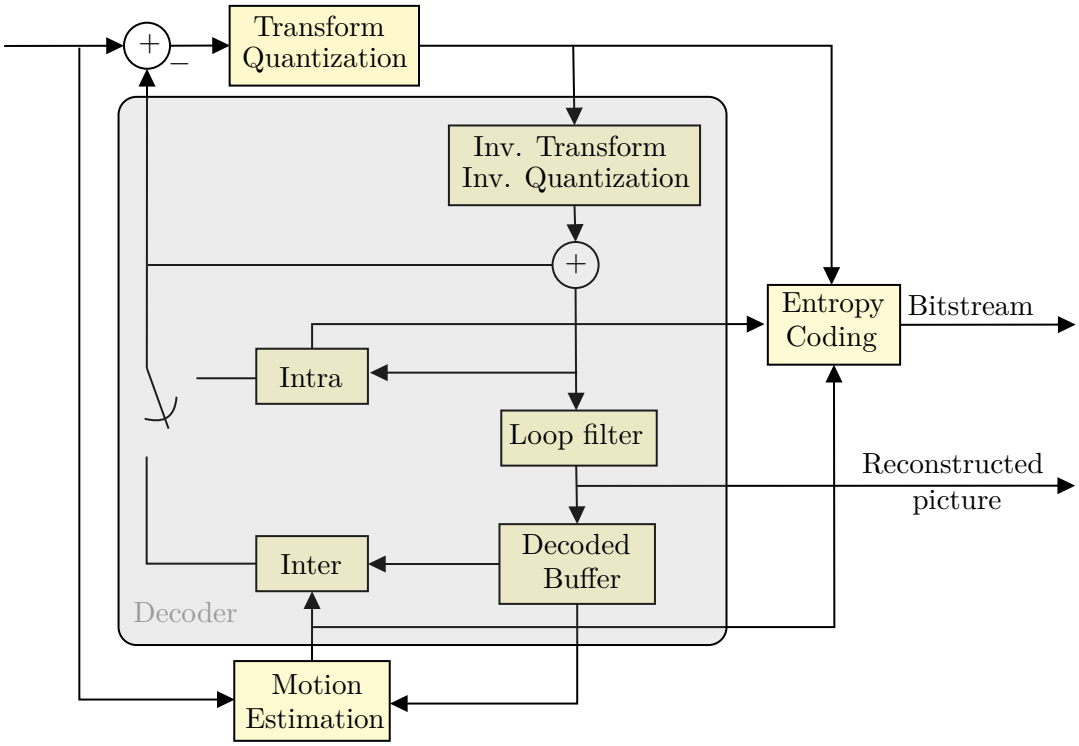


Figure 1-1: Diagram of a block-based hybrid video coding system [2].

and VVC. The units that are separated in rectangular boxes in this figure, represent different modules of a codec. Some of the important modules will be discussed with more details in the rest of this chapter.

Containing various tools in a hybrid video coding, each of which targeting specific type of content, provides flexibility in terms of coding choices, adapted for different conditions of signal. These content-dependent conditions constantly change in the spatial domain (i.e. within a single frame) and temporally (i.e. through frames). Therefore, the encoder needs to react to these changes by making different coding decision once the content condition is changed.

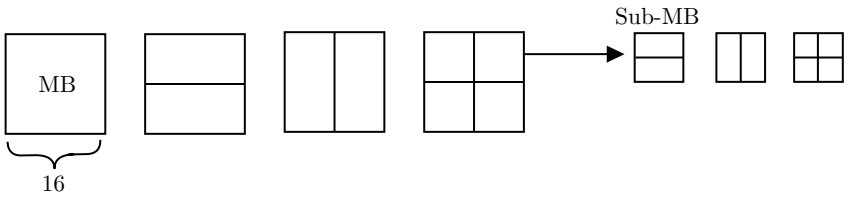
Block partitioning provides spatial isolation of regions that require different coding decisions. For instance, assume that a video frame contains a motionless background with a regular texture along with a moving object. In this situation, the encoder can partition these regions into two blocks and use different decorrelation tools on each block. To this end, different non-overlapping rectangular block partitioning have been used in standards. As will be explained later, depending on the standard, shape, size and even name of this rectangular region are different. In this thesis, each partition unit is simply referred to as a "block", except when a specific standard is being discussed.

Two main factors have determined the partitioning specification of the past video standards: computational capacity and video resolution. With respect to these aspects, different partitioning schemes have been adopted in the recent video coding standards to meet their requirements. Here, the progress of these schemes is discussed in a chronological order to provide a clear understanding of its evolution.

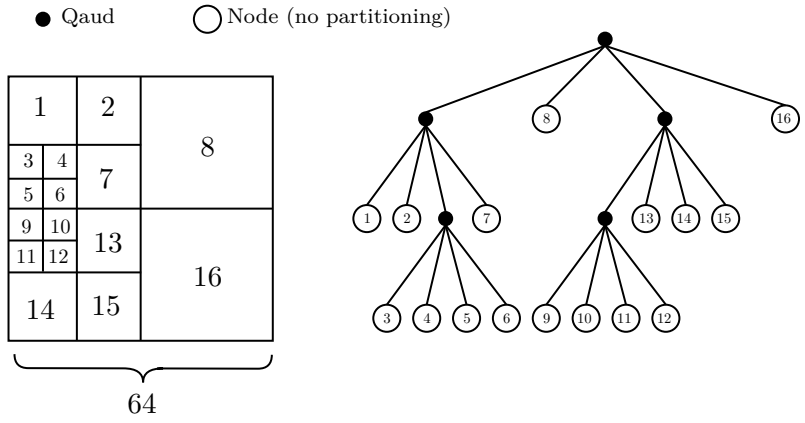
2.1 Block partitioning in AVC

In AVC, Macroblocks (MB) of size 16×16 are the bases of coding. They allow further block splitting down to 4×4 Sub-MB (SMB) to capture high detail regions. In the middle of this size range, the 8×8 and four rectangular sizes of 16×8, 8×16, 8×4 and 4×8 block sizes are also

a) AVC: Macroblocks (MB), maximum size of 16.



b) HEVC: Quad-Tree (QT), maximum size of 64.



c) VVC: QT with a nested multi-type tree using Binary-Tree (BT) and Ternary-Tree (TT), maximum size of 256.

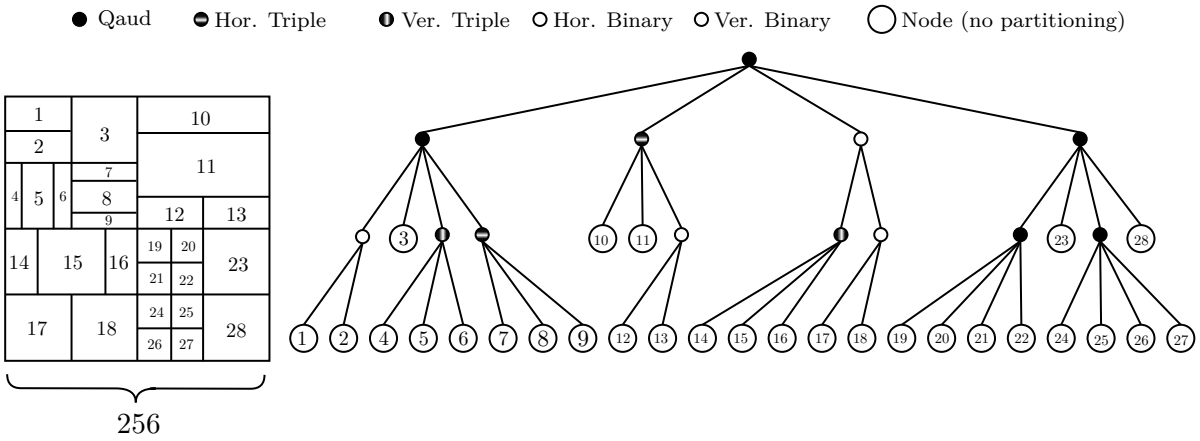


Figure 1-2: Block partitioning used in three video coding standards of AVC, HEVC and VVC.

permitted. Figure 1-2 (a) depicts all possible MB size choices.

There were two main reasons in limiting the partitioning to this MB size for the block partitioning during the standardization of the AVC. Firstly, enabling larger MB sizes (e.g. 32×32) would have required more computation and hence, more encoder side complexity. Second, in the early 2000's and when the AVC was being standardized, the perspective of higher resolutions, such as Ultra High Definition (UHD), was too far and was considered to be handled in future standards. Therefore, the current MB size is a result of a compromise between coding performance, computational complexity and the standardization schedule.

2.2 Block partitioning in HEVC

A Coding Tree Unit (CTU) in HEVC corresponds to a MB in AVC, with two main differences. First, the maximum CTU size is 64×64, which enables more efficiently coded high resolution videos. Second, it performs a Quad-Tree (QT) block partitioning scheme. This partitioning scheme considers the CTU as the root of a tree, where each node can either be a leaf node, or have 4 children. Each node in this tree structure is called a Coding Unit (CU) and is processed separately.

Figure 1-2 (b) shows an example of quad-tree partitioning on a 64×64 CTU in HEVC. Through this simple example, it can be understood that beside higher resolution blocks (e.g. CUs), the quad-tree also provides more flexibility in the shape of final partitions. However, this complexity results in a higher computational cost as well as signaling. Compared to AVC, this flexibility allows a higher coding efficiency both on complex and plain video contents, especially on higher resolutions (e.g. High Definition (HD), UHD).

2.3 Block partitioning in VVC

Block partitioning in VVC is even more flexible than in HEVC. First, a CTU in VVC can have up to 128×128 pixels. This feature helps VVC to properly adapt to higher resolutions. Another new feature in VVC is the possibility apply to three different CTU splitting approaches, namely: Quad-Tree, Binary-Tree and Ternary-Tree (QTBTTT). Figure 1-2 (c) demonstrates the adopted block partitioning in VVC. To apply this partitioning, each CU can be further split using one of the following partitions:

1. Quadratic partitioning: the square CU is partitioned into 4 CUs. This is similar to the QT partitioning of HEVC explained in previous section.

2. Binary partitioning: each leaf node is split into two CUs. This step requires the encoder to explicitly signal whether horizontal or vertical direction has been used.

3. Ternary partitioning: each leaf node is split into three CUs. This partitioning should split the leaf CU into one CU of half its size in the middle and two other CUs of its quarter size, on either side. Similar to the binary partitioning, the encoder needs to signal the split direction in a ternary tree too.

3 Decorrelation modules and compression techniques

There are mainly two domains of redundancies in video signals: 1) spatial, 2) temporal. In each domain, several algorithms have been developed to exploit the redundancy and further compress the signal. In the spatial domain, a possible regularity in the texture within a single frame is modeled. while in the temporal domain, the possible motion of blocks is modeled. In both cases, the encoder is given different models for which it performs specific prediction methods to optimize the parameters associated to that model.

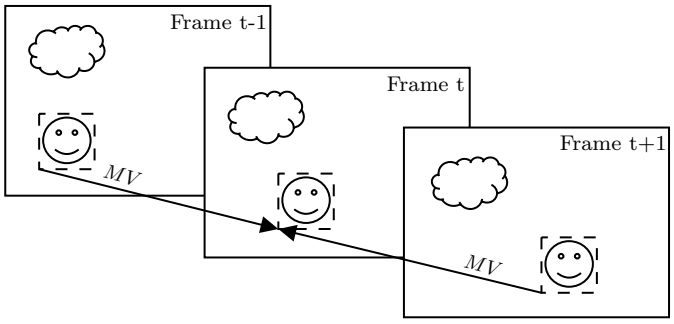


Figure 1-3: The ME process to find the best Motion Vector (MV). Here, t denotes current frame and $t\pm1$ are reference frames.

Even the most accurate prediction models introduce an error. This error, represented in a signal called the residual, R, is computed as the difference between the original uncompressed input signal and its predicted model. Eq. 3.1 expresses this residual calculation, where O and P are the original and prediction signals, respectively.

$$R = O - P. \tag{3.1}$$

In order to control the level of distortion, the encoder has to transmit the residual signal. In the lossy residual coding, the encoder transmits an estimation of the residual in Eq. 3.1 as $\hat{R}$, where $\hat{R} \neq R$. At the decoder side, the prediction model selected by the encoder is reproduced and added to the decoded residual signal $\hat{R}$. This provides the final reconstructed signal C, as expressed in Eq. 3.2. Depending on the amount of information loss during the lossy residual coding, the approximation error at the right side of Eq. 3.2 can vary. In case of lossless residual coding, where $\hat{R} = R$, this equation turns to $C = O$.

$$C = P + \hat{R} \approx O \tag{3.2}$$

Below, essential modules associated to the above prediction models and the coding of their residual error are briefly presented. Furthermore, in the following chapters, more detailed discussions will be provided in the domains on which this thesis proposes novel algorithms.

3.1 Temporal redundancy: Inter-frame prediction

In each scene cut of a video, the objects and regions of the content approximately contain similar pixels, though slightly displaced. The idea of the inter-frame prediction is to model the displacement of these regions or objects, restricted to rectangular blocks.

The algorithm that performs the above motion modeling is called Motion Estimation (ME) and takes two inputs: the original block to model and a reference frame shifted in time (e.g. previous or next frame). ME finds a MV among a set of candidates by maximizing the similarity between the original block and its displaced block in the reference. Figure 1-3 shows a simple example of ME from either previous or next reference frame.

3.2 Spatial redundancy: Intra-frame prediction

A regular pattern in texture of natural images can be represented by geometric models. This feature is used in intra prediction methods in order to remove redundancies in the spatial domain. Assume that the encoder aims at performing intra prediction on the green block of the current frame as shown in Figure 1-4, given all previously coded CTUs shown in gray as reference. The intra prediction takes a subset of reference pixels from previous blocks and applies different

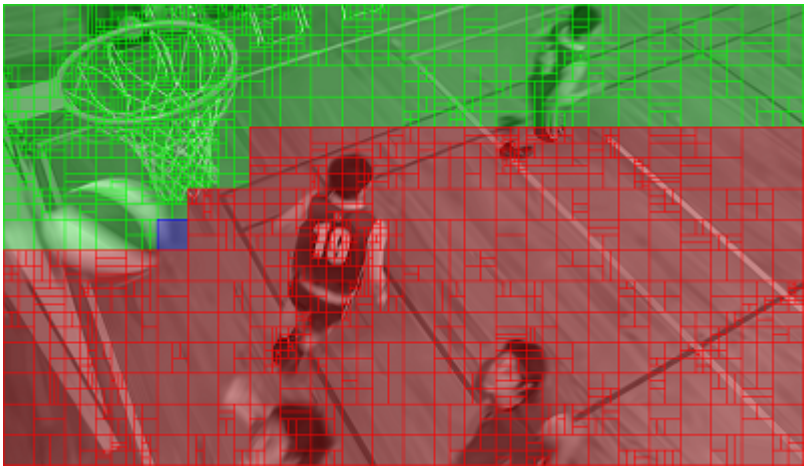


Figure 1-4: Raster scan of the intra blocks in an image. The blue block is the current one, while the green and red ones are respectively, the past and future blocks.

angular reference projections. Similar to the inter prediction, the objective of the intra prediction is also to find the best model parameters that maximize the similarity to the original block. In the next chapter, more details of the intra prediction algorithm are presented.

3.3 Transform coding of the residual error

Residual transmission uses transform coding. This is due to the fact that the residual energy is mostly concentrated in low frequency regions, and hence into a few non-zero transform coefficients. This interesting property can significantly help the encoder decorrelate the remaining redundancies in residual signals.

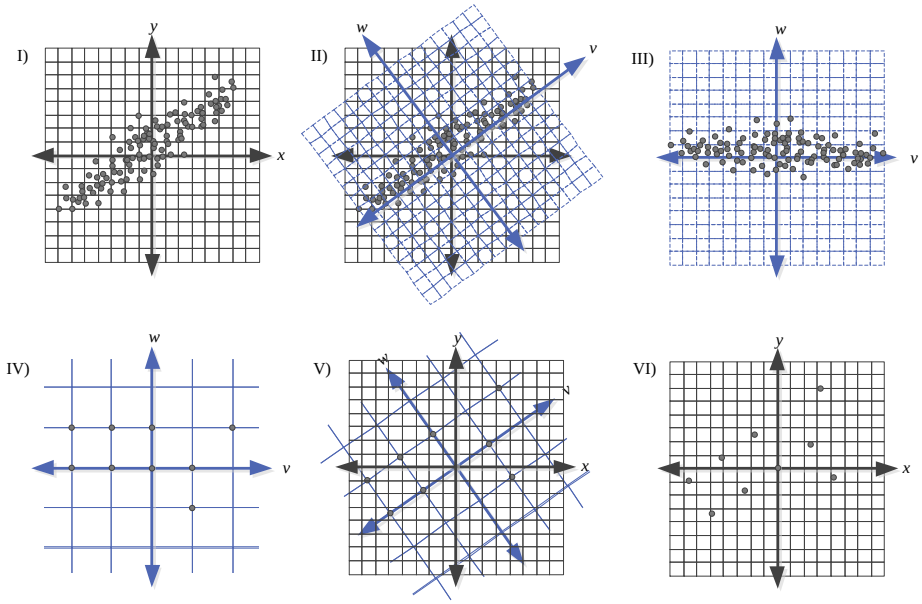
Transform coding also enables lossy compression techniques to compress the signal by eliminating their least informative parts. In other words, by mapping a signal into the transform domain, one can extract a part of the signal that, if removed, minimizes artifacts that would be noticed by the human visual perception system. In lossy image and video coding schemes, this step is carried out by quantization of the transform coefficients.

To understand the impact of lossy transform coding, assume that an encoder is given an input signal as shown in Fig 1-5-I. This signal is in 2D and contains samples in the xy plane, while there is an evident angular correlation in the vw plane, as can be seen in Fig 1-5-II. However, the current xy plane is unable to properly exploit this correlation. In this situation, applying a proper transform function on the samples projects them in the new vw plane, as shown in Fig 1-5-III. The projected samples are now called coefficients of the transform that was just applied. The final step in a lossy encoder is to quantize the vw plane and to estimate the coefficients with their nearest quantized levels. This representation, shown in Fig 1-5-IV, brings further compression rate with the cost of introducing a quantization distortion.

At the decoder side, the quantized coefficients are parsed and represented in the same way as Fig 1-5-IV. As the decoder is aware of the applied transform in step II, it can calculate the inverse transform to project the parsed coefficients back to the xy plane (Fig 1-5-V). Finally, the decompressed samples look like Fig 1-5-VI.

The transform coding of the residual error in video coding performs a similar scheme to the above example. The main difference is that the dimension of the input signal must be as large as the number of pixels in the block. For instance, for 8×8 blocks, a transform $T : \mathbb{R}^{64} \rightarrow \mathbb{R}^{64}$ is required. Figure 1-6 shows a real example of applying the transform and quantization steps on an 8×8 residual block. In this figure, the impact of different levels of quantization on the information loss and the coefficients energy is presented. As can be seen, in the first scenario

Figure 1-5: A simple 2D input signal to be compressed by transform and quantization. I) Input. II) Directional correlation. III) Transformation. IV) Quantization in the transform domain. V) Inverse transform to the pixel domain. VI) Lossy reconstructed signal.



in the top of the figure, applying a coarse quantizer results in a few non-zero coefficients in the transform domain. Its impact on the amount of information loss is visible in the reconstructed signal at bottom-right. On contrary, the second scenario at the bottom of the figure applies a finer quantizer in comparison to the first scenario. This results in a better reconstruction at the cost of more non-zero coefficients in the transform domain (i.e. a higher rate).

3.4 Entropy coding of symbols

Once coding decisions (e.g. block size, prediction model, model parameters etc.) are made at the encoder side, their corresponding symbols need to be entropy coded. For this purpose, different entropy coding methods have been proposed. Entropy coders are able to exploit the remaining correlation in symbols according to their probability.

According to Shannon's source coding theorem, the optimal code length of a symbol is $-log_b P$, where b is the number of symbols needed to make output codes and P is the probability of the input symbol [3]. This theorem means that symbols with a probability close to 0 have higher code length (rate) than symbols with a probability closer to 1. In other words, the more likely a symbol value is to happen, the shorter its code length become.

Two of the most common entropy coding methods based on the Shannon‘s theorem are Huffman coding [4] and arithmetic coding [5]. Both methods are also implemented for video coding with Context Adaptive Variable Length Coding (CAVLC) and Context Adaptive Binary Arithmetic Coding (CABAC), respectively. These two methods enable video codecs write/read binary symbols with a rate close to their optimal rate according to the Shannon‘s theorem. However, the coding performance of CABAC is significantly superior to CAVLC, at the cost of a higher computational complexity both at encoder and decoder side. This is due to the

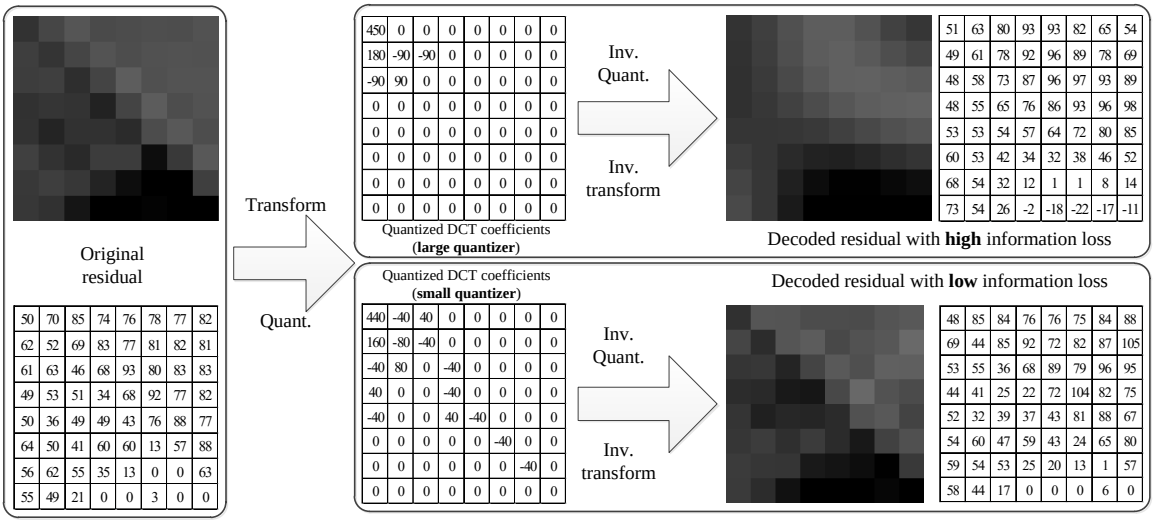


Figure 1-6: Transform and quantization of a 8×8 block and their impact on the decoded signal.

fact that CAVLC uses an integer number of bits to code a symbol, while CABAC can "share" bits between symbols. Due to this inefficiency of CAVLC and the fact that electronic devices have higher computational capacity nowadays, CABAC has completely replaced CAVLC since HEVC.

In CABAC, context models are integrated to separately track the statistical behavior of binary symbols with significantly different probabilities. For this purpose, non-binary symbols are first binarized to produce coding "bins". Each bin is a binary symbol and is associated with a CABAC context model. This context model can either be exclusively dedicated to that bin or shared between more than one bin, with similar statistical behavior.

Theoretically, a finer source separation results in a higher compression efficiency. However, this usually requires an excessive number of CABAC context models which imposes an unfeasible memory and computational complexity overhead. Therefore, a practical video codec needs to make a compromise between complexity and compression efficiency, by tuning the number of CABAC context models.

4 Encoder control

Given all the coding modules containing different coding tools, the encoder needs to determine when and how to use each tool. Therefore, the encoding flow can be seen as a comprehensive procedure of decision making. Some examples of high level questions leading to different decisions in different domains are:

- Given a neighborhood of pixels, what block size should be used for its content? Should the neighborhood be split into small blocks to make fine decisions? Or is a large block adequate for a coarse modeling of the content?
- Given the block size, what prediction scheme should be applied to the content? Should the content be modeled by its relative displacement with respect to its temporary shifted reference? Or should its texture be modeled within the same frame, using an angular projection?
- Given the inter mode as the prediction scheme, what MV should be selected to represent the motion?

- Given the intra mode as the prediction scheme, what pre-defined pattern should be selected for texture modeling?
- Given the prediction model and its residual error, what transform should be applied? And how finely coefficients of the selected transform should be quantized?

The above list can be further extended by adding numerous questions in lower level. However, the decision making procedure is quite similar in response to all questions and tests: The encoder has to test different choices and to pick the one that optimizes an objective function.

The key point is which objective criterion is qualified for the evaluation of each choice? One might assume that fidelity metrics such as Squared Error (SE) applied on the final reconstructed signal, would be a proper criterion. However, the coding efficiency measurement is rather a bilateral problem taking into account not only the decoded signal quality, but also the transmission rate.

The example presented in Figure 1-6 of previous section, simply explains this bilateral optimization problem. As explained, the first scenario in the top outputs a high distortion signal with a low rate. On the contrary, the second scenario in the bottom provides a low distortion signal with a relatively high rate. None of these scenarios is definitively superior to the other. For instance, in a low bit-rate live video conference application, where a stable and low delay transmission is required, the first scenario probably is probably preferred. However, in a video streaming application, where the high quality of video is critical, the second scenario will be chosen.

The compromise between rate and distortion is determined by the user and is given to the encoder as input. The parameter to balance this compromise is called the Quantization Parameter (QP). The main functionality of QP is to feed two internal functions of the encoder: 1) the rate-distortion cost calculation and 2) the transform coefficient quantization.

4.1 Rate-distortion cost

The rate-distortion cost calculation is used all along the encoding flow, in order to a apply the compromise that the chosen QP imposes. To this end, QP makes the optimum balance between the rate and the distortion by defining a weight λ as expressed in Eq. 4.1.

$$\lambda = 0.85 \times 2^{(QP-12)/3}. \tag{4.1}$$

One simple way of understanding how the encoder controls the rate-distortion balance, is to explain the prediction mode decision at the block level. Assume that the encoder is given λ and aims at determining the optimum prediction mode $m = \{intra, inter\}$. As described earlier in this section, in both predictions the encoder models the block and transmits all the syntax elements corresponding to the prediction parameters in addition to a residual error.

Depending on the statistics of these syntax elements, coded by CABAC, this step would result in two different rates for the two prediction modes, say R_{intra} and R_{inter}. Moreover, the lossy quantized error introduces two different amounts of distortion, D_{intra} and D_{inter}, measured in terms of SE. At this point, the rate-distortion cost, J, for each prediction mode is calculated by Eq. 4.2.

$$J_m = D_m + \lambda \times R_m \quad , m = \{inter, intra\}. \tag{4.2}$$

The best mode decision ,m^*, between inter and intra prediction, for this simplified example, is then performed by minimizing the rate-distortion cost:

$$m^* = \arg\min_m J_m \quad , m = \{inter, intra\}. \tag{4.3}$$

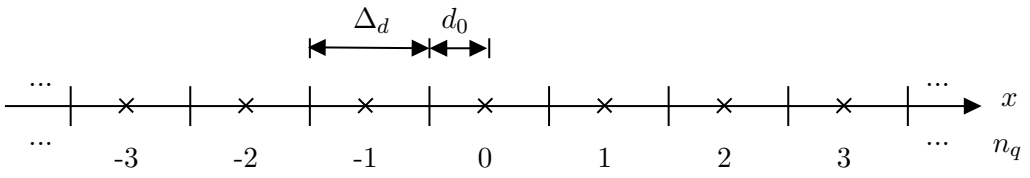


Figure 1-7: Example of a uniform distribution quantizer [2].

The above scheme is called Rate-Distortion Optimization (RDO) and can be extended to all other coding decisions. From the high-level point of view, RDO can be simply considered as a complex nested loop. Inside this loop, as many tests as possible are performed on the provided tools with their different internal configurations and joint combinations. RDO is expected to find the best combination of tools among the ones it has tested in its limited encoding time. In fact, RDO is a particular case of Lagrange optimization applied to lossy signal compression [**?**].

4.2 Coefficient quantization

The quantization is an inherently non-linear lossy process that maps transform coefficient amplitudes to a predefined set of representative values [2]. However, by a proper control of the quantization process within the coded frame and over the coded video sequence, the proportional amount of removed irrelevant information compared to relevant information, can be optimized.

In quantization, an input value x is mapped to the n_q-th reconstruction value representing the interval which the input value x falls into, as show in Figure 1-7. n_q is called the quantizer level. The intervals are scaled to achieve finer or coarser quantization according to the application needs. The transform quantizer scaling is expressed by the quantizer step size Δ_{tq}. In a uniform quantizer, the level n_q for a quantized value x is calculated as:

$$n_q = sgn(x).\lfloor \frac{|x|}{\Delta_d} + d_0 \rfloor, \tag{4.4}$$

where sgn denotes the sign of coefficient and d_0 is an offset which determines the position of interval boundaries, as shown in Figure 1-7 . Finally, Δ_d value is calculated as:

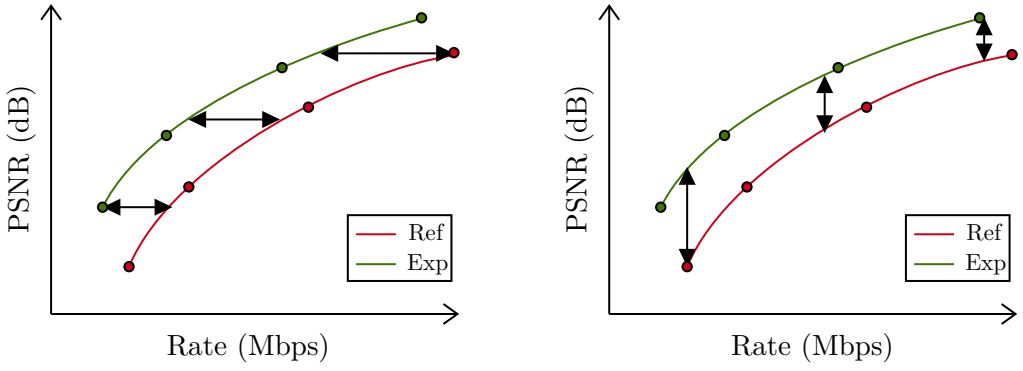
$$\Delta_d = 2 \times \sqrt{\lambda}. \tag{4.5}$$

5 Encoder performance with Bjontegaard Delta (BD)

The syntax specified by integrated tools in an encoder defines its identity. In other words, two encoders that produce different syntax are considered as different encoders and supposedly have dissimilar performance. Performance measurement of different encoders is a critical task. Similar to the block level decision performed by the rate-distortion cost, two main elements are involved in the encoder performance on a given video signal: 1) the transmission rate which is actually the number of written bits in the bitstream and 2) the reconstruction quality which is measured by the Peak-Signal-to-Noise-Rate (PSNR) measure as:

$$PSNR = 10\log_{10}(\frac{MAX_D^2}{MSE_{O,D}}), \tag{5.1}$$

In Eq. 5.1, MAX_D is the upper bound of error (i.e. for the bit-depth of b, equals to $2^b - 1$) and MSE is the mean squared error of compression and is calculated by:



a) BD-Rate b) BD-PSNR

Figure 1-8: Interpretation of BD-R and BD-PSNR from an RD-curve. The curves Exp and Ref indicate an experiment and a reference encoders, respectively.

$$MSE_{O,D} = \frac{1}{W \times H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} [O(x,y) - D(x,y)]^2. \tag{5.2}$$

Similar to the rate-distortion cost problem, the rate and the PSNR are not able to solely quantify the encoder performance. Therefore, different measurement tools are proposed to jointly consider the rate and the PSNR of an encoder and output a single value as its performance index. One important use of such measurement is to evaluate the impact of addition or removal of a new tool to a codec. In this case, one can quantify the encoder performance with or without the specific tool and decide about its existence. This can also be applied when the difference between the two encoders is more than one single tool.

Two encoder performance evaluations are common in the literature: the Bjontegaard Delta Rate (BD-R) and the Bjontegaard Delta PSNR (BD-PSNR) [86]. One outcome of these measures is a plot of rate against PSNR, known as RD-curve. The main steps of BD-R and BD-PSNR computation are:

1. Running the encoder under study on 4 different values of QP=22, 27, 32 and 37 and plotting a curve

2. Based on these four points, finding an expression for the integral of the curve.

3. Computing the average difference between the integrals divided by the integration interval.

Applying the third step on the rate and PSNR axes provides BD-R and BD-PSNR, respectively. Figure 1-8 shows the visual interpretation of BD-R and BD-PSNR, between RD-curves of a reference and an experimental encoder.

It is well known that HEVC standard saves 50% rate against to AVC (i.e. BD-R). The same objective is also set for VVC agains HEVC. These measurements can now be tested with the given information. For this purpose, reference softwares of these three standards have been used for encoding two sequences following the BD instruction. Figure 1-9 shows RD-curves corresponding to this test. As can be seen, a BD-R gain of about 50% has already been achieved for HEVC against AVC. However, VVC performance against HEVC in terms of BD-R gain is about 25% still needs to be improved.

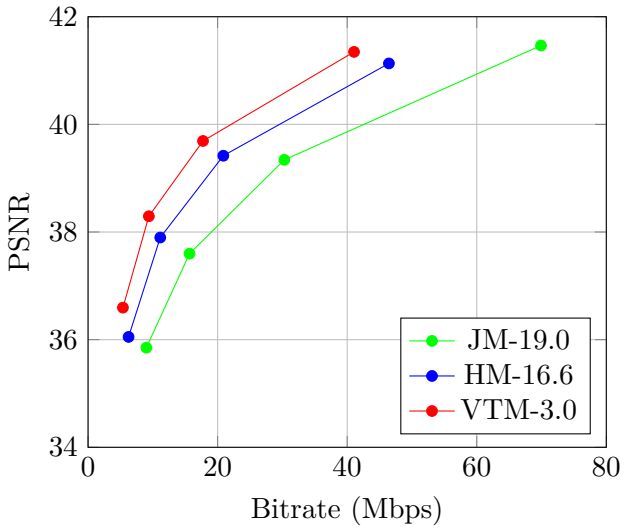


Figure 1-9: Performance comparison of VVC, HEVC and AVC, computed from their latest reference softwares in All Intra (AI) mode: Joint Model (JM) for AVC, HEVC test Model (HM) for HEVC and Versatile Test Model (VTM) for VVC. All tests have been carried out on the BasketballDrive sequence with the resolution of 1080×1920.

6 Conclusion

This chapter has presented the principles of a hybrid and block-based video coding system. Such system deals with signals that carry a lot of of redundancy. To provide a clearer view of the main modules, a historical narrative has been used to compare similar modules in the three standards of AVC, HEVC and VVC.

Block partitioning plays an important role in achieving a desired video compression efficiency. Throughout the time, different ideas for block partitioning have been adopted. These ideas are designed to adapt with the video format requirements as well as implementation restrictions.

Prediction methods help in modeling a correlation with a few number of parameters. Depending on the content, two types of correlations have been defined: temporal and spatial. In each domain, different tools and technologies have been developed in the past 30 years to properly decorrelate a signal.

The coding of prediction residual is the only lossy module of a video codec. This module benefits from signal properties and applies different transformations on it in order to represent it with a limited number of coefficients. This representation of the signal provides the lossy residual coding with the opportunity to throw away parts of the signal which less damages the quality.

In the next chapter, more details about the coding technologies in the intra coding and transform coefficient coding will be presented. This chapter will provide a background for the algorithms proposed in Part II and Part III of this thesis.

CHAPTER 2

Intra coding

Contents

1 Introduction

The use of intra frames is critical in video coding. One important application of intra coding is to provide random-accessibility in the time axis [6]. For instance, assume a 2-hour compressed video sequence that uses only inter frames, except for the first frame that has to be intra coded. A viewer wants to skip the first half of the sequence and start watching from the beginning of the second part. Without regular access points intra frames, the viewer has to decode the entire first hour of the video file in order to start watching it from there. Intra frames can provide random-accessibility due to their temporal decoding independence from other frames.

Another important role of intra coding is error propagation control [7]. For instance, in the previous example, if an error happened in a frame (e.g. due to a packet loss in transmission network), the whole video after that frame will be erroneous and probably unwatchable. This issue can also be avoided either by putting intra frames to completely refresh the sequence, or by regularly putting intra blocks for a partial refresh.

The performance of intra coding plays an important role in the compression efficiency of entire sequence. In other words, an inefficient intra coding significantly limits the entire coding performance of the sequence. This is due to the fact that intra frames usually take significantly higher rate for the same level of quality, compared to inter frames. For instance, Figure 2-1 shows the number of bits used for coding the first 16 frames from a 1080×1920 sequence, in 2 QP values. In both plots of this figure, the first bar corresponds to the single intra frame of the group, while all others correspond to inter frames. As can be seen, in both QPs, the intra frames are responsible for most of the rate. This means that any contribution with a coding improvement capacity of intra frames would have a noticeable impact on the coding performance of entire sequence,

The reconstruction accuracy of intra frames has a direct impact on the coding performance of inter frames. This is due to the high dependency of inter frames on temporal references, which eventually end up in using intra frames as reference. Therefore, having high fidelity intra frames (e.g. in terms of PSNR), would also improve the efficiency of inter prediction.

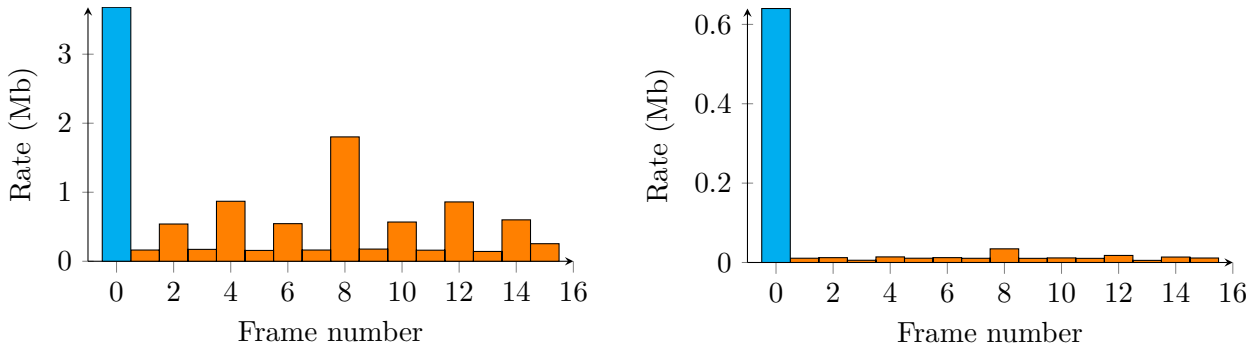
Last but not least, intra coding has an important role in video post-production applications. In post-production, it is highly preferred to have a zero level of temporal dependency in order to provide access to all pixels in all frames with a minimum decoding effort. This requirement is usually met by using All Intra (AI) coding mode. In AI mode, one can access any arbitrary frame within a sequence, edit it, and finally re-encode it, without having to involve other frames. Such property not only facilitates the post-processing tasks, but also avoids unnecessary re-compression of unchanged frames, which is extremely harmful for the visual quality of the video [8].

Now that the necessity of efficient intra coding is clarified, the rest of this chapter describes the principles of two domains: intra prediction and its residual coding. For this purpose, the state-of-the-art algorithms are explained in both domains. Moreover, advanced coding techniques under study by the VVC standardization committee are discussed. Finally, a brief discussion about the contributions of this thesis in each domain is presented.

2 Texture modeling in VVC

A strong correlation usually exists in natural textures. Intra coding aims at removing this type of correlation. The main idea is to represent regular patterns of an image in a less expensive manner than transmitting all texture pixels. Such representation is known to the decoder as a pre-defined model and enables it to reproduce the pattern with a few model parameters, transmitted by the encoder [9].

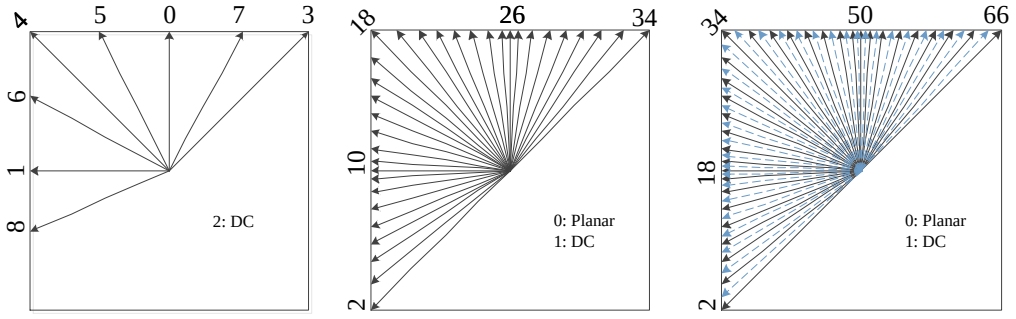
In modern video coding standards, spatial decorrelation of image is achieved by defining flexible texture modeling to allow generating a diverse set of angular and uniform patterns. These



(a) QP 22 (b) QP 37

Figure 2-1: The comparison between the rate of inter and intra frames in the first 16 frames of BQTerrace sequence (1080×1920), in two QP values.

Figure 2-2: The evolution of IPM sets in the three recent video coding standards of AVC, HEVC and VVC.



(a) AVC (b) HEVC (c) VVC

patterns are called Intra Prediction Modes (IPM) and the entire texture modeling algorithm using an IPM set is called intra coding. Generally, an efficient IPM set must be large enough to allow model specification for blocks with both angular and plain texture.

IPM sets vary in different standards. In AVC, a total of 9 IPMs, including 8 angular modes and one plain mode, called DC, were used [10]. The choice of having a reasonably small IPM set has been made mostly due to the software and hardware complexity restrictions at the time. In HEVC, this set was extended from 9 to 35 IPMs, including 33 angular modes, DC, and a new plain mode, called Planar. This improvement brought a huge compression gain to intra coding domain, especially for the new video formats and resolutions such as HD and UHD. In VVC, intra prediction accuracy has been further improved with a set of 67 IPMs. This set includes 65 angular modes, along with the same DC and Planar modes as HEVC. Figure 2-2 visualizes the evolution of the IPM sets from AVC to HEVC and VVC. For a better comparison between the IPM sets of HEVC and VVC, the extended angular IPMs of VVC are shown with dotted blue lines.

Nowadays, the intra coding module of modern video coding standards is considered as a powerful competitor to traditional still image compression standards [11, 12]. Therefore, a still image compression standard based on the state-of-the-art coding technologies of HEVC is currently under development by MPEG. This standard is called High Efficiency Image File Format

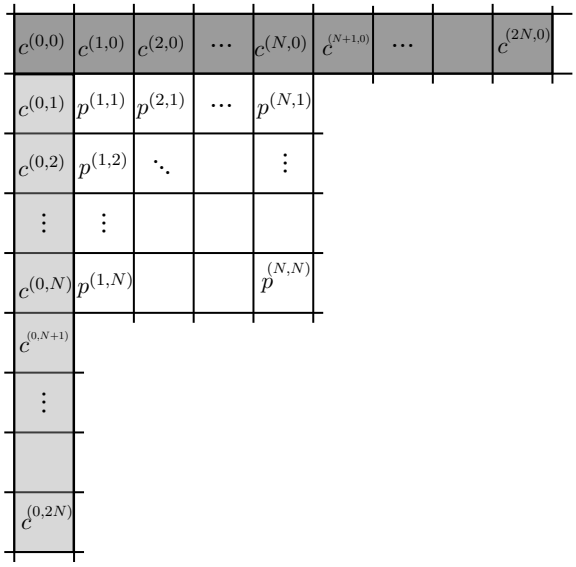


Figure 2-3: Reference pixels $c^{(x,y)}$ used to obtain predicted pixel $p^{(x,y)}$ for a block of size $N \times N$.

(HEIF) and claims that can store significantly more information than the Joint Photograpy Expert Group (JPEG) format in the same file size [13, 14].

2.1 Intra block prediction

All IPMs in VVC use the same basic set of references for prediction, selected from blocks above and to the left of the target to predict. In this section, these references are denoted by C and are indexed as $c^{(x,y)}$, with (x, y) having its origin one pixel above and to the left of block. Similarly, P denotes the prediction signal and is indexed as $p^{(x,y)}$ at position (x, y), as shown in Figure 2-3. As can be seen, in addition to left and above, pixels at above-right and below-left samples ($c^{(N+1,0)..c^{(2N,0)}}$ and $c^{(0,N+1)}..c^{(0,2N)}$) are also used as prediction references.

Angular intra prediction in VVC aims at modeling different directional structures, typically presented in video and image content. For this purpose, IPM number 18 and 50 are defined as the pure horizontal and pure vertical modes, respectively. Other angular modes (i.e. 2 to 66) are correspondingly divided into horizontal modes (i.e. 2 to 33) and vertical modes (i.e. 34 to 66). Depending on the orientation of IPMs, the angular intra filter is performed by extrapolating reference pixels in C. More precisely, the horizontal IPMs utilize left column and the vertical IPMs utilize the above row. According to the selected direction, each predicted sample $p^{(x,y)}$ is calculated by projecting its location to a sub-pixel position between two reference pixels. A linear interpolation is then performed at 1/32 pixel accuracy as:

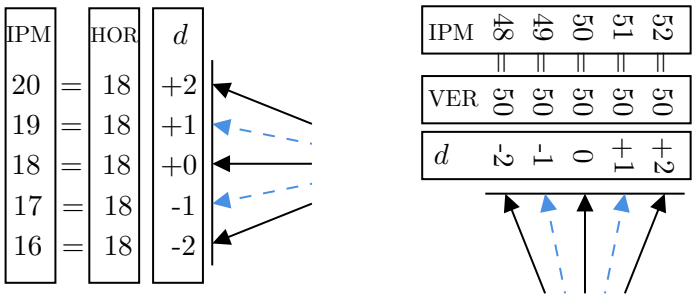
$$p^{(x,y)} = ((32 - \omega_y).c^{(i,0)} + \omega_y.c^{(i+1,0)} + 16) >> 5, \tag{2.1}$$

where ω_y is the weight corresponding to two references surrounding the projected sub-pixel position, namely $c^{(i,0)}$ and $c^{(i+1,0)}$. Moreover, $>>$ denotes a bit shift operation to right. In this equation, the reference sample at (i) and the weighting parameter ω_y are computed using the projection displacement d which is associated to the angle of the selected IPM. This projection displacement describes the tangent of the prediction direction in units of 1/32 samples and can have values between -32 and +32, as shown in Figure 2-4. Finally, ω_y and i are calculated as:

$$\omega_y = (y.d)\&31. \tag{2.2}$$

and:

$$i = x + ((y.d) >> 5), \tag{2.3}$$



(a) Horizontal (b) Vertical

Figure 2-4: Intra projection displacement d, around pure horizontal (18) and pure vertical (50) in VVC. Dotted blue lines correspond to the extended IPM representation in Figure 2-2

where & denotes a bit-wise AND operation [9].

Not all texture contents fit an edge modeling such as the one explained above. Therefore, the DC and Planar modes are used for this aim. The DC mode simply copies the arithmetic mean of reference pixels for all values in P. On the other hand, the Planar mode implements a more complex model for continuous changes in intensity level by using the average of two linear predictions:

$$p^{(x,y)} = (p_V^{(x,y)} + p_H^{(x,y)} + N) >> (log_2(N) + 1), \tag{2.4}$$

where $p_V^{(x,y)}$ and $p_H^{(x,y)}$ are the vertical and horizontal linear models and calculated as:

$$p_V^{(x,y)} = (N - y).c^{(x,0)} + y.c^{(0,N+1)}, \tag{2.5}$$

and:

$$p_H^{(x,y)} = (N - x).c^{(0,y)} + x.c^{(N+1,0)}. \tag{2.6}$$

The above methodologies provide a potentially efficient texture modeling tool. However, its proper use at the encoder side requires a special attention, which is discussed below.

2.2 Fast IPM selection

The additional model flexibility brought by the 67 IPMs of VVC compared to HEVC, imposes more tests to find the optimal IPM. Ideally, such IPM can be determined by a brute-force search over all IPMs in the set for each block. However, this process is usually too complex to be implemented in practice. Therefore, non-normative fast IPM selection algorithms have been proposed. Here, we explain the algorithm used for fast IPM decision in the VVC reference software, the VVC Test Model (VTM).

The IPM search complexity is reduced in two ways: 1) testing less modes, and 2) performing simpler operations for each tested mode. Figure 2-5 shows the reference algorithm for fast IPM selection in VTM. For the mode reduction in this algorithm, the first pass is performed on available IPMs in half precision (i.e. solid black lines in Figure 2-2), plus DC and Planar modes. In this pass, which is called fast mode check, L best IPM candidates are selected and sent to the second pass. In the second pass, called extended angular check, each angular mode m among the candidates are further refined by testing $m \pm 1$ (i.e. among dotted blue lines in Figure 2-2). After testing each mode, the candidate list is updated with respect to cost of tested mode. In both passes, an estimation of rate-distortion cost, based on the Hadamar transform and rate estimation, is used [15]. This estimation consists of simpler operations and provides a reliable correlation with the actual rate-distortion cost. Finally, the third pass, called full RD check,

performs a full rate-distortion cost calculation on the L candidate modes in the list. The mode with the least rate-distortion cost is then selected as the IPM of the block.

Non-normative nature of the IPM selection allows different implementations with arbitrary compromises between performance and complexity. For instance, an encoder which is designed to operate offline and on a powerful machine, would probably prefer to avoid any simplification and perform the full rate-distortion cost on the entire IPM set. In an experiment in this project, the VTM reference software has been modified to perform this test, which resulted in gaining about 1% compression efficiency with the cost of 250% encoder time complexity.

2.3 IPM coding

The Probability Distribution Function (PDF) corresponding to the IPMs random variable is far from uniform, both locally in a part of an image and globally among all sequences. One example of the local concentration of a specific IPM is a textured background in an image, which is split into several neighboring blocks with a similar IPM. In this situation, there is a high correlation in the syntax elements representing that IPM and can be be exploited for further compression by the encoder.

Moreover, the general PDF of IPMs tends to have an extremely non-uniform shape. Figure 2-6 shows the histogram of IPMs in HEVC and VVC. As can be seen, the first two IPMs, corresponding to Planar and DC modes, respectively, have the highest probability (i.e. IPM numbers 0 and 1 in both diagrams). The third and fourth tallest spikes in the middle of the histogram correspond to the horizontal and vertical modes, respectively (i.e. IPM number 10 and 26 in HEVC and 18 and 50 in VVC). This behavior was expected as most of the angular textures in natural contents are either horizontal or vertical. Finally, the fifth, sixth and seventh most popular IPMs are roughly the numbers 2, 18 and 33 in HEVC and 2, 34 and 66 in VVC. These modes respectively represent the three diagonal angles of south-west, north-west and north-east.

One way to decorrelate the IPM information, for further compression, is to use direct binary codes (i.e. 7 bits for binarization of 67 IPMs in VVC) with a full CABAC context model tree. Theoretically, this would guarantee reaching the entropy rate of the IPMs. However, since this approach requires the unfeasible number of 127 CABAC contexts, it is not practical.

A practical algorithm that has been used since HEVC, is the Most Probable Mode (MPM) list construction [16, 17]. Here, we focus on the IPM coding scheme of VVC, at the time of writing this document. The goal is to accommodate the increased number of modes and for this purpose an IPM coding method with 6 MPMs is used [18]. Two major technical aspects are involved: 1) the derivation of 6 MPMs in the list, and 2) the entropy coding of MPMs and non-MPM modes (i.e. all IPMs that are not in the MPM list). VVC puts IPMs in the MPM list from one the following three groups:

- Neighboring IPMs, which are observed in the neighborhood of the block.
- Derived IPMs, which are obtained by applying ± on the neighboring IPMs.
- Default IPMs, which are considered as the most popular IPMs in general.

Figure 2-7 shows five blocks in the local neighborhood of the current block at left (L), above (A), above-left (AL), above-right (AR) and bottom-left (BL). These blocks are the main constructors of an MPM list. To start, an initial MPM list is formed by including 5 neighboring intra modes as well as Planar and DC modes. A pruning process is then used to remove duplicated modes. The order in which the initial modes are included is: 1) L, 2) A, 3) Planar, 4) DC, 5) BL, 6) AR, and finally 7) AL.

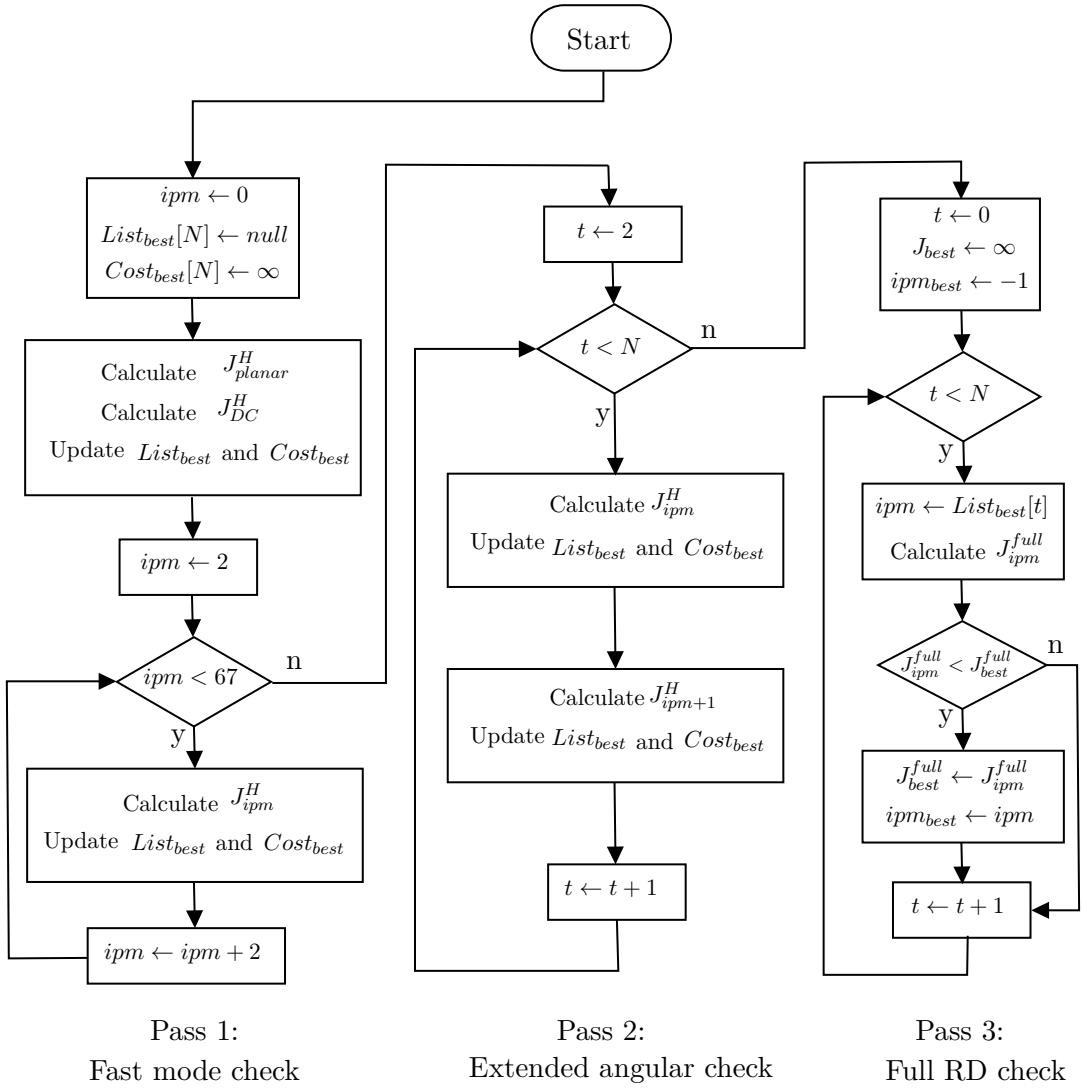
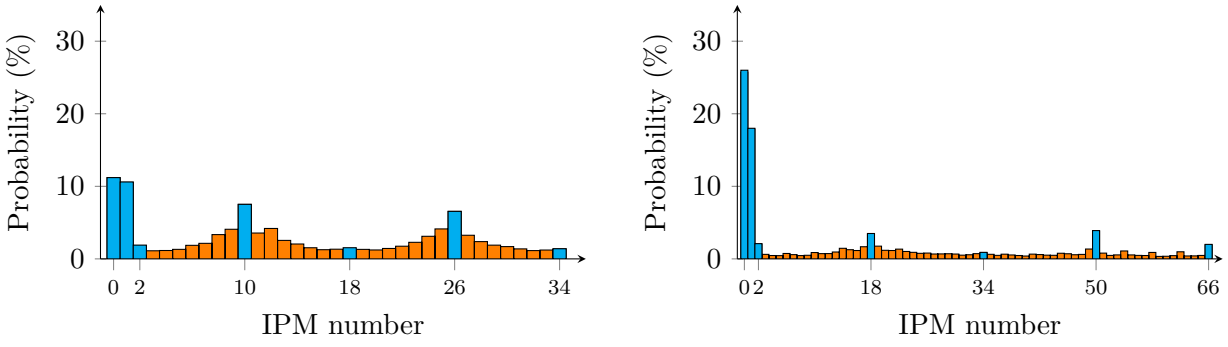


Figure 2-5: Fast IPM selection algorithm of VVC, consisting of three passes. In this diagram, J denotes rate-distortion cost and this can be calculated either as an estimation with Hadamar transform (J^H), or fully (J^{full}). Moreover, *List* and *Cost* store the number of and the cost of best IPMs, respectively. The output of this algorithm is selected in the third pass as ipm_{best} and has the lowest full rate-distortion cost among the candidate IPMs in *List*.



(a) HEVC (b) VVC

Figure 2-6: Histograms of the IPMs in HEVC and VVC. Blue bars indicate popular modes.

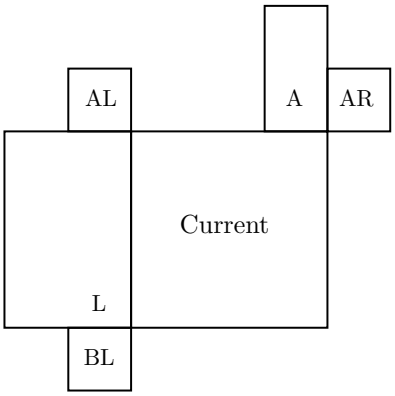


Figure 2-7: Neighboring blocks at left (L), above (A), above-left (AL), above-right (AR) and bottom-left (BL) for the MPM list construction.

After applying the above steps and if there are still less than 6 MPM candidates in the list, then derived IPMs are added. These IPMs are obtained by adding ±1 only to the angular IPMs (i.e. >1) that are already added to the list.

Finally, if the list is still not complete, a set of default IPMs are added, if they do not already exist in the list. The order in which the default modes are checked is as follows: vertical (50), horizontal (18), the south-west diagonal (2) and the north-west diagonal mode (34). This process guarantees a unique list of 6 IPMs in the MPM list:

A truncated unary binarization scheme is used for entropy coding of the current mode using an MPM list. The first three bins are coded with CABAC contexts that depend on the MPM mode related to the bin currently being signaled. The MPM mode is classified into one of three categories and accordingly, three contexts are used to signal the MPM index based on this classification.

- The modes that are predominantly horizontal (i.e. the MPM mode number is less than or equal to the mode number for the diagonal direction 34),
- The modes that are predominantly vertical (i.e. the MPM mode is greater than the mode number for the diagonal direction 34),
- The the non-angular modes (DC and Planar).

In case that the chosen IPM does not appear in the final MPM list, a list of 61 non-MPMs is used to code the index to the chosen IPM. This process is performed as follows. The non-MPM list is first divided into two sets: 1) selected modes set, and 2) non-selected modes set. This division is hard-coded in the algorithm and is independent from block specifications. The selected modes set contains 16 modes and the rest (45 modes) are assigned to the non-selected modes set. The mode set that the current IPM belongs to, is signaled with a flag. If the mode is within the selected modes set, the index is signaled with a fixed-length code of 4-bits, and if it is within the non-selected set, the index is signaled with a truncated binary code.

3 Advanced tools in VVC

In this section, some of the notable intra coding tools of VVC are presented. These tools are mostly VVC-specific and did not exist, at least in their current form, during the standardization of HEVC.

3.1 4-tap filters

In HEVC, a 2-tap linear interpolation filter was used to project reference pixels into intra prediction block in the directional prediction modes. In VVC, two types of 4-tap intra interpolation filters are used: 1) cubic interpolation filters for blocks with size smaller than or equal to 64 samples, and 2) Gaussian interpolation filters for block with size larger than 64 samples. The filter parameters are fixed depending on block size, and the same filter is used for all predicted samples, in all directional modes [18, 19].

3.2 Boundary prediction

Boundary filter prediction is a post-prediction process for certain IPMs. In HEVC, this method is only applied on vertical and horizontal modes, for which the left-most column or above-most row of prediction samples are further adjusted, respectively [20–22]. In VVC, this method is further extended to several diagonal intra modes, and boundary samples up to four columns or rows are further adjusted using a 2-tap or 3-tap filter, depending on the IPM [18, 19].

3.3 Position Dependent intra Prediction Combination (PDPC)

The Position Dependent Prediction Combination (PDPC) algorithm proposes a set of recursive 2-D filters for intra prediction [23]. The idea of recursive filters for intra prediction has always been a promising, yet unfeasible approach, in terms of complexity. The main contribution of this algorithm is to reduce the complexity and to propose a reasonably simple method. Moreover, another interesting feature of PDPC is its parallel-friendliness.

In PDPC, predicted values by different IPMs are combined using reference samples. The model parameters of PDPC are depending on block information such as IPM and size. Furthermore, the combination model of PDPC is conservative about radical changes and tries to avoid them. All specifications of current PDPC algorithm, including the parameter sets, have been determined through a training procedure.

PDPC has currently been adopted by VVC. Experiments show that about 1% of the compression gain of VVC comes from PDPC, with a negligible complexity overhead.

3.4 Cross-component linear model prediction

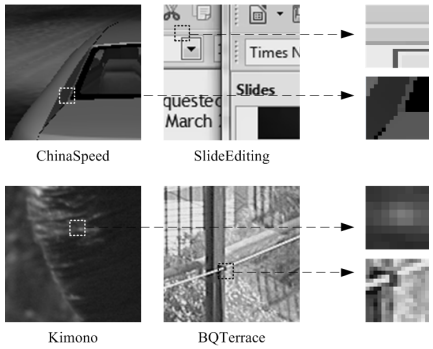
The early cross-component prediction methods were initially proposed during the development of HEVC Range Extention (REext) [24]. The main idea was to combine the block residual information from different components (i.e. Y, Cb and Cr) and to operate adaptively on the spatial or temporal prediction of residuals using a linear model [25]. In VVC, in order to reduce the cross-component redundancy, a Cross-Component Linear Model (CCLM) prediction mode is used for which the chroma samples are predicted based on the reconstructed luma samples of the same block by using a linear model.

The CCLM luma-to-chroma prediction mode is added as an additional chroma prediction mode to VVC [18, 19]. At the encoder side, one more iteration of RDO is performed for each chroma components block to calculate the rate-distortion cost of CCLM, during intra prediction. When intra prediction modes other than the CCLM luma-to-chroma prediction mode is used for the chroma components of a block, CCLM will be performed as an intra-component model for Cb-to-Cr prediction.

4 Distant reference inaccuracy problem

Although the new intra coding system of VVC, including all the new tools, significantly improve that of HEVC, it is still unable to properly code certain types of texture. Specifically, this poor performance is evident in high detail textures, where reference pixels usually have a low

Figure 2-8: DRI problem examples from natural and screen contents.



correlation to in-block pixels. As discussed, the first impact of such low correlation in block-based video codecs is that the encoder would have to further split the block. However, deeper block partitioning always comes with the cost of transmitting extra block level syntax elements. Even more problematic, this issue would cause further rate cost when the encoder reaches its smallest block size (i.e. 4×4) and is still unable to accurately predict the block. In this situation, the low prediction accuracy would result in a relatively large residual, which is expensive to transmit. This problem is referred to as the Distant Reference Inaccuracy (DRI) in the rest of this manuscript. Subsequently, a block that has a relatively high level of texture complexity and is more likely to have low correlated references, is called a DRI block. Figure 2-8 shows some examples of DRI blocks from different video sequences.

The DRI problem in the lossy intra prediction has been sparsely studied in the literature. Here, we briefly review some notable researches that have tried to address the problem.

4.1 Existing solutions

In this section, some of the existing intra coding technologies are briefly introduced. These technologies are somehow related to the problem definition of this thesis.

Short Distance Intra Prediction (SDIP)

The Short Distance Intra Prediction (SDIP) was first proposed to JCT-VC in 2011 [26]. This algorithm tries to reduce the distance between reference pixels and predicted pixels, in order to address a similar problem to DRI.

In SDIP, the prediction remains at the CU level by forcing a split into thin rectangular blocks, expecting that the small height/width of the block solves DRI problem. The partitioning by SDIP can be either horizontal or vertical. Figure 2-9 shows an example of partitioning by SDIP on top of the quadtree partitioning of HEVC.

In SDIP, square blocks smaller than 32×32 can be divided into several lines or non-square blocks with rectangle shape. In each block, pixels are predicted and reconstructed line by line or rectangle by rectangle. For residual coding, the existing transform matrices (i.e. 2×2, 4×4, 8×8 and 16×16) are reused, with an additional normalization for the quantization step to adapt the actual transform size. At the entropy coding stage, for 2×8, 1×16 and 4×16 partitions, the coefficients are first scanned into a 1D buffer and then reorganized into a 4×4 or 8×8 block and then entropy coded [27]. The initial version of this algorithm had a reasonable compression gain, when implemented on the HM.

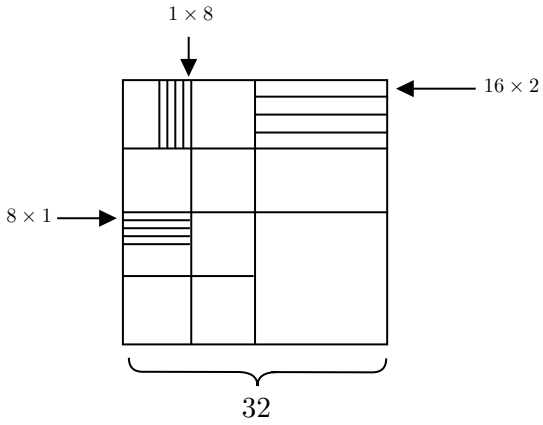


Figure 2-9: Example SDIP block partitioning.

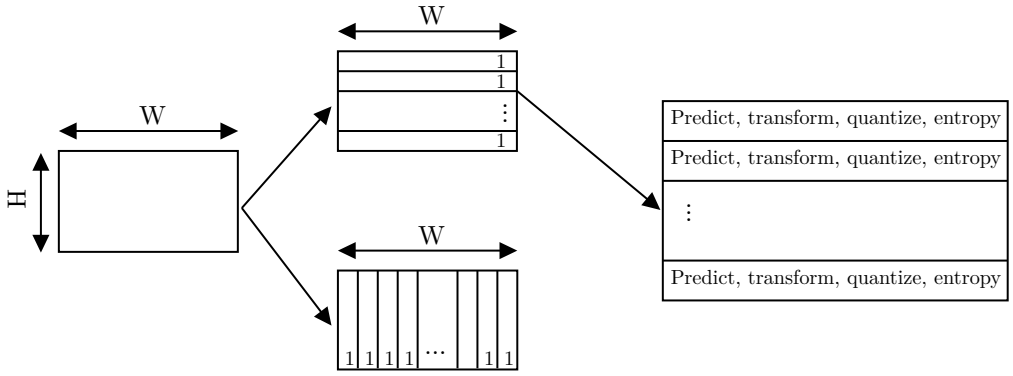


Figure 2-10: Example of horizontal and vertical partitioning by LIP.

Line-based Intra Prediction (LIP)

This method is very similar to SDIP and is currently under consideration for VVC standardization. LIP mode divides luma intra-predicted blocks into 1-D partitions. Hence, a W×H dimensional block can be partitioned either into H rows of size W×1 or W columns of size 1×H, as presented in Figure 2-10. This mode can be applied to any available block size in VVC [28, 29].

As can be seen in Figure 2-10, residual calculation and coding by LIP is carried out on each line separately. For this purpose, the full chain consisting of prediction, transformation, quantization and entropy coding is kept independent for lines. For transformation, LIP utilizes 1-D transform cores and can benefit from multiple transform tools of VVC [30, 31].

Experimental results on implementing LIP on top of VTM show a compression gain on different sequences. On natural content, this gain is between 0.5% and 1%, while on screen content, it brings between 3% and 5% gain. In both cases, LIP introduces a noticeable encoder side complexity.

Combined Intra Prediction (CIP)

CIP is another algorithm attempting to address DRI problem. This algorithm performs an additional inner-block pixel-level prediction and combines it with the traditional prediction using a weighted average [32–34].

The technology of CIP was a part of the response to the Call for Proposals of HEVC from BBC [35]. Its algorithm consists of a weighted combination of a closed-loop angular prediction

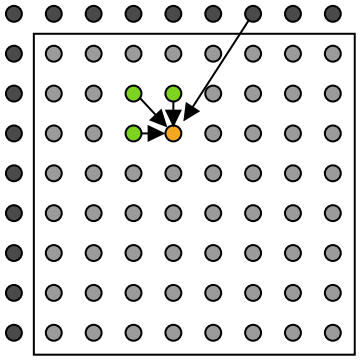


Figure 2-11: Example of CIP utilizing a local mean and angular prediction

together with an open-loop local mean prediction. The local mean is obtained as the average of neighboring pixels. The selected neighbors are defined using a pre-defined set of templates. An example of this prediction combination is presented in Figure 2-11. In this figure, the template utilizes three in-block reference pixels at left, above-left and above of the current one.

At the decoder side, the values in the template are the reconstructed values after the inverse quantization and transform. On the other hand, the encoder side uses original pixels of the template. This is due to the fact that at the encoder side, the reconstructed values are not available. since the quantization and transform of the prediction residue have not yet been performed.

The main problem with CIP methodology is the mismatch due to the use of original pixels at the encoder side. This choice makes the decoder use a different set of references in the template, as it does not have access to the original template. Despite the fact that this mismatch is partially compensated after addition of residual, it still limits performance of CIP.

DPCM-based prediction solution

In this algorithm which is presented by Gao *et al.*, the use of a DPCM-based pixel-level prediction is proposed for addressing the DRI problem [36]. This algorithm uses in-block pixels for intra prediction. The reconstruction step in order to prepare the in-block references is proposed to be performed by the regular decoded residual.

The CIP algorithm and the proposed DPCM-based prediction of this research both share a similar problem definition. This problem is the use of in-block pixels as reference. As discussed, the CIP proposes the use of original pixels, with the cost of a reconstruction mismatch, while the DPCM-based algorithm actually reconstructs in-block references. However, since the residual calculation for reconstruction must be performed outside of the prediction loop, the DPCM-based algorithm introduces another type of mismatch. This mismatch, which is related to the reconstruction accuracy of in-block pixels, will be discussed extensively in the next chapter.

Intra Block Copy (IBC)

IBC is one of the most successful algorithms from the family of Current Picture Referencing (CPR) [37]. In CPR algorithms, various types of block matching algorithms are applied using the same frame as reference [38]. This is in contrast with the conventional block matching algorithms, most notably ME, where the reference frame is shifted on the time axis.

Unlike the previous methods, the DRI problem is targeted indirectly by IBC. In fact, IBC puts its specific focus on text compression of screen content coding. Hence, it has been adopted in an HEVC extension, called Screen Content Coding (SCC) [39]. In this method, a spatially restricted area is defined around each block and a search is performed to find the most similar

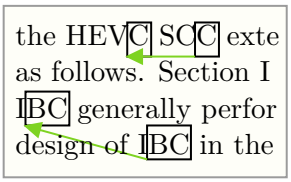


Figure 2-12: Example of using IBC on a screen content containing pure text.

block, in that area, to the current block. Similar to ME, the non-normative search process can use a distortion-based measure (e.g. Sum of Squared Error (SSE), Sum of Absolute Difference (SAD) etc.) in order to find the best displacement vector. This vector is transmitted to act as MV at the decoder side, by copying the found block in the search area. Figure 2-12 shows an example of finding the displacement vector for two blocks containing pure text [40].

Experimental results show that IBC has an outstanding performance on screen content coding, in term of BD-R. However, its complexity is excessively large [41]. This complexity comprises both encoder-side run-time of the block matching algorithm and the encoder/decoder-side memory overhead of storing the entire spatial neighborhood. The second problem can be severe particularly when the decoder is implemented on a device which has strict memory constraints.

Palette mode

Palette mode is another tool proposed for texture compression of screen content [42–44]. Similar to IBC, Palette mode has been adopted by HEVC-SCC [39].

Palette mode in HEVC-SCC draft specifications is comprised of three main components: 1) palette table derivation, 2) palette table coding, and 3) palette index coding. In this algorithm, a palette is referred to a set of entries to act as representative pixel values for a block. For a block coded with Palette mode, a palette is first transmitted. Then, the pixel values within the block are represented and transmitted as indexes to their corresponding palette entries.

This tool is currently under consideration to be adopted by VVC. Therefore, different versions of it, bringing various compromises between performance and complexity have been proposed. For instance, in one version of Palette mode, the entry index rate is reduced. To this end, a number of consecutive locations (in the scanning order) that have a similar palette index, are coded once with a run-length code [45].

A significant compression gain can be achieved on screen content, with Palette mode. From the complexity-performance point of view, Palette mode is considered less efficient and less complex than IBC.

4.2 Proposed intra prediction

A framework is proposed to address the DRI problem in Part II of this thesis. The main idea of our proposed intra coding framework is to use in-block pixels rather than conventional out-block ones, as prediction references.

Two algorithms are devised within the proposed in-block intra prediction framework. Similar to the CIP and the DPCM-based algorithms, these algorithms have to deal with a major data dependency challenge for the reconstruction of in-block references. However, both algorithms are integrated with an additional residual signal to remove this dependency. As will be discussed in next part, the main difference between the proposed algorithms is their residual representation and coding.

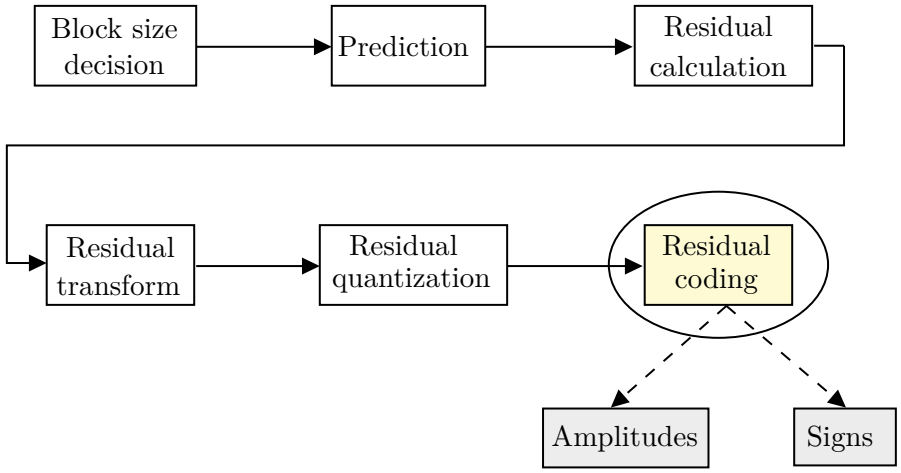


Figure 2-13: The encoder side modules before residual coding.

5 Residual coding

In this section, the state-of-the-art residual coding algorithm used in HEVC and VVC is discussed. Moreover, some related researches either to improve or replace this algorithm are introduced.

5.1 Problem definition

The task of residual coding is typically referred to the encoding and decoding of quantized coefficients in the transform domain [46]. Without loss of generality, here the focus is put on the encoder side only. As shown in Figure 2-13, the corresponding module to residual coding is placed at the end of the encoder chain.

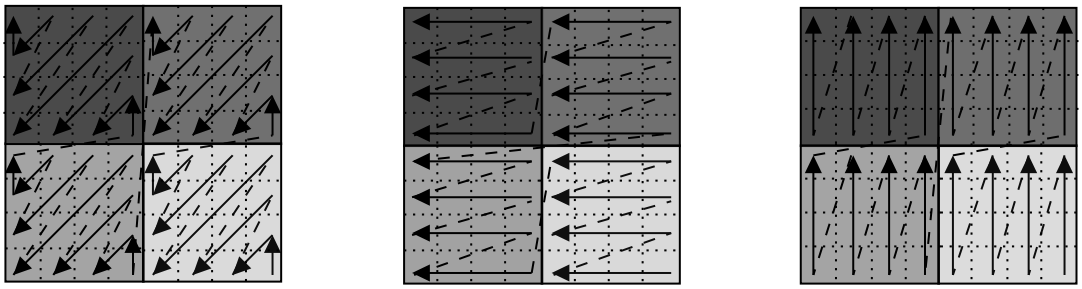
Quantized transform coefficients still carry a large amount redundancies. These redundancies are left from the prediction step and need to be carefully treated for further compression. Therefore, the main objective in residual coding is to first detect redundancies and then integrate tools to remove them.

Residual coding is supposedly a lossless process. This makes it appropriate for applying entropy coding techniques in order to benefit from the statistics of the residual signal. For this purpose, transform blocks are decomposed into amplitudes and signs. As information in the sign part carries almost no redundancies, it is usually bypassed (i.e. coded without compression). Therefore in the rest of this section, the amplitude part of transform blocks is studied and unsigned quantized coefficients to be coded are referred to as amplitude blocks.

5.2 Residual coding of VVC and HEVC

The residual coding algorithm in VVC is very similar to that of HEVC [46]. In this method, once the prediction is finished and the amplitude block is computed, a nested quad-tree structure, called Residual Quad-Tree (RQT), is used [47, 48]. RQT is responsible for further partitioning the amplitude block to be coded separately. After the RQT structure determines an amplitude block and its coding order, a Coded Block Flag (CBF) is associated to it. This flag signals whether any significant (i.e. non-zero) amplitude exists in the block or not. In case there is no significant amplitude, the rest of the residual coding algorithm is skipped. Otherwise, amplitudes in the block are coded one by one.

The amplitude coding of non-zero blocks consists of four steps: 1) scan, 2) last significant amplitude coding, 3) significance map coding, and finally 4) amplitude levels coding [46].



Diagonal Horizontal Vertical

Figure 2-14: Three scan patterns applied both at the CG level and amplitude level, inside an 8×8 amplitude block.

Scan

The scan step of residual coding maps a 2-D block of amplitudes into a 1-D array and defines the process order of its individual amplitudes. This step also defines 4×4 subsets of an amplitude block, called Coefficient Groups (CG). A CG is a set of 16 consecutive amplitudes, corresponding to its 4×4 subset block, in the given scan order. With this technique, if there are more than one CGs in the amplitude block, they are scanned with the given scan pattern. Both a CG and amplitudes inside it, can be scanned either diagonally, horizontally or vertically. Figure 2-14 shows these three scan patterns.

Last significant position

Signaling the position of last significant amplitude usually reduces the number of coded bins. In HEVC and VVC, the (x, y) coordinates of the last position are calculated and coded explicitly. For the last significant position coding, the amplitude block is scanned in the forward direction until the last significant amplitude is visited. Then, the (x, y) coordinates are binarized independently, each with a prefix and a suffix. The prefix represents an index to an interval and has a truncated unary code. The bins in this code are coded in the regular mode (i.e. compressed by CABAC context modeling). The suffix is responsible for providing an offset within the interval that is determined by the prefix. This part uses a fixed length code, since its interval length is known to both encoder and decoder side. Unlike the prefix bins, the suffix bins are coded in bypass mode (i.e. no compression).

Table 2-1 shows the binarization of the last significant position. In this table an amplitude block of size 32 is used, which can be applied similarly to width or height of the block. As can be seen, there are certain prefix values for which no suffix in defined, since their suffix interval is empty. For the rest, a fixed length suffix is denoted with Ys, which indicate how many additional bypassed bins should be coded.

Significance map

There are two means to carry information about non-zero amplitudes: CBF that is already discussed and amplitude significance. In fact, the amplitude significance is signaled just when the CBF is set to one. In this case, a significance map which is interleaved with the last significant position information, is sent. The role of the last significant position is to exclude the significance bit coding of zero amplitudes after the last position.

For a better bin source separation, different CABAC context derivations are used for significance map coding. The factors involved in the context derivation of a bin are usually the frequency band of corresponding amplitude and the CG depth in the RQT structure. As will

Position	Prefix	Suffix	Suffix range
	Truncated unary (regular mode)	Fixed length (bypass mode)	
0	0	-	-
1	10	-	-
2	110	-	-
3	1110	-	-
4-5	11110	Y	0 to 1
6-7	111110	Y	0 to 1
8-11	1111110	YY	0 to 3
12-15	11111110	YY	0 to 3
16-23	111111110	YYY	0 to 7
24-31	1111111110	YYY	0 to 7

Table 2-1: Binarization of last significant position in an amplitude block size of 32. This scheme can be used for both x and y coordinates.

be discussed later, a potential compression gain can be achieved by a more efficient context derivation scheme.

Levels

As soon as the position of all significant amplitudes are coded, their remaining levels are coded. The first pass of this step codes one bin per significant amplitude, in order to signal whether it is "*greater_than_one*" or not. This bin is coded in normal mode. After this pass is run for all significant amplitudes, the second pass is run only on amplitudes with *greater_than_on*=1. Similarly to the first pass, here a bin is coded to signal whether the amplitude is "*greater_than_two*" or not.

The third and last pass codes the remaining levels only for amplitudes with *greater_than_two*=1. This pass specifies the remaining absolute values using Exp-Golomb codes [49, 50]. Bins in these codes are coded in bypass mode in order to increase the throughput.

5.3 Alternative methods for VVC

A handful of residual coding schemes have been introduced in the context of MPEG standardization. These schemes benefit from different characteristics to remove the remaining redundancies in residual signals and provide a wide range of compression gain on different video signals. Zhang *et al.* have evaluated different proposals, that were not adopted by HEVC, for consideration in VVC standardization [51]. The experiments in this research demonstrates that all alternative approaches can bring an acceptable coding performance, where some even achieve up to -1.7% BD-R gain. However, due to complexity concerns, both in hardware and software implementation, they have been excluded from the final standard specification of HEVC.

In this section, four alternative residual coding methods are introduced. These methods are related either to the algorithm that was explained above, or to the contributions of this thesis. The first two methods propose further improvement in the existing algorithm. While the other two present different residual coding methods that perform without the transformation step.

Adaptive scan order by Chiang *et al.*

This work proposes an adaptive approach to generate a scan pattern dynamically [52]. Unlike the existing algorithm in HEVC, that sorts a 2-D array of coefficient positions according to the

frequency appearance of non-zero levels only, this algorithm utilizes a topological sort. This sorting order fully accounts for the spatial constraints due to the context dependency of the entropy coding. The algorithm provides a streamlined scan order design for both inter and intra prediction modes, and for all the transform size and kernel combinations.

The proposed method by Chiang *et al.* combines frame-level coefficient statistics and context dependencies. For this purpose, a draft scan order is first generated by ranking all coefficient positions in descending order of their probabilities. The context dependencies are then translated into a direct acyclic graph. This graph integrates a topological sort to optimize the initial draft order into the final scan order for the frame.

Improved context modeling by Gao *et al.*

The algorithm presented in their work specifically attempts to improve the transform coefficient coding efficiency of HEVC [53]. In the proposed improved context modeling scheme for transform coefficient levels, the model index for the significance map depends on the number of significant neighbors. To limit the total number of context models for the significance map, transform blocks are split into different regions based on the coefficient positions. Regions in different blocks then share a same CABAC context model set.

The proposed algorithm performs only on the first two bins of the truncated unary codes of amplitudes. In particular, the context model index for the first bin is determined by the number of neighbors covered by the local template with an absolute magnitude equal to 1 or larger than 1. Moreover, for the second bin, the context model is determined by the number of neighbors covered by the local template with an absolute magnitude larger than 1 or larger than 2. Experimental results, presented in this work, show a BD-R gain of about 0.8%, 0.6% and 0.4% in AI, RA and LD modes, respectively.

Transform skipping mode (TSM)

The TSM technology has been adopted in HEVC, to specifically target screen content coding [54, 55]. As the integration of TSM in HEVC has showed an outstanding performance, its adoption for VVC is very likely.

The main philosophy of TSM is that the transformation of the residual signal could negatively influence the rate and quality of screen content residual signals. This is due to the existence of sharp and unnatural edges in screen content that make conventional transformations inefficient. In TSM, as the name suggests, residual coding skips the transformation step and adapts the rest of the coding process to the nature of residual signal in the pixel domain.

The adaptation of TSM quantization consists of normalization of the step size to be used in the pixel domain. This normalization depends on the TSM configuration that is used for a residual block. Three configurations are proposed in the main algorithm: 1) vertical transform skipped, 2) horizontal transform skipped and 3) both vertical and horizontal transforms skipped. The experiments demonstrate a significant compression gain on top of HEVC, when applied on screen content.

The above idea of pixel-domain quantization has been widely used in the proposed algorithm of Chapter 5. As will be seen, due to a similar problem definition, the integration of TSM quantizer normalization is very helpful for addressing the main challenge of the proposed algorithm in that chapter.

Residual DPCM

Another residual coding algorithm that works in the absence of transformation is a DPCM-based residual coding, called Residual DPCM (RDPCM). In this method, which is originally proposed for lossless coding, residual values are predicted with predefined predictors [56].

The main idea of RDPCM is to perform a sample-based prediction of residual signal in the pixel domain. This is useful when transform coding of residual is inefficient and fails in rate reduction, which is common in screen content coding. Different proposals based on initial RDPCM idea have implemented its various specifications.

In one of the lossless versions of RDPCM, the residual values are predicted using their contextual information. This information is gathered from neighboring residual values in the pixel-domain and hopefully help at proper edge detection without signaling [56]. The same context adaptive predictor has been been used in two contributions of this thesis, presented in Chapter 4 and Chapter 5.

The lossy version of RDPCM mainly focuses on screen content coding and visually lossless applications (e.g. medical imaging). In these methods, a sample-based DPCM is applied to residual values either in vertical or horizontal direction. Then, the differentially coded residual samples are signalled. At the decoder side, the original residual samples are directly reconstructed from the parsed samples without waiting for the reconstruction of the previous residual sample, so that the throughput problem of the sample-based prediction method can be avoided. [57, 58]

5.4 Residual coding in other standards

Residual coding is not uniquely used in the MPEG family of standards. On the contrary, since almost all modern video coders benefit from spatio-temporal prediction, they have to address similar problems for residual transmission. In this section, two famous video coding standards are briefly discussed from the transform coefficient coding aspect.

Alliance for Open Media (AOM) Video 1 (AV1)

AV1 is an open royalty-free video coding standard and is considered as the primary opponent of VVC. The main application of AV1 is video over the Internet. However, it is supposedly a general-purpose codec and able to code any video signal [59–61].

In a work by Han *et al.*, carried out in the WebM Codec Team of Google, the weak correlation between magnitude of transform coefficient and a number of factors, is exploited [62]. These factors are namely the frequency band, the neighboring coefficients magnitudes and the luma/chroma planes. The proposed method in this work was under consideration for adoption in AV1. However, its methodology is generalizable to other standards, including VVC.

The main objective of this method is to propose an efficient source separation scheme of residual bins. Such scheme would define categories, each of which representing bins with similar statistical behavior. From this aspect, the proposed algorithm of Chapter 8 has a lot in common with this algorithm.

A level-map amplitude coding is proposed to make a compromise between coding efficiency and model size. In this model, the binarization step performs a series of "$greater\_than\_X$" tests on amplitude values and produces its bins with respect to the test outcomes. In case that an amplitude is larger than a pre-defined threshold, the remaining absolute value is coded separately. The rest of the proposed coefficient coding consists of the following stages: 1) significance map and last significant position flag for each significant coefficient, 2) multiple runs on levels to code the bins until the threshold, 3) remaining level after the threshold.

One important contribution of this paper is the definition of the contextual information derivation. This information is used for deciding about the entropy coding context model of each bin. For this purpose, a compact modeling is used to map each contextual situation to a proper context model. Experimental results of this algorithm on top of AV1 shows a noticeable compression gain on a large set of test sequences.

AVS2

Audio-Video coding Standard (AVS) is a series of compression specifications formulated by the AVS workshop in China, according to international rules [63]. Currently, its second generation called AVS2 is available. For residual coding, AVS2 uses a specific entropy coding engine, called Coefficient-base Binary Arithmetic Coding (CBAC). The CBAC design consists of the following modules: 1) scan, 2) Level-Run coding, 3) End-of-block flag coding, and finally 4) sign coding. Three transform block sizes are allowed in AVS2: 8×8, 16×16, and 32×32.

The CBAC design of AVS2 represents transform coefficients as Level-Run pairs. In this design, Level is the magnitude of a nonzero coefficient and Run is the number of consecutive zeros preceding the corresponding non-zero coefficient in the diagonal order. The Level-Run pairs are coded in the reverse diagonal order. Moreover, sign bins are coded in bypass mode.

In a paper by Wang *et al.*, a coefficient group based transform coefficient coding design for AVS2 is proposed [64]. This algorithm includes two main coding tools, where both have been adopted into the AVS2 working draft:

1. Two-level coefficient coding for accurate coefficient position information use in the context model design.

2. Intra-mode based context design for further improving coding performance by utilizing the intra mode information.

Experimental results of the proposed design in this paper show consistent coding gains as well as encoder simplifications on top of AVS2. More precisely, the algorithm is able to significantly reduce the encoding time for low QP values.

5.5 Proposed residual coding

A residual coding framework is proposed in Part III of this thesis. In this framework, a flexible model for deriving contextual information is proposed.

Based on the proposed framework, an initial algorithm is designed for residual coding of intra blocks in VVC. This algorithm benefits from informative local features of a block. These features hopefully help in efficient source separation of residual bins. As will be discussed in detail, the preliminary results, particularly in screen content coding, are promising and highly motivate us to further improve this framework.

6 Conclusion

In this chapter, the main elements of the intra prediction and its residual coding have been discussed. The presented information can provide a background for an easier understanding of the contributions of this thesis.

In the first part of this chapter, different aspects of intra coding have been discussed in detail. A unique property of intra coding is its temporal independence. This interesting feature allows video transmission systems to address challenges such as error propagation through the time, decoding delay of temporal seeking and frame editing in post-production applications. More importantly, the compression efficiency of intra coding proved to be very crucial for entire sequence-level performance. All these have motivated us to design a more efficient intra coding tool.

The state-of-the-art intra prediction is based on texture modeling. By studying the evolution of intra coding algorithms through the history of video coding standards, one can notice that major changes are mostly related to the modeling capacity. Currently, several advanced modeling tools are available to be adopted for VVC. Thanks to these tools, VVC can perform higher flexibility and outperform HEVC in the intra coding domain. However, it is still unable to

properly model certain high detail content. This problem, which is referred to as the DRI in this thesis, is specifically targeted in this research. The main objective is to add an extra intra coding mode to improve the performance in the existence of the DRI problem.

The second part of this chapter has been dedicated to residual coding. For this purpose, first, the existing algorithm adopted in VVC was explained in details. Then, some of the notable researches in this domain have been reviewed. These research works consist of a wide range of algorithms, from VVC-related ones to those of other standards. Moreover, some methods that exclude the transformation step from the residual coding of screen contents, have been discussed.

Part II:
Proposed Intra Prediction Tools

CHAPTER 3

Framework of In-Loop Residual (ILR) coding intra prediction

Contents

1 Introduction

In the literature, most studies regarding intra coding target well-shaped and smooth texture. However, a noticeable portion of video frames usually contain high detail texture, that, as discussed in Chapter 2, causes the DRI problem for prediction. Experiments show that the existing intra coding tools perform significantly worse on DRI blocks, than on well-shaped ones. In terms of rate-distortion behavior, performing the state-of-the-art intra coding algorithm on a DRI block would result in a higher rate-distortion cost than on a block with a simple texture of the same size. This is due to the poor prediction accuracy of the DRI block which produces a high energy residual.

Figure 3-1 visualizes the rate-distortion cost of regions with different textures complexities in the first frame of three sequences. For this test, the best rate-distortion cost of each block from the encoding phase was stored. To assign intensity levels to pixels inside each block, these costs are divided by the number of block pixels and then mapped into the range of 0-255, with respect to the maximum cost in each frame. As can be seen, regions with higher texture complexity have significantly higher rate-distortion cost. This experiment confirms that VVC, as the future standard, shall pay a special attention to the coding of this type of content, as it highly constrain the rate-distortion behavior of the intra coding.

Our proposed framework specifically targets DRI blocks in intra coding. To address the problem, we propose to use closer in-block references instead of regular out-block ones. This would hopefully increase the correlation between references and pixels to be predicted. Such high correlation, if achieved, can improve the prediction accuracy which usually results in less residual bits to transmit.

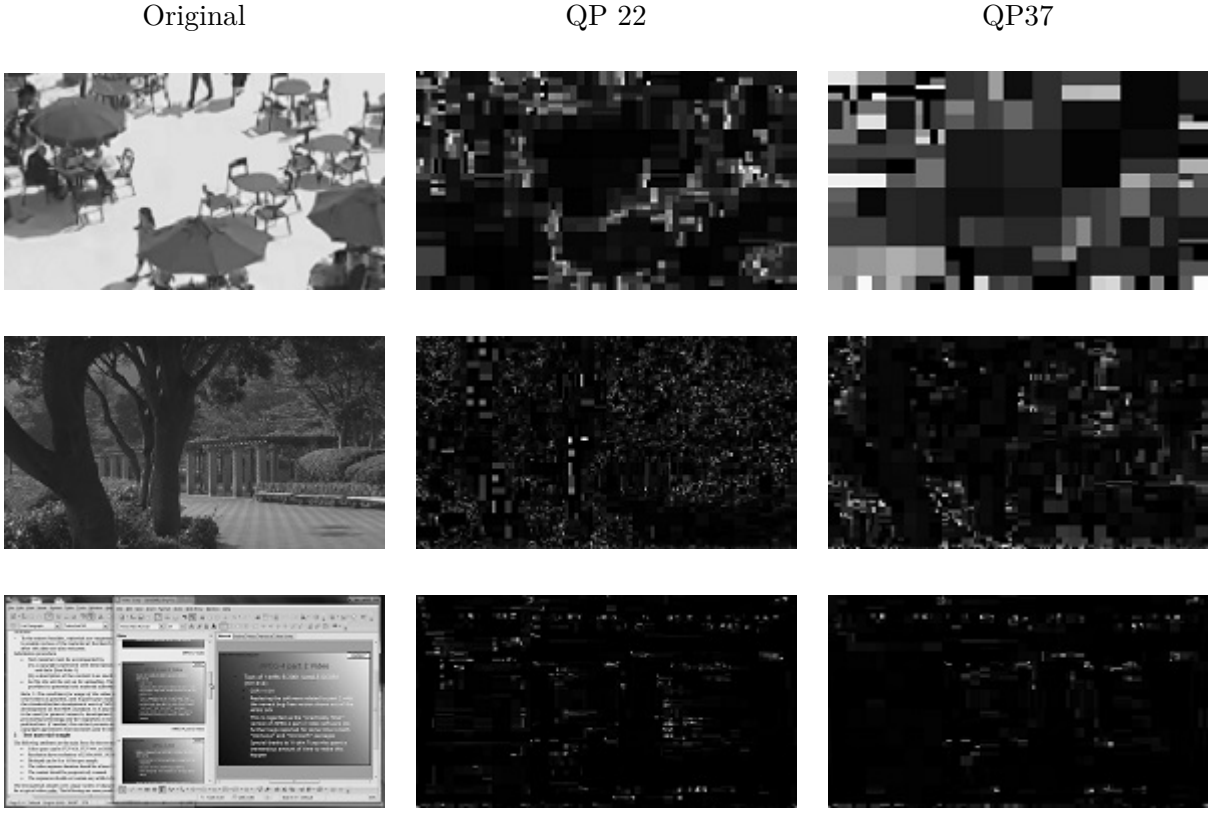


Figure 3-1: The heatmap of rate-distortion cost of intra coding in two QP values of 22 and 37. Brighter pixels mean higher rate-distortion cost.

Table 3-1: Overhead of transmitting a dummy flag per block in terms of BD-R and complexity overhead.

Block size	only 4×4	up to 8×8	up to 16×16	up to 32×32	up to 64×64
BD-R	+1%	+2%	+3%	+4%	+5%
Complexity	+1%	+2%	+3%	+4%	+5%

2 A bi-algorithm intra coding system

Although the existing intra coding scheme operates poorly on DRI blocks, it still has an outstanding rate-distortion performance on smooth blocks. Hence, it is not reasonable to completely replace it with a new algorithm that specifically targets DRI blocks. Although such full replacement may improve the coding efficiency of DRI blocks, it significantly deteriorates the performance on smooth blocks.

The proposed framework integrates a new algorithm with in-block references as an additional mode alongside the existing modes. This design imposes a competition at the encoder side between the two block prediction algorithms with a rate and complexity overhead. An estimation of these two types of overhead is presented in Table 3-1 and discussed below.

2.1 Rate overhead

The rate overhead is due to the necessity of transmitting an extra block-level flag to inform the decoder about the selected intra algorithm. This necessity has always been a big challenge for performance evaluation of new tools. Usually, a new tool is added to a codec in order to target a specific content condition in which the existing tools are inefficient. Such tool may cause a rate-distortion cost reduction, if applied to its desired blocks. However, in all other blocks, it could impose a rate cost overhead, even if not used. This is due to the rate overhead of signaling a flag to inform the fact that the new tool was not used. To understand the approximate impact of transmitting one extra flag per block, Table 3-1 shows the performance drop when a dummy flag is encoded (without compression) at the end of each block. Each column in this table presents a limit on the size of blocks for which the dummy flag is coded. As a reminder, a positive BD-R expresses a rate increase percentage for the same level of quality. The BD-R increase in Table 3-1 confirms the concern about a new coding tool which requires signaling.

In the existing DRI problem, the signaling rate overhead can be more harmful when the number of non DRI blocks (i.e. smooth blocks) is significantly higher than the number of DRI blocks. In this situation, a big portion of blocks have to carry an additional flag that is rarely set to one. Although a proper flag compression (i.e. by using CABAC context models) would reduce the rate of this flag, it still limits the performance by making the flag more expensive particularly for the proposed algorithm. The only situation in which the proposed algorithm is worth transmitting a flag, is when the compression gain achieved on DRI blocks is more than the flag overall overhead on the entire sequence (including both DRI and non-DRI blocks).

Developing a new tool that requires additional signaling usually consists of two phases. In the first phase, a baseline version is implemented in its best configurations in order to compensate the rate overhead and reach the performance of the reference encoder, in terms of BD-R. Upon achieving an acceptable baseline version, further improvements can be carried out in the second phase in order to bring additional BD-R gain on top of the reference encoder. In other words, the second phase attempts to make the flag overhead rewarded by a lower rate-distortion cost of the blocks that are coded properly by the new tool.

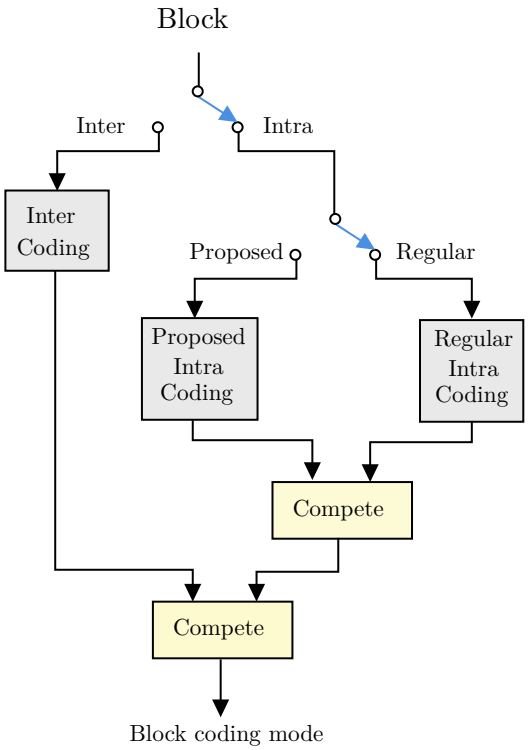


Figure 3-2: The block level algorithm competition with the proposed intra coding algorithm integrated to compete with the regular intra algorithm.

2.2 Complexity overhead of the encoder

To decide whether or not to use a coding tool, the encoder needs to actually evaluate it. As explained in Chapter 1, this is implemented by further expanding the RDO loop during the encoding phase, which results in an encoding complexity overhead. An example of such complexity overhead at the encoder side is shown in Table 3-1. Here, a dummy iteration is added to the encoder loop of IPMs and the complexity is computed in terms of encoding run-time. To impose a complexity load to the dummy iteration, the best IPM is re-tested, in order to avoid changes in the coding decisions. This experiment provides an estimation of complexity overhead we should expect from integration of the proposed algorithms. However, as will be discussed later, this complexity overhead may be significantly reduced by avoiding unnecessary computations at the encoder side, when the content is clearly inappropriate for the proposed algorithm.

3 Proposed in-block pixel prediction framework

In this section, the principles of our proposed framework are explained. This framework allows a competition between the proposed and the state-of-the-art intra coding algorithms within the RDO loop of each block. The winner of this competition is determined by the rate-distortion cost of using each algorithm as shown in Figure 3-2.

The “Proposed Intra Coding” module in the high level diagram of Figure 3-2 is a flexible placeholder to allow different editions of the proposed algorithms. In the next two chapters, we introduce different intra coding alternatives, implementing the idea of in-block references. As will be discussed in detail, depending on the algorithm specification, various sets of syntax elements needed to be transmitted. The goal is to benefit from the framework flexibility in order to integrate and test algorithms with different rate-distortion behaviors.

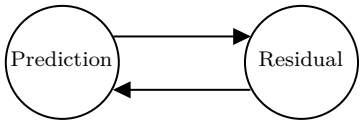


Figure 3-3: The chicken-and-egg data dependency problem of using in-block reference pixels.

3.1 The chicken-and-egg problem

The use of in-block references for addressing the DRI problem sounds promising at the first glance. However, there is a major data dependency obstacle at the encoder side: during the block prediction phase, the closest fully reconstructed pixels are the out-block references from previous blocks, as used in the conventional intra prediction. Therefore, in order to exploit in-block references, one must reconstruct them beforehand. This requires the residual signal to be available before the prediction phase. However, the residual signal is traditionally accessible only after the block prediction is finished. This initial chicken-and-egg dependency problem of using in-block references for the intra prediction is visualized in Fig. 3-3 and can be expressed as:

The prediction signal depends on the residual signal and subsequently, the residual signal also depends on the prediction signal.

Here, the above data dependency problem is formulated and its possible solutions are discussed. For this purpose, we define the following notation: assume a pixel position denoted as ($\bullet$) inside a 4×4 block. The proposed in-block references for intra prediction of pixel at this position are chosen at its left *(l)*, above-left *(al)* and above *(a)*. In this process, five 4×4 signals are involved, as shown in Figure 3-4:

1. The original block O is the input block to be predicted and coded.
2. The prediction block P is the output of the proposed intra prediction phase by applying a prediction function f on a subset of in-block reference pixels.
3. The original residual block R_{org} is the prediction error calculated as $R_{org} = O - P$.
4. The decoded residual block R is the residual block received at the decoder. In the lossless coding mode $R = R_{org}$. However, due to the quantization step, in lossy coding mode $R \neq R_{org}$.
5. The reconstructed block C is the ultimate output of the coding expressed as:

$$c^{(\bullet)} = p^{(\bullet)} + r^{(\bullet)}. \tag{3.1}$$

To avoid any mismatch between encoder and decoder, all reference pixels for prediction have to be selected from the signal C, since it is what both sides have in common access. This means that in-block reference pixels to predict $p^{(\bullet)}$ need also to be selected from the reconstructed signal:

$$p^{(\bullet)} = f(c^{(l)}, c^{(al)}, c^{(a)}). \tag{3.2}$$

3.2 The chicken-and-egg problem in the lossless mode

The solutions to the chicken-and-egg problem are different in lossy and lossless modes. In the lossless coding mode (LS), thanks to the exact reconstruction of the original signal at the decoder side, the above data dependency is automatically removed:

$$R = R_{org}. \tag{3.3}$$

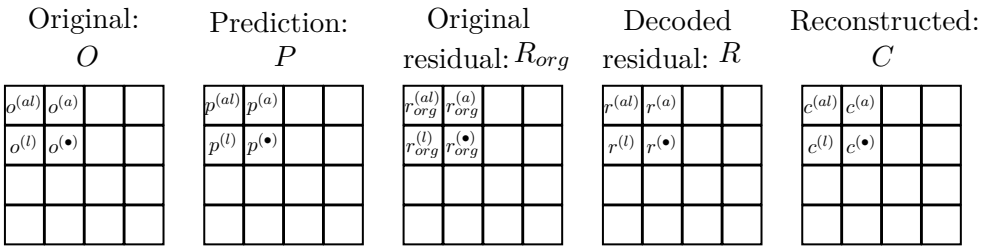


Figure 3-4: Five main blocks involved in the intra prediction with the proposed framework. In this figure, pixel at $(\bullet)$ intends to use its in-block reference at its left *(l)*, above-left *(al)* and above *(a)*.

Therefore, the lossless reconstruction of in-block references at the encoder side (denoted as C_{LS}), would result in the same values as in the original signal O:

$$C_{LS} = P + R = P + R_{org} = O. \tag{3.4}$$

Due to this property, Eq. 3.2 can be re-written for the lossless mode as:

$$p^{(\bullet)}_{LS} = f(o^{(l)}, o^{(al)}, o^{(a)}). \tag{3.5}$$

3.3 The chicken-and-egg problem in the Lossy mode

In lossy coding mode, the above solution is not applicable, since $R \neq R_{org}$. Therefore, in order to exploit in-block references in the lossy mode, the encoder has to have an estimation of the decoded residual signal R that is not yet produced. However, the regular residual calculation is unable to properly address the problem of in-block references. Here, a simple example is presented to explain it.

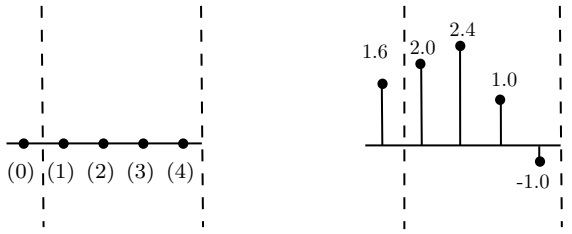
A simple approach for using the regular residual calculation for estimating R before the prediction at the encoder side, can be carried out as follows:

1. Perform the short distance block prediction in the lossless mode (i.e. by using the signal in O as reference) and produce the prediction block in lossless mode P_{LS}, as explained in Section 3.5.
2. Calculate the original residual block by $R_{org} = O - P_{LS}$.
3. Transform, quantize and then inverse quantize, inverse transform R_{org} to obtain R.
4. For the second round, perform the short distance block prediction, but in the lossy mode. The output of this step is the final prediction signal P that uses lossy reconstruction of in-block references, instead of the original ones. This reconstruction is performed by Eq. 3.1.

However, the above solution in the lossy mode introduces an inaccuracy in the reconstruction, which can propagate through the block. This is due to the fact that in step 4, we reconstruct each prediction pixel by adding a residual that is calculated from the lossless prediction of step 1. However, a proper reconstruction is achieved when the residual and prediction signals are calculated in the same process.

Let us elaborate the above inaccuracy propagation with a simple 1-D example. Assume a 1×4 original block $O = \{o^{(1)}, o^{(2)}, o^{(3)}, o^{(4)}\}$ and its last reconstructed pixel from previous block $c^{(0)}$, as shown in Figure 3-5.

The goal is to perform the above steps in order to apply an in-block intra prediction with a DPCM-based predictor function f^{1D}. At each pixel position inside block $(\bullet) = (1), .., (4)$, this



(a) Pixel positions$_{(\bullet)}$ (b) Original signal O

Figure 3-5: An original 1×4 block (positions (1) to (4) and its last reconstructed pixel from the previous block (0), where (a) denotes the position indexes and (b) shows original pixel values. The current 1×4 block is surrounded by two vertical dashed lines.

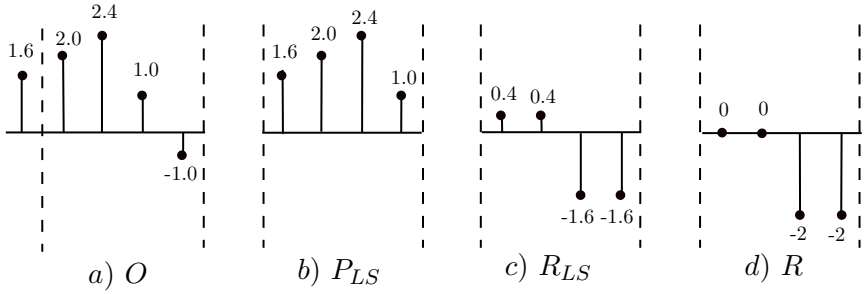


Figure 3-6: Calculation of P_{LS} from the original signal O (i.e. step 1) and then obtaining the lossy decoded residual R by quantizing original residual $R_{org} = O - P_{LS}$ in the spatial domain.

function simply takes the previous reconstructed pixel and copies it in $p^{(\bullet)}$. Applying step 1 with f^{1D} gives the lossless prediction signal P_{LS} as shown in Figure 3-6(b). Also, applying step 2 gives the original residual error R_{org} of this prediction signal as shown in Figure 3-6(c). The decoded residual R in Figure 3-6(d) can then be calculated from step 3 with a pixel domain integer rounding.

At this point, the error propagation can be seen by applying step 4 pixel by pixel. For this purpose, Figure 3-7 starts from the first pixel in the 1×4 block and applies the f^{1D} predictor on the lossy reconstructed reference of each pixel.

As can be seen from Figure 3-7, due to the reconstruction inaccuracy propagation, the final reconstructed block after the last stage is significantly different from the original signal. Figure 3-8 compares these two signals.

The main reason that the above methodology causes a poor performance, is the fact that R was calculated "outside the in-block prediction loop". On the contrary, a desired R signal needs to be prepared somehow "inside" the loop. Such residual would reconstruct each in-block reference pixel independently from others to control the inaccuracy propagation. However, what happened in the above example was that each residual value in R partially carried the quantization error of all of its previous pixels, since it had used them as its in-block reference.

3.4 In-loop residual for data dependency removal

To properly address the chicken-and-egg data dependency problem, the proposed framework integrates an additional residual signal which is computed either before or during the prediction phase. This residual signal needs to be transmitted to the decoder side to allow the same prediction. The methodology of obtaining the new residual is basically different and independent from the regular residual in the conventional video coding. Therefore, for a better representation of the proposed ideas, here we use the term In-Loop Residual (ILR) for the proposed additional

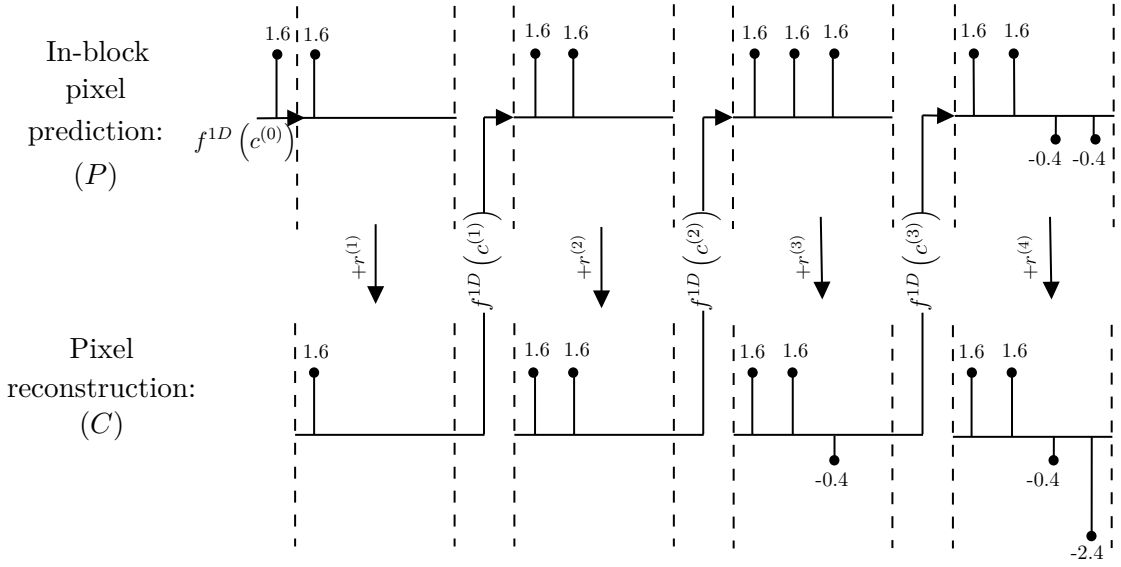
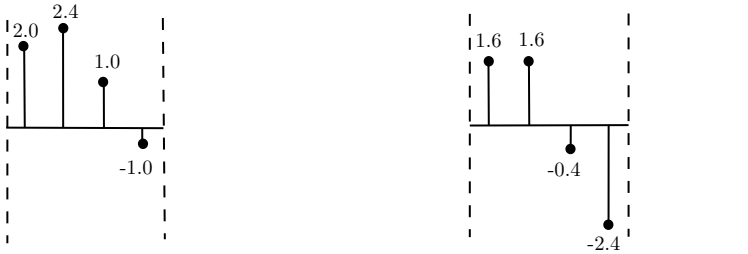


Figure 3-7: Applying the second round of in-block pixel prediction with the lossy R (i.e. step 4).



a) Original signal O b) Reconstructed signal C

Figure 3-8: Comparison of original 1D block O with its reconstructed block C, predicted by in-block pixel references and an estimation of residual.

residual, as opposed to the conventional Out-Loop Residual (OLR).

The main difference between the ILR and OLR is that the ILR signal is designed to be calculated either during or before the prediction stage. However, the OLR signal is traditionally calculated at the last stage and when the prediction is finished. Moreover, in addition to pixel reconstruction during the prediction, the ILR signal is also responsible for controlling the reconstruction inaccuracy propagation. The importance of this task was shown with the previous 1-D example.

From a high level design perspective, the ILR signal is deployed to remove the chicken-and-egg dependency between the prediction signal and the OLR signal. This solution can convert the diagram of Figure 3-3 into an OLR-independent prediction shown in Figure 3-9. As can be seen, an intra prediction algorithm integrating the ILR signal proposes that in-block references would be reconstructed by ILR instead of OLR.

As will be discussed in the next chapters, the reconstruction accuracy of ILR can be less effective than OLR, in terms of distortion. Thus, after the in-block prediction is finished at the block level, the OLR signal can still be used to improve the reconstruction. Therefore, in order to avoid confusion in the rest of this thesis, we differentiate between the pixel reconstruction by ILR and OLR. For this purpose, the former will be called "pixel correction" and the latter will be kept called "pixel reconstruction".

The decoder has to be able to reproduce the same prediction signal as that of the encoder side, therefore, we must transmit ILR signal to allow the decoder to access the same in-block ref-

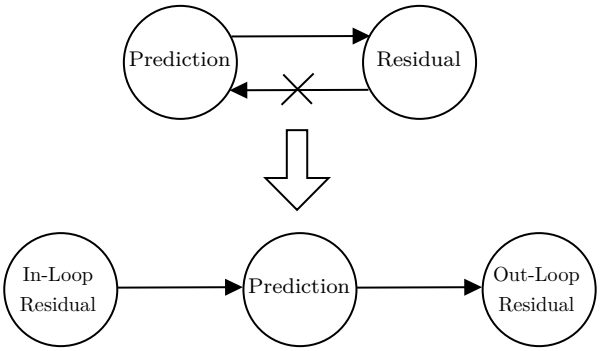


Figure 3-9: Removing the chicken-and-egg data dependency problem by integrating ILR signal.

erences. For this purpose, two different ILR representations and coding algorithms are proposed in the next two chapters:

- Vector quantization: A finite dictionary of ILR signals is trained and provided to both encoder and decoder sides. At the encoder side, the best dictionary codeword is selected in terms of rate-distortion cost and its index in the dictionary is transmitted.
- Scalar quantization: The encoder bypasses the transform and directly quantizes the residual amplitudes in the pixel domain. The quantized levels are then transmitted in a similar fashion as the transform coding scheme.

In the next chapters, the characteristics of these two ILR coding schemes are discussed in detail.

3.5 Block prediction using ILR

Once the chicken-and-egg data dependency problem is addressed by introducing the ILR signal, one can apply different in-block pixel prediction schemes on 2-D blocks. Here, a simple approach is explained to complete the chain of operations in the proposed framework. In this approach, major aspects of the proposed block prediction are discussed and different design choices are explained from a high level point of view.

3.5.1 Reference selection

Once all previous in-block pixels are available, one can consider different combinations of references for predicting pixels. There are mainly two factors involved in determining in-block references: 1) the number of references, and 2) the distance of references. Altogether, these two factors are limited by the block size and the position of each pixel in the block. For instance, in a 4×4 block, the pixel at the last scan position can have up to 15 in-block references (i.e. all previously scanned pixels). However, the first scan position has no in-block reference available and has to use out-block references from previous blocks.

There is also another aspect which further limits the number of in-block references. From the implementation point of view, having a high number of reference pixels imposes hardware and/or software implementation complexity. It is highly recommended not to use an excessive number of rows and columns from previously reconstructed pixels, although it is usually more efficient to use as many references as possible. In the proposed algorithms, a proper compromise will be made between the compression performance and the codec complexity.

3.5.2 Pixel scan order

The choice of pixel scan order within the block has a direct impact on the availability of reference pixels. Assume that three pixel scan orders, the horizontal, vertical and diagonal are possible

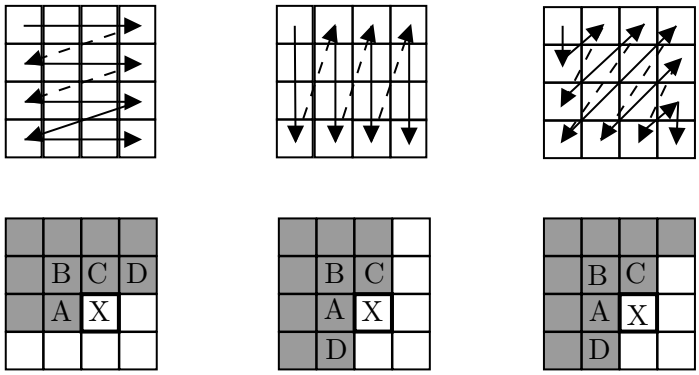


Figure 3-10: Availability of in-block reference pixels with a distance of one, when different scan orders are applied.

for each block. Moreover, the in-block predictor function only allows the use of reference pixels with distance of 1 (i.e. above row and left column). This restriction results in three different reference sets corresponding to each scan order as shown in Figure 3-10. Each scan order can be useful for a certain type of texture. Therefore its decision and whether the encoder should allow multiple scan orders or not, are two important design choices. This question will also be discussed later.

3.5.3 In-block pixel predictor function

There are different aspects to be considered while determining an in-block pixel predictor function.

One factor affecting the performance of the pixel predictor is the number and position of reference pixels and their distance from the current pixel. As explained, the number of references can vary depending on the scan position. With a higher number of reference pixels, the goal of the pixel predictor is to exploit as much as possible texture information while remaining in an acceptable complexity level.

Another aspect is the possibility of having different pixel predictors. In the case of multiple predictors, another question can be raised regarding the necessity of signaling. For instance, one can allow different predictors for different blocks but avoid signaling the selected function. Instead, an algorithm can be used to derive the best predictor based on local texture information. This derivation may use information in the residual or in the neighborhood.

4 Conclusion

This chapter has introduced a framework for exploiting in-block pixels as reference for intra prediction. The proposed framework potentially allows the encoder to compress high-detail blocks more properly. For this purpose, an alternative algorithm is proposed in the form of a placeholder which can be replaced by different implementations. This algorithm will compete with the state-of-the-art intra algorithm in terms of rate-distortion cost, at the block level, and will transmit a flag to signal the selected algorithm. The ultimate goal is first, to compensate the rate overhead of the flag and then, to further improve the BD-rate gain.

The main difficulty of using in-block references is their data-dependency to the residual. For this purpose, the proposed framework integrates an additional residual signal to remove this dependency. As the efficiency of such residual is critical for the performance, we will propose two different algorithms in the next chapters. These algorithms conform to the general framework of this chapter and have different characteristics.

CHAPTER 4

ILR-IP with vector quantization

Contents

1 Introduction

In this chapter, we propose an algorithm to represent and encode the In-Loop Residual (ILR) signal, described in the previous chapter. As discussed, the ILR signal has to be computed beforehand, or during, the prediction phase. This new algorithm benefits from the Vector Quantization (VQ) method to prepare a list of ILR candidates in the form of a codebook before the encoding phased.

Quantization is the process taking a continuous/discrete scalar/vector, produced by a source, and representing it by a reduced set of representative scalar/vectors. There are mainly two viewpoints to reach this goal: the Vector Quantization (VQ) and the Scalar Quantization (SQ). Theoretically, the most efficient coding scheme for a set of quantized symbols is VQ [65]. Compared to SQ, there is a potential capacity in VQ that allows exploiting correlation directly. The proposed algorithm in this chapter benefits from the VQ technique to represent a list of candidates for the In-Loop Residual (ILR) signal, as described in previous chapter.

The rest of this chapter is organized as follows. First, the principles of the VQ technique are introduced. Then, the details of the proposed algorithm are discussed. As the problem under study possesses unique features in terms of data dependency, a special codebook training is explained. Finally, the results of implementing the proposed algorithm on top of the JEM reference software are presented.

2 Principles of VQ techniques

VQ is a lossy data compression technique that allows the Probability Distribution Function (PDF) modeling by scattering a set of prototype vectors [66]. To put it simply, VQ works by splitting a large set of points in an Euclidean space into a fewer number of clusters. The objective is usually to minimize the average distance between any point and the closest prototype. Samples are then represented by the centroids of their clusters, here called codevectors.

Let Q be a vector quantizer to map the signal from the K-dimensional Euclidean space $\mathbb{R}^K$ into a subset of it we call C. This subset is defined as $C = c_i, i = 1, .., N$ and is a set of N points, where $c_i \in \mathbb{R}^K$. The quantizer Q is expressed as:

$$Q : \mathbb{R}^K \rightarrow C. \tag{2.1}$$

In the above notation, the subset C is usually called a codebook and its N elements c_i are called codevectors. Each point x in $\mathbb{R}^K$ is approximated by one of the N codevectors:

$$Q(x) = c_i, 0 \leq i \leq N. \tag{2.2}$$

The lossy nature of VQ coding can be reflected in the fact that each approximation produces a quantization error that is calculated as:

$$\varepsilon(x, c_i) = \|x - c_i\|_2^2. \tag{2.3}$$

Let $\mathbb{R}^K$ be split into N non-overlapping partitions $R_1, R_2, ..., R_N$, represented by their codevector in the codebook $C = c_i, i = 1, ..., N$. In the standard VQ design, two conditions should be satisfied to obtain the optimal codebook, that is, the codebook that minimizes the distortion C [65]:

- Nearest neighbor condition: For each partition R_i, the optimal partitioning should satisfy:

$$R_i \subset \{x : \varepsilon(x, c_i) \leq \varepsilon(x, c_j); \text{ for all } j \neq i\}. \tag{2.4}$$

  That is to say:

$$Q(x) = c_i \text{ only if } \varepsilon(x, c_i) \leq \varepsilon(x, c_j), \text{ for all } j \neq i. \tag{2.5}$$

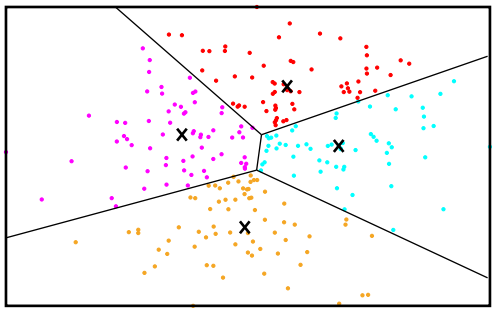


Figure 4-1: An example of partitioning by VQ. There are four codevectors, each representing approximately 64 points in a 2D Euclidean space.

- Centroid condition: For a given partition R_i with the length L_i, the optimal codevector must minimize the inner-partition distortion criterion. This can be done by applying a centroid function that calculates the arithmetic mean of the points in the partition:

$$c_i = cent(R_i) = \frac{1}{L_i} \sum_{x_m \in R_i} x_m. \tag{2.6}$$

VQ can be viewed as the cascade of an encoder and a decoder, both using a shared codebook stored in the form of a look-up table [67]. Given a K-dimensional uncompressed vector x as input, the encoder simply looks up in the table and picks the best matching codevector c^* in the codebook. With the condition of minimizing the distortion, this step is performed as:

$$c^* = \arg\min_{c_i \in C} \varepsilon(x, c_i). \tag{2.7}$$

Upon selecting c^*, the encoder obtains its index in the codebook. This index is then binarized and transmitted to the decoder. The decoder which has access to the same codebook with the same indexing scheme, retrieves the codevector in the codebook as the reconstructed version of the source vector.

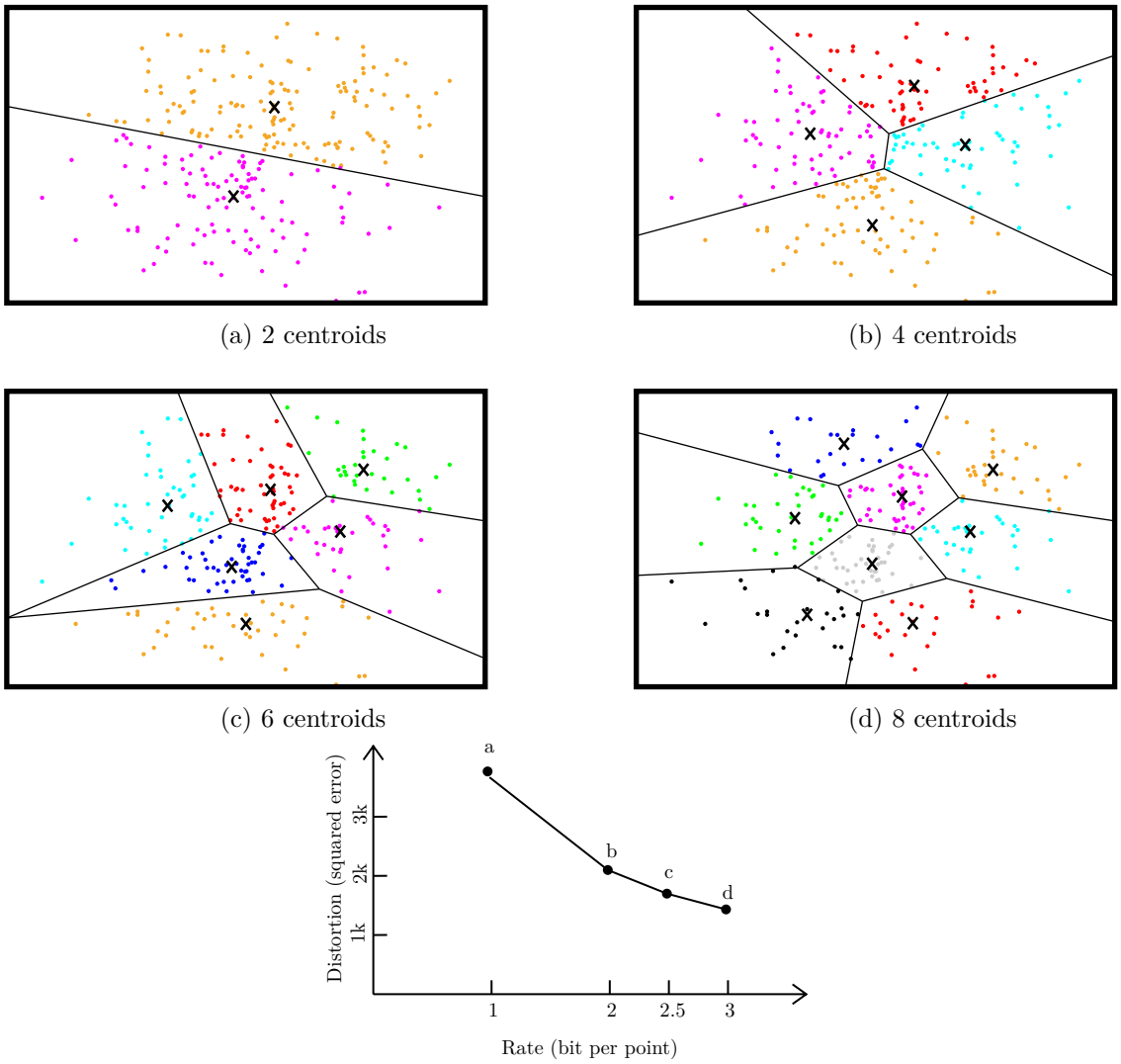
With a source vector consisting of M points in the $\mathbb{R}^K$ space, a vector quantizer with N codevectors can provide a compression ratio of R_{VQ}, calculated as:

$$R_{VQ} = \frac{log_2^M}{\lceil log_2^N \rceil}. \tag{2.8}$$

An example of optimal partitioning by VQ is shown in Figure 4-1. In this example, a subset of $\mathbb{R}^2$ space is split into 4 partitions. In this figure, there are almost 64 points in each partition, represented by one codevector shown by a black cross. According to Eq. 2.8, the compression ratio of such vector quantizer is $log_2(4 \times 64)/log_2 4 = 6$.

An important design choice to be made for a VQ system in video coding is the codebook size N. As can be deduced from Eq. 2.8, N is the only parameter for imposing different rate-distortion behaviors. For instance, in case of a coarser quantizer, one wants the encoder to operate in higher compression ratio compared to a finer quantizer. This flexibility can be provided to the encoder by training variable size codebooks in different quantizer granularities. More precisely, with coarser quantizers, smaller codebooks that result in higher distortion and lower rate are preferred. Conversely, with finer quantizers, larger codebooks having the opposite behavior are desired.

An example of tuning the rate-distortion behavior with the number of codevectors is shown in Figure 4-2. In this figure, a set of $M = 200$ data points in $\mathbb{R}^2$ are given to vector quantizers with 2, 4, 6 and 8 codevectors, respectively. At the bottom of the figure, the rate-distortion



(a) 2 centroids

(b) 4 centroids

(c) 6 centroids

(d) 8 centroids

(e) Rate-Distortion behavior of above four VQ settings.

Figure 4-2: A vector quantizer with four different rate-distortion behaviors, imposed by its number of codevectors $N = 2, 4, 6, 8$. The curve in the bottom demonstrates the rate-distortion behavior corresponding to these four settings.

curve of this vector quantizer is also provided. Here, the rate shown by the x-axis is calculated as:

$$Rate = M \times \lceil log_2^N \rceil. \tag{2.9}$$

Moreover, the distortion in Figure 4-2 is calculated as:

$$Distortion = \sum_{j=1}^{M} \varepsilon(x_j, Q(x_j)). \tag{2.10}$$

In the rest of this chapter, we introduce an intra coding algorithm that employs a variation of VQ coding. As will be seen, some slight modifications have to be made on the general concept of VQ coding, in order to adapt it to the existing problem.

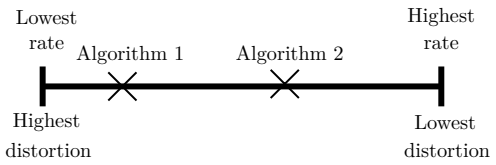


Figure 4-3: The rate-distortion spectrum between two extremes of ILR granularity
The rate-distortion spectrum between two extremes of very low and very high ILR granularity.

3 Proposed ILR-VQ algorithm

In this section, the details of the proposed algorithm based on the principles of the previous section are presented.

3.1 Expected rate-distortion behavior of ILR-VQ

The choice of the ILR signal plays an important role in the performance of the proposed in-block prediction framework. At the pixel level, providing a proper ILR value would improve the accuracy of the predictor and this results in a better texture decorrelation. However, such accuracy usually comes with the cost of a finer quantization, hence, a higher rate. Therefore, like other compression tools, the performance of an ILR signal for in-block intra prediction must be measured in terms of rate-distortion cost.

A desired ILR signal should optimize the rate-distortion cost by making a balance between 1) the transmission rate of the codevector index and 2) the distortion of the best codevector, acting as the ILR signal during the intra prediction. For this purpose, a spectrum between two extremes in the rate-distortion sense is defined. Depending on the ILR coding methodology, an algorithm may cover different ranges on this spectrum. On one extreme, a very low rate budget may be considered for the ILR signal, by applying a coarse-grained quantizer. This usually results in a poor performance in terms of distortion. On the other extreme, an ILR signal can be quantized at finer granularity with a higher rate budget, which results in less distortion. For instance, the lossless short distance intra coding algorithm discussed in chapter 3 stands exactly at this extreme.

Figure 4-3 visualizes the rate-distortion spectrum for different ILR coding schemes. As shown, two different schemes stand on different positions of this spectrum. However, neither side of the spectrum necessarily means more efficient compression. Instead, the local Lagrangian rate-distortion cost at the block level and the global BD-R measure at the sequence level determine the performance of different algorithms and allow comparisons.

The proposed ILR-VQ algorithm stands rather near the “lowest rate/highest distortion” extreme of the spectrum in Figure 4-3. Basically, the performance of ILR-VQ is essentially limited to the number of codevectors in the codebook. Moreover, according to Figure 4-2, the only way to allow higher rates in VQ-based algorithms is increasing the codebook size. However, as will be discussed in detail, the codebook size has direct impact on the encoder side complexity. Therefore, the best efficiency/complexity compromise leaves the ILR-VQ rather near the left extreme in the spectrum.

3.2 Design choices of ILR-VQ

In this section, we introduce a short distance intra prediction algorithm that engages a VQ technique to code the ILR signal of each block. Similar to the problem in chapter 3, four main questions shall be answered during the algorithm design. These questions, representing different components of ILR-VQ, are namely:

- Given a block, a scan order and an ILR signal, how do we perform pixel prediction?

Figure 4-4: Three reference pixels at $c^{(l)}$, $c^{(a)}$ and $c^{(al)}$ for in-block pixel prediction at position (•), by LOCO-I.

- Given a block, how do we calculate the best ILR signal? Or, what options are provided to the encoder as the ILR signal?
- Once the ILR signal is decided, how do we transmit the best ILR signal to the decoder?
- Given the rate and distortion of ILR-VQ, in which manner does it compete with the existing intra coding algorithm?

The next sections in this chapter answers the above questions regarding the design choice of ILR-VQ.

3.3 Notation

For clarification, here we present the notation that is used in the rest of this chapter. There are mainly four signals involving the ILR-VQ process: prediction signal P, ILR signal R, reconstructed signal C and original signal O. Each signal is used in the shape of a block of size (W, H) and consists of $W \times H$ values. These values are denoted with (•) and complies with the scan order. For instance, $p^{(3)}$ is the 3^{rd} predicted value with respect to the scan order, or, $c^{(al)}$ means the reconstructed value positioned at above-left of the current position.

3.4 Pixel prediction

The main problem of DRI blocks is the fact that they do not contain a regular texture pattern. This makes conventional block-level angular modes inefficient for their prediction. An ideal predictor function must be able to adapt with content change and predict pixels by looking at their in-block references and their relative combination.

One way to adapt with in-block content change is to signal the change. For instance, assume a DRI block which has roughly a horizontal direction in the beginning and a vertical direction at the end. One can signal the change and its location in the block to inform the decoder about it. However, experiments show that this approach is extremely expensive in terms of rate. Especially in cases where more than one change happen inside a block.

In ILR-VQ, a context-based pixel predictor is used which does not require further signaling. This method is called LOw COmplexity LOssless COmpression for Images or LOCO-I and uses three reconstructed reference pixels at left C_l, above C_a and above-left C_{al}, as shown in Figure 4-4.

Context derivation by LOCO-I is performed by evaluating possible correlation in the three references as expressed in Eq. 3.1:

$$p^{(\bullet)} = \text{LOCO-I}(c^{(l)}, c^{(a)}, c^{(al)}) = \begin{cases} min(c^{(l)}, c^{(a)}), & \text{if } c^{(al)} \geq max(c^{(l)}, c^{(a)}). \\ max(c^{(l)}, c^{(a)}), & \text{if } c^{(al)} \leq min(c^{(l)}, c^{(a)}). \\ c^{(l)} + c^{(a)} - c^{(al)}, & \text{Otherwise.} \end{cases} \tag{3.1}$$

The contexts of LOCO-I are defined by the three conditions in Eq. 3.1 and are used for direction detection. That way, vertical and horizontal directions are detected by the first two conditions, respectively, where the last condition defines "no direction" context. For instance, assume three different settings of Figure 4-4 as:

1. $(c^{(l)}, c^{(a)}, c^{(al)}) = (10, 240, 20)$: condition 1 detects a vertical edge and returns the prediction value $p^{(\bullet)} = 20$.

2. $(c^{(l)}, c^{(a)}, c^{(al)}) = (10, 240, 250)$: condition 2 detects a horizontal edge and returns the prediction value $p^{(\bullet)} = 10$.

3. $(c^{(l)}, c^{(a)}, c^{(al)}) = (10, 240, 110)$: conditions 3 detects no edge and returns the prediction value $p^{(\bullet)} = 140$.

3.5 Block prediction

For completing prediction at the block level, ILR-VQ needs ILR signal R in addition to the prediction function (i.e. LOCO-I). Assuming that R is given as ILR signal, ILR-VQ scans pixels in block and performs two steps at each pixel position i:

1. Pixel prediction by LOCO-I and calculating $p^{(\bullet)}$ by 3.1.

2. Pixel correction by adding residual:

$$c^{(\bullet)} = p^{(\bullet)} + r^{(\bullet)}. \tag{3.2}$$

The ILR signal R allows applying the above two steps on each pixel progressively. This makes each pixel available as an in-block reference for its next pixels in the scan. The main idea in ILR-VQ is to provide a codebook of candidates for R and let the encoder choose the best one. Therefore, the encoder performs a brute-force search on the candidates and tries each codevector as the input R to the above algorithm. Figure 4-5 demonstrates this process for a 4×4 block. In this figure, the input DRI block and its regular out-block references are shown at top-left corner. On the right, the codebook of ILR candidates is presented to feed the ILR-VQ block prediction with R. Given R, the block prediction progresses through all pixels in the 4×4 block and performs the pixel prediction and pixel correction steps, as explained above. The output of this process is the prediction block shown in the last row.

Different block scans are applicable on the process of Figure 4-5. In the experiments of this chapter as well as in the codebook training, three scan orders of horizontal, vertical and diagonal were tested. However, all of them performed similarly in terms of compression efficiency. Therefore, it is the horizontal raster scan that has been chosen in the final ILR-VQ design.

3.6 ILR coding

Fixed-length coding with direct binary codes has been used for ILR coding. For this purpose, different codebooks of size $N = 2^n$ have been used to enable binary codes of length b for signaling the selected codevector.

3.7 Algorithm competition

The proposed framework of Chapter 3 provides a block level choice for the encoder to choose between two intra coding algorithm: 1) the conventional algorithm of VVC, 2) the ILR-based algorithm (e.g. ILR-VQ). This flexibility costs one flag per eligible block, called ILR flag. For instance, an ILR-VQ design which applied in-block prediction only on 4×4 blocks, does not code the ILR flag on blocks larger than 4×4.

Figure 4-6 demonstrates the algorithm competition at the encoder side for an eligible block. As can be seen, intra coding provides the ILR-VQ branch in addition to the standard branch, where the ILR flag determines the selected coding algorithm. On the standard branch, the conventional intra coding, explained in Chapter 2, is performed an "IPM loop" by iteration over all available IPMs and chooses the one with the least rate-distortion cost J_{Std}. This cost

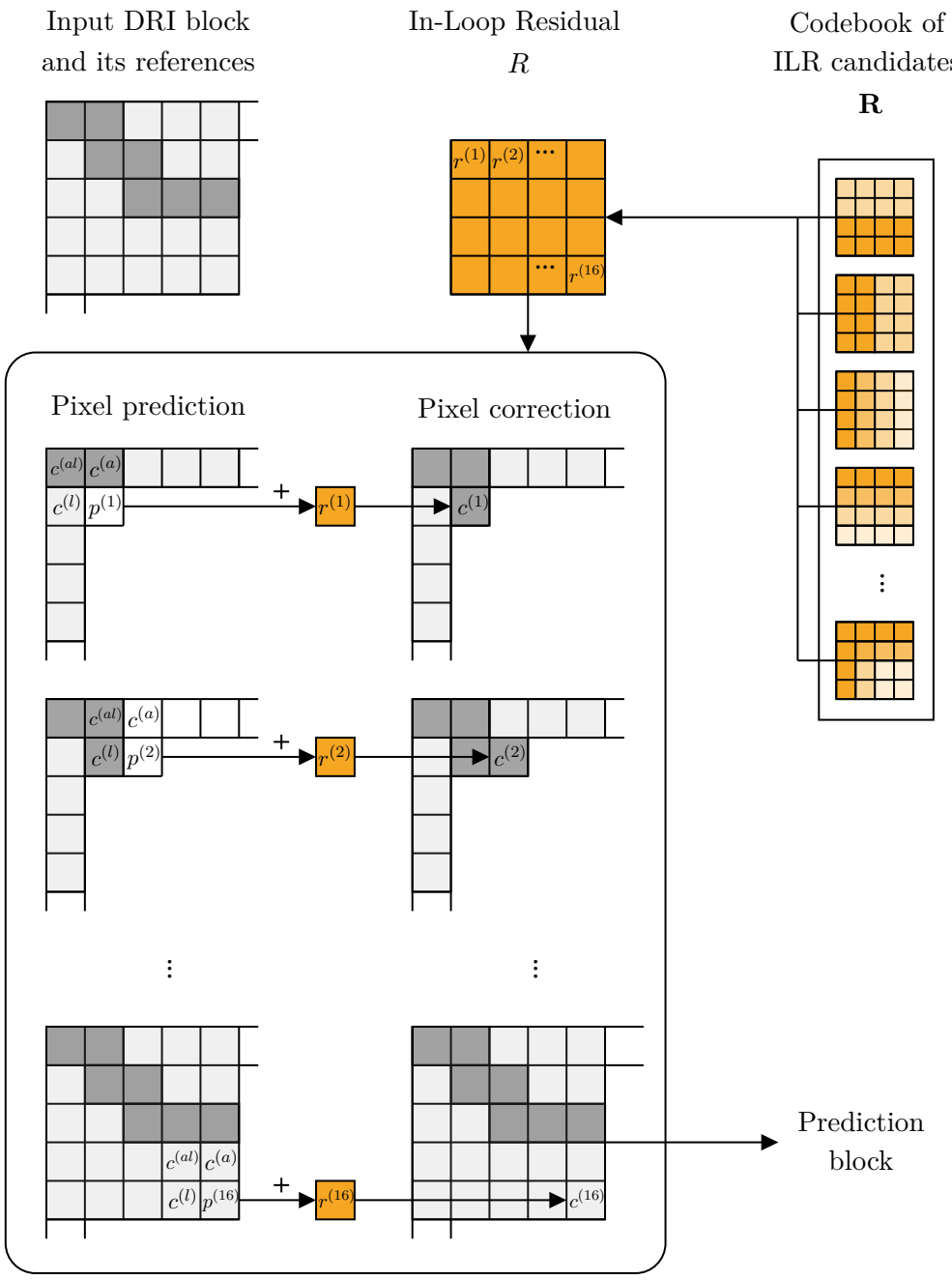


Figure 4-5: Diagram of the proposed block prediction with ILR-VQ.

is calculated by applying the standard intra prediction in "Std. IP" with out-block references. However, the ILR-VQ branch performs differently as it does not benefit from the regular IPM set. Instead, it has an "ILR loop" in which it iterates over all codevectors provided in the codebook. In each iteration, ILR-VQ performs in-block prediction denoted as "ILR-VQ IP" in this figure. After this prediction, the Out-Loop Residual (OLR) is also calculated and added to the prediction block. Finally, the rate-distortion cost of the ILR-VQ branch is computed as J_{ILR-VQ}. At this point, the encoder chooses between the standard and ILR-VQ intra coding algorithms by comparing J_{Std} and J_{ILR-VQ}.

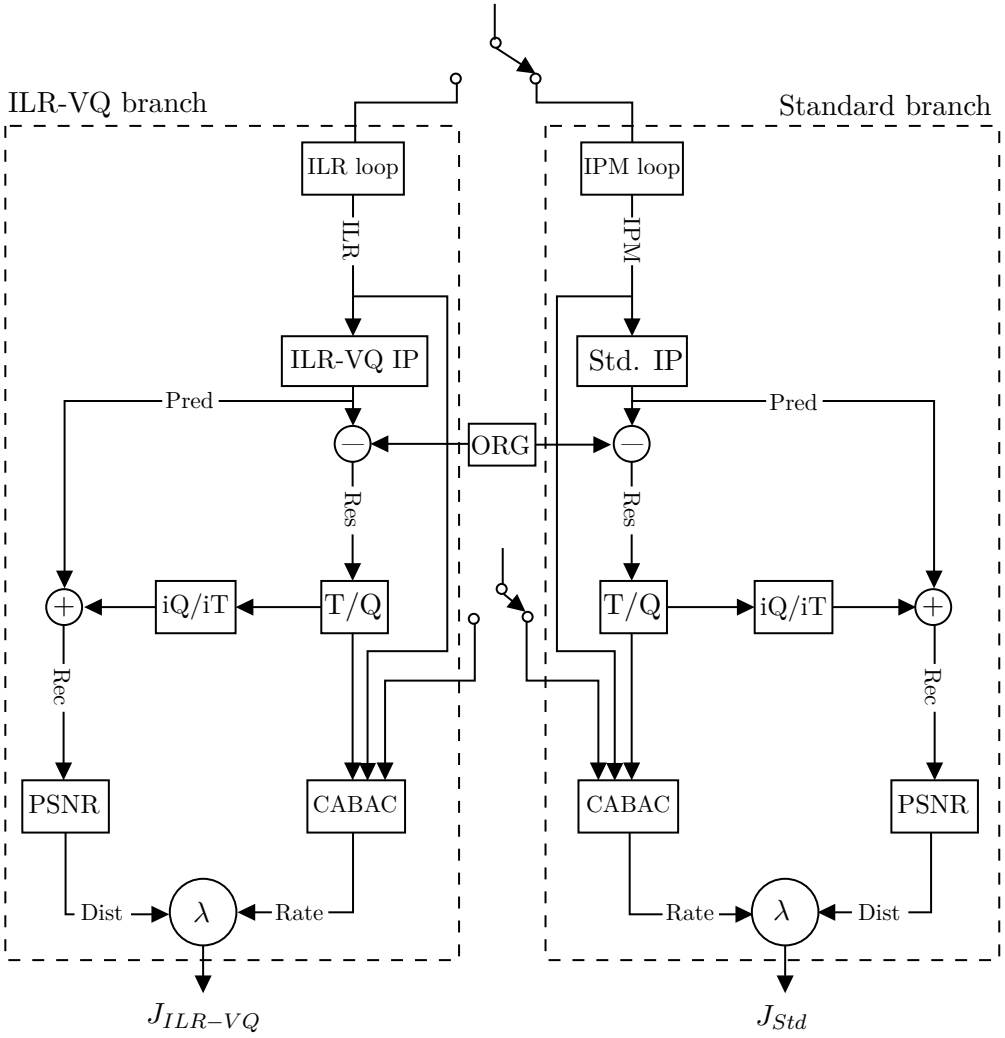


Figure 4-6: Intra coding algorithm competition diagram with ILR-VQ.

4 Codebook training

An ideal codebook is able to provide any arbitrary block with the exact ILR signal that it needs. However, this would require an exceedingly large codebook which imposes an unfeasible transmission rate. In practice, VQ systems make a compromise between the codebook size and the compression ratio. In this section, a codebook training scheme is introduced to be used in ILR-VQ given its codebook size. This scheme is based on a well-known algorithm called Linde-Buzo-Gray (LBG) [68].

4.1 Standard LBG algorithm

LBG is an iterative algorithm to optimize a codebook with a given size on an adequately large training sequence $\mathbf{S}$. The goal is to optimize a codebook with N codevectors $\mathbf{R} = \{R_j; j = 1, 2, ..., N\}$, where R_j is the j-th ILR signal candidate in the codebook of ILR-VQ. Iterations are considered in LBG to update the codebook $\mathbf{R}$ until it converges. Therefore, the codebook at each iteration is denoted as $\mathbf{R}^t = \{R_j^t; j = 1, 2, ..., N\}$. Standard LBG starts its first iteration with a random codebook.

Each iteration t consists of two main steps:

Classification

Given the codebook at iteration t with N codevectors $\mathbf{R}^t = \{R_j^t; j = 1, 2, ..., N\}$ and the dateset $\mathbf{S}$ of samples, the classification step determines the best codevector for each sample in $X \in \mathbf{S}$ from a set of N categories $\mathbf{G} = \{G_1, ..., G_N\}$:

$$G_j^t = \{X : d(X, R_j^t) < d(X, R_k^t); \text{all } k \in [1, N] \text{ and } k \neq j\}, \tag{4.1}$$

where $d(X, R)$ is the objective function for calculating the distance of sample X from the codevector R.

The standard LBG is normally used for representing a large set of points in an Euclidean space by a limited number of codevectors. Hence, a ρ-norm distance measure (e.g. Euclidean distance) can be used as the distance function d. However, in ILR-VQ, categorizing a block into a class $\mathbf{G}_i$ means selecting codevector R_i as its ILR signal. Therefore, it is more appropriate to define d based on the rate-distortion cost of using R_j on the block X. For instance, $d = J_{ILR-VQ}$, explained in Figure 4-6, can be a proper classification objective function.

Update

After the classification of all samples at iteration t, each codevector in $\mathbf{V}^t$ is updated by a centroid function that basically minimizes a global distance measure based on d:

$$\mathbf{R}^{t+1} = \{\text{cent}(G_j^t); j = 1, ..., \text{N}\}. \tag{4.2}$$

In the standard LBG that uses the Euclidean distance as d, the arithmetic mean can be used as centroid function. Such algorithm with the above specification can guarantee a convergence by implementing an adequate number of iterations on the classification and update steps, expressed in Eq. 4.1 and Eq. 4.2 [69]. However, the codebook training in the ILR-VQ is slightly more complicated, as all decisions are made through a non-linear prediction process. This means that the choice of the Euclidean distance and the arithmetic mean for the classification and update steps, respectively, no longer guarantees a convergence. This is due to the fact that the progressive pixel prediction in ILR-VQ, which imposes in-block dependencies, makes it theoretically possible to decrease prediction error while increasing Euclidean distance from the optimal ILR signal.

4.2 Modified LBG algorithm

Due to the above limitation, a slight, yet important modification is made on the centroid function of the LBG algorithm. In our new method, each centroid is updated pixel-by-pixel. This is comparable to the block-level updating process of the standard LBG, explained in the previous section. The only difference is that, in the pixel-level updating, when one value of a codevector is updated, it will immediately be used for updating the next values of the same codevector. However, in the block-level updating, all values at all scanning positions are updated at once by using the old codevectors from the previous iteration. In the rest of this section, all discussions for the centroid updating are only focused on the j-th codevector of the codebook R_j, where $j = 1, 2, ..., N$.

Let $\mathbf{S}_j^t = \{X_l; l = 1, ..., L\}$ be the set of samples that were classified in the i-th codevector at the t-th iteration by Eq. 4.1. Each sample in this set is an H×W block. Moreover, a scanning order specifies the order of predicting pixels as well as the updating of codevector values. Therefore, given the scanning order and the training samples in $\mathbf{S}_i^t$, the l-th sample can be vectorized as:

$$X_l = [x_l^{(1)}, ..., x_l^{(\text{H}\times\text{W})}]. \tag{4.3}$$

According to Eq. 4.1, at iteration t, the codevector associated to the above sample is R_j^t, which can be vectorized similarly:

$$R_j^t = [r_j^{t,(1)}, ..., r_j^{t,(\mathrm{H}\times\mathrm{W})}]. \tag{4.4}$$

Moreover, the prediction pixels may be calculated by applying Eq. 3.1, given R_j^t:

$$P_l^t = [p_l^{t,(1)}, ..., p_l^{t,(\mathrm{H}\times\mathrm{W})}]. \tag{4.5}$$

Note that the superscript t on each predicted pixel p indicates that its corresponding inner-block references at its top, left and top-left, were corrected by the codevector at iteration t. By referring to Figure 4-4, Eq. 3.1 and Eq. 3.2, this can be written:

$$p_l^{t,(\bullet)} = f(c_l^{t,(a)}, c_l^{t,(l)}, c_l^{t,(al)}) = f(p_l^{t,(l)} + r_i^{t,(l)}, p_l^{t,(a)} + r_i^{t,(a)}, p_l^{t,(al)} + r_i^{t,(al)}). \tag{4.6}$$

The adopted centroid function performs a pixel-level update with respect to the scanning order as:

1. It starts from the first scanning position of the centroid at iteration t.
2. It updates the centroid value at that position.
3. It replaces the updated value in the ongoing centroid of iteration t.

After performing above steps for all positions of the scanning order, the entire centroid is updated and ready for the next iteration $t+1$. Eq. 4.7 formulates the pixel-level update at position $(\bullet)$ that requires performing the pixel prediction for all L samples of the class at that position:

$$r_j^{t+1,(\bullet)} = \frac{1}{L}\sum_{l=1}^{L}(o^{(\bullet)} - p_l^{t+1,(\bullet)}), \tag{4.7}$$

where $o^{(\bullet)}$ is the original pixel value at the scanning position $(\bullet)$. As shown, p_l^{t+1} belongs to the iteration $t+1$ instead of t. According to Eq. 4.6, this indicates that its inner-block references were corrected by the centroid of iteration $t+1$ that was just updated in the ongoing update phase. This is possible since the scanning positions at top, left and top-left of the (p) are precedent to (p), thus, their corresponding centroid values are already updated in the current update phase.

After Eq. 4.7 is performed on all scan positions $(\bullet)$ in the block, the j-th codevector at iteration $t+1$ is expressed as:

$$R_j^{t+1} = [r_j^{t+1,(1)}, ..., r_j^{t+1,(\mathrm{H}\times\mathrm{W})}]. \tag{4.8}$$

The above steps explain the centroid update step in one iteration of the codebook construction process for one centroid of the codebook. The goal is to repeat both the classification and update steps until convergence.

Figure 4-7 visually summarizes the above steps for a 2×2 codevector R_j, with L samples in S_j and the given scanning order. For a better visualization, the index j has been removed from symbols S, R and r, as no other codevector is involved in the process of this figure. Moreover, scan positions at each stage are highlighted by gray squares allowing to remove $(\bullet)$ used in equations, too.

In Figure 4-7, the update process starts with the R^t (i.e. R_j^t) in the first row and first column. Each column $l = 1, 2, ..., L$ corresponds to sample $X_l \in S_j$. Moreover, each row explains the pixel level update process at its highlighted scan position which performs the update $r^{t,(\bullet)} \rightarrow r^{t+1,(\bullet)}$. As can be seen, at the end of each row, the highlighted codevector value is updated and replaced in the R^t for the next row. This process repeats until the last position is updated and the whole codevector is updated to R^{t+1}, as shown as the output at the bottom-right corner.

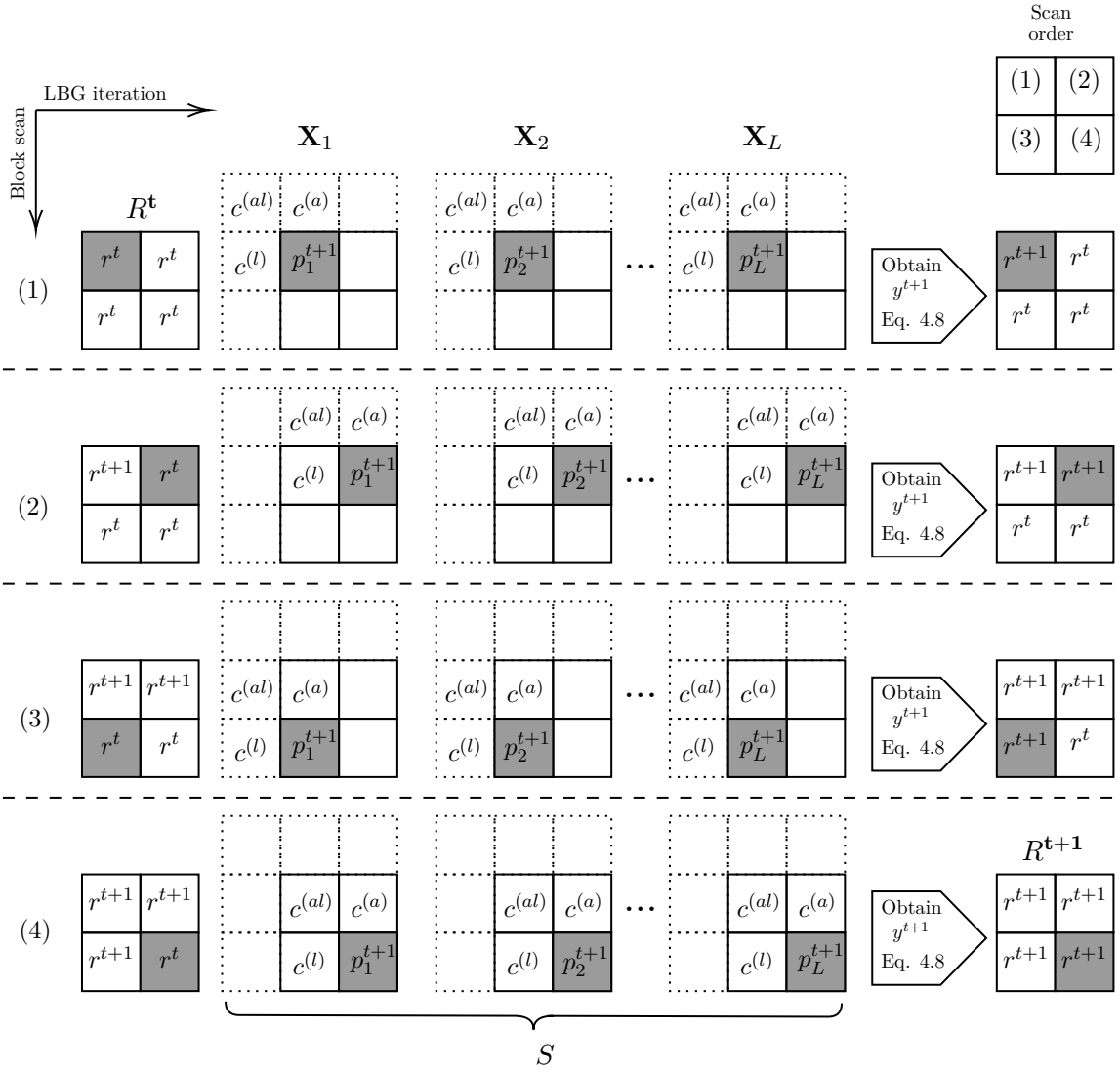


Figure 4-7: One iteration of the modified LBG algorithm for updating the j^{th} codevector R_j. The block size is considered to be 2×2 for simplicity and LBG is given a set of L samples in S_j that are all classified into R_j codevector. Four stages of the pixel-level centroid update are presented with respect to the given scan order at top-right of the figure. At each stage, the gray pixel indicates the scan position (•) of the codevector that is being updated. The out-block prediction references of the 2×2 blocks from previous blocks are shown by dotted squares.

5 Results

In this section, the performance of ILR-VQ is presented. For this purpose, the impact of different design parameters is evaluated. There are three main parameters involved in the performance of ILR-VQ: Block size, codebook size and LBG settings. According to the performance evaluation results of different configurations, the recommended configuration of ILR-VQ has been determined to use a block size of 4×4 with 128 codevectors in its codebook.

Regarding the block size, two configurations are used in the experiments of this section. The first configuration uses ILR-VQ only for 4×4 blocks, while, the second configuration uses it for all block sizes up to 8×8 (i.e. 4×4, 8×4, 4×8 and 8×8). In each configuration, both encoder and decoder have to store one codebook for each activated block size. After performing different tests to evaluate the impact of the codebook size on the performance, it was concluded that the codebook size should increase as the block size (i.e. the number of pixels in the block) increases.

For this purpose, here we define a Codebook Size Factor (CSF), which determines the number of codevectors in each codebook. This number is selected from the set $CSF = 1, 2, 4, 8, 16, 32$ and determines the number of codevectors in the codebook of a block size $W \times H$ as:

$$N_{W \times H} = CSF \times W \times H, \tag{5.1}$$

Another perspective of the ILR-VQ performance depends on LBG. Here, we also evaluate the convergence of LBG and its impact on the final performance of ILR-VQ in different configurations.

All the results provided in this section are obtained from an implementation within the JEM5 reference software. For this purpose, an intra algorithm competition has been added to the code in order to provide JEM with a choice between ILR-VQ algorithm and regular intra coding algorithm, as discussed in Chapter 3.

Codebook size

Figure 4-8 presents the impact of the codebook size for 4×4 blocks on the BD-R gain. As can be seen, there is a general tendency toward higher BD-R gain when the codebook size is increased. However, there are certain sequences on which, using ILR-VQ with smaller codebooks performs better. This can be due to the fact that contents of those sequences might not require an excessively diverse set of ILR candidates in the codebook. Therefore, using large codebooks for them would just impose an unnecessary rate overhead, as codevector index range becomes larger with higher codebooks.

Another interesting result presented in Figure 4-8 is the significantly high performance of ILR-VQ on screen contents. More precisely, the three sequences of SlideShow, SlideEditing and ChineseEditing that contain pure screen content, provide about 1.5% gain. This can be justified by the nature of these videos, where edges are sharp and contain patterns which are relatively more regular than natural contents. In this situation, a single ILR value at one pixel can completely compensate a content change which is difficult to be coded by the regular intra coding algorithm.

Complexity of larger blocks and codebooks

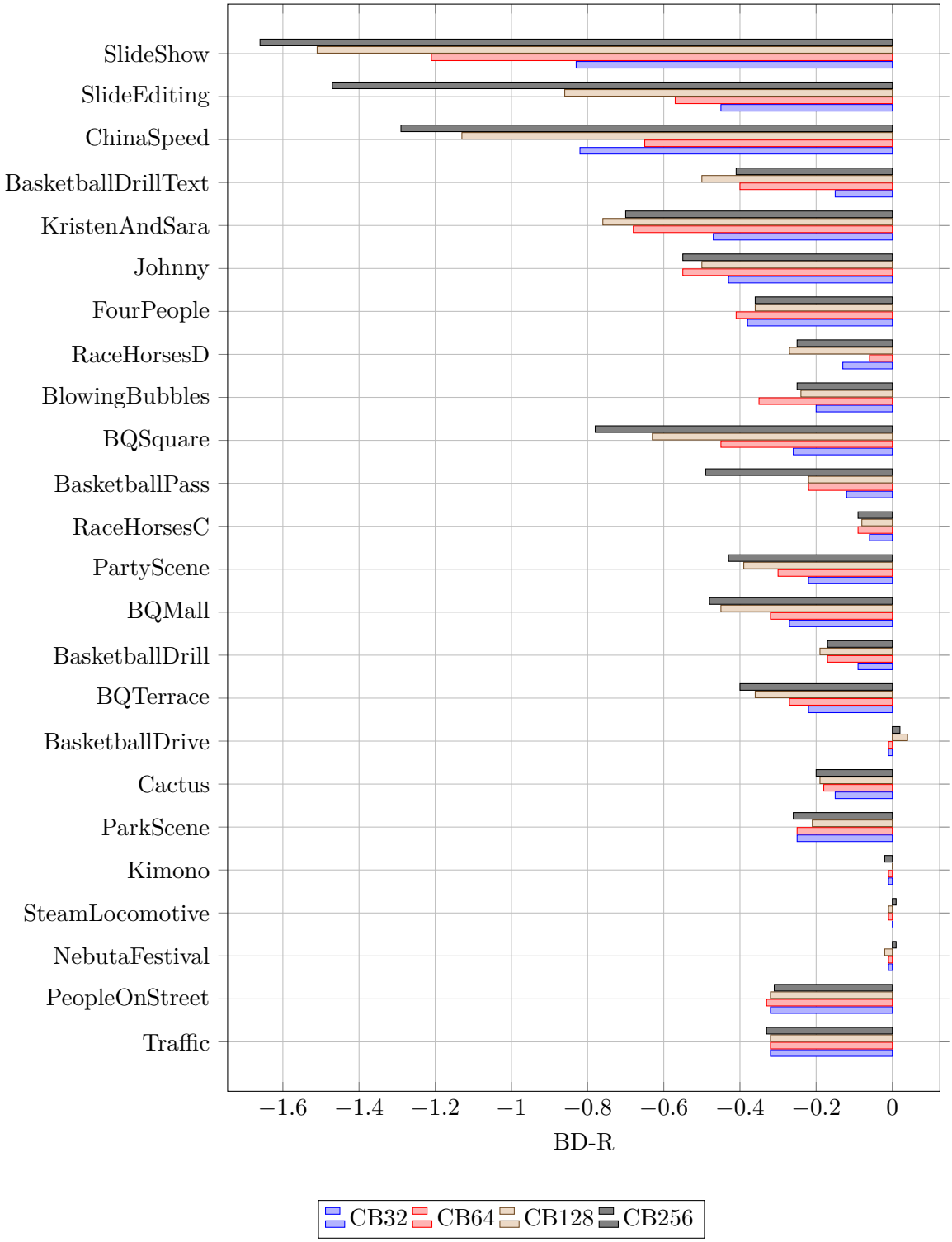
A compromise between BD-R and complexity is crucial for the optimum ILR-VQ design. There are two types of complexities involved regarding this issue which are directly impacted by block size and codebook size: 1) Encoding time complexity and 2) Encoder/Decoder memory overhead. For this purpose, Table 4-1 provides a summary of performance and complexity when excessively large blocks and/or codebook are used. In this table, "EncT" is the encoder side complexity and "Mem" in terms of encoder run-time and memory. It is important to note that in order to achieve the BD-R values in this table, the modified LBG algorithm has been performed multiple times until convergence.

As can be seen in Table 4-1, a BD-R gain of about 1% can be achieved with an unfeasible complexity overhead. Based on these results, the recommended configuration of ILR-VQ determined to be the use of only 4×4 blocks $CSF = 8$, which results in a single codebook of size $4 \times 4 \times 8 = 256$. This configuration is reasonable compromise between BD-R and complexity.

LBG iterations

Achieving the full potential of a given codebook size strongly relies on the quality of optimized codebooks. To guarantee that nothing is missed in the codebook optimization step, adequate number of iterations have been performed. Figure 4-9 shows the evolution of average BD-R gain in different codebook sizes for 4× 4 blocks. In this experiment, convergence was interpreted both in terms of BD-R gain consistency and codevector updates. More precisely, LBG iterations have

Figure 4-8: BD-R gain of ILR-VQ with optimized codebooks in different sizes.



been continued until a point that no change in BD-R gain was visible and also, the update step of LBG would not significantly change codevectors.

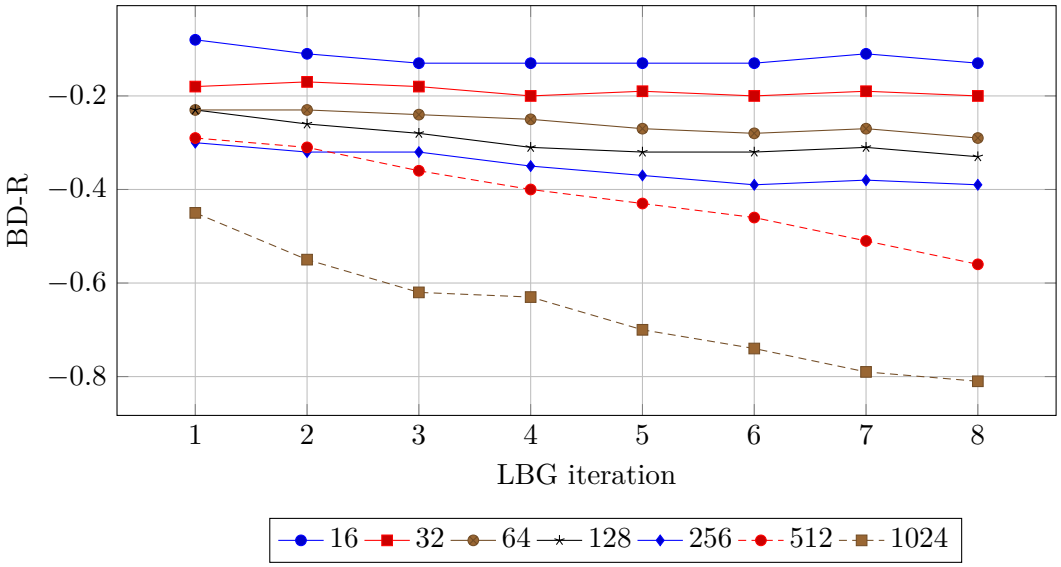
6 Conclusion

In this chapter, an intra coding algorithm based on the framework of Chapter 3 was proposed. This algorithm, called In-Loop Residual coding with Vector Quantization (ILR-VQ), benefits

Table 4-1: Summary of the performance and complexity in different block sizes and codebook sizes.

CSF	Only 4x4			Up to 8x8		
	BD-R	EncT	Mem	BD-R	EncT	Mem
1	-0,19	117%	16	NA	NA	NA
2	-0,26	126%	32	-0,29	210%	288
4	-0,32	137%	64	-0,41	320%	576
8	-0,4	165%	128	-0,65	630%	1152
16	-0,45	216%	256	-0,95	2300%	2304
32	-0,76	500%	512	NA	NA	NA
64	-0,9	2000%	1024	NA	NA	NA

Figure 4-9: Convergence of the modified LBG algorithm with different codebook sizes.



from a VQ-based technique to transmit the ILR signal to the decoder.

The first part of this chapter presented a brief introduction of VQ to give the necessary theoretical background for understanding the proposed method. In this part, the main goals of VQ and its potential compression efficiency were introduced. Furthermore, the impact of the codebook size on compression in different situations was briefly discussed.

In the second part, key components of the proposed ILR-VQ algorithm were introduced with detail. For this purpose, first, a perspective of expected rate-distortion behavior of ILR-VQ was provided with respect to potentials of VQ as well as limitations of the intra coding problem. Then, the block prediction algorithm of ILR-VQ, given an arbitrary codebook was explained. For codebooks optimization part, the Linde-Buzo-Gray (LBG) algorithm was used. This limitation has led us to design a specific optimization algorithm that complies with the characteristics of our problem.

CHAPTER 5

ILR-IP with scalar quantization

Contents

1 Principles of Scalar Quantization

Scalar Quantization (SQ) is usually considered as the counterpart of Vector Quantization (VQ). As discussed in chapter 4, the main goal of quantization is to compress symbols from a data source in a reproducible manner, so that a decoder would be able to approximate the input symbol with a reasonable error. More precisely, a random variable x is distributed over some source alphabet X, and we would like to encode it efficiently while ensuring that we can then reproduce another variable y which is distributed over a reproduction alphabet Y and approximates x [70]. The quality of the approximation is determined by a distortion measure:

$$d : X \times Y \to [0, \infty]. \tag{1.1}$$

where $d(x, y)$ represents the loss incurred when a source value $x \in X$ is reproduced as $y \in Y$. Since X and Y are implicitly specified by d, the quantization problem is determined by the pair (x, d).

Let (x, d) be a quantization problem with source alphabet X and reproduction alphabet Y. To approximate x, the encoder represents a source value $x \in X$ by a binary message which the decoder maps to some quantized value $y \in Y$. A scalar quantizer is therefore a mapping $q : X \to Y$ such that $|q(X)|$, the number of possible quantized values, is either finite or countably infinite.

In the context of video coding, scalar quantization is the standard manner to compress transform coefficients. The quantization problem in the transform coding is defined as a function that takes an individual coefficient amplitude and maps it into an approximation which is less costly to transmit.

In this chapter, an idea similar to transform coefficient compression is adopted to represent the ILR signal.

2 Block prediction with baseline ILR-SQ

In this section, the main elements of the block prediction process by ILR-SQ are introduced. The inputs to this process are a set of regular reference pixels from the previous blocks as well as a scan order for traversing the pixels within the block. The expected outputs are a prediction signal and an ILR signal. This algorithm scans the block according to the given order and and performs three steps on each pixel position.

The block prediction algorithm of ILR-SQ consists of a scanning loop over all pixels within the block. At each position, three steps are performed in the following order:

1. Pixel prediction
2. Residual calculation
3. Pixel correction

Notation

As discussed in Chapter 3, the two terms of "reconstruction" and "correction" are differently used to refer to the additive error compensation by a residual. More precisely, "reconstruction" is used for error compensation by OLR and "correction" for ILR.

Moreover, the following notation is used in the rest of this chapter:

- R: the ILR signal that is computed and transmitted.
- R_o: the original residual signal, before quantization.

Figure 5-1: Three corrected reference pixels at left $c^{(l)}$, above $c^{(a)}$ and above-left $c^{(al)}$ for in-block pixel prediction at scan position (•), by LOCO-I.

- P: the prediction signal.
- C: the reconstructed/corrected signal, depending whether the pixel position falls inside or outside of block.
- O: the original signal for computing the residual signals.

Each of the above signals is represented as a vector of values and can be indexed by (•), where (•) can be any scan position within that block. For instance, $r^{(3)}$ is the 3^{rd} value in the ILR signal. Similarly, $c^{(al)}$ denotes the reconstructed/corrected pixel at above-left position of the current pixel.

2.1 Pixel prediction

The main idea of the ILR-based intra coding algorithms is to utilize in-block references, as close as possible to the predicted pixels. Given such short distance references, different predictor functions can be applied. One can simply use the conventional IPM set provided by standards (e.g. 67 IPMs of VVC). However, ILR-based algorithms mostly target contents with high level of texture complexity, on which conventional IPMs work quite inefficiently, as discussed in Chapter 3.

Similar to the ILR-VQ algorithm of Chapter 4, ILR-SQ benefits from context-based linear predic tors for pixel prediction. The LOCO-I algorithm is one of the most operational predictor function for integration in ILR-SQ. The important benefits of LOCO-I are its simplicity, in terms of implementation complexity, and efficiency, in terms of prediction accuracy.

To apply LOCO-I for pixel prediction by ILR-SQ, three short distance references are needed. Figure 5-1 shows how these references are located at left $c^{(l)}$, above $c^{(a)}$ and above-left $c^{(al)}$ of a pixel at (•). As discussed before, depending on whether(•) falls inside or outside a block, the short distance references can be selected from the previous blocks or the current block.

The prediction function LOCO-I is performed adaptively and based on texture type. Eq. 2.1 shows how three contextual situations are detected by LOCO-I. In this equation, the first and second conditions check the existence of an edge with a vertical or horizontal direction, respectively, and the last condition defines "no direction".

$$p^{(\bullet)} = \text{LOCO-I}(c^{(l)}, c^{(a)}, c^{(al)}) = \begin{cases} min(c^{(l)}, c^{(a)}), & \text{if } c^{(al)} \geq max(c^{(l)}, c^{(a)}). \\ max(c^{(l)}, c^{(a)}), & \text{if } c^{(al)} \leq min(c^{(l)}, c^{(a)}). \\ c^{(l)} + c^{(a)} - c^{(al)}, & \text{Otherwise.} \end{cases} \tag{2.1}$$

2.2 Residual calculation

Once the prediction value at (•) is computed, its original residual value $r_0^{(\bullet)}$ is computed by referring to the original signal O:

$$r_o^{(\bullet)} = o^{(\bullet)} - p^{(\bullet)}. \tag{2.2}$$

The original residual value $r_o^{(\bullet)}$ is able to losslessly correct $p^{(\bullet)}$. However, this requires an unfeasible transmission rate. To reduce its rate, a spatial domain quantizer is integrated in

ILR-SQ to perform a linear quantization on original residual values. Fig. 5-2 shows how original residual values are mapped into quantized values. In this figure, a quantization step size of $\Delta_{qs} = 64$ has been used. As this step will be further discussed in the next section, here we refer to it as a black box denoted by:

$$r^{(\bullet)} = Q(r_o^{(\bullet)}). \tag{2.3}$$

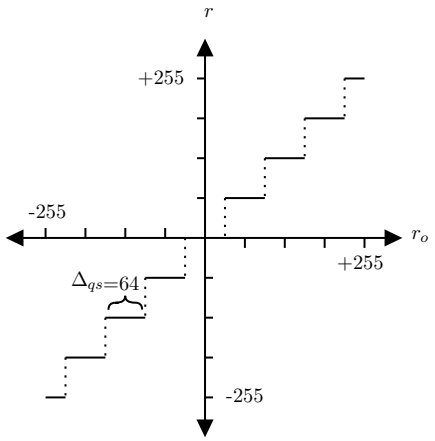


Figure 5-2: Linear quantizer in the spatial domain with quantization step size $\Delta_{qs} = 64$

2.3 Pixel correction

In the first row and the first column of each block, short distance references fall outside of block and are chosen among decoded pixels which are fully reconstructed. However, in the rest of the block, short distance references must be selected from the same block. Therefore, this step takes the residual value that is computed with Eq. 2.3 and corrects the prediction by:

$$c^{(\bullet)} = p^{(\bullet)} + r^{(\bullet)}. \tag{2.4}$$

At this stage, pixel prediction at $(\bullet)$ is finished. This means that, in the similar processes for the next pixels in this neighborhood, corrected value $c^{(\bullet)}$ can serve as in-block reference.

3 ILR quantization in the spatial domain

As discussed in the previous sections, the quantization Q plays a key role in the functionality of ILR-SQ. In fact, Q applies the lossy aspect of the compression by ILR-SQ and consists of a pixel-level quantization by a uniform step size Δ_{qs}, in the spatial domain. Such quantization by ILR-SQ should comply the rate-distortion optimization. In other words, in each QP value, the amount of information loss caused by the function Q must be in the same range as the regular quantization in the transform domain. However, compared to the transform domain quantization, Q deals with an input which is different in nature. Therefore, it cannot simply utilize the same quantization step size as Δ_{qt}. As this problem has already been studied in the Transform Skip Mode (TSM), we simply adopt its quantization scheme for ILR-SQ [54, 55, 71, 72].

The fundamental difference between the spatial and transform domain inputs is in their range. More precisely, all transformations (e.g. DCT, DST etc) apply a scaling step on their input signal and map them into a new dynamic range. Subsequently, Δ_{qt} has been properly adapted to this scaling by applying a re-scaling in the calculation. However, spatial domain

quantization of ILR-SQ does not perform the above scaling. This means that the re-scaling step of Δ_{qt}, must be removed from its calculation in order to guarantee a dynamic range compliant with the QP.

In order to neutralize the above scaling by Δ_{qt} and map the signal in a proper dynamic range before spatial domain quantization, a normalization is applied to obtain Δ_{qs}:

$$\Delta_{qs} = (\Delta_{qt}.bscale)/2^{bshift}, \tag{3.1}$$

In Eq. 3.1, *bscale* is calculated as *scale*(width).*scale*(height) using the factors presented in Table 5-1. Moreover, *bshift* is calculated as $s_1 + s_2 + \delta$, where δ denotes internal bit depth increment relative to 8-bit, and s_1 and s_2 are bit shifts of the inverse transforms on columns and rows, respectively:

Table 5-1: Scaling factors of Δ_{qs} for different block sizes.

	width/height			
	4	8	16	32
scale	128	181	256	362

Usually, the quantization has a know linear impact on rate and distortion. However, the impact is not linear due to the in-block pixels dependency.

4 ILR transmission

Similar to ILR-VQ algorithm of Chapter 4, ILR-SQ has to transmit its ILR signal. This will allow the decoder to perform the exact same prediction process using in-block reference pixels corrected by the transmitted ILR.

The transform coefficient coding module of VVC performs the exact same functionality as is expected from the ILR coding module. Therefore, technically, it is able to code ILR signal as well, as in both cases, signed quantized coefficient blocks are coded. However, there are fundamental differences between the two inputs, which makes it inefficient to adopt the same coding module for ILR signal coding. As explained in Chapter 2, the transform coefficient coding module of VVC is highly optimized to perform specifically on transform coefficient blocks. This has been carried out by using technologies to exploit the energy compaction property of the input signal due to the transformation step; a property that the ILR signal does not necessarily possess. Therefore, sharing the existing module for both uses is inefficient not only for ILR-SQ, but also for transform coefficient coding, too. This is due to the fact that the source separation by shared CABAC context would become poorer, which results in a higher rate. Although deploying more CABAC context models might partly address this issue, it would impose an undesirable complexity overhead. In the following section, the proposed ILR coding module is presented, which uses different binarization and coding methods than the regular transform coefficient coding of VVC.

4.1 Binarization

The binarization of a coding symbol does not impact the coding efficiency, as long as the entropy of that symbol is "fully" exploited. This can be achieved by utilizing the finest possible source separation on a symbol in the signal. However, such source separation scheme can be quite expensive in terms of complexity. Therefore, depending on the limitations of a coding system, the best binarization scheme might be different.

In video coding, the cost of finer source separation of a symbol is the complexity overhead of having extra CABAC context models. This overhead can be interpreted both in hardware and software implementation of the codec. To control this complexity overhead, it is always preferred to keep the number of contexts as low as possible.

The main purpose of binarization in data compression is to represent most of the energy in as few number of of bins as possible. This directly involves the statistical distribution of the data. In ILR-SQ, experiments show that quantized coefficients rarely become excessively large.

In ILR-SQ, unary codes are used for binarization. For this purpose, each quantized ILR amplitude m, calculated from its corresponding ILR coefficient r as an integer value of $m = \text{abs}(r/\Delta qs)$, is represented by $m+1$ bins $b_0 b_1 ... b_m$ where abs is the absolute function. Each bin b_i is calculated as:

$$b_i = \begin{cases} 1, & \text{if } i < m. \\ 0, & \text{if } i = m. \end{cases} \tag{4.1}$$

Fig. 5-3 visualizes an example of binarization by unary codes on a 4×4 block.

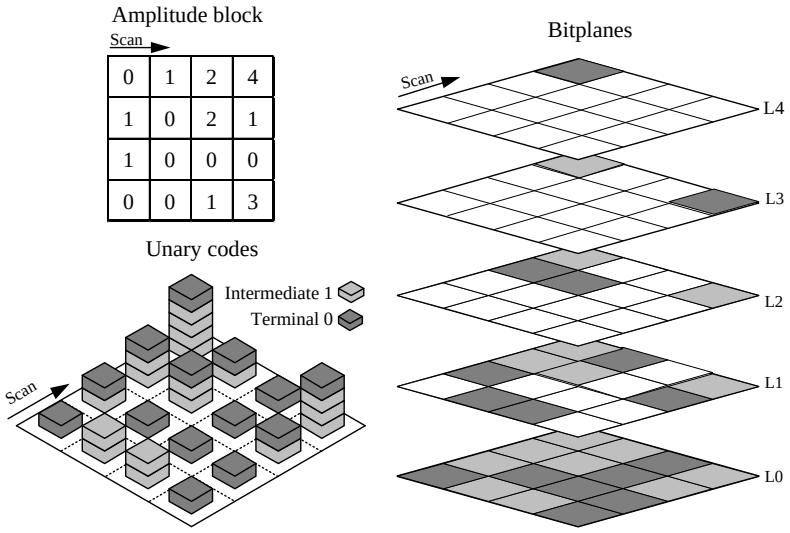


Figure 5-3: Binarization of a 4×4 block of unsigned quantized ILR amplitudes with unary codes.

4.2 Coding

All bins corresponding to amplitudes, calculated in previous step, are compressed in ILR-SQ. For this purpose, a simple layer-based context derivation method is used for ILR bin compression. In this lossless method, a total of N context models have been added to the existing CABAC engine of VVC.

Unlike amplitude bins, coefficient sign bins are not compressed ILR-SQ. Instead, the entropy coding is bypassed and equi-probable bin coding is used. Usually, bypassed mode is used when the binary PDF of a bin is close to $P(0) \simeq P(1) \simeq 0.5$. In this mode, which is also used in VVC transform coefficient sign coding, each bin takes exactly 1 bit due to the fact that the entropy is not exploited. As ILR coefficients signs follow a same statistical behavior, the bypass mode was determined to be used for their coding.

5 Advanced tools

In this section, three additional tools are presented on top of the baseline configuration of ILR-SQ. The use of these tools is completely independent from the main body of ILR-SQ and its functionality. In other words, one can simply switch on/off any of these tools without making any change on the main algorithm of ILR-SQ.

5.1 Rate-Distortion Optimized Spatial Quantization (RDOSQ)

Rate-Distortion Optimized Quantization (RDOQ) has been widely used in different video codecs. RDOQ is a non-normative method due to the fact that it does not change the stream syntax of standards [73, 74]. Instead, RDOQ allows making better decisions within the scope of a predefined syntax. More precisely, by using RDOQ, an encoder can opt for choosing a different quantized level than the closest one to each coefficient.

For example, assume that an original transform coefficient of 11 is sent to a linear quantizer with $\Delta = 5$, which allows the following quantized levels: $\{..., 2.5, 7.5, 12.5, ...\}$. Without any RDOQ strategy, the quantized level should be decided by mapping 11 to the closest quantized, which is 12.5. Applying such quantization strategy to the whole transform block would guarantee the minimum distortion (i.e. information loss) compared to the original block. However, a lower distortion does not necessarily mean a more efficient coding, as the rate-distortion cost is also influenced by the rate. More specifically, it is technically very common to achieve a lower rate-distortion cost by increasing the distortion in order to save rate.

In conventional RDOQ, the decision about quantization level of each transform coefficient does not have any impact on other coefficients. In fact, the conventional RDOQ process is completely independent from prediction and residual calculation processes.

The dependency of in-block pixel correction in the ILR-SQ makes the conventional RDOQ inefficient for ILR quantization. This is due to the fact that ILR quantization process is not independent from prediction process. On the contrary, they are both carried out in an interleaved manner in the block level. Therefore, to compute the rate-distortion cost of each alternative quantized level for an ILR coefficient, one must redo the prediction process, since the new quantized level results in a new corrected pixel, hence a new in-block prediction reference.

The proposed Rate-Distortion Optimized Spatial Quantization (RDOSQ) integrates testing of alternative levels within the prediction process of ILR-SQ. For this purpose, RDOSQ performs a loop over all scan positions in the block. At each iteration, corresponding to one scan position, the entire baseline ILR-SQ algorithm is performed in a similar way than explained in the previous section, except that the function Q in Eq. 2.3 is modified to operate with a quantization map with three functionalities on original residuals r_o: ceiling (cl), flooring (fl) and rounding (rn).

The quantization map enables ILR-SQ to test alternative levels and has three partitions: current, past and future. The current position is the scan position that is associated to the ongoing iteration, while past and future are defined relatively to this scan position. The goal of the iteration at each scan position ($\bullet$) is to decide the optimal functionality of Q as $q^{*,(\bullet)}$. This choice is made among two alternatives "cl" and "fl", given the optimal functionalities in the past and the default "rn" functionality in the future. Given these instructions, ILR-SQ is performed and the rate distortion cost of both alternatives are computed as $J^{cl,(\bullet)}$ and $J^{fl,(\bullet)}$. Finally the optimum level is chosen with respect to the rate-distortion costs and stored for future iteration.

Figure 5-4 visualizes an example of RDOSQ for a simplified 1×4 block. In this figure, four iterations corresponding to four scan positions are presented in columns. The scan positions are provided under the x-axis of the first plot. At each column, two tests are performed according to their quantization maps showed above their x-axis. As can be seen, each iteration computes $q^{*,(\bullet)}$ and passes it to next iteration. This resulted in a final optimum quantization map of $q^* = \{fl, fl, cl, fl\}$

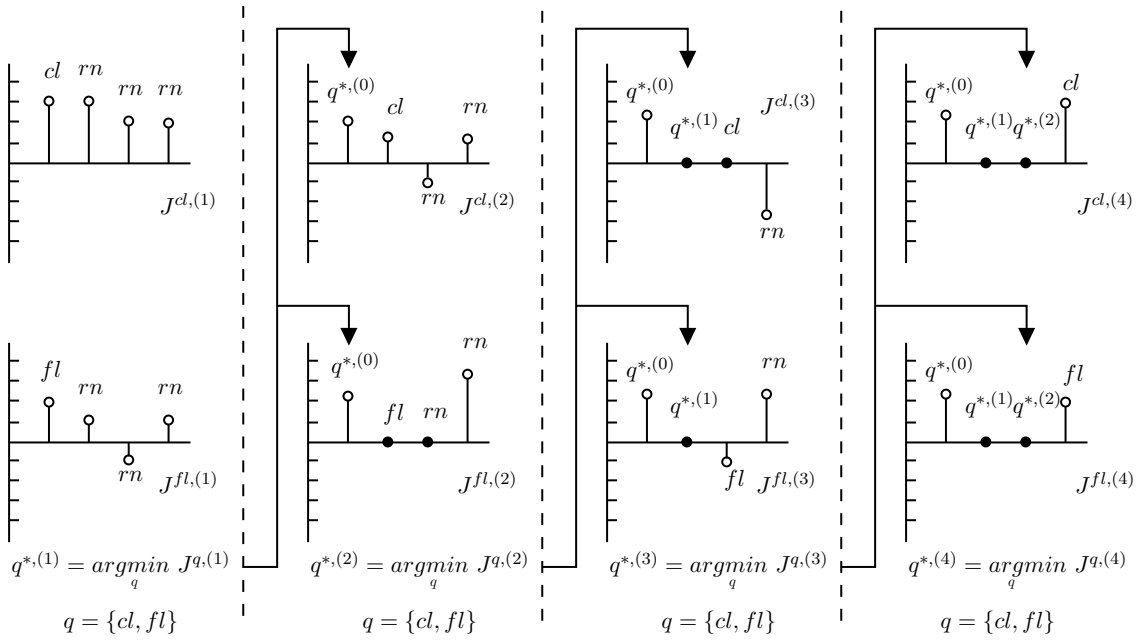


Figure 5-4: An example of using RDOSQ on a 1×4 block. Each column corresponds to one pixel position on the x-axis and contains two diagrams for the two quantization decisions of that pixel (i.e. ceiling *cl* and floor *fl*). The encoder chooses one of the two quatnization decisions by optimizing the rate-distortion cost J and advances to the next pixel position. It is important to note that at each diagram, the previous pixel positions use their optimal quantization decisions, while the future ones simply use rounding (rn).

5.2 Hierarchical coded block flag (CBF) tree

Zero values in quantized coefficients can cost an unnecessary rate, if not properly handled. In conventional transform coefficient coding, two main tools are used to address this problem: 1) last significant position coding and 2) Coded Block Flag (CBF) [46].

Unlike the last significant position coding, that is only beneficial for transform coefficients, the CBF coding can be adopted for ILR-SQ. For this purpose, a hierarchical CBF coding tree is used. In this method, the same idea as the regular CBF is applied on square sub-blocks. Fig. 5-5 shows an example of using the proposed hierarchical CBF tree on an 8×8 block, capable of signaling the CBF flag down to 2×2 blocks.

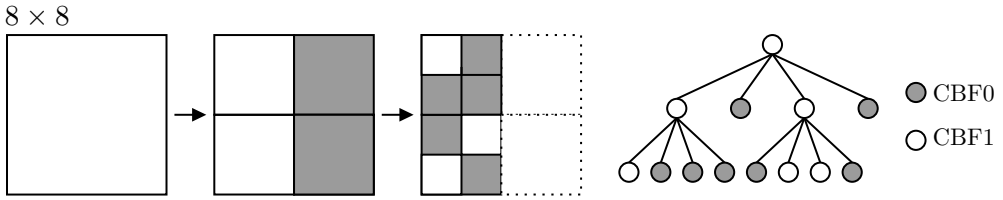


Figure 5-5: CBF tree of an 8×8 block with the depth of 2.

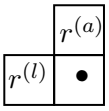
5.3 Context derivation based on neighborhood

Further source separation in unary bins coding is used. Experiments show that a reasonable amount of correlation usually remains in the unary codes of neighboring amplitudes in a block.

More precisely, an ILR amplitude is more likely significant when surrounded mostly by significant neighbor amplitudes than non-significant ones.

To exploit the remaining correlation in ILR unary codes, new contextual situations are defined. Each contextual situation depends on the other ILR values in the neighborhood. There are various ways to define such situations for source separation of bins. However, a proper way must make a compromise between the complexity, in terms of number of added CABAC context models, and the coding efficiency.

Fig. 5-6 shows an example CABAC context derivation for significance bit which is the unary bin b_0 in Eq. 4.1. In this figure, two neighboring ILR values at left $r^{(l)}$ and above $r^{(a)}$ have been used to define three situations.



$r^{(a)}$	nsig	nsig	sig	sig
$r^{(l)}$	nsig	sig	nsig	sig
ctx	1	2	2	3

Figure 5-6: Proposed context derivation for the significance bit at position (•).

6 Results

The proposed ILR-SQ algorithm has been implemented in two different VVC-related reference softwares, namely Joint Exploration Model (JEM) and VVC Test Model (VTM). Table 5-2 summarizes the results provided in this section. As can be seen, apart from the reference software, different coding modes as well as test sequence sets are used for the result demonstration in this section.

Software	Natural			Screen		
	AI	RA	LD	AI	RA	LD
JEM	Figure 5-7			Figure 5-9		
VTM	Figure 5-8			Figure 5-10		

Table 5-2: Summary of results in the JEM and VTM with three coding modes of All Intra (AI), Random Access (RA) and Low Delay (LD) and on two different content types (natural and screen).

Due to the structural differences between the JEM and VTM in terms of implementation, the ILR-SQ specifications in these two references softwares are different. Table 5-3 shows 5 tools/parameter that are used in each software. In this table, the "Block size" column indicates the maximum size for which the ILR-SQ has been activated. The impact of this parameter has been evaluated separately and will be presented in next section. Moreover, the Out-Loop Residual (OLR) column indicates whether or not the regular residual has been used along with the ILR. As will be discussed, this parameter also has a specific behavior when integrated with the ILR-SQ.

6.1 Compression efficiency

In this section, a performance comparison is carried out by different implementations of ILR-SQ is provided in terms of BD-R. Tables 5-7 to 5-10 compare extensively the BD-R performance of

	Block size	RDOSQ (Sec. 5.1)	OLR	Hier. CBF (Sec. 5.2)	Cxt. derivation (Sec. 5.3)
JEM	4×4	Used	Used	Used	Not used
VTM	32×32	Not used	Not used	Used	Used

Table 5-3: The ILR-SQ specifications used in JEM and VTM.

ILR-SQ according to the summary presented in Table 5-2.

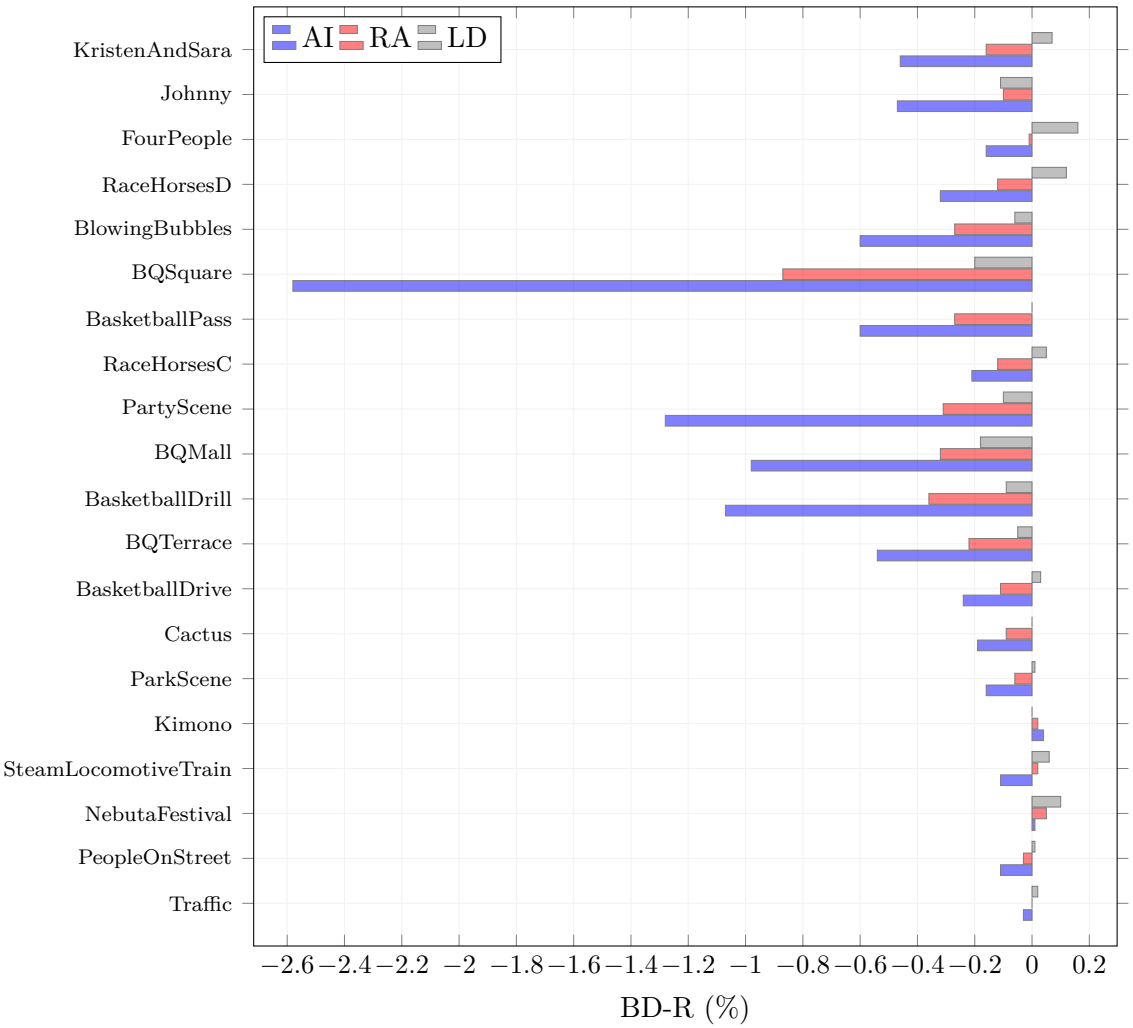


Figure 5-7: BD-R (%) performance the ILR-SQ, implemented on top of the JEM and applied on natural content sequences. Negative values of BD-R indicate compression gain achieved by the ILR-SQ.

Natural content versus screen content

The ILR-SQ performance is significantly superior on screen content than on natural content. This is mainly due to sharper edges in screen content, which makes the ILR correction more efficient. More precisely, the rate cost of ILR signal has a larger reward in terms of prediction accuracy. For instance, assume two in-block content changes in Figure 5-11. In the screen content block, the ILR signal is very small, yet efficient in adapting to the sharp in-block change.

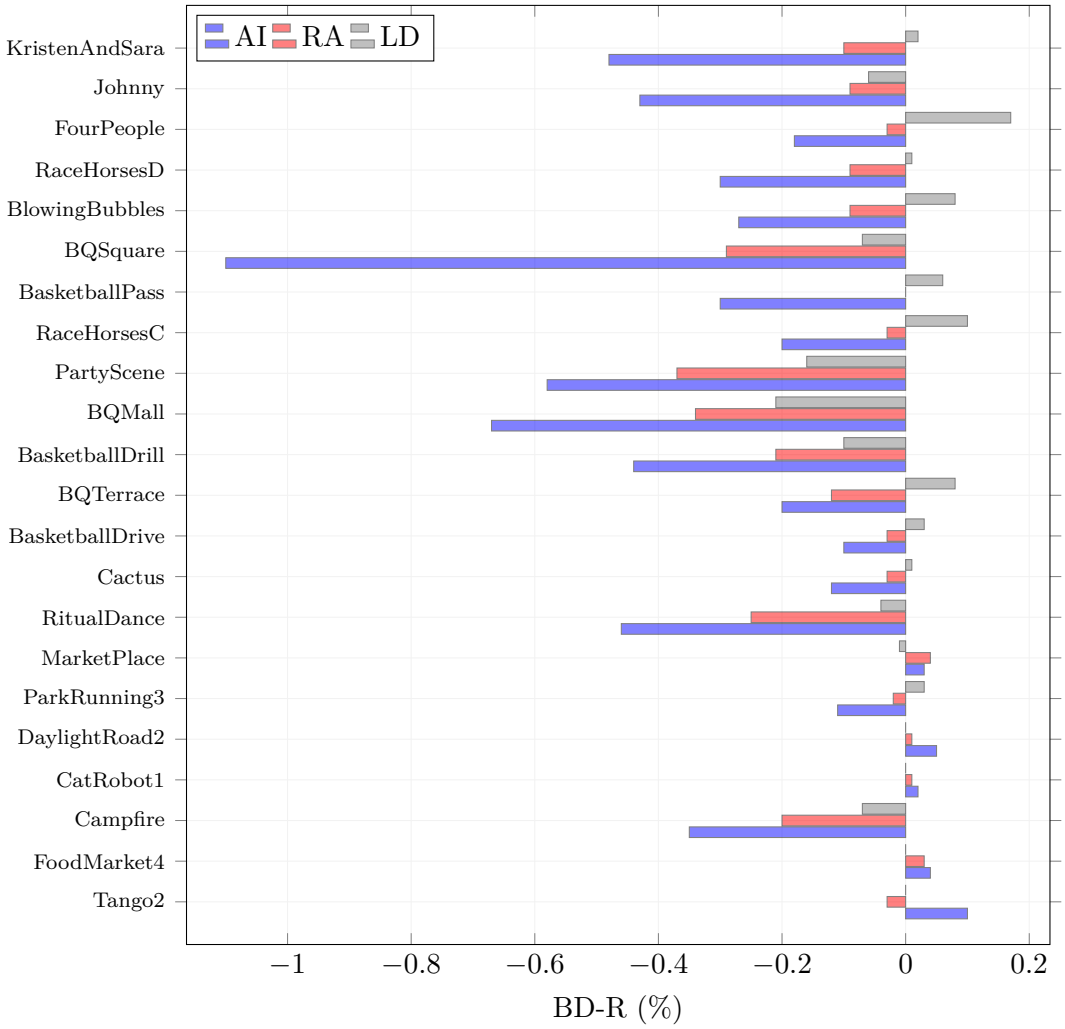


Figure 5-8: BD-R (%) performance the ILR-SQ, implemented on top of the VTM and applied on natural content sequences. Negative values of BD-R indicate compression gain achieved by the ILR-SQ.

However, the content change in the natural block is too complicated and consequently, its ILR signal is very big.

The VTM version versus the JEM version

The slight difference between the VTM and JEM versions of ILR-SQ lies on many factors. First, JEM and VTM basically benefit from variant tools which have different impacts on the performance of ILR-SQ. For instance, block partitioning of VTM is more sophisticated than that of JEM, while in JEM a more efficient residual transformation algorithm is used.

6.2 Complexity

As was discussed in previous chapters, the modifications in the intra prediction domain, proposed by the ILR framework, may impose both encoder and decoder side complexities. In this section, we present this side effect in different configurations. It is useful to note that for all experiments, we have used the VTM version of ILR-SQ and we compare the complexity to the pure VTM. Numbers are all in the form of percentage of run-time and values less than 100% indicate a complexity (run-time) reduction by ILR-SQ and vice versa.

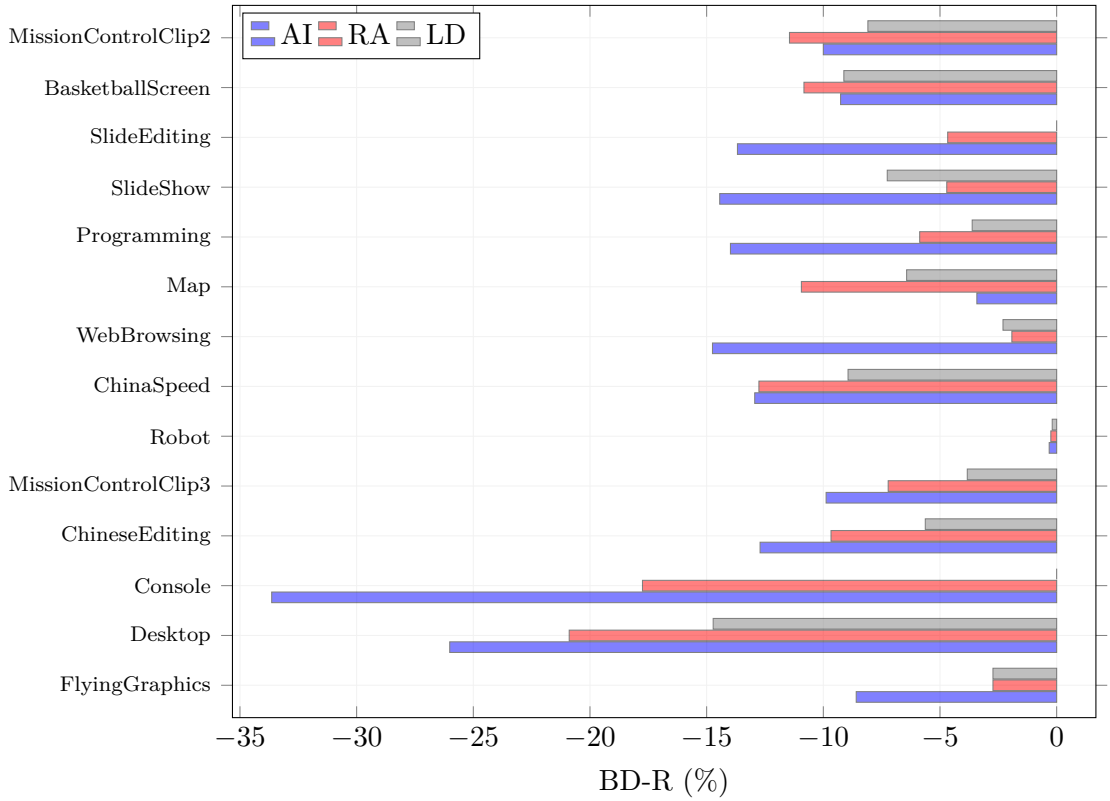


Figure 5-9: BD-R (%) performance the ILR-SQ, implemented on top of the JEM and applied on screen content sequences. Negative values of BD-R indicate compression gain achieved by the ILR-SQ.

Encoder side complexity

The encoder side complexity is mainly caused by the block-level algorithm competition. More precisely, an encoder which is integrated with ILR-SQ has to perform an additional comparison on top of its IPM search algorithm, in order to decide whether or not the best IPM is still better than ILR-SQ. The overall complexity overhead of ILR-SQ depends on two man factors: 1) How often does the encoder performs the above additional test? 2) How complex is each trial of ILR-SQ?

Impact of the maximum ILR-SQ block size on the encoder complexity

One of the important factors impacting on the first above aspect is the maximum eligible block size of ILR-SQ. Usually, with activating larger block sizes for ILR-SQ, we make the encoder more complex. However, there are some rare situations in which allowing a larger block size for ILR-SQ would reduce the encoder side complexity. For instance, assume a certain content on which a 32×32 ILR-SQ block can perfectly work. It is likely that the low rate-distortion cost of ILR-SQ at this block size would stop the encoder to check smaller block sizes for both ILR-SQ and the regular intra algorithm. However, if one limits the maximum block size to 8×8 for the same content, the block partitioning may stop later, causing more encoder side complexity on that part of the video. Experiments show that this situation happens quite rarely. Table 5-4 summarizes the impact of maximum allowed ILR-SQ block size on the encoder complexity in different classes.

In addition to the aforementioned impact of allowing larger block sizes on the complexity growth, another impact can also be seen in Table 5-4. In each column, corresponding to a certain

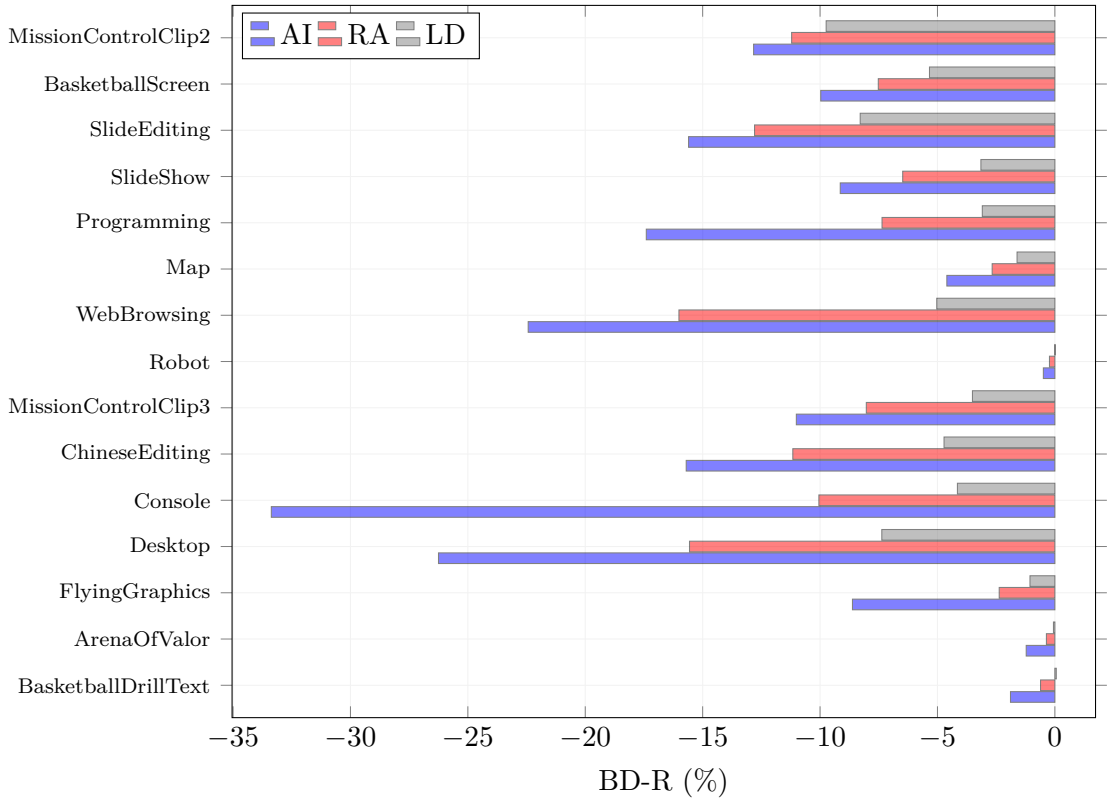


Figure 5-10: BD-R (%) performance the ILR-SQ, implemented on top of the VTM and applied on screen content sequences. Negative values of BD-R indicate compression gain achieved by the ILR-SQ.

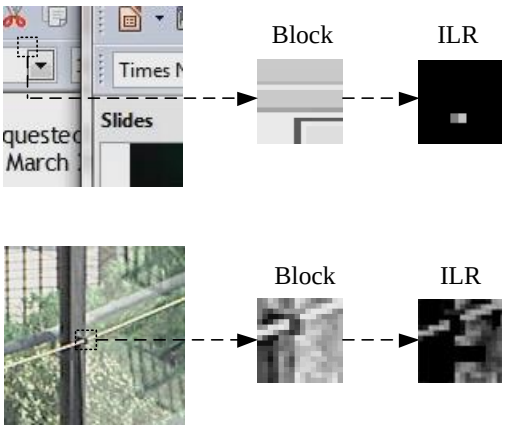


Figure 5-11: Comparison of ILR signal when an in-block content change happens in natural and screen contents. Bright pixels in the ILR signals indicate larger amplitudes.

maximum block size of ILR-SQ, the encoder complexity seem to be larger on low resolution classes (see Appendix B for more details on CTC classes). The justification for this behavior is that in high resolution classes (e.g. A1 and A2), the encoder stops partitioning at larger block sizes than the low resolution classes (e.g. C and D). Therefore, it happens more often that ILR-SQ is never tested for a parts of a high resolution sequence.

Sequence	Maximum ILR-SQ block size				
Class	4×4	8×8	16×16	32×32	64×64
Class A1	101%	101%	102%	102%	102%
Class A2	101%	102%	102%	104%	104%
Class B	103%	103%	103%	104%	105%
Class C	105%	106%	106%	109%	110%
Class D	107%	110%	112%	113%	112%
Class E	103%	105%	104%	106%	106%
Class F	116%	115%	119%	121%	120%
Class TGM	113%	121%	124%	127%	128%

Table 5-4: The impact of the maximum ILR-SQ block size on the encoder complexity of the VTM.

ILR-SQ imposes higher encoder complexity on screen content sequences of classes F and Text-Graphic-Motion (TGM). The experiments show that this can be due to the fact that regular transform coding of residual signal performs poorly on screen content. Therefore, the encoder usually has to continue the block partitioning until the smallest size (i.e. 4×4), which requires more ILR-SQ tests in the block level.

Impact of the Quantization Parameter (QP) on the encoder complexity

The QP has an indirect impact on the frequency of checking the ILR-SQ test. The main connection between the QP and the frequency of the ILR-SQ tests, is the block partitioning. In general, smaller block sizes are checked less often when relatively large QP values are selected. This means in these QP values, the ILR-SQ algorithm is performed less often and therefore, the encoder side complexity overhead is reduced. Table 5-5 shows how the encoder complexity decreases in larger QP values.

QP	Maximum ILR-SQ block size				
	4×4	8×8	16×16	32×32	64×64
22	108%	107%	108%	109%	110%
27	105%	105%	104%	107%	107%
32	102%	105%	104%	105%	108%
37	101%	101%	102%	101%	103%

Table 5-5: The impact of the QP on the encoder complexity of the ILR-SQ, integrated within the VTM.

Decoder side complexity

The decoder side run-time of ILR-SQ depends on two processes: 1) Residual parsing/decoding. 2) Block prediction. Experiments show that the ILR-SQ is less complex in the later domain, and slightly more complex in the latter one. In overall, the decoder of ILR-SQ is less complex that the regular intra decoder. Table 5-6 compares the complexity of the ILR-SQ decoder in different QP values and different test types of contents (i.e. natural and screen).

Sequence type	QP			
	22	27	32	37
Natural	99%	99%	95%	93%
Screen	81%	78%	74%	73%

Table 5-6: The decoder side run-time of ILR-SQ in different QP values and content types.

6.3 Statistics

Introducing a completely different intra coding mode by ILR-SQ has made significant changes in coding the statistics of JEM and VTM. In this section, some notable aspects of these changes are presented.

Selection rate

The frequency of selecting ILR-SQ mode for blocks is the main indicator of its performance. More precisely, when there are not many ILR-SQ blocks in a sequence, no BD-R gain should be expected. On the contrary, having many selected blocks means that ILR-SQ was successful in the rate-distortion cost reduction of those blocks. In this situation, it can be expected that the algorithm flag overhead has been compensated and even further BD-R gain is achieved on top of it. To prove this point, Table 5-7 shows the ILR-SQ selection rate (i.e. the algorithm flag PDF) for four sequences with QP of 22 and 37, where two sequences (RaceHorsesC and Robot) had a poor performance and the other two (BQMall and Console) had a solid performance with the ILR-SQ. Each number in this table is the percentage of ILR-SQ blocks in the given block size and corresponds to the sequences QP at that row and column, respectively. These statistics were taken from the VTM implementation of ILR-SQ. In two sequences of BQMall and Console on which ILR-SQ has a high BD-R gain, the selection rate of ILR-SQ is high with different block sizes. On the contrary, the other sequences, RaceHorsesC and Robot, have a low selection rate which explains the poor performance of ILR-SQ.

Selection Rate (%)									
Type	Sequences	QP 22				QP 37			
		4×4	8×8	16×16	32×32	4×4	8×8	16×16	32×32
Natural	RaceHorsesC	7.3	0.5	0.0	0.0	6.3	4.9	0.0	0.0
	BQMall	16.2	4.2	7.1	5.1	16.9	9.2	9.3	4.2
Screen	Robot	11.3	3.0	0.4	0.0	3.6	4.7	4.8	0.0
	Console	45.6	31.5	19.3	26.0	43.8	39.6	28.4	35.0

Table 5-7: Selection rate of ILR-SQ in different block sizes. The poor performance of ILR-SQ on RaceHorseC and Robot sequences is reflected in their low selection rate, when compared to their counterparts (i.e. BQMall and Console, respectively). The numbers are obtained from the VTM implementation of ILR-SQ.

Visual inspection

Most ILR-SQ blocks are among high complexity textures. This is shown in Figure 5-12. In this figure, which is generated from the VTM version of the ILR-SQ, the block partitioning of one natural and one screen content sequence, coded with two QP values, are shown in different

sub-figures. In each sub-figure, gray and white rectangles indicate blocks coded by the regular and ILR-SQ intra algorithms.

Figure 5-12: Visualizing ILR-SQ selected blocks in a natural (BQTerrace) and a screen content (SlideShow) sequence, in 2 different QPs.

As can be seen in Figure 5-12, there are also plain areas in screen content videos that are coded by ILR-SQ blocks. In first glance, this seems to be in contrast with the initial claim of the ILR framework, discussed in Chapter 3. However, there is an interesting justification for this behavior. Statistically, in almost all plain blocks coded by the ILR-SQ, the Coded Block Flag (CBF) of the ILR-SQ is set to zero (i.e. the ILR signal is zero). As discussed in Section 5.2, this would result in applying the LOCO-I predictor without the correction step by ILR signal. In this condition, the LOCO-I performs its internal edge detection scheme on the entire block, without needing to transmit an angular mode (i.e. IPM). By reminding the fact that edges are much sharper in screen contents than in natural contents, we can conclude that ILR-SQ is being selected for these plain areas, since it is able to perform the same prediction while saving the IPM rate.

To elaborate the above impact, Figure 5-13 visualizes the spatial distribution of two different

types ILR-SQ blocks: CBF0 and CBF1. In this figure, white and dark gray rectangles indicate ILR-SQ blocks with CBF0 and CBF1, respectively, while bright gray ones still indicate blocks coded with the regular intra algorithm. By comparing each sub-figure with its original image on top, it can be confirmed that IRL-SQ blocks with CBF0 occur mostly on plain areas and screen content sequences.

Figure 5-13: Visualizing ILR-SQ selected blocks with CBF0 and CBF1, along with regular intra blocks, in two screen content sequences.

Impact of ILR-SQ block size

Allowing larger block sizes with ILR-SQ results differently in natural and screen content. Three impacts are evaluated from different values of the block size parameter: 1) BD-R, 2) encoding time, and 3) decoding time. These impacts are shown Table 5-8. According to this table, the BD-R gain increases by enabling larger ILR-SQ blocks. However, above 16×16, the gain is saturated and barely improves, while the encoder side complexity keeps growing, especially on screen content. On the decoder side, this impact is inverse in screen contents, meaning that enabling larger block sizes can reduce the complexity.

ILR-SQ size	Screen content			Natural content		
	BD-R	Enc-T	Dec-T	BD-R	Enc-T	Dec-T
4	-7.66%	102%	91%	-0.20%	103%	103%
8	-9.88%	103%	91%	-0.22%	103%	103%
16	-11.29%	106%	89%	-0.23%	104%	102%
32	-11.49%	110%	88%	-0.25%	104%	104%
64	-11.54%	112%	87%	-0.25%	105%	104%

Table 5-8: The impact of ILR-SQ block size on BD-R performance and codec complexity. The numbers are obtained from the VTM implementation of ILR-SQ.

Probability of zero OLR signal (%)								
Seq. Type	QP 22		QP 27		QP 32		QP 37	
	Regular	ILR-SQ	Regular	ILR-SQ	Regular	ILR-SQ	Regular	ILR-SQ
Natural	11	97	15	97	24	98	58	99.6
Screen	21	95	23	91	43	92	45	95

Table 5-9: Probability of having a zero residual after quantization with the regular intra coding algorithm and ILR-SQ, in four QPs.

One justification for the difference in the relatively lower decoder-side complexity of ILR compared to OLR. Notably, OLR decoding has to perform parsing, inverse transformation and inverse quantization, while ILR decoding only consists of parsing and inverse quantization. Moreover, the linear quantization of ILR is basically simpler than that of OLR.

Impact of using OLR within ILR-SQ

The use of ILR for pixel correction leaves a smaller residual energy after prediction by ILR-SQ. This is shown in Table 5-9. In this table, which is obtained from the JEM implementation of ILR-SQ, the percentage of zero OLR signals is provided for ILR-SQ blocks and regular intra blocks. To calculate these statistics, sequences with different resolutions have been used. As can be seen, the OLR signal of a ILR-SQ blocks is more likely to be zero than for regular intra blocks, in all QP values.

7 Conclusion

In this chapter, another intra coding algorithm based on the framework of Chapter 3 was proposed. This algorithm, called In-Loop Residual coding with Scalar Quantization (ILR-SQ), performs in-block pixel prediction and skips the transformation for coding the residual.

The first part of this chapter was devoted to the principles of scalar quantization in general. Then, key modules of ILR-SQ have been explained in detail and the full chain of block prediction has been clarified. In a next part, the main contribution of the chapter on residual coding has been explained. In this part, the adoption of scalar quantization for coding ILR values in the pixel domain has been introduced. After explaining the baseline version of ILR-SQ, including both prediction and ILR coding parts, some advanced tools on top of the baseline version have been introduced. This included non-normative improvement of the quantization step, empty ILR coding tool and the CABAC context derivation algorithm based on neighborhood density.

The performance of the proposed ILR-SQ algorithm has been evaluated in VVC-related reference softwares, namely JEM and VTM. For this purpose, two categories of sequences have

been studied separately: natural and screen content. On natural contents, ILR-SQ brings a slight BD-R gain with a negligible complexity overhead at both encoder and decoder sides. However, on screen contents, experiments show that ILR-SQ has a significant performance both in terms of BD-R gain and decoder side complexity reduction.

In the next chapter, other aspects of both proposed ILR-based intra coding algorithms will be discussed. This includes limitations as well as potentials future works. Moreover, a comparison will be presented for evaluating the differences and similarities of the two algorithms.

CHAPTER 6

Perspective of ILR coding

Contents

1 SQ vs. VQ

In this chapter, we provide a thorough discussion regarding the future work in the domain of the proposed ILR framework. For this end, some aspects that still have room to improve are discussed and the focus is put on the two proposed algorithms introduced in Chapter 4 and Chapter 5.

Before presenting the potential improvements of the ILR-VQ and the ILR-SQ algorithms, first a general comparison is provided. This comparison is helpful for understanding the basic limitations of each approach for future considerations. For this purpose, we thoroughly compare the use of SQ and VQ techniques for the ILR representation and coding.

1.1 Different behavior on the rate-distortion spectrum

Generally, the VQ-based and SQ-based ILR coding methods stand differently on the rate-distortion spectrum. More concisely, in the VQ-based method, the codebook size directly determines both the compression ratio and the distortion level. For instance, a codebook with a few codevectors would result in a very low rate with the cost of a relatively high distortion. On the contrary, the SQ-based approaches, that quantize and code the ILR signal in the spatial domain, usually results in a relatively larger rate when the conventional QP values are applied. This rate overhead is the cost of the flexibility that the SQ-based ILR coding provides to the blocks, which is supposedly rewarded by a lower distortion compared the VQ-based version.

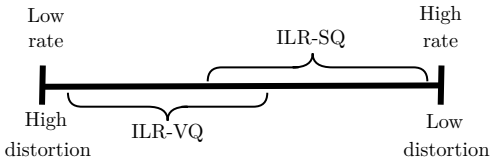


Figure 6-1: Different rate-distortion behaviors of the VQ-based and the SQ-based ILR methods.

Figure 6-1 visualizes the above difference in the nature of the VQ-based and the SQ-based ILR methods on a rate-distortion spectrum. As can be seen, for obtaining a low distortion by spending a higher rate, the SQ-based method seems to be a better option. Conversely, for staying in a very low rate with the cost of high distortion, the use of a VQ-based method can be recommended.

1.2 Drawbacks of VQ

In this section, some major differences in favor of the SQ-based ILR coding method are listed.

Flexibility in choosing the ILR signal

Despite the fact that the VQ-based ILR coding method can always extend its codebook size, an SQ-based method is still more efficient for having a wider range of ILR choices. For instance, in the bit-depth of 8, each pixel in the ILR signal can have any integer number between $(-2^8, 2^8)$, which consists of 511 different values. Therefore, for a block of size $W \times H$, the Euclidean space consisting of all possible ILR signals would include $(W \times H)^{511}$ points. This makes a total of $2^{4\times4\times511} = 2^{8178} = 2,01 \times 10^{615}$ points for the block size of 4×4. A comparison between this number and the size of the largest feasible codebook integrated in the ILR-VQ of Chapter 4 with 1024 codevectors, confirms the extreme sparsity of a vector quantizer for ILR coding.

The main problem with the extremely sparse ILR representation with a VQ-based algorithm is that the encoder is usually unable to transmit a proper approximation of its preferred ILR signal. The result of such restriction is that a given block for which the codebook does not

contain a proper ILR candidate, either is not selected, or is selected with a high distortion. However, this problem can be significantly addressed by an SQ-based ILR coding method such as ILR-SQ.

Control on the encoder complexity

The rate-distortion performance of a VQ-based ILR coding method is directly affected by its encoder side complexity constraints. This is due to the fact that a VQ-based encoder must evaluate as many codevectors as possible to find a proper ILR signal, and such search can be computationally complex.

One way to address the above problem is to ignore parts of the codebook during the block coding. For instance, one can blindly select a subset of the codebook and ignore the rest in order to accelerate the encoder search. However, such acceleration could significantly compromise the coding performance. A more sophisticated acceleration requires a fast, yet efficient, method for the performance estimation of each codevector before applying it. Although this approach provides a better performance than the blind codevector removal, experiments show that it is still unable to control the performance drop due to the acceleration.

An SQ-based method is usually less complex for the ILR signal coding. Unlike the VQ-based approach which has to perform the entire block prediction for each codevector in the codebook, an SQ-based method can make all coding decisions of a block only in one pass and still provide a reasonable performance. For instance, the standard transform coefficient coding of VVC, which uses a similar SQ-based coder to that of the ILR-SQ, finishes the quantization and encoding of transform coefficients in a single pass.

Memory overhead

One important drawback of the VQ-based ILR coding methods is the obligation of storing a relatively large codebook at both the encoder and the decoder sides. In Chapter 4, only four QP values at 22, 27, 32 at 37 were considered. However, a practical algorithm must be able to operate in any QP in the allowed range (i.e. between 1 and 63 in VVC). This would require storing several codeooks, otherwise different QPs would have to share QPs that are not necessarily optimized for them.

The above issue is also problematic when it comes to the technology adoption process during the standardization. More precisely, the hardware manufacturers are usually unable to implement such additional memory requirement, unless its overhead brings significant coding gain. As opposed to the VQ-based method, using a scalar quantizer for ILR coding requires no or a small amount of additional memory. Such advantage guarantees that any possible coding gain would be appreciated in the standardizaion community without the counteract of the memory overhead.

Difficult codebook training

It is not straightforward to implement a codebook training algorithm to reach or come close the upper bound of the theoretical performance. Practically, several factors affect the output of the codebook training process. As discussed with details in Chapter 4, there are two major obstacles to carry out a proper codebook training. First, due to the obligation of applying the modified LBG instead of the standard algorithm, the training process is usually very time-consuming and in some cases unfeasible to perform. Second, regardless of the set of sequences used for training sample extraction, there is always the chance that the optimized codebook would not properly serve a new sequence that have not been used in the training phase.

1.3 Drawbacks of SQ

Despite the long list of advantages in using the SQ-based methods compared to the VQ-based ones, there are still some aspects in which the VQ-based version of the ILR framework is preferred.

Decoder-side complexity

The current SQ-based ILR decoder is more complex than the VQ-based one. The sources of this complexity overhead are mainly the parsing and dequantization steps that have to be performed at the pixel level. Although the complexity difference is not significant, it still can be considered as a barrier as it happens at the decoder-side.

It is useful to remind that keeping the decoder-side complexity at a low level always has a higher priority than that of the encoder-side. This is due to the fact that the decoder of a standard has to be implemented in a wide range of devices with different processing capacity, which is in contrast with the encoder implementation. Besides, by nature, a decoder has to operate at real-time speed with a very little buffering.

Very low bit rate coding

The use of the regular coefficient coding method of the VVC, within the proposed ILR framework, is not as efficient as it is in the transform coefficients coding of the regular residual. This inefficiency is mainly due to the fact that the this method has been highly optimized for a signal with a nature in the frequency-domain and possess the energy compaction property, which is not the case for the spatial-domain residual of the ILR signal. On the contrary, experiments show that ILR signals can contain larger amplitudes around their bottom-right corner. This property makes the use of the existing coefficient coding method of VVC even more inefficient for the ILR coding.

2 Common future work

Regardless of the fundamental differences between the VQ-based and the SQ-based methods, there are a few common aspects which still have rooms to improve. In this section, three common future tracks within the proposed ILR framework of Chapter 3 are discussed.

2.1 IPM derivation

Introducing new tools as an alternative for an existing tools can always introduce unintentional side effects. In the proposed ILR framework, one important side effect is damaging the regular IPM coding scheme, that is highly optimized for the existing intra coding framework of VVC [9, 16, 17].

Let us elaborate the above problem with an example. According to the discussions in Section 2.3, the existing IPM coding of VVC exploits the information in a spatial neighborhood to construct an MPM list and predict the current selected IPM. For this purpose, the selected IPMs of the neighboring blocks are used. One side effect happens when a neighbor is coded by the proposed ILR framework and does not carry an IPM. In this situation, the performance of IPM coding will degrade as less contextual information is available for the current IPM prediction.

A similar problem occurs in a B or P slice, when an intra block is surrounded by inter blocks. In this situation, the inter coded neighbors are simply marked as unavailable and are excluded from the MPM list construction. However, the experiments showed that using this solution is harmful when applied in intra slices with ILR blocks. Here, two methods for addressing the IPM unavailability are introduced.

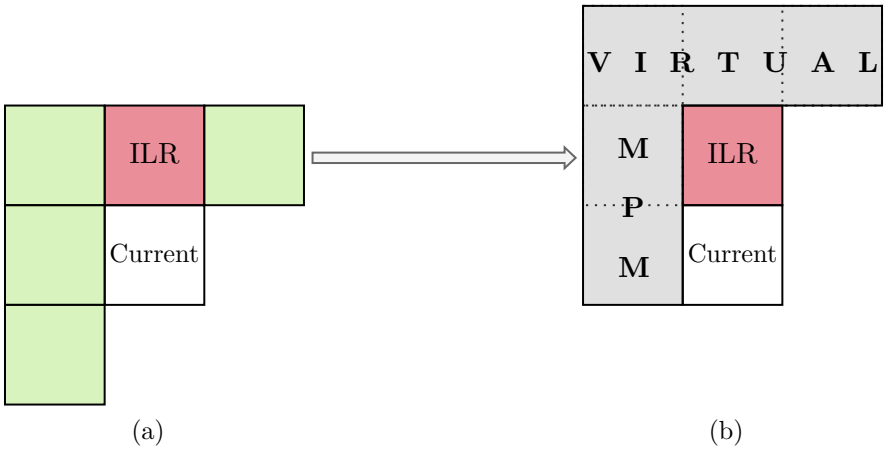


Figure 6-2: Constructing a virtual MPM list for an ILR-coded block. (a) "Current" block inquires "ILR" block about its IPM. (b) "ILR" block constructs its virtual MPM list to provide "Current" block with an IPM.

Texture analysis

Texture information in a block can provide useful information about its suitable IPM, even if it is coded with the proposed ILR framework [75, 76]. For instance, once a regular intra block inquires the non-existing IPM of an ILR-coded neighbor for its MPM list construction, one can examine the texture of that ILR block in order to estimate the best IPM.

One major problem of texture analysis for IPM derivation is the parsing dependency due to the access to the reconstructed pixels [77, 78]. More precisely, it is highly recommended that two phases of 1) syntax elements parsing and 2) pixel decompression and reconstruction would be kept independent from each other. This requirement, if met, would allow high throughput parallelism for the decoder implementation. However, the proposed texture analysis makes the parsing phase of the IPM information dependent to the reconstructed pixels in the neighborhood.

One can explore the possibility of a texture analysis while parsing and independently from the decompression. For instance, the residual signal in the transform domain, which is available during the parsing phase, might carry useful information about the texture of its corresponding block.

Virtual MPM list

To address the parsing dependency problem, one must use only the information that is available during the parsing phase. A method that has been proposed during the development of the ILR framework, constructs a virtual MPM list for an ILR-coded block. The first IPM in the virtual MPM list is then used as the IPM of that block as it was coded by the regular intra coding. Figure 6-2 shows how an ILR block refers to its neighbors to construct its virtual MPM list.

Another benefit of the virtual MPM list compared to the texture analysis method is its simplicity. As explained in the previous section, the texture analysis requires the decoder side to perform an IPM derivation method on the reconstructed block. However, the virtual MPM list only needs to access the IPM information of neighbors that are already parsed and decoded, which is significantly simpler than any texture analysis method.

2.2 Pixel predictor design

In the current design of both the ILR-VQ and the ILR-SQ, a single predictor is used. This predictor, called LOCO-I, does not impose any signaling rate. Instead, it is embedded with a

u_2	u_3	u_4
u_1	x	

Figure 6-3: Four in-block references, $u_1, ..., u_4$, around the current pixel x.

context-adaptive predictor function that enables implicit edge detection. The current context derivation scheme of the LOCO-I predictor has been originally designed for the lossless texture compression. Therefore, one might study if there is a more efficient context derivation scheme for the lossy mode of the proposed ILR framework.

In this section, a context-based predictor design algorithm is introduced. This algorithm has been briefly studied in different periods of this thesis, yet it is still open as a future work. Here, we describe the algorithm with an example configuration. This configuration is flexible and can adapt to different needs of the proposed ILR framework.

There are mainly two questions to answer, concerning the new pixel predictor: 1) Should we keep using one predictor or should we integrate multiple predictors? 2) How should we design the context derivation scheme of each predictor? The following example provides a mean to answer these questions.

Notation

The following notations have been used in the rest of this section:

- $\mathbb{P} = \{P_i | i = 1, .., L^{\mathbb{P}}\}$ denotes the set of $L^{\mathbb{P}}$ pixel predictor functions. Each predictor P_i has four coefficients $p_{i,1}, .., p_{i,4}$ to apply a weighted average on the four reference pixels in Figure 6-3 as:

$$P_i(u_1, ..., u_4) = p_{i,1} \times u_1 + ... + p_{i,4} \times u_4, \text{ where } i = 1, .., L^{\mathbb{P}}. \tag{2.1}$$

  In the current example, three values of 0, ± 1 are allowed for each coefficient, with a normalization constraints such that $p_{i,1} + p_{i,2} + p_{i,3} + p_{i,4} = 0$. This would make a total of $L^{\mathbb{P}} = 19$ different linear predictors as:

$$\mathbb{P} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{19} \end{bmatrix} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,3} \\ p_{2,1} & \cdots & p_{2,4} \\ \vdots & \ddots & \vdots \\ p_{19,1} & \cdots & p_{19,4} \end{bmatrix} = \begin{bmatrix} -1 & -1 & +1 & +1 \\ -1 & 0 & 0 & +1 \\ \vdots & \vdots & \vdots & \vdots \\ +1 & +1 & -1 & -1 \end{bmatrix}. \tag{2.2}$$

- $\mathbb{S} = \{S_j | i = 1, .., L^{\mathbb{S}}\}$ denotes the set of $L^{\mathbb{S}}$ contextual situations, defined for each pixel by its in-block reference pixels. In the current example, situations in $\mathbb{S}$ are categorized with respect to the appearance order of the in-block references in their sorted list. More precisely, at each pixel, the intensity level of the neighbors are sorted in the ascending order. Then, depending on the resulting combination, one of the following $L^{\mathbb{S}} = 4! = 24$ situations are selected:

$$\mathbb{S} = \begin{bmatrix} S_1 : u_1 \leq u_2 \leq u_3 \leq u_4, \\ S_2 : u_1 \leq u_2 \leq u_4 \leq u_3, \\ \vdots \\ S_{24} : u_4 \leq u_3 \leq u_2 \leq u_1. \end{bmatrix}. \tag{2.3}$$

  The use of the operation $\leq$ makes overlaps on the situations defined by Eq. 2.3. For instance, $u_1 \leq u_2 \leq u_3 = u_4$ belongs to both S_1 and S_2. In these cases, the situation that has a smaller number (e.g. S_1 in this example) is selected.

- $\mathbb{D} = \{D_k | k = 1, ..., L^{\mathbb{D}}\}$ denotes a large dataset of $L^{\mathbb{D}}$ samples, each of which representing an actual coded pixel along with its in-block references, denoted in Eq. 2.4. As can be seen, each entry D_k in this dataset contains intensity level of the k^{th} sample as x_k as well as its four references, as $u_{k,t}$ with $t = 1, 2, 3, 4$.

$$\mathbb{D} = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} x_1 & u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ x_2 & u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{2.4}$$

- G is a function that takes a sample in $\mathbb{D}$ as the input returns the situation that is defined by the ascending order of its references, according to Eq. 2.3:

$$G : \mathbb{D} \rightarrow \mathbb{S}. \tag{2.5}$$

- F is the situation to predictor mapping function that is aimed to be optimized:

$$F : \mathbb{S} \rightarrow \mathbb{P}. \tag{2.6}$$

Optimization

The goal of the optimization step is to find the best mapping function F^* that minimizes a cost function. Let F^t be the situation to predictor mapping function of the current iteration t, that assigns one of the $L^P = 19$ predictors to each of the $L^S = 24$ situations. The optimization process of obtaining F^{t+1} is an iterative algorithm, where each iteration consists of two steps on each situation S_j , $j = 1, ..., 24$:

- Classification: The function G partitions the entire dataset $\mathbb{D}$ into L^S non-overlapping classes corresponding to situations in Eq. 2.3:

$$\mathbb{D}_{|j} = \{D \in \mathbb{D} \mid D = \{x, u_1, u_2, u_3, u_4\},\ G(D) = S_j\}\ ,\ j = 1, .., L^S. \tag{2.7}$$

  According to above classification, it can be derived that the predictor $P_i = F^t(S_j)$ is currently used for class j and is expressed as $P_i = \{p_{i,1}, ..., p_{i,4}\}$.

- Update: Given P_i as the predictor of class j, the distortion based cost function for $\mathbb{D}_{|j}$ can be calculated as:

$$E_{j,i} = \sum_{D \in \mathbb{D}_{|j}} (x - P_i(u_1, u_2, u_3, u_4)),\ \text{where } D = \{x, u_1, u_2, u_3, u_4\}. \tag{2.8}$$

  Finally, the predictor of class j is updated for the next iteration as:

$$F^{t+1}(S_j) = \arg\min_{P_i} \sum_{D \in \mathbb{D}_{|j}} E_{j,i},\ \text{where } i = 1, ..., L^P. \tag{2.9}$$

Repeating the above two steps on all L^S classes updates the entire situation to predictor function F^t to F^{t+1}.

Preliminary results

The above predictor optimization has been implemented on a large dataset of samples (i.e. $\mathbb{D}$), obtained from actually coded sequences with ILR-SQ. The result of this optimization after a few iterations is shown in Table 6-1. The third column of this table represents the first best predictor for each situation, as discussed earlier. Moreover, the second best predictor is also optimized for each situation. For this purpose, the first best predictor is excluded from $\mathbb{P}$ at the end of each iteration and then the same process of that iteration is carried out to provide the second best predictor.

Table 6-1: The optimal situation to predictor function $F : \mathbb{S} \rightarrow \mathbb{P}$ after a few iterations.

Situation	References order	1^{st} best predictor	2^{nd} best predictor
1	$u_1 \leq u_2 \leq u_3 \leq u_4$	$u_1 + u_3 - u_2$	$u_1 + u_2 - u_3$
2	$u_1 \leq u_2 \leq u_4 \leq u_3$	$u_1 + u_3 - u_2$	u_2
3	$u_1 \leq u_3 \leq u_2 \leq u_4$	u_1	u_2
4	$u_1 \leq u_3 \leq u_4 \leq u_2$	$u_1 + u_3 - u_2$	u_2
5	$u_1 \leq u_4 \leq u_3 \leq u_2$	u_1	$u_1 + u_3 - u_2$
6	$u_1 \leq u_4 \leq u_2 \leq u_3$	u_1	u_2
7	$u_2 \leq u_1 \leq u_3 \leq u_4$	u_1	u_2
8	$u_2 \leq u_1 \leq u_4 \leq u_3$	u_3	u_4
9	$u_2 \leq u_3 \leq u_1 \leq u_4$	u_3	u_4
10	$u_2 \leq u_3 \leq u_4 \leq u_1$	$u_1 + u_3 - u_2$	u_2
11	$u_2 \leq u_4 \leq u_3 \leq u_1$	u_1	$u_1 + u_3 - u_2$
12	$u_2 \leq u_4 \leq u_1 \leq u_3$	u_1	u_2
13	$u_3 \leq u_2 \leq u_1 \leq u_4$	$u_1 + u_3 - u_2$	u_4
14	$u_3 \leq u_2 \leq u_4 \leq u_1$	u_1	u_2
15	$u_3 \leq u_1 \leq u_2 \leq u_4$	u_1	u_2
16	$u_3 \leq u_1 \leq u_4 \leq u_2$	u_3	u_2
17	$u_3 \leq u_4 \leq u_1 \leq u_2$	u_3	$u_4 + u_2 - u_3$
18	$u_3 \leq u_4 \leq u_2 \leq u_1$	u_3	$u_3 + u_1 - u_2$
19	$u_4 \leq u_2 \leq u_3 \leq u_1$	u_1	u_2
20	$u_4 \leq u_2 \leq u_1 \leq u_3$	$u_1 + u_3 - u_2$	$u_2 + u_3 - u_1$
21	$u_4 \leq u_3 \leq u_2 \leq u_1$	$u_1 + u_3 - u_2$	u_4
22	$u_4 \leq u_3 \leq u_1 \leq u_2$	$u_1 + u_3 - u_2$	u_2
23	$u_4 \leq u_1 \leq u_3 \leq u_2$	u_1	u_2
24	$u_4 \leq u_1 \leq u_2 \leq u_3$	u_1	u_2

2.3 Encoder acceleration by fast algorithm competition

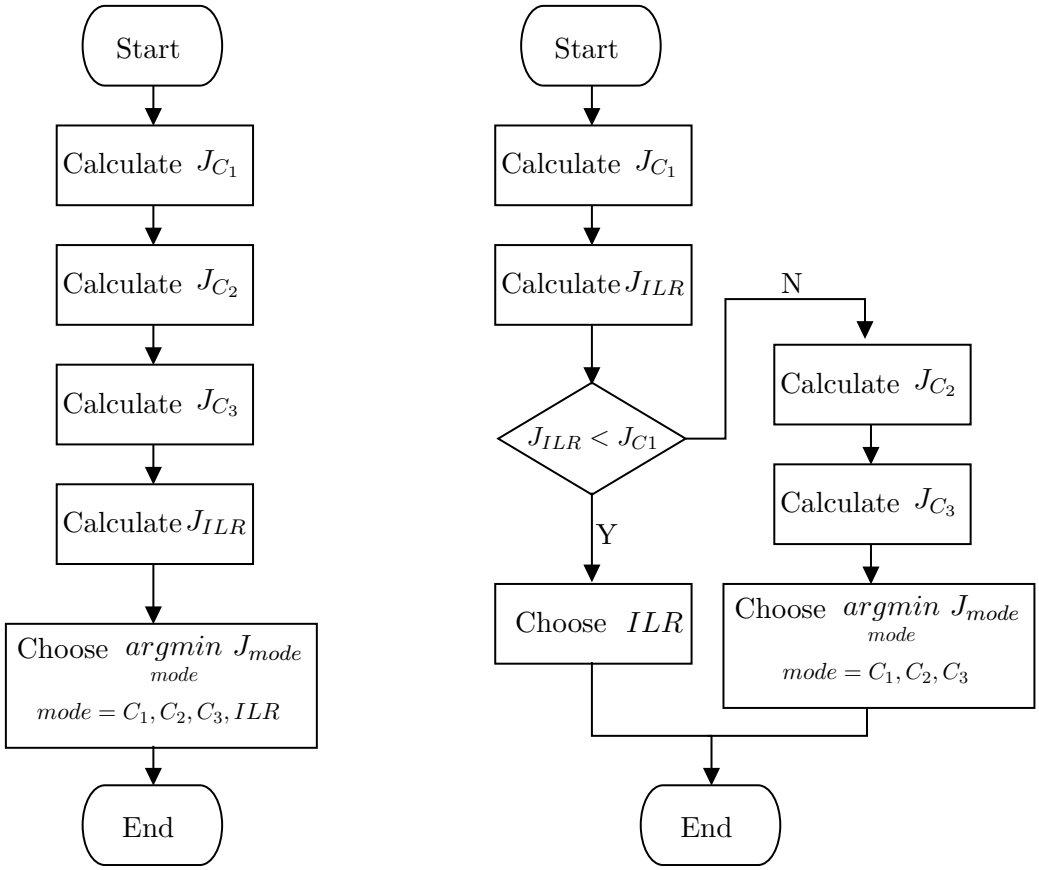
As was discussed, the block-level algorithm competition of the proposed ILR framework imposes an encoder side complexity overhead. This is due to the fact that the encoder has to evaluate a new tool on top of all other existing tools. Until the end of this thesis, this aspect of the proposed ILR framework has been remained untouched. That is to say, the main objective has always been obtaining the highest possible compression gain, regardless of its encoder side complexity overhead.

A possible future work can be studying different ways to reduce the encoder side complexity of the algorithm competition, integrated in the proposed ILR framework. Here, we present one example algorithm that has been developed within the ILR framework to serve as an encoder shortcut for acceleration.

Early termination of the IPM search

The concept of incomplete IPM search for fast mode decision at the encoder side has been widely studied in the literature [79–83]. Current implementation of both ILR-VQ and ILR-SQ adds an iteration at the end of the IPM search algorithm, explained in Section 2.2 of Chapter 2. In this implementation, first a set of IPM candidates are selected among all available IPMs, based on a rough estimation of their rate-distortion costs. Then, the ILR-based algorithm competes with these candidates in terms of actual rate-distortion cost.

Let $[C_1, C_2, C_3]$ be a list of three IPM candidates that are selected in the rough search mode and are sorted ascendingly, based on their rate-distortion cost estimation. Figure 6-4-(a) shows the normal IPM search without termination.



(a) Normal IPM check (b) Fast IPM check with early termination

Figure 6-4: Normal and fast IPM check in existence of 3 IPM candidates (C_1, C_2 and C_3) and ILR-SQ.

The proposed early termination of the IPM search fully trusts the rate-distortion cost estimation of the rough search mode and stops the search if the ILR-based algorithm is better than C_1. In a statistical experiment, we studied the outcome of the competition of Figure 6-4-(a), particularly when the ILR-based algorithm outperforms the best candidate C_1. We observed that when the actual rate-distortion cost of the ILR-based algorithm is better than that of C_1, it is very likely that the ILR-based gets selected for that block. Based on this experiment, an early termination scheme was designed for the above algorithm competition, shown in Figure 6-4-(b). As can be seen in this figure, we propose to check the ILR-based algorithm right after checking C_1. Then by comparing the rate-distortion costs of these two modes, we decide whether or not to continue the search.

Preliminary results of ILR-SQ with early termination in VTM

The proposed early termination algorithm has been implemented for the ILR-SQ algorithm in VTM. Table 6-2 shows the results of this algorithm. As can be seen in the screen content classes, namely class F and Text-Graphic-Motion (TGM), this simplification can save a noticeable amount of encoder side complexity with a negligible drop in the BD-R. However, in all other classes that contain natural content, the ratio between the BD-R drop and complexity reduction is less favorable.

Class	Normal IPM check		Early termination	
	BD-R	Encoder Time	BD-R	Encoder Time
A1	-0,07%	101%	-0,02%	98%
A2	0,00%	101%	0,03%	99%
B	-0,11%	101%	-0,03%	99%
C	-0,27%	100%	-0,14%	95%
D	-0,54%	101%	-0,22%	93%
E	-0,19%	102%	-0,15%	99%
F	-8.58%	106%	-8.46%	77%
TGM	-16,29%	110%	-16.14%	61%

Table 6-2: The impact of the early termination of the IPM check on the encoder side complexity and the BD-R gain.

Future work

The early results of ILR-SQ in VTM showed that there might still have room to improve the early termination algorithm of the IPM check. As a future work, one can change the scheme of Figure 6-4-(b) in order to make the termination condition $J_{ILR} < J_{C_1}$ less harsh. For instance, a weighting factor $\omega > 1$ can make a milder termination condition as $\omega \times J_{ILR} < J_{C_1}$.

Another idea would be checking the early termination condition later in the diagram of Figure 6-4. For instance, one can check C_1 and C_2 first and then the ILR-based algorithm. At this point, a condition such as $J_{ILR} < \max(J_{C_1}, J_{C_2})$ can postpone the IPM search termination. This delay can hopefully reduce the amount of BD-R drop in the natural content coding.

2.4 Other common tracks

During the development phase of the ILR framework, a few other problems were left open for future work. Similar to the IPM derivation and predictor design, these problems are applicable to both the VQ-based and the SQ-based algorithms. Here, a brief list of these problems are presented.

- **Larger block sizes**: Both proposed algorithms are extendable to larger block sizes without any significant change. However, experiments showed that their performance degrades in larger block sizes. There are different possible justifications for this behavior. In ILR-VQ, this inefficiency is mainly due to the difficult codebook training process, while in ILR-SQ, the high rate of the spatial domain residual coding is causing this inefficiency.

- **Flag compression**: The concentration of ILR-coded blocks is significantly more visible in high detail areas of frames. This property was shared between both the ILR-VQ and the ILR-SQ algorithms and can be exploited for further compression. One idea can be using the flag information in the neighborhood. For instance, one can simply predict the flag from neighboring flags and send the prediction error instead of the flag itself, which might have a lower entropy. Another example would be using two different CABAC context models for areas with sparse and dense population of ILR-coded blocks.

- **Other encoder acceleration methods**: The density of ILR-coded blocks in a neighborhood can also be used for determining whether a block is potentially appropriate for the proposed methods or not. For instance, one can make the encoder evaluate a local neighborhood in terms of flag density and/or texture complexity and determine whether or not the corresponding ILR method should be checked.

3 Future of the ILR-VQ

In this section, two possible future work in the domain of the ILR-VQ are discussed.

3.1 Variable length coding of the codebook index

The statistical analysis of blocks coded by ILR-VQ showed a non-uniform selection of codevectors in the optimized codebooks. One example is shown in Figure 6-5. In this figure, the histogram of 256 codevectores within the codebook of 4×4 blocks in QP 37 is presented. As can be seen, the codevectors in this codebook can roughly be divided into three groups:

1. Small set of highly popular codevectors in the beginning of the histogram.
2. Large set of relatively less probable codevectors in the middle of the histogram.
3. Large set of highly improbable codevectors at the end of the histogram.

Although, the skewness level of the histogram in Figure 6-5 is not ideal, it might still be possible to compress the codevector index by using variable length codes. One idea would be coding one flag to indicate whether the selected codevector belongs to the first group or not. If yes, a unary code can be sent to encode the short index in the first group. Otherwise, another flag can be coded to choose between the second and the third groups. Finally, a binary code can be sent as the long index within the selected group.

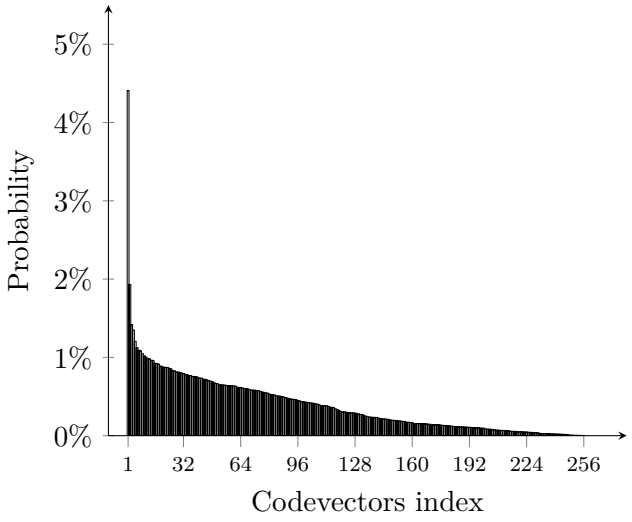


Figure 6-5: Probability of codevectors index in the codebook of QP 37.

3.2 Flexible codebook size during training

Retaining a constant size during the codebook training is appropriate when a fixed-length coding is used for coding the codevectors index. Otherwise, it is more efficient to allow the codebook training to change its size. For instance, one might start from an inital codebook size and allow the LBG algorithm to throw away or add codevectors during its training.

4 Future of the ILR-SQ

In this section, possible future work in the domain of ILR-SQ are discussed. Among all interesting ideas, one main idea has been thoroughly introduced, while some others are briefly listed.

4.1 Foreground/Background separation for screen content

When the original pixel value is far from its prediction, the prediction scheme in the baseline ILR-SQ algorithm introduces a relatively large residual signal. In screen content coding, textures often contain separate background and foreground layers and this problem usually happens when references belong to a different layer than the pixel to be predicted. In this situation, the available information in references is not adequate for an accurate prediction to the opposite layer. Here, this problem is called "inter-layer transition" and a pixel associated with it, is called a "transition pixel".

A novel algorithm for ILR-SQ is presented here to further exploit the redundancies in the transition pixel. This algorithm currently is in its early stages of development and has rooms to improve in future. The main idea is to derive information about the layers from a local neighborhood. This information, along with an adaptive transition detection method, helps at reducing the residual energy of transition pixels. Figure 6-6 shows examples of the layer information availability from three screen contents. As can be seen, the local neighborhood (i.e. green squares) usually contains useful information about the intensity level of foreground and background within the current block (i.e. red squares).

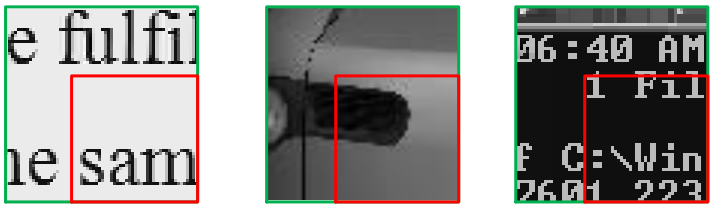


Figure 6-6: Three blocks of texture with their informative neighborhoods regarding the intensity levels of background and foreground layers.

The context-adaptive predictor function of LOCO-I that has been used in the ILR-SQ usually produces a large residual on texture edges in screen contents. This causes a large rate and negatively impacts the coding performance, especially when the intensity level difference between the background and the foreground is large. In this case, a large residual value must be transmitted to make a transition from one layer to the other.

Implicit layer detection

In order to derive layer information, one can use an implicit method that does not require signaling. For this purpose, the encoder may use the texture information of previously coded blocks for foreground and background layer separation. This property enables implicitness of layer detection at the decoder side. Such approach is in contrast with the explicit methods, such as Palette mode [42, 43, 45], in which the intensity level information is directly signaled to the decoder.

The amount of texture information used from previous blocks is a compromise between the complexity and the layer separation accuracy. The complexity is defined both in terms of buffering memory and access of previously coded blocks as well as the computation overhead of processing those blocks. For a more practical implementation, the current design simply uses one row and one column of buffered texture from previous blocks, as is the case in the regular intra prediction.

The proposed implicit layer detection algorithm uses a Gaussian Mixture Modeling (GMM) of the texture histogram. However, the conventional GMM parameter estimation methods such as Expectation-Maximization are too complex to perform on each block during coding. Hence, a low-complexity greedy algorithm is used for this purpose [84]. Each trial of this algorithm spots the highest spike of the histogram and considers the neighborhood around it as the most popular intensity level. Then, it estimates the PDF parameters of the neighborhood by a Gaussian PDF. In the current problem, we perform two trials of this algorithm for background and foreground layers, respectively. This process is summarized in Figure 6-7. As can be seen in this figure, a threshold *thr* is also defined after determining about the intensity levels of the layers. This threshold will be used later in the proposed coding of inter-layer transition.

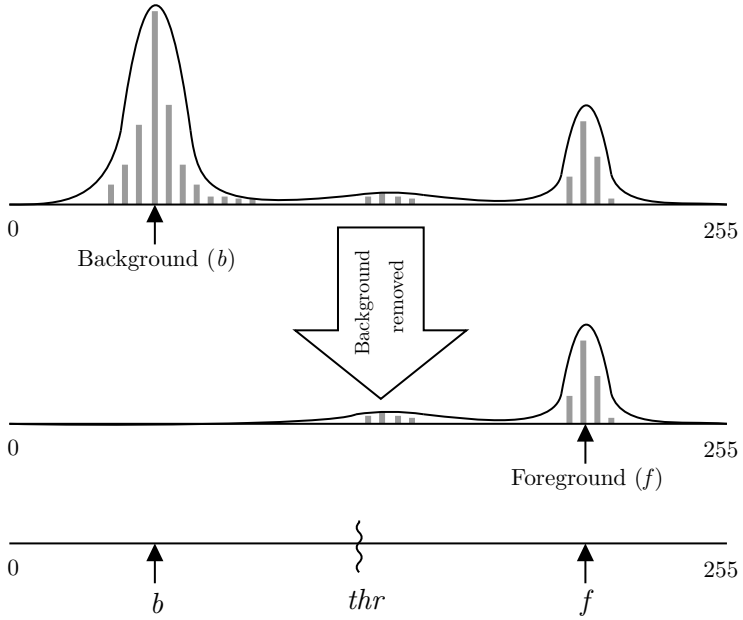


Figure 6-7: Two trials of the GMM-based background and foreground layers separation and determination a transition threshold between them.

Transition states

Here, we define three transition states to determine whether an inter-layer transition flag should be signaled or not. At the parsing stage, the decoder first derives the state of each pixel based on its residual value. Then depending on the derived state, the decoder decides whether or not an inter-layer transition flag should be parsed. For this purpose, we define the following three transition states, depending on the local neighborhood around a pixel and its residual value:

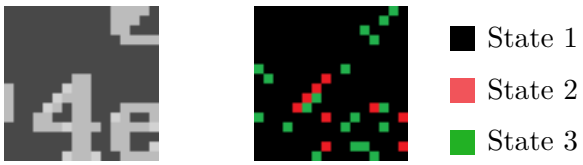
State 1) Intra-layer plain: areas on which the LOCO-I predictor produces zero residual after the quantization. The inter-layer transition flag is not signaled in this condition and the decoder implicitly derives the state after parsing a zero residual value.

State 2) Intra-layer low-complexity: areas which are usually handled by a moderately accurate prediction by LOCO-I, along with a small non-zero residual. In this situation, the residual value is not large enough to define an inter-layer transition. Therefore, the inter-layer transition flag of zero is explicitly signaled.

State 3) Inter-layer high-complexity: areas on which the context-adaptive predictor of LOCO-I performs poorly and produces a significantly large residual. In this situation, the inter-

layer transition is set to one and is signaled. Moreover, the predicted value by the LOCO-I is replaced by the intensity level of the further layer. Finally, the new residual value is re-calculated based on the new predicted value.

An example of a 16×16 block is presented in Figure 6-8, which consists of a bright text on a dark background. The transition state map, at the right of this figure, shows how the above three states are assigned to pixels with different levels of LOCO-I prediction error.



a) Original block *b*) Transition states

Figure 6-8: An example 16×16 block containing clean background/foreground layers with different transition states of its inner pixels.

Encoder/Decoder design

In this section, we explain how the encoder and decoder of the ILR-SQ can incorporate the above state definitions. For simplicity, the decoder side process is explained first and then the order of the encoder side process to match the decoder is provided. In the rest of this section, the following notation is used. Symbols p, c and o respectively present prediction, reconstruction and original values of each pixel in a block. Moreover, each residual value is associated with its signed decoded residual r which is composed of its amplitude a and sign sgn, while its transition state and flag (if any) are denoted as s and t, respectively. Finally, block level symbols b and f represent intensity levels of background and foreground layers, respectively.

The layer separation algorithm adds a few extra steps to regular ILR-SQ decoder, as shown in Algorithm 1. In the parsing phase, a combination of explicit and implicit derivation of the transition state is added. More precisely, the following steps are performed for parsing each residual value: 1) parsing the amplitude a (similarly the regular ILR-SQ), 2) parsing the sign sgn if a is non-zero (similarly the regular ILR-SQ), 3) Implicit state derivation of $s = 1$, if the amplitude is zero, or 4) explicit state derivation of $s = 2$ or 3, if a is non-zero. This is carried out by parsing the transition flag t. Each pixel at the end of this process has its signed residual value r, state s and transition flag t (if $s \neq 1$).

Once the residual amplitudes, states and transition flags are parsed, the decoder is able to perform the decompression phase for pixel reconstruction. In this phase, first the layer derivation is performed using the GMM-based method explained in Section 4.1. This step provides the background b and foreground f intensity levels for the entire block. Then the following steps are carried out for each pixel: 1) prediction of p with LOCO-I, 2) in the case of being in State 3, making the inter-layer transition. For this purpose, first the distances of p from both b and f is calculated and then p is updated with the intensity level of the layer with longer distance. Finally, 3) pixel c is reconstructed normally and by adding residual r.

An encoder conforming the above decoder syntax would properly determine pixel states and their transition flags. Algorithm 2 summarizes this process. In this algorithm, the regular ILR-SQ prediction is performed on each pixel first. Then a condition is checked to determine whether or not p and o fall on different sides of thr, in order to determine the necessity of an inter-layer transition. More precisely, if p and o fall on different sides of thr, then the inter-layer transition flag is set to one and the state of current pixel is set to 3. In this case, the distance of p from

Algorithm 1 The decoder of ILR-SQ with layer separation.

```
1: procedure DECODER
2:     Set background b and foreground f as explained above
3:     for each pixel in block do
4:         Parsing
5:         Parse amplitude a
6:         if a = 0 then
7:             r ← 0
8:             s ← 1 (implicit)
9:         else
10:            Parse sign sgn
11:            r ← sgn × a
12:            Parse transition flag t
13:            if t = 0 then
14:                s ← 2 (explicit)
15:            else
16:                s ← 3 (explicit)
17:        Decompression
18:        Predict p with LOCO-I
19:        if s = 3 then
20:            if p − b < p − f then
21:                Background to foreground transition
22:                p ← f
23:            else
24:                Foreground to background transition
25:                p ← b
26:        Reconstruct c ← p + r
```

both b and f, are compared and then it is replaced by the further one. Otherwise (i.e. if p and o fall on the same side of thr), the residual value r is used to determine whether a transition flag $t = 0$ is explicitly needed to be signaled (i.e. $r \neq 0$), or it can be derived implicitly (i.e $r = 0$). Finally, no matter in which state the pixel is, its residual amplitude a and sign sgn are transmitted separately.

Preliminary results

A preliminary performance evaluation of the proposed layer separation algorithm is carried out within the VVC Test Model (VTM) reference software [18]. For this purpose, the algorithm has been implemented on top of the VTM and its performance has been calculated against two anchor encoders: 1) pure VTM, and 2) VTM + ILR-SQ. Table 6-3 compares the performance of the proposed algorithm against these two anchor encoders, under the constraints of the Common Test Conditions (CTC) [85]. As can be seen, the proposed method improves the pure VTM by about 13%, in terms of BD-Rate gain [86].

The results in Table 6-3 also confirms that the proposed layer separation improves the baseline ILR-SQ algorithm. As can be seen, an average BD-Rate gain of about 1.2% is achieved.

Future work

The algorithm that was explained in this section is a simple implementation of the general idea of layer separation. The main purpose was to demonstrate the fact that the texture information

Algorithm 2 The encoder of BDPCM with layer separation.

```
1: procedure ENCODER
2:     Set background b and foreground f as explained above
3:     for each prediction pixel in block do
4:         Obtain p using the function with LOCO-I
5:         if p < thr < o or o < thr < p then
6:             s ← 3
7:             t ← 1
8:             Encode t (explicit)
9:             if p − b < p − f then
10:                Background to foreground transition
11:                p ← f
12:            else
13:                Foreground to background transition
14:                p ← b
15:            Obtain quantized residual r
16:        else
17:            Obtain quantized residual r
18:            if r = 0 then
19:                s ← 1 (implicit)
20:            else
21:                s ← 2
22:                t ← 0
23:                Encode t (explicit)
24:        Decompose r insgn and a
25:        Encode sgn and a
```

in screen content can help the baseline ILR-SQ for residual coding. Here, a list of possible future steps in the domain of layer separation for ILR-SQ is presented:

- **Complexity reduction:** Current computational complexity overhead, especially at the decoder side, makes its challenging for real-time implementation. The experiments show that almost all of this complexity overhead is due to the histogram analysis with GMM, that has to be performed at each block. One idea to reduce this complexity might be layer detection during the reconstruction of each block. More precisely, each block in the neighborhood of an ILR-coded can particularly process its bottom row or right column during reconstruction, in order to calculate its two popular intensity levels. Later, this information can be used at the ILR-coded block to replace the GMM-based layer detection method.

- **Smart thresholding:** Currently, the transition threshold thr is simply calculated as the average intensity level of the two layers. However, the experiments show that this choice causes redundant transition flag t signaling in higher QPs. Therefore, one might optimize thr or in general, the transition state definition to avoid such problems.

- **Adaptive quantization step offset:** The texture information in the neighborhood can also provide information about a more efficient spatial domain quantization. For instance, one can use the difference between background and foreground intensity levels as the quanzization step size.

Table 6-3: Performance of the proposed layer separation for ILR-SQ against pure VTM as well as against VTM+ILRSQ, in terms of BD-rate gain (%) and coding time (%).

Resolution	Sequence	vs. VTM		vs. VTM+ILRSQ	
		BD-rate	ET/DT	BD-rate	ET/DT
2560	Basketball_Sc	-8.95	135/142	-1.28	139/139
×	MissionCtrlClip2	-7.60	135/148	-1.35	144/142
1440	**Average**	**-8.27**	**135/145**	**-1.31**	**142/141**
	FlyingGraphics	-7.21	149/153	-2.59	144/149
	Desktop	-20.42	150/127	-0.41	140/130
	Console	-19.06	145/133	-1.25	130/125
1920	ChineseEditing	-9.94	141/142	-0.94	149/129
×	MissionCtrlClip3	-8.92	136/110	-0.24	126/123
1080	Robot	-4,34	144/133	-1.96	139/120
	ChinaSpeed	-9.42	137/136	-2.09	130/146
	TencentAOV7	-1.49	127/106	-0.28	119/112
	Average	**-6.58**	**141/130**	**-1.22**	**134/129**
	Web_browsing	-14.01	131/115	-0.08	142/126
	Map	-4.15	146/139	-1.26	139/127
1280	Programming	-11.91	159/144	-2.50	150/162
×	SlideShow	-12.36	141/117	-0.52	142/121
720	SlideEditing	-12.26	134/142	-1.17	129/130
	BasketballDrillText	-1.05	114/102	-0.00	113/109
	Average	**-9.29**	**137/126**	**-0.92**	**135/129**
	Total Average	**-9.56**	**139/131**	**-1.05**	**144/130**

4.2 Other future works in ILR-SQ

Here, a list of possible future works in the ILR-SQ domain, is presented.

Quantization offset

In the regular transform domain quantization, we are allowed to adjust QP for a better subjective quality perception [87–89]. The goal of the transform domain QP adjustment is improving the subjective quality. In the ILR-SQ domain, one can apply a similar idea in order to improve the rate-distortion behavior. There are two approaches to implement the quantization step size adjustment:

- Explicit: Block level flags carry information about the adjustment parameters (e.g. offset). Such parameter values are determined by the encoder and during the RDO process.
- Implicit: Pre-defined rules are shared between the encoder and decoder to derive the quantization adjustment parameters. On example was the block level quantization adjustment based on reference pixels that was explained in Section 4.1. Another example is a pixel level quantization adjustment. For instance, one might quantize residual of pixels at top-left corner more finely than bottom-right corner, since more pixels are dependent on them.

Block filtering

As discussed, the current context-adaptive pixel predictor of LOCO-I is vulnerable to misleading references. However, this problem might be partly addressed by integrating some in-loop

normative steps [90–92] or non-normative pre-processing steps [93–95] before the prediction.

- De-noising: Assume that there is a salt-paper-like noisy pixel in a block. The damage of this pixel is double-fold. First, a large residual must be transmitted when the noisy pixel is incorrectly predicted by its references. Second, it causes another incorrect prediction, when it is used as a misleading in-block reference. Therefore, one can remove the noise from this pixel by filtering the block before the prediction in order to avoid both problems.
- Thresholding: In screen content coding, it is common that the transition between background and foreground is gradual. However, in cases such as text coding, removing the gradual transition might cause a minor damage in terms of subjective quality. Therefore, one idea is to make the transitions sharper in order to help LOCO-I predictor. This idea can specifically be efficient when used with the above quantization step adjustment.

Improved ILR coding

Current residual coding of ILR-SQ uses a unary representation for the coding. This choice has been made during the ILR-SQ design, since ILR signals do not usually possess the energy compaction property, as transform domain residual signals. However, the current coding algorithm still has rooms to improve. Here a list of possible future works is presented briefly.

- Adopting the transform coefficient coding method [46]: Although this algorithm is highly optimized for coding of residual signals possessing the energy compaction property, it still has tools that might be helpful for ILR coding (e.g. coefficient groups, adaptive scans etc.). Therefore, one idea is to utilize a modified version of this algorithm and adapt it with the characteristics of the ILR framework. For instance, one can remove the coding of last significant position and simply encode all coefficients in a block. Another modification can be ILR rotation in order to produce a signal that has concentrated its zero amplitudes at the bottom-right corner. This modification may allow adopting the existing coefficient coding algorithm as is.
- Re-ordering amplitudes: There are usually a noticeable number of zero coefficients in most ILR signals, yet not locally concentrated in the block. Therefore, one might apply a re-ordering scheme to partially or completely shift zero coefficients to the end of block. The additional signaling of the re-ordering scheme may hopefully be rewarded with lower rate of ILR coding.
- Residual DPCM [57, 58]: As discussed in Chapter 2, the RDPCM algorithm benefits from a DPCM-based prediction of residual values in the spatial domain. One can also apply this algorithm on the ILR signal and losslessly code the residual of its prediction by RDPCM.

Part III
Proposed Transform Coding Tools

CHAPTER 7

Framework of Unary Bitplane Coding (UBC) of transform coefficients

Contents

1 Introduction

The general idea in this section is introducing a flexible bin coding framework with a more efficient source separation capability. Such framework potentially allows the encoding of arbitrary syntax elements of the standard. However, here, we specifically target the existing amplitude coding scheme of the VVC to be replaced by the proposed algorithm.

In a nutshell, the proposed framework first obtains as much contextual information as possible from a coding bin (e.g. binarized amplitudes). Then it studies the probability behavior of the bin in different contextual situations in order to optimize a merging scheme for clustering them. For this purpose, a set of contextual features are extracted and a look-up table is trained to map each contextual situation to an actual CABAC context.

2 Unary bitplanes representation

The unary bitplane representation plays a key role in achieving a proper source separation by the proposed framework. In this section, the representation of amplitude blocks by unary bitplanes is explained.

2.1 Binarization

Like any other telecommunication system, symbols and syntax elements have to be binarized before encoding. In the proposed framework, two main characteristics are specifically considered in the block-level: unary codes and bitplanes.

To compute unary bitplanes of a transform amplitude block, first, each amplitude is independently binarized. To do so, a decimal amplitude value of C is turned to a string of $C + 1$ bins, starting with C bins of 1 and a single terminal bin of 0. For instance, three amplitude values of 6, 2 and 0, have unary codes of 1111110, 110 and 0, respectively.

Once all bins of an amplitude block are computed, its bitplane representation can simply be performed by building a three-dimensional cube, where the Z-axis is the plane number. Fig. 7-1 visualizes how a 4×4 amplitude block is first converted to a block of unary codes and then cube of bitplanes. As can be seen, in each bitplane L, amplitudes can have three states:

1. Available with an intermediate bin of 1 (bright gray): the amplitude is larger than or equal to L.

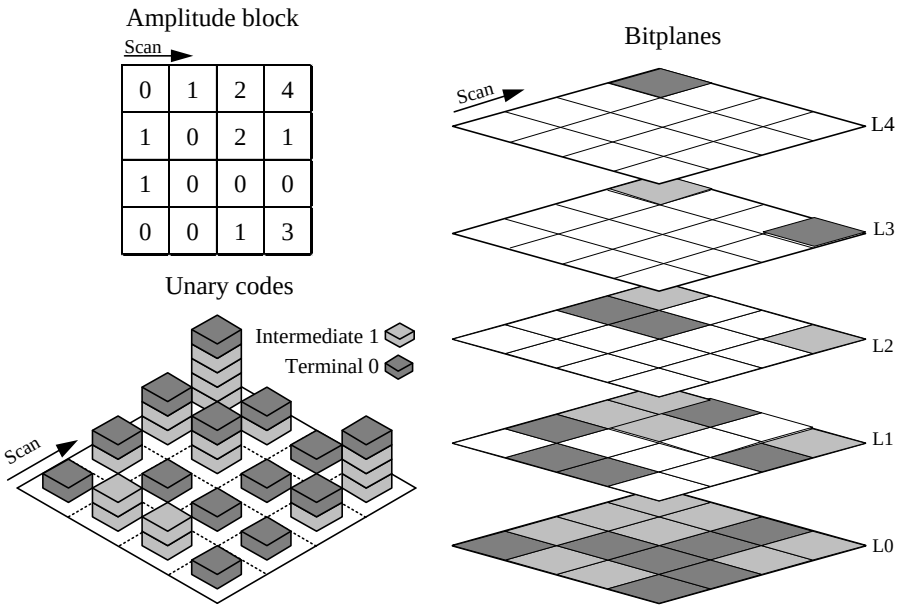
2. Available with the terminal bin of 0 (dark gray): the amplitude equals to L.

3. Non-available (white): the amplitude is smaller than L.

2.2 Why bitplane representation

Theoretically, the entropy of a bin is the only factor impacting its rate. In other words, the representation schemes such as unary, binary or in general, N-ary codes do not change the rate of a syntax element as long as its entropy is unchanged. Therefore, the use of bitplanes does not impose any additional cost in terms of rate. Bitplane representation, however, is designed for easy access to local information of the neighboring bins. As explained in Chapter 5, in a certain bitplane, one can simply obtain an approximation of other amplitudes in its neighborhood simply by evaluating their bin values up to the current layer. This property perfectly fulfills the need for local feature extraction in order to separate sources.

Figure 7-1: Unary bitplane binarization of a transform amplitude block.



2.3 Why unary bitplanes

Compared to the binary representations, unary codes instantly provide more information about the significance of a neighbor. Simultaneous use of unary codes and bitplane representation allows maximum contextual information access with a reasonable complexity.

Assume that the bins are encoded bitplane-by-bitplane and the encoder is currently in the L-th bitplane. This means that all the bin values of all other amplitudes in the lower bitplanes are visible. Now, in order to extract contextual information from a bin at the level L of a certain amplitude, the encoder aims to know the value of its neighbor amplitude given the information up to the bitplane *L-1*. By accessing an intermediate bin of '1' at the (*L-1*)-th level of the neighbor amplitude, the encoder can immediately infer that the actual value of the neighbor is at least *L-1*. On the contrary, to obtain the same knowledge about the neighbor amplitude, the encoder has to access all the bins between bitplanes 0 and *L-1*.

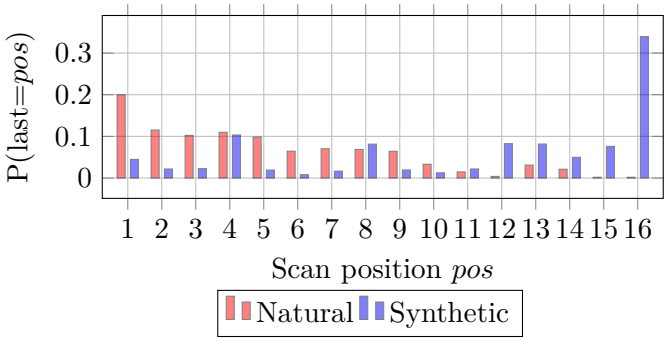


Figure 7-2: Last position histogram

3 Source separation of coding bins

Despite the well-know efficiency of the state-of-the-art coefficient coding scheme, we believe that there are still some correlations remained in the transform coefficient signals [proof]. Therefore, after applying unary bitplane binarization for representing transform amplitude blocks, the proposed algorithm studies different statistical behavior of bins with respect to local features with potential capability of source separation.

Given a feature space defined on the unary bitplanes, each point represents a feature vector and can be associated to any of the bins. In other words, different points can define different contextual situations of a bin. Therefore, for the sake of simplicity in the rest of this document, we use the term "situation" to indicate the contextual setting of a bin that is defined by its feature vector.

By investigating adequate number of coding bins in each situation, probability distribution function (PDF) of each situation can be used as an indicator of its statistical behavior. In the proposed framework, this information is exploited to categorize the situation and put statistically similar situations in the same cluster. Such clustering allows sharing CABAC to share contexts with all situations in each cluster and hopefully, enable a more efficient source separation.

As an example of source separation, assume the "coefficient significance" bin, that is coded for each transform coefficient to indicates whether its amplitude is non-zero. One feature with potential impact on the statistics of significance bin is the coefficient position in the block (i.e. frequency). Generally, low frequency coefficients around the top-left corner of the transform blocks are more likely significant, than high frequency coefficients around the bottom-right corner. Therefore, simply by distinguishing between high and low frequencies, one can potentially achieve compression gain. Another discriminative feature is the local density around the coefficient. More precisely, in a certain frequency, a coefficient that is surrounded mostly by significant neighbors is more likely significant than a coefficient surrounded mostly by non-significant neighbors.

A simple feature space formed by two features of frequency and density provide four possible situations are possible for a significance bit: 1) high frequency and high density, 2) high frequency and low density, 3) low frequency and high density and 4) low frequency and low density. This simple situation derivation scheme can be further improved either by increasing the number of features, or finer distinction of features (e.g. very low density, moderate density and very high density).

To show the impact of above source separation example, a dataset of samples from the actual coded streams was prepared. This dataset includes more than 100,000 samples of significance bits of 4×4 blocks from four sequences of class C in the quanization parameter of 22. Let B the binary random variable of coding bins and F and D the random variables of frequency and density, respectively. To compute the probabilities, we consider the frequency variable F as low, if located above the diameter that connects the top-right corner of the block to its bottom-left corner. Moreover, the neighborhood density of a coefficient D is considered as low density, if less than half of its available neighbors are significant. Using these two features, four source separation configurations can be applied on the dataset to derive the following probability distribution functions:

$$\begin{aligned} &\text{Cfg 1) No separation:} && P_u(B) = P(B), \\ &\text{Cfg 2) Only frequency:} && P_f(B) = P(B|F), \\ &\text{Cfg 3) Only density:} && P_d(B) = P(B|D), \\ &\text{Cfg 4) Both:} && P_{fd}(B) = P(B|F, D). \end{aligned} \tag{3.1}$$

To evaluate the above source separation configuration, the entropy of each configuration is

calculated as:

$$H = -\sum_{b=0,1} P_{\mathrm{Cfg}}(B=b) \times \log_2 P_{\mathrm{cfg}}(B=b), \tag{3.2}$$

where, cfg indicates the one of the four source separation configurations in Eq. 3.1. On the given dataset, the following entropy values where computed as in Table 7-1:

Table 7-1: Entropy of the significance bin with different source separation configurations.

cfg	Contextual features	Number of situation	P_{cfg}	H_{cfg}
u	None	1	$P_u(B) = P(B)$	0.98
f	F	2	$P_f(B) = P(B\|F)$	0.92
d	D	2	$P_d(B) = P(B\|D)$	0.67
fd	F,D	4	$P_{fd}(B) = P(B\|F,D)$	0.64

As can be understood from the above simple example, proper source separation by using discriminative features can reduce the average bins rate. Fig. 7-3 visualizes the impact of the above four source separation configurations on the histograms (i.e. PDF) of each sub-source.

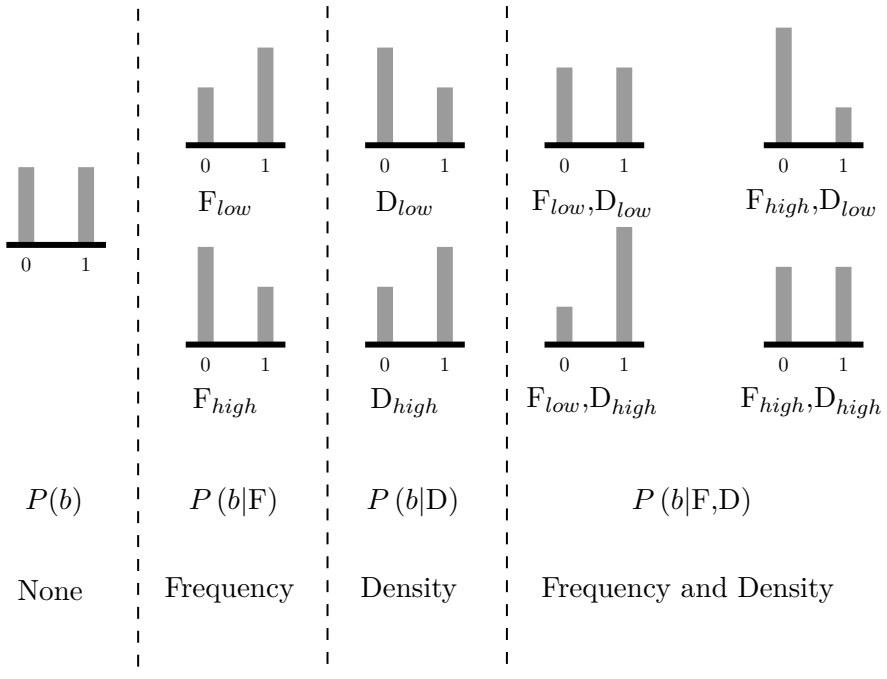


Figure 7-3: Four granularity levels of source separation and their impact on the probability distribution function of the significance bit.

In the proposed transform amplitude coding, we aim at using the bitplane representation of transform blocks to provide simple contextual information access for feature extraction. A classifier is then trained on the extracted features to optimize the source separation scheme constrained by the final number of clusters.

4 Situation clustering by K-Means

Increasing dimensions of the feature space by involving more relevant features in the previous example, can further increase the compression gain. However, due to the high cost of storing several CABAC context models at a time, this potential gain would rapidly become unfeasible in higher dimension feature space. To ease this restriction, second step of the proposed algorithm clusters the large set of situations into a small set of contexts. In practice, a working system may have up to millions of situations and the objective is to cluster them into a very limited number of CABAC contexts (e.g. 8, 16, 32 etc.). For this purpose, the proposed framework integrates an offline implementation of the K-Means on a big dataset of transform coefficient samples.

Let D be a dataset of actual amplitude samples from coded video streams in different QPs. Each entry of this dataset, corresponds to one transform block and provides the following information:

- Block width W and height H,
- Unary bitplanes of the $W \times H$ transform block as shown in Fig. 7-1,
- Other block-level contextual information such as intra prediction mode (IPM), transform type etc.
- Number of bits used for writing the coefficients to the bitstream. (do we really need it at this stage?)

Assuming that D contains a total of N_{bin} bins, one can define and extract a set of features from each bin and form a feature space. As explained earlier, the N_{sit} points in this feature space define different situations that can be associated to each bin. In fact, by considering the bins in D as the main source of information, the situations defined by the feature space separate this source into N_{sit} number of sub-sources. The main goal of K-Means is to optimize a coarser source separation and output a look-up table T, for mapping N_{sit} situations into N_{ctx} final contexts. Algorithm 3 describes the proposed K-Means scheme to optimize the situation to context mapping table.

As can be seen in Algorithm 3, the context of each situation sit is assigned after minimizing a rate function. This greedy approach uses the latest context assignment stored in the table T and attempts to update $T[sit]$ with a new context ctx^* with lowest rate, if possible. The goal is to repeat the above K-Means algorithm for adequate number of iterations until the table T is converged so that it can be integrated in the actual encoder.

In order to compute the rate of the signal in D, given the situation to context table T, the algorithm uses an entropy-based rate of the signal. Let B be the binary random variable of bins in D and $G(B)$ a function that extracts features of B and returns its situation index. Also define D_i as the subset of D that contains all the bins whose situations are mapped to context ctx_i:

$$D_i = \{B | B \in D, T[G(B)] = ctx_i\}. \tag{4.1}$$

The probability distribution function of the sub-source separated by context ctx_i can be expressed as:

$$P_i(B) = P(B | B \in D_i), \tag{4.2}$$

moreover, the entropy-based rate of the sub-source D_i is computed as:

$$Rate_i = -Len(D_i) \sum_{b=0,1} \log_2 P_i(B = b), \tag{4.3}$$

Algorithm 3 K-Means algorithm for partitioning N_{sit} situations into N_{ctx} clusters.

procedure K-MEANS
    Random initialization of T entries between 1 and N_{ctx}.
    $sit \leftarrow 1$ // iterator over the situations
top:
    if $sit > N_{sit}$ **then**
        return T
    $ctx \leftarrow 1$ // iterator over contexts
    $ctx^* \leftarrow -1$
    $Rate^* \leftarrow \infty$
loop:
    if $ctx > N_{ctx}$ **then**
        $T[sit] \leftarrow ctx^*$ // assigning the best context to the current situation
        $sit \leftarrow sit + 1$
        goto *top*
    else
        $T[sit] \leftarrow ctx$
        $Rate \leftarrow getRate(T)$
        if $Rate < Rate^*$ **then**
            $Rate^* \leftarrow Rate$
            $ctx^* \leftarrow ctx$
        $ctx \leftarrow ctx + 1$
    goto *loop*

where $Len(D_i)$ is the number of bins in D_i such as:

$$\sum_{i=0}^{N_{ctx}-1} Len(D_i) = N_{bin}. \tag{4.4}$$

Therefore, the total rate of the signal in D is calculated as:

$$Rate = \sum_{i=0}^{N_{ctx}-1} Rate_i = -N_{bin} \sum_{i=0}^{N_{ctx}-1} \sum_{b=0,1} \log_2 P_i(B = b) \tag{4.5}$$

5 Integration in the codec

Once the look-up table T is optimized by Algorithm 3, it can be integrated and used quite similarly at both encoder and decoder sides. For this purpose, the residual coding modules will be equipped with T and the unary bitplane binarization algorithm. This allows them to scan bins in bitplanes and perform the following steps:

- Extract feature values of each bin (e.g. frequency, density etc.)
- Compute their pre-defined production and obtain the situation number of the bin
- Access the situation to context table of T and extract the CABAC context number
- Encode/Decode the bin with the extracted CABAC context

Figure 7-4 visualizes the above process. In this figure, M features, having $N_1, N_2, ..., N_M$ different values, are used. Four above steps are shown with gray boxes in this figure and are repeated for each bin in the unary bitplane representation. As can be seen, the first step extracts

M features and passes them to the second step. Then, one of the N_{sit} possible values is selected as the situation number in the second step. In the third step, a CABAC context among N_{ctx} is selected using the table T. Finally, this CABAC context is used in the fourth and last step for coding.

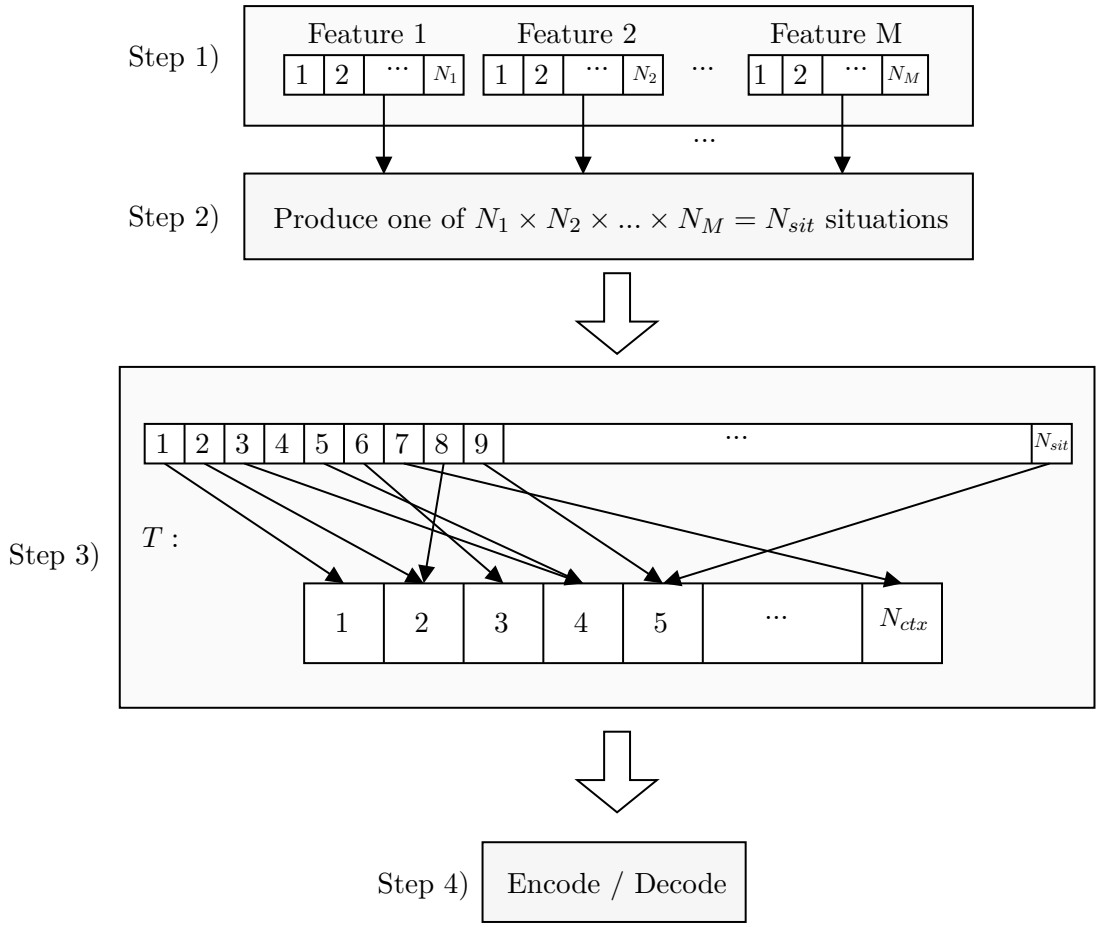


Figure 7-4: Diagram of the proposed UBC framework.

6 Conclusion

A general framework for source separation of coding bins was introduced. With two properties of unary coding and bitplane representations, this framework allows coding of several syntax elements. However, here the focus is mainly put on the transform coefficient coding task. In the next chapter, one implementation of this framework will be explained.

CHAPTER 8

Proposed UBC design

Contents

1 Introduction

In this chapter, a transform coefficient coding algorithm is proposed. This algorithm has been designed within the UBC framework, explained in Chapter 7. Without loss of generality, the proposed UBC design in this chapter only considers 4×4 intra residual blocks. However, a similar methodology can also be applied to other blocks sizes as well as the inter mode.

2 Feature space

A set of features with potential source separation power is introduced in this section. As will be seen, the main focus has been put on the feature extraction methodology, rather the nature of the features. In particular, it is more important to know how to manage the dynamic range of features, than which features to use. Once a proper feature space representation is determined, the nature of features could also be studied to improve the performance.

The features introduced in this section, form a feature space, as explained in the previous chapter. Each feature vector is shown by V and consists of four features representing density, bitplane height, frequency band and transform type. A feature vector associated to a bin in the UBC framework is then denoted as $V = \{v_d, v_l, v_f, v_t\}$. Moreover, B denotes a binary random variable (i.e. $B = 0, 1$) that represents coding bins from the unary bitplane representation.

2.1 Neighborhood density v_d

As discussed earlier, one informative feature with potential source separation capacity is the density of a neighborhood. Here, this feature is quantified with the number of significant coefficient around an amplitude and is denoted as the bin-level feature v_d. The bin-level aspect of v_d means that, density of bins in different bitplanes of a same coefficient is computed differently. More precisely, in a certain bitplane, a neighbor is considered as significant, if its unary code at that bitplane is 1. With this logic, significance of a neighboring coefficient at a certain bitplane is validated if that coefficient is not smaller than the bitplane.

At the decoder side, significance information of neighbors are available partially. Although it directly depends on the coding order of bins in unary bitplanes, it is technically impossible to be aware of all neighbors significance, given above significance definition. To address this problem, a 3×3 neighborhood is defined around each bin, as shown in Figure 8-1. Depending on the causality of a neighbor in this figure, the starred bin in the center looks at either current bitplane (i.e. neighbors at 0, 1, 2 and 3) or previous bitplane (i.e. neighbors at 4, 5, 6 and 7). The obtained bin information is then used to evaluate significance of each neighbor.

Given eight neighboring bins, $b_i,\ i = 0, 1, ..., 7$, as shown in gray background in Figure 8-1, the density feature can have $2^8 = 256$ different values and is calculated as:

$$f_d = \sum_{i=0}^{7} 2^{b_i} \tag{2.1}$$

It is important to note that there are two cases that some of the neighbors are unavailable. For the simplicity, in both below cases, the neighboring bin value is considered 0 for calculations:

1. At the block borders, some neighbors in both bitplanes are located outside of the block.
2. In the bitplane L_0, all non-causal neighbors at the bitplane $L - 1$ are unavailable.

Figure 8-2 shows the probability of $P_d(B = 1)$ in two different QP values of 22 and 37. For an easier visual inspection, a density index is defined as the number of significant neighbors with a v_d value. For instance, v_d values of $192 = b_0 : 0, b_1 : 0, b_2 : 0, b_3 : 0, b_4 : 0, b_5 : 0, b_6 : 1, b_7 : 1$ and $144 : b_0 : 0, b_1 : 0, b_2 : 0, b_3 : 0, b_4 : 0, b_5 : 1, b_6 : 0, b_7 : 1$ both have density index of 2. As can be seen in this figure, probability of having a significant bin is moderately dependent to density index. Therefore, it can be used as a proper feature for source separation by UBC.

Figure 8-1: The 3×3 neighborhood in bitplane L to model the spatial density v_d around the starred bin in the center.

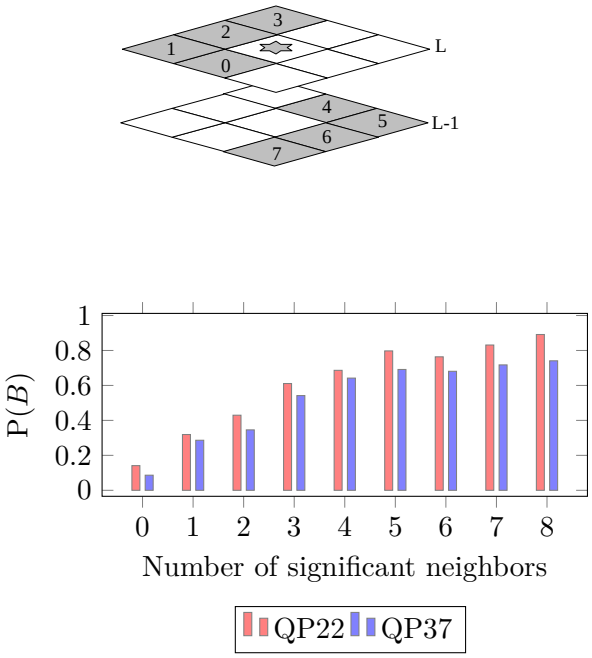


Figure 8-2: Probability of $P(B = 1)$, given the number of significant neighbors, determined by v_d.

2.2 Bitplane number v_l

The second proposed feature to extract from bin in UBC is simply the bitplane number. This feature is denoted as v_l. Experiments show that there is a strong, yet complicated correlation between v_l and $P(B)$. To understand this complexity, Figure 8-3 shows the relationship between v_l and $P(B = 1)$. As can be seen, the correlation model is completely different in two QPs. This complexity in the correlation model requires v_l to be considered jointly with other features, which will be achieved when v_l is integrated with other proposed features.

Technically, the upper bound of transform coefficient amplitudes is 32768. However, high amplitudes rarely occur, even in very low QP values. Therefore, a limitation is applied on the value of v_l, using the following threshold thr. By using this threshold, UBC can share one value of v_l for all bins above thr. In the proposed configuration of UBC, $thr = 32$ has been used.

$$v_l = \begin{cases} v_; & : v_l < \text{thr}, \\ \text{thr} & : otherwise. \end{cases}$$

2.3 Transform index v_t

One of the new tools in VVC standardization is the diversity of transform choices. This tool, called Explicit Multiple Transforms (EMT), allows the encoder to choose the best transformation for each residual. The best transform choice depends on different factors, most importantly, the prediction performance and shape of its residual, which directly depends on coded content. Therefore, a set of separable transforms are optimized and integrated within the VVC codec. This set consists of DCT-II as the default transform, and 4 additional transforms, namely

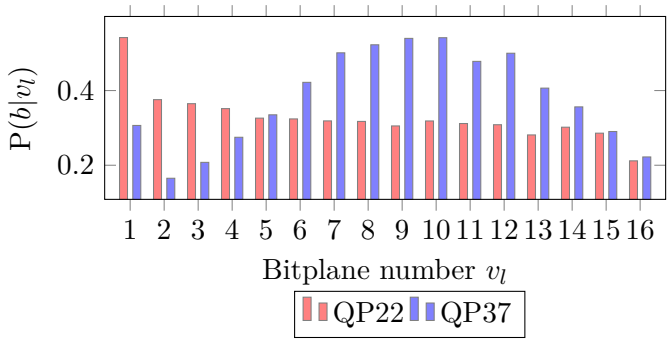


Figure 8-3: Joint probability of bin B and bitplane number v_l, in two QPs.

DST-VII, DCT-IX, DST-I and DCT-V. Each of these transforms can be applied separately in horizontal or vertical directions of a residual signal.

EMT in intra mode, allows explicitly signaling a transform pair (T_{hor}, T_{ver}) for each block, with respect to the selected IPM. However, the number of all possible pairs is too large (i.e. $\binom{5}{2} \times 67{=}670$), hence too expensive in terms of rate. Therefore, possible pairs for each are limited with respect to some block specifications. For this purpose, an EMT flag is first transmitted, if this flag is set to 0, DCT-II is used both for vertical and horizontal directions. Otherwise, an EMT index between 0 and 3 is also transmitted to choose one of the four possible pairs of (T_{hor},T_{ver}) for the block. Table 8-1 shows how the EMT index chooses the transform pair for each selected IPM. Combination of IPM and the EMT index.

A block-level feature, based on the selected transform of a block is defined in the proposed UBC design. The value of this feature, denoted as v_t is derived directly from the selected transform pairs in Table 8-1. In this table, only 12 unique pairs are used with EMT. One additional pair is also considered as the default transform of a block, when EMT flag is not set. This pair is $(DCT-II, DCT-II)$. Therefore, the proposed v_t feature can have a total of 13 values, as specified in Table 8-2.

It is important no not that different v_t values are weakly correlated to selected IPM of blocks associated with residual. This can hopefully result in a achieving a better source separation capacity by using v_t. Figure 8-4 shows probability of $P(B = 1|v_t)$, in two different QP values. As can be seen, there is a weak correlation between the selected transform and the probability of $P(B)$.

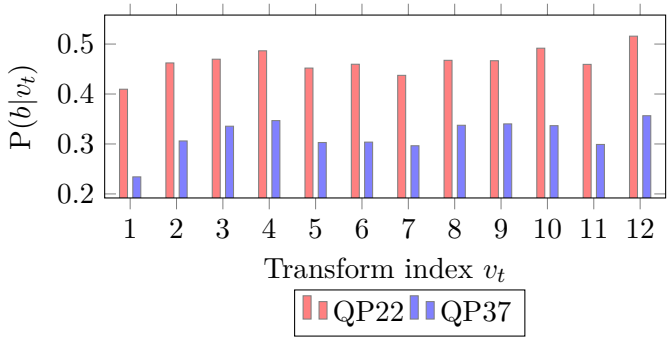


Figure 8-4: The relationship between the transform feature v_t of a bin B and its probability of $P_v(B = 1)$, in two QPs.

EMT index	Planar	DC	IPM 2/66	3/65	4/64
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DCT-V,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)
2	(DST-VII,DCT-V)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)
3	(DCT-V,DCT-V)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)

EMT index	5/63	6/62	IPM 7/61	8/60	9/59
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)
2	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)
3	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)

EMT index	10/58	11/57	IPM 12/56	13/55	14/54
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-V,DST-VII)
2	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)
3	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-V,DCT-IIX)

EMT index	15/53	16/52	IPM 17/51	18/50	19/49
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DCT-V,DST-VII)	(DCT-V,DST-VII)	(DCT-V,DST-VII)	(DCT-V,DST-VII)	(DCT-V,DST-VII)
2	(DST-VII,DCT-IIX)	(DST-VII,DCT-IIX)	(DST-VII,DCT-IIX)	(DST-VII,DCT-IIX)	(DST-VII,DCT-IIX)
3	(DCT-V,DCT-IIX)	(DCT-V,DCT-IIX)	(DCT-V,DCT-IIX)	(DCT-V,DCT-IIX)	(DCT-V,DCT-IIX)

EMT index	20/48	21/47	IPM 22/46	23/45	24/44
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DCT-V,DST-VII)	(DCT-V,DST-VII)	(DCT-V,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)
2	(DST-VII,DCT-IIX)	(DST-VII,DCT-IIX)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)
3	(DCT-V,DCT-IIX)	(DCT-V,DCT-IIX)	(DCT-V,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)

EMT index	25/43	26/42	IPM 27/41	28/40	29/39
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)
2	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)
3	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)

EMT index	30/38	31/337	IPM 32/36	33/35	34
0	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)	(DST-VII,DST-VII)
1	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)	(DST-I,DST-VII)	(DCT-IIX,DST-VII)
2	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)	(DST-VII,DST-I)	(DST-VII,DCT-IIX)
3	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)	(DST-I,DST-I)	(DCT-IIX,DCT-IIX)

Table 8-1: Horizontal and vertical transforms in each IPM, signaled by EMT index, denoted as (T_{hor},T_{ver}).

2.4 Frequency band v_f

Benefiting from the energy compaction property of transforms is a motivation to use it as a feature. This property makes it possible to represent most of information with a few transform

		Vertical				
		DST-VII	DCT-IIX	DST-I	DCT-V	DCT-II
Horizontal	DST-VII	1	5	8	10	-
	DCT-IIX	2	6	-	11	-
	DST-I	3	-	9	-	-
	DCT-V	4	7	-	12	-
	DCT-II	-	-	-	-	0

Table 8-2: v_t feature value based on the transform pair (T_{hor}, T_{ver}) from Table 8-1. Empty entries, denoted with '-', are combinations that did not appear in Table 8-1.

coefficients in low frequencies. Therefore, one can exploit the high correlation of coefficient frequency band and probability of B.

The frequency band feature, denoted as v_f, is used in the proposed UBC design. v_f is defined as a coefficient level feature and uses the an index of the frequency band as its value. Figure 8-5 demonstrates two 4×4 blocks of transform coefficient unsigned amplitudes, corresponding to two QPs. In this figure, the lowest and higher frequency bands are shown at the top and bottom of the 3D bar plots. Moreover, the spectrum legend provided at left, guides at interpreting colors into different values of $P(B = 1)$. As can be seen, different frequency bands represented with different values of v_f can help in producing sub-sources of unary bins with various probabilities.

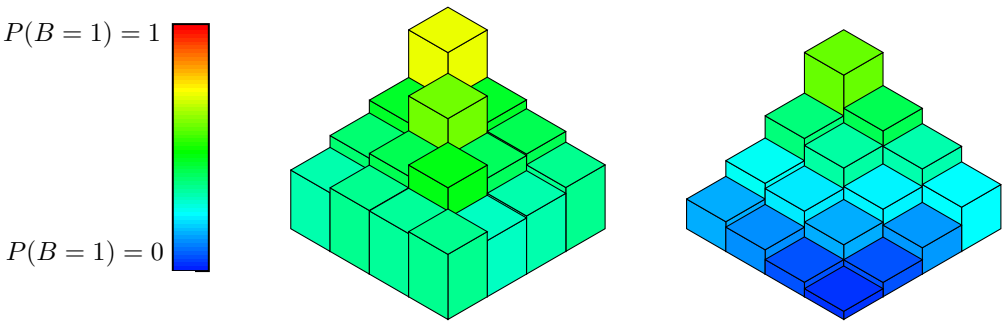


Figure 8-5: The relationship between the frequency band of a coefficient in 4×4 blocks and probability of $P(B = 1)$, in two QPs. 3D bars at the top and bottom of each plot represent low and high frequency bands, respectively.

3 Feature space reduction by chunking

Capturing the full potential of source separation capacity by all features requires an unfeasible amount of memory. Particularly, four introduced features in this section, form the following feature space of V for 4×4 transform coefficient blocks:

$$V \in (0 \leq v_d < 256) \times (0 \leq v_l < 32 \times (0 \leq v_t < 13) \times (0 \leq v_f < 16). \tag{3.1}$$

The above ranges produce a 4D feature space of $256 \times 32 \times 13 \times 16 = 1,703,939$ different feature vectors. In other worlds, using feature vector V in its full range would produce an excessively fine set of contextual situations for bins in unary bitplanes of 4×4 blocks. However,

this number of contextual situations in the UBC framework requires storing a huge and unfeasible situation to context table with over 1,7 million entries.

In this section, a chunking algorithm is presented to bucketize features in V. This algorithm aims at separately reducing the range of each feature, with minimum performance sacrifice in terms of compression efficiency. This step is crucial for a practical implementation of the proposed UBC algorithm, as it reduces the size of situation to context table that has to be stored at the encoder and decoder sides. Figure 8-6 shows where the feature chunking module stands with respect to other modules showed in Figure 7-4.

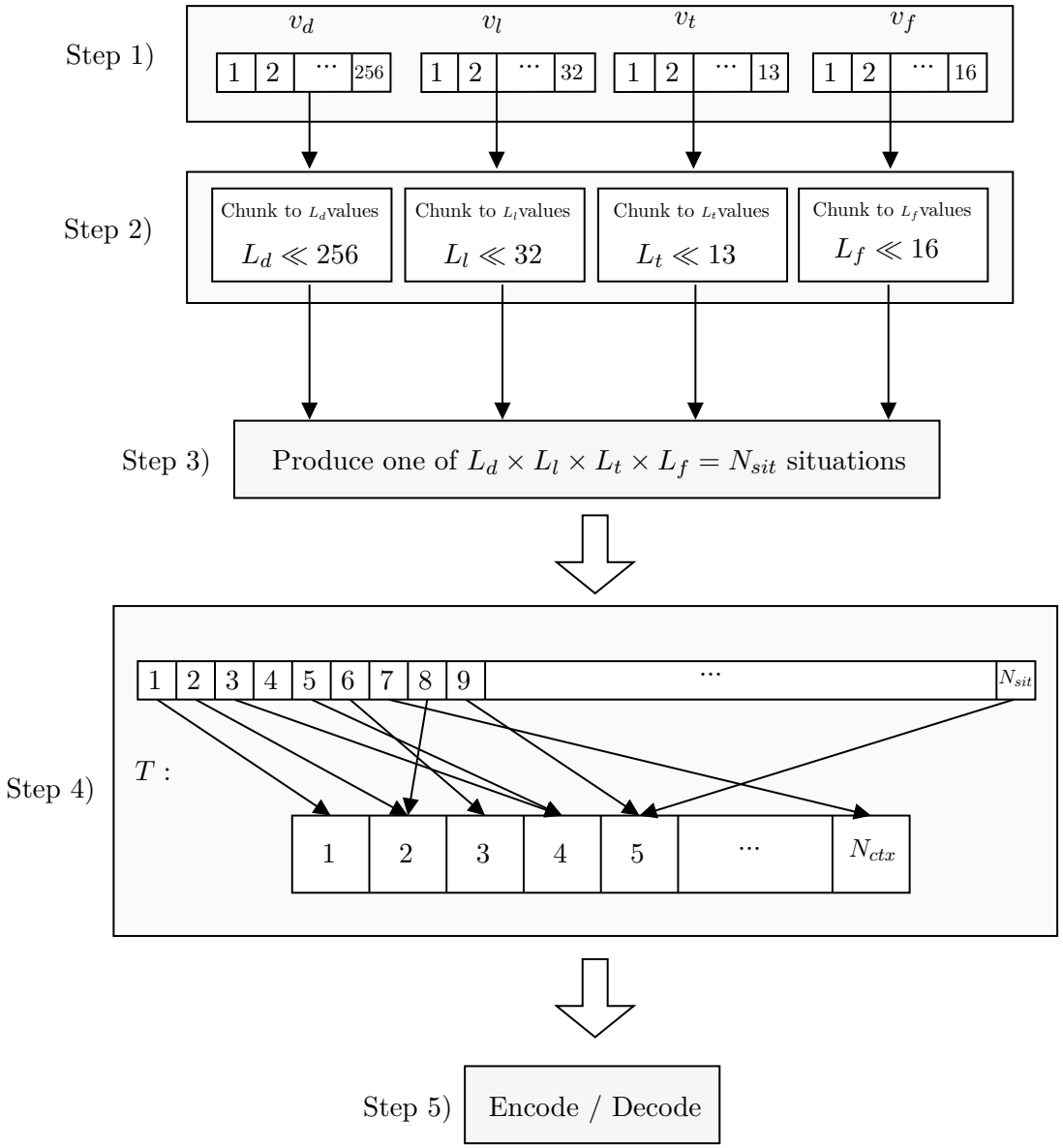


Figure 8-6: Chunking

According to Figure 8-6, a UBC design integrated with feature chunking performs the following five steps for coding each bin:

- Extract feature values of v_d, v_l, v_t and v_f from the bin.
- Perform chunking on each feature independently and reduce their range to L_d, L_l, L_t and L_f values, respectively. This can be performed by storing small look-up tables for each feature to map from the initial range to the chunked range.

- Compute the pre-defined production of chunked feature values and obtain the situation number of the bin.
- Access the situation to context table of T and extract the CABAC context number.
- Encode/Decode the bin with the extracted CABAC context.

In order to obtain the chunking look-up tables of features, an optimization has been carried out. This process is very similar to the K-means scheme explained in Algorithm 3 of Chapter 7.

3.1 Rate measurement

Each chunking scheme on the feature space of V has a different impact on the final performance of the codec implementing it. Therefore, in order to cope the offline nature of the proposed K-means algorithm, an offline rate measure is used. This measure is very similar to the *getRate* function in Algorithm 3 that operates on the situation to context table T. The only difference is that in the existence of a chunking step, table T is smaller, hence, contextual situations of bins are defined coarser.

As discussed earlier, coarser representation of contextual situations always results in a lower performance in terms of entropy coding. Therefore, the offline rate measurement is used to make a compromise between the size of the chunking tables and the coding performance.

3.2 Performance drop of chunking

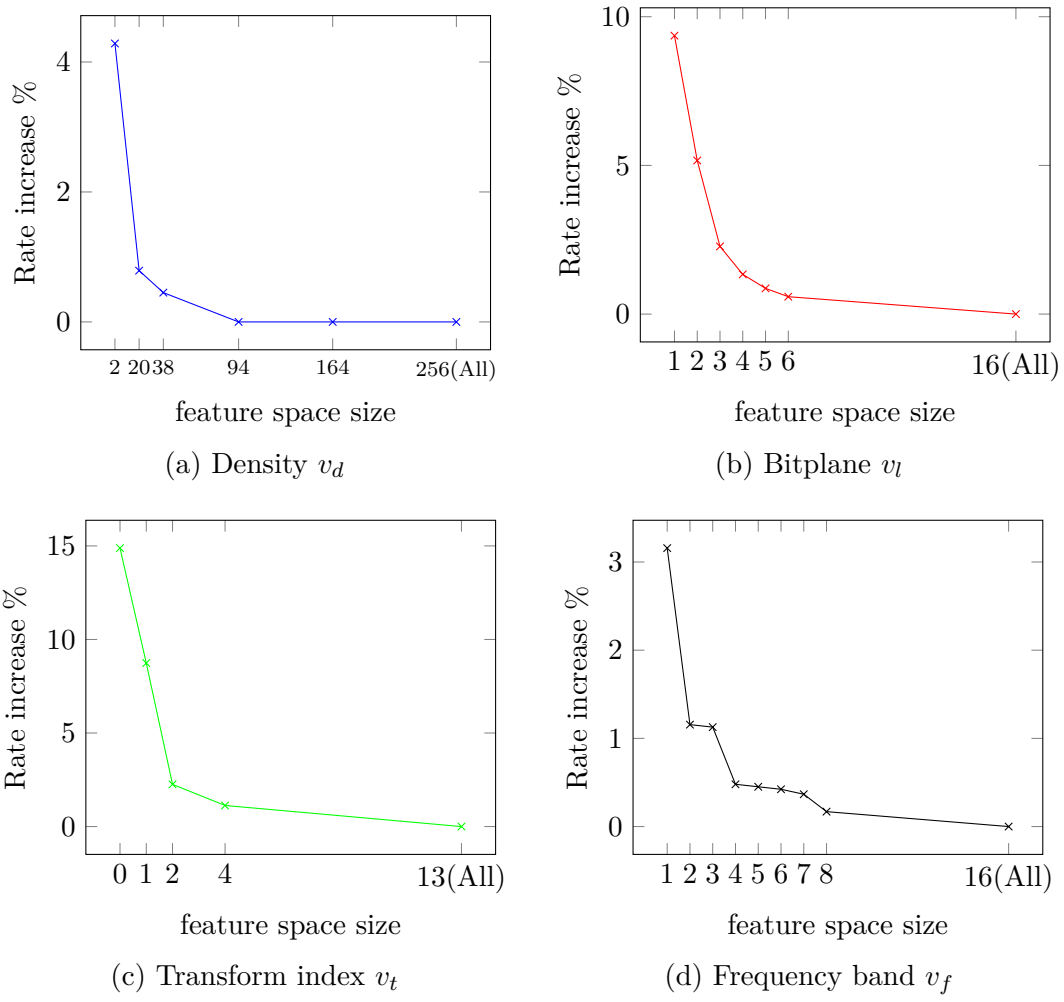
Different chunking schemes have been applied on the four features in V in order to produce a wide range of contextual situation granularity. This is performed on each feature separately. Here, the performance drop of each feature in different granularity levels are presented. Figure 8-7 shows the performance drop due to the chunking of each feature in V. In each curve, x-axis presents a range of its feature that has been tested during the chunking step. The last value on each x-axis denotes "no chunking" setting that uses all values in range of the feature. The corresponding value on y-axis shows the rate increase on the dataset, compared to "no chunking" setting, which has been calculated with the offline measure explained above.

3.3 Recommended chunking scheme

Based on the results presented in Figure 8-7, a chunking scheme is optimized. This scheme has compromised the coding performance in order to reduce shrink situation to context table T. The optimized chunked sizes are shown in Table 8-3. As can be seen, this chunking scheme reduces the size of T from 1,7 million to 2432 entries, which is easily implementable.

Table 8-3: Reduced range of features in V and table T, before and after chunking.

Feature	Original range	Reduced range
v_d	256	38
v_l	32	4
v_t	13	4
v_f	16	4
V (All)	1,703,939	2432



(a) Density v_d (b) Bitplane v_l

(c) Transform index v_t (d) Frequency band v_f

Figure 8-7: Performance drop due the chunking step on each feature in V.

4 Proposed UBC codec

Once the situation to context table is ready and integrated in the codec, it can be used for coding quantized transform coefficient. At the encoder side, the following steps are performed in the given order to guarantee the desired contextual information access:

1. Binarization using unary bitplanes in Figure 7-1.

2. Loop over bitplane levels.

2.1. At each bitplane level lvl, loop over coefficients in the raster scan.

2.1.1. At each coefficient cf at level lvl, access corresponding bin b:

2.1.2. Extract feature vector as $V =< v_d, v_l, v_t, v_f >$.

2.1.3. Quantize V.

2.1.3. Compute situation index.

2.1.4. Extract CABAC context number ctx, using the situation to context table T.

2.1.5. Encode b with ctx CABAC context.

At the decoder side, the UBC algorithm performs similar operations, except that in the beginning, the unary bitplanes are empty. In fact, the decoder side gradually builds the unary bitplanes as it decodes bins in the above order.

5 Results

In this section, we present results of integrating the proposed UBC algorithm in JEM. For this purpose, choices had to be made in the design of UBC algorithm. Here is a brief list of these design choices:

- Only $4times4$ blocks. Other block sizes keep using the existing coefficient coding algorithm of JEM.
- Only intra residuals. Inter residual blocks keep coded by the existing coefficient coding algorithm of JEM.
- Four features of v_d, v_l, v_t and v_f, with a chunking scheme explained in Section 3. Hence a situation to context table T of size 2432 entries.
- 16 final CABAC contexts.

5.1 Natural content

Table 8-4 shows the performance of UBC on natural sequences of JVET, in terms of BD-R gain. As can be seen, UBC is able to slightly improve the coding performance of JEM, with a cost of encoding and decoding complexity. However, it is important to note that unlike the existing algorithm in JEM, which has been optimally implemented, the current implementation of UBC is not mature and has lots of rooms to improve.

5.2 Screen content

As discussed, existing transform coefficient coding of VVC on screen content is not as efficient as it is on natural content. For this purpose, the performance of UBC has been evaluated separately on a large set of synthetic contents. Table. 8-5 shows the result of using UBC on this set of sequences. For a better comparison, these results are visualized in Figure 8-8. As can be seen, except on sequence (Robot), a consistent performance over all synthetic sequences has been achived with UBC. In this sequence, the texture is very close to natural content, hence more difficult for UBC to improve.

6 Conclusion and future work

6.1 General eyesight

Current UBC framework of Chapter 7 and the proposed design of Chapter 8 are premature and still have room for improvements. The existing method is applicable only to a subset of blocks that have certain conditions (e.g. intra blocks of size 4×4). The drawback of such restrictions is two-folded. First, it keeps us from exploiting other blocks to improve the coding efficiency. Second, it still requires the codec to implement the regular residual coding algorithm for other blocks. This aspect is in contrast with the initial goal of codec complexity reduction with UBC.

Here, a brief list of possible future tracks in the domain of UBC is presented.

Table 8-4: The performance of the proposed UBC configuration on natural content, when applied on transform coefficients of 4×4 intra blocks.

Class	Sequence name	BD-R	Enc. Time	Dec. Time
Class A	Traffic	-0.03%	0,00%	0,00%
	PeopleOnStreet	-0.11%	0,00%	0,00%
	NebutaFestival	0.01 %	0,00%	0,00%
	SteamLocomotive	0.00 %	0,00%	0,00%
	Average	-0.03%	0,44%	0,54%
Class B	Kimono	0.04%	-0,22%	-0,13%
	ParkScene	-0.16%	0,59%	0,71%
	Cactus	-0.19%	0,28%	0,16%
	BasketballDrive	-0.04%	0,66%	-0,77%
	BQTerrace	-0.42%	1,95%	0,97%
	Average	-0.15%	0,44%	0,54%
Class C	BasketballDrill	0.12%	-0,35%	0,85%
	BQMall	-0.03%	0,24%	2,75%
	PartyScene	0.04%	1,39%	1,16%
	RaceHorses	0.08%	0,03%	0,40%
	Average	0.05%	0,44%	0,54%
Class D	BasketballPass	-0.13%	1,25%	0,63%
	BQSquare	-0.18%	2,34%	1,39%
	BlowingBubbles	0.24%	0,29%	1,15%
	RaceHorses	0.21%	1,09%	1,10%
	Average	0.04%	0,44%	0,54%
Class E	FourPeople	-0.16%	0,37%	-0,41%
	Johnny	-0.47%	-0,66%	2,55%
	KristenAndSara	-0.46%	1,37%	0,32%
	Average	-0.36%	0,44%	0,54%
All		-0,26%	0,44%	0,54%

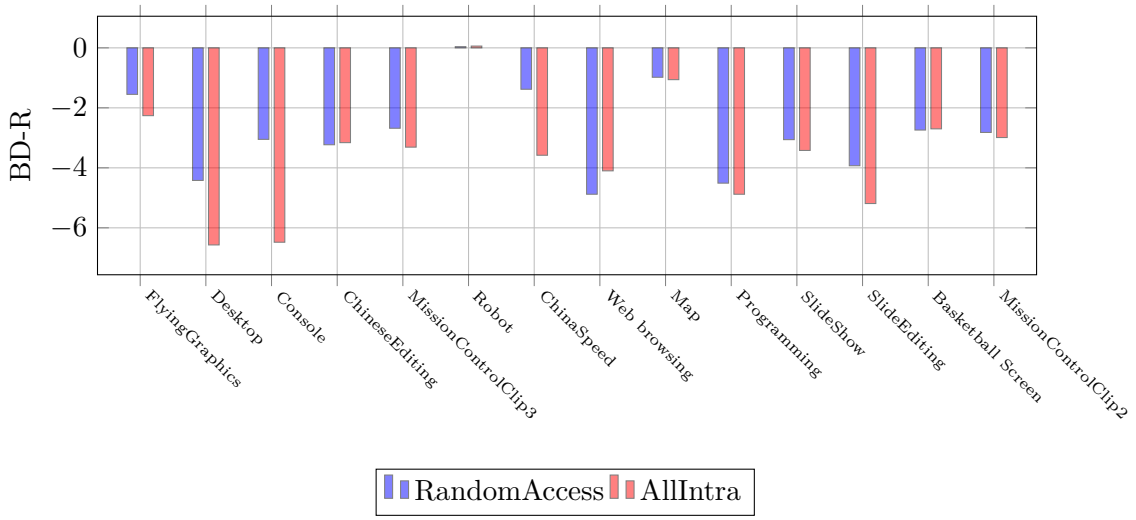


Figure 8-8: The BD-R performance of UBC on screen content.

Table 8-5: The BD-R performance of UBC on screen content.

Res.	SequenceName	Randomaccess		Allintra	
		BD-rate	ET/DT	BD-rate	ET/DT
1920 × 1080	FlyingGraphics	-1.55	107/103	-2.26	225/102
	Desktop	-4.42	103/103	-6.57	220/101
	Console	-3.05	105/103	-6.48	198/102
	ChineseEditing	-3.23	101/102	-3.16	234/100
	MissionControlClip3	-2.68	106/105	-3.31	221/100
	Robot	+0.04	114/103	+0.06	209/100
	ChinaSpeed	-1.38	107/103	-3.58	107/100
	Average	**-2.32**	**106/103**	**-3.68**	**217/101**
1280 × 720	Webbrowsing	-4.88	101/104	-4.10	195/100
	Map	-0.98	110/103	-1.06	229/101
	Programming	-4.51	106/104	-4.88	216/100
	SlideShow	-3.06	109/103	-3.42	179/98
	SlideEditing	-3.93	104/102	-5.19	229/100
	Average	**-3.47**	**107/103**	**-2.84**	**208/100**
2560 × 1440	BasketballScreen	-2.74	105/102	-2.70	209/99
	MissionCtrlClip2	-2.82	104/101	-2.99	206/101
	Average	**-2.78**	**105/102**	**-2.84**	**207/100**
	TotalAverage	**-2.8**	**105/103**	**-3.4**	**213/100**

6.2 Larger blocks, inter blocks

One can expand the scope of UBC by removing basic restrictions, such as block size and block type. As discussed, this can potentially help at improving both the coding efficiency of the codec and its implementation complexity. More precisely, by efficiently enabling UBC for all block sizes and types, one can entirely replace the existing coefficient coding and simplify the codec. In addition to codec simplification, such replacement might also improve the coding efficiency.

6.3 Natural content

As was discussed, the proposed UBC design currently brings gain only for screen contents. However, it has a worse performance than the existing method of VVC, on natural contents. Therefore, another future track can be optimizing a universal design that hopefully performs efficiently on both natural and screen contents.

6.4 Applications on other syntax elements

The general idea of coding a syntax element, with respect to the amplitude of similar syntax elements in its neighborhood, can be extended to coding modules than coefficient coding. For instance, Motion Vectors (MV) coding can use a similar concept to benefit from the correlation between MV of neighboring blocks.

Thesis conclusion

Video coding, in general, takes advantage of basic science and understanding of human perception, in order to efficiently give information to viewers in tractable units. In the past three decades, a handful of international video coding standards have been released. Some widely adopted and changed the broadcast industry, some disappeared shortly after their failure. As of the time of this thesis, the next generation standard, called Versatile Video Coding (VVC), is planned to be released in 2020 in order to significantly improve its predecessor, High Efficiency Video Coding (HEVC), both in terms of coding efficiency and diversity in input video format.

The objectives of this thesis have been highly aligned with those of the VVC standardization. This research has been conducted within two main modules of VVC, attempting at improving their coding efficiency on different video formats. The coding modules that have been focused on, are mainly intra prediction and its residual coding. Regarding the signal format, screen video contents have been specifically targeted in this thesis, along with natural contents. Some interesting results, achieved during different periods of this research, have been published in the forms of international conference and journal papers, patents and standardization documents.

Intra prediction

The intra coding track of this research proposes new methods for coding high detail textures. Traditionally, this type of texture have been challenging to compress, since the conventional methods are unable to properly model them. To address this problem, a flexible framework with the use of short distance prediction has been proposed, first. Based on the principles of this framework, two intra coding algorithms have been proposed to improve the coding efficiency of high detail textures in VVC. Experiments show a coding gain, especially on screen contents.

One of the algorithms in the intra prediction domain is currently under consideration for adoption in the VVC. This technology, called Block Differential Pulse-Code Modulation (BDPCM), is entirely designed based on the ILR-SQ algorithm which has been discussed in Chapter 5. As of today, some other research entities (e.g. LG Electronics, HHI etc.) are developing the proposed algorithm within a standardization Core Experiment (CE) that has been dedicated to screen content coding.

Thanks to the numerous features and specifications of the proposed framework of this Part, a handful of open questions and modification ideas have remained as the future work. These tracks include alternative algorithms in pixel prediction, residual representation, encoder acceleration and so on. The preliminary results and statistics, presented in Chapter 6, show potential improvements within the proposed intra prediction framework.

Transform coding

As the prediction framework imposes significant modifications on the residual representation and coding, some attempts to improve VVC in this domain have also been carried out. With this regard, the entropy coding aspect of residual transmission has been thoroughly studied.

Similar to the intra coding track, first a general framework has been designed in order to allow

implementing ideas in various forms. In this framework, the concept of using Unary Bitplane Codes (UBC) for binarization and coding has been investigated. Based on this idea, one residual coding method has been proposed. The experimental results show that the proposed method brings promising gain both in terms of coding efficiency and codec complexity reduction, on screen content coding.

Similar to the prediction part of this thesis, there seems to be much rooms for improvement in the transform coding domain as well. There are mainly two tracks on which the UBC framework can extend: 1) Applying it on wider range of block sizes/types. 2) Applying it on coding elements other than transform coefficients e.g. motion vectors, block partitioning etc.

Implementations

All implementations have been carried out in standard reference softwares, namely Joint Exploration Model (JEM) and VVC Test Model (VTM). For this purpose, the reference under study was used as a skeleton and proposed methods were added on top of it.

In order to conform the standardization regulations, the Common Test Conditions (CTC) have been used for evaluating the performance in different stages of this thesis. This includes the test sequences, codec parameters, coding types, performance measurements etc.

APPENDIX A

Publications

Papers

1. M. Abdoli, F. Henry, P. Brault, P. Duhamel and F. Dufaux, "Intra Prediction Using In-Loop Residual Coding for the post-HEVC Standard", MMSP - IEEE 19th International Workshop on Multimedia Signal Processing, Luton, UK, 2017.
2. M. Abdoli, F. Henry, P. Brault, P. Duhamel and F. Dufaux, "Short Distance Intra Prediction of Screen Content in Versatile Video Coding (VVC)", IEEE Signal Processing Letters, 2018.
3. M. Abdoli, F. Henry, P. Brault, F. Dufaux and P. Duhamel, "Transform Coefficient Coding for Screen Content in Versatile Video Coding (VVC)", ICASSP - IEEE 19th International Conference on Acoustics, Speech, and Signal Processing, Brighton, UK, 2019.
4. M. Abdoli, F. Henry, P. Brault, F. Dufaux and P. Duhamel, "Intra Block-DPCM With Layer Separation of Screen Content in VVC (VVC)", ICIP - IEEE 26th International Conference on Image Processing, Taipei, Taiwan, 2019.

Patent applications

1. F. Henry, M. Abdoli, "Video Coding Using Progressive Prediction", patent application INPI 1754688, May 2017.
2. F. Henry, M. Abdoli, "Video Coding Using Intra Prediction and In-Loop Prediction", patent application INPI 1855791, June 2018.
3. F. Henry, M. Abdoli, "Video Coding Using Associated Intra Mode", patent application INPI 1855792, June 2018
4. F. Henry, M. Abdoli, G. Clare, P. Philippe, "Image Coding Using Syntax Subset", patent application INPI 1855792, September 2018.
5. F. Henry, M. Abdoli, G. Clare, P. Philippe, "Image Coding Using Multiple Modes and Block Size Limitation", patent application INPI 1855792, September 2018.
6. F. Henry, M. Abdoli, G. Clare, P. Philippe, "Image Coding Method and Device", patent application INPI 1855792, September 2018.
7. F. Henry, M. Abdoli, G. Clare, P. Philippe, "Image Coding Method and Device, patent pending", October 2018.

Standardization contributions

1. M. Abdoli, G. Clare, F. Henry, P. Philippe, (AHG11): Block DPCM for Screen Content Coding, Document JVET-L0078, Macau, China, October 2018.
2. M. Abdoli, G. Clare, F. Henry, P. Philippe, CE8: BDPCM with LOCO-I and independently decodable areas, Document JVET-M0056, Marrakesh, Morocco, January 2019.
3. M. Abdoli, G. Clare, F. Henry, P. Philippe, CE8: BDPCM with horizontal/vertical predictor and independently decodable areas, Document JVET-M0057, Marrakesh, Morocco, January 2019.
4. M. Abdoli, G. Clare, F. Henry, P. Philippe, CE8: BDPCM with modified binarization, Document JVET-M0058, Marrakesh, Morocco, January 2019.

APPENDIX B

Common Test Condition (CTC) sequences

Class A1

CampfireParty 3840×2160

Drums 3840x2160

Tango 4096x2160

ToddlerFountain 4096x2160

Class A2

CatRobot 3840x2160

DaylightRoad 3840x2160

RollerCoaster 4096x2160

TrafficFlow 3840x2160

Old class A (2560×1600)

Traffic

SteamLocomotiveTrain

PeopleOnStreet

NebutaFestival

Class B (1920×1080)

RitualDance

MarketPlace

BasketballDrive

BQTerrace

Cactus

Kimono

ParkScene

Class C (832×480)

BasketballDrill

BQMall

RaceHorses

PartyScene

Class D (416×240)

BasketballPass

BlowingBubbles

BQSquare

RaceHorses

Class E (1280×720)

FourPeople

Johhny

KristenAndSara

Class F

SlideShow (1280×720)

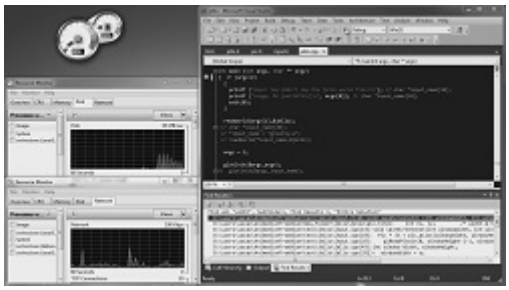
Slideediting (1280×720)

Chinaspeed (1024×768)

Basketballdrilltext (832×480)

Class TGM

Robot (1280×720)

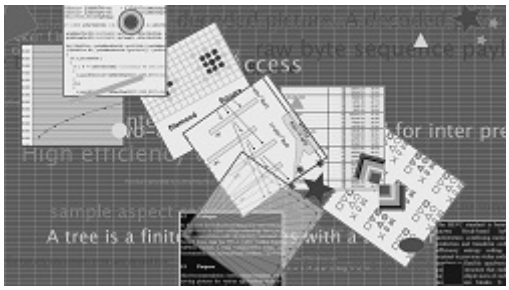


Programming (1280×720)

MissionControlClip3 (1920×1080)

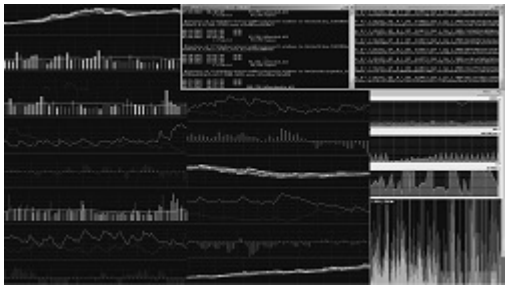
MissionControlClip2 (2560×1440)

Map (1280×720)

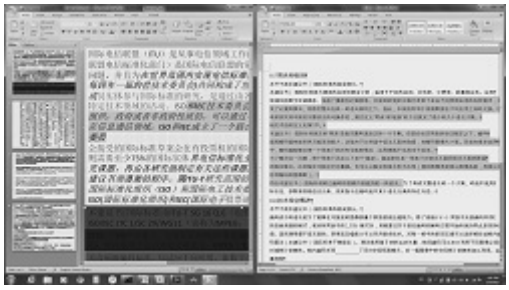


FlyingGraphics (1920×1080)

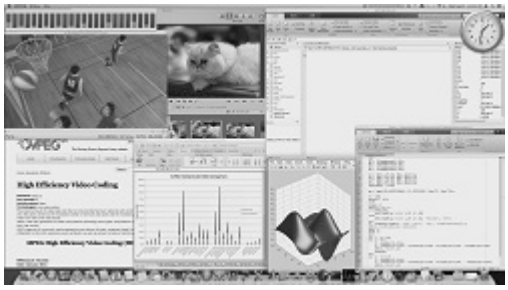
Class TGM



Console (1920×1080)



ChineseEditing (1920×1080)



Basketballscreen (2560×1440)

Web-browsing (1280×720)

Bibliography

[1] Cisco Visual networking Index. Forecast and methodology, 2016-2021, white paper. *San Jose, CA, USA*, 1, 2016.

[2] M. Wien. *High efficiency video coding, Coding Tools and specification*. Springer, 2015.

[3] Claude E Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1, 1959.

[4] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[5] Jorma J Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3):198–203, 1976.

[6] Hannuksela M., Wang Y., and Gabbouj M. Isolated regions in video coding. *IEEE Transactions on Multimedia*, 6(2):259–267, 2004.

[7] Zhang R., Regunathan S., and Rose K. Video coding with optimal inter/intra-mode switching for packet loss resilience. *IEEE Journal on Selected Areas in Communications*, 18(6):966–976, 2000.

[8] Lee T. A new frame-recompression algorithm and its hardware design for mpeg-2 video decoders. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(6):529–534, 2003.

[9] Lainema J., Bossen F., Han W., Min J., and Ugur K. Intra coding of the hevc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1792–1801, 2012.

[10] Wiegand T., Sullivan G., Bjontegaard G., and Luthra A. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[11] Cai Q., Song L., Li G., and Ling N. Lossy and lossless intra coding performance evaluation: Hevc, h. 264/avc, jpeg 2000 and jpeg ls. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–9, 2012.

[12] Nguyen T. and Marpe D. Performance analysis of hevc-based intra coding for still image compression. In *Picture Coding Symposium (PCS), 2012*, pages 233–236, 2012.

[13] Miska M Hannuksela, Jani Lainema, and Vinod K Malamal Vadakital. The high efficiency image file format standard [standards in a nutshell]. *IEEE Signal Processing Magazine*, 32(4):150–156, 2015.

[14] Jani Lainema, Miska M Hannuksela, Vinod K Malamal Vadakital, and Emre B Aksu. Hevc still image coding and high efficiency image file format. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 71–75, 2016.

[15] Tu Y., Yang J., and Sun M. Efficient rate-distortion estimation for h. 264/avc coders. *IEEE Transactions on circuits and systems for video technology*, 16(5):600–611, 2006.

[16] Reuze K, Philippe P., Deforges O., and Hamidouche W. Intra prediction modes signalling in hevc. In *IEEE Picture Coding Consortium (PCS2016)*, 2017.

[17] Reuze K., Philippe P., Deforges O., and Hamidouche W. Intra prediction modes signalling in hevc. In *Picture Coding Symposium (PCS), 2016*, pages 1–5, 2016.

[18] Kim S. Chen J., Ye Y. Algorithm description for versatile video coding and test model 2 (VTM). *Document JVET-K1002*, Ljubljana, Slovenia, July 2018.

[19] J. Chen, E. Alshina, G. J. Sullivan, J.-R. Ohm, and J. Boyce. Algorithm description of joint exploration test model 7 (JEM). *Document JVET-G1001*, Italy, Turino, July 2017.

[20] Zhang X. and Cohen R. Scce3 test d.2: Independent uniform prediction (IUP) mode. *Document JCTVC-R0200*, Sapporo, Japan, June-July 2014.

[21] Van der Auwera G., Wang X., and Karczewicz M. Ce6.e: Mode-dependent intra smoothing modifications. *Document JCTVC-F126*, Torino, Italy, July 2011.

[22] Fan K., Wang R., Wang Y., Li G., and Gao W. Improved intra boundary filters for hevc. In *Visual Communications and Image Processing (VCIP), 2017 IEEE*, pages 1–4, 2017.

[23] Said A., Zhao X., Karczewicz M., Chen J., and Zou F. Position dependent prediction combination for intra-frame video coding. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 534–538, 2016.

[24] Flynn D., Marpe D., Naccari M., Nguyen T., Rosewarne C., Sharman K., Sole J., and Xu J. Overview of the range extensions for the hevc standard: Tools, profiles, and performance. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):4–19, 2016.

[25] Khairat A., Nguyen T., Siekmann M., Marpe D., and Wiegand T. Adaptive cross-component prediction for 4: 4: 4 high efficiency video coding. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3734–3738, 2014.

[26] Cao X., Peng X., Lai C., Wang Y.and Lin Y., Xu J., Liu L., Zheng J., He Y., Yu H., and Wu F. CE6.b1 report on short distance intra prediction method. *Document JCTVC-E278*, Geneva, Switzerland, April 2013.

[27] Cao X., Lai Ch., Wang Y., and He Y. Short distance intra coding scheme for HEVC. In *Picture Coding Symposium (PCS)*, pages 501–504, 2012.

[28] De Luxan Hernandez D., Schwarz H., Marpe D., and Wiegand T. CE3: Line-based intra coding mode. *Document JVET-K0049*, Ljubljana, Slovenia, July 2018.

[29] De Luxan Hernandez S., Schwarz H., Marpe D., and Wiegand T. Line-based intra prediction for next-generation video coding. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 221–225, 2018.

[30] Zhao X., Chen J., Karczewicz M., Zhang L., Li X., and Chien W. Enhanced multiple transform for video coding. In *Data Compression Conference (DCC), 2016*, pages 73–82, 2016.

[31] Biatek T., Lorcy V., Castel P., and Philippe P. Low-complexity adaptive multiple transforms for post-HEVC video coding. In *Picture Coding Symposium (PCS), 2016*, pages 1–5, 2016.

[32] Gabriellini A., Flynn D., Mrak M., and Davies T. Combined intra-prediction for high-efficiency video coding. *IEEE Journal of selected topics in Signal Processing*, 5(7):1282, 2011.

[33] Dias A., Blasi S., Mrak M., and Izquierdo E. Improved combined intra prediction for higher video compression efficiency. In *Picture Coding Symposium (PCS), 2016*, pages 1–5, 2016.

[34] Mrak M, Davies T., Flynn D., and Gabriellini A. CE6: Report and evaluation of new combined intra prediction settings. *Document JCTVC-D191*, Daegu, South Korea, January 2011.

[35] Davies T. Video coding technology proposal by BBC and samsung. *Document JCTVC-A126*, Dresden, Germany, April 2012.

[36] Gao W., Jiang M., He Y., Song J., and Yu H. Differential pulse code modulation intra prediction for high efficiency video coding, November 7 2017. US Patent 9,813,733.

[37] Xu X., Müller K., and Wang L. CE8 summary report on current picture referencing. *Document JVET-K0028*, Ljubljana, Slovenia, July 2018.

[38] Barjatya A. Block matching algorithms for motion estimation. *IEEE Transactions Evolution Computation*, 8(3):225–239, 2004.

[39] Xu J., Joshi R., and Cohen R. Overview of the emerging hevc screen content coding extension. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):50–62, 2016.

[40] Xu X., Liu S., Chuang T., Huang Y., Lei S., Rapaka K., Pang C., Seregin V., Wang Y., and Karczewicz M. Intra block copy in hevc screen content coding extensions. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(4):409–419, 2016.

[41] Kwon D. and Budagavi M. Fast intra block copy (IntraBC) search for hevc screen content coding. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 9–12, 2014.

[42] Xiu X., He Y., Joshi R., Karczewicz M., Onno P., Gisquet C., and Laroche G. Palette-based coding in the screen content coding extension of the hevc standard. In *Data Compression Conference (DCC), 2015*, pages 253–262, 2015.

[43] Guo L., Pu W., Zou F., Sole J., Karczewicz M., and Joshi R. Color palette for screen content coding. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 5556–5560, 2014.

[44] Pu W., Guo X., Lai P. Onno P. and, and Xu J. Suggested software for the ahg on investigation of palette mode coding tools. *Document JCTVC-P303*, Sam Jose, CA, USA, January 2014.

[45] Onno P., Xiu X., Huang Y., and Joshi R. Suggested combined software and text for run-based palette mode. *Document JCTVC-R0348*, Sapporo, Japan, June-July 2014.

[46] Sole J., Joshi R., Nguyen N., Ji T., Karczewicz M., Clare G., F. Henry, and Duenas A. Transform coefficient coding in hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1765–1777, 2012.

[47] Tan Y., Yeo C., Tan H., and Li Z. On residual quad-tree coding in hevc. In *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, pages 1–4, 2011.

[48] Nguyen T., Helle P., Winken M., Bross B., Marpe D., Schwarz H., and Wiegand T. Transform coding techniques in hevc. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):978–989, 2013.

[49] Rice R. *Some practical universal noiseless coding techniques*. Jet Propulsion Lab., California Inst. of Tech., Pasadena, CA, United States, 1991.

[50] Li L. and Chakrabarty K. On using exponential-golomb codes and subexponential codes for system-on-a-chip test data compression. *Journal of Electronic Testing*, 20(6):667–670, 2004.

[51] Zhang H., Song L., Yang X., and Luo Z. Evaluation of beyond-hevc entropy coding methods for dct transform coefficients. In *Visual Communications and Image Processing (VCIP), 2016*, pages 1–4, 2016.

[52] Ching-Han Chiang, Jingning Han, and Yaowu Xu. A constrained adaptive scan order approach to transform coefficient entropy coding. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 1298–1302. IEEE, 2017.

[53] Min Gao, Xiaopeng Fan, Debin Zhao, and Wen Gao. An enhanced entropy coding scheme for hevc. *Signal Processing: Image Communication*, 44:108–123, 2016.

[54] Marta Mrak and Ji-Zheng Xu. Improving screen content coding in hevc by transform skipping. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1209–1213. IEEE, 2012.

[55] Gabriellini A., Naccari M., Mrak M., Flynn D., and Van Wallendael G. Adaptive transform skipping for improved coding of motion compensated residuals. *Signal Processing: Image Communication, Elsevier*, 28(3):197–208, 2013.

[56] Yih Han Tan, Chuohao Yeo, and Zhengguo Li. Residual dpcm for lossless coding in hevc. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2021–2025. IEEE, 2013.

[57] M Nacarri, A Gabriellini, M Mrak, S Blasi, and E Izquierdo. Inter prediction residual dpcm. *JCTVC-M0442, Apr*, 2013.

[58] R Joshi, J Sole, and M Karczewicz. Residual dpcm for visually lossless coding. *JCTVC-M0351, Incheon, Korea*, 2013.

[59] Nguyen T. and Marpe D. Future video coding technologies: A performance evaluation of av1, jem, vp9, and hm. In *2018 Picture Coding Symposium (PCS)*, pages 31–35, 2018.

[60] Y. Chen, Murherjee D., Han J., Grange A., Xu Y., Liu Z., Parker S., Chen C., Su H., and Joshi U. An overview of core coding tools in the av1 video codec. In *Picture Coding Symposium (PCS)*, pages 24–27, 2018.

[61] Topiwala P., Krishnan M., and Dai W. Performance comparison of vvc, av1 and hevc on 8-bit and 10-bit content. In *Applications of Digital Image Processing XLI*, volume 10752, page 107520V, 2018.

[62] Jingning Han, Ching-Han Chiang, and Yaowu Xu. A level-map approach to transform coefficient coding. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 3245–3249. IEEE, 2017.

[63] He Z., Yu L., Zheng X., Ma S., and He Y. Framework of avs2-video coding. In *2013 IEEE International Conference on Image Processing*, pages 1515–1519, 2013.

[64] Jing Wang, Xiaofeng Wang, Tianying Ji, and Dake He. Transform coefficient coding design for avs2 video coding standard. In *Visual Communications and Image Processing (VCIP), 2013*, pages 1–6. IEEE, 2013.

[65] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[66] C-Q Chen, S-N Koh, and P Sivaprakasapillai. Codebook generation for vector quantisation. *Electronics Letters*, 31(7):522–523, 1995.

[67] Bihong Huang. *Second-order prediction and residue vector quantization for video compression.* PhD thesis, Université Rennes 1, 2015.

[68] Yoseph Linde, Andres Buzo, and Robert Gray. An algorithm for vector quantizer design. *IEEE Tran. on communications*, 28(1):84–95, 1980.

[69] Mahmoud El Chamie, Ji Liu, and Tamer Başar. Design and analysis of distributed averaging with quantized communication. *IEEE Tran. on Automatic Control*, 61(12):3870–3884, 2016.

[70] Alon Orlitsky. Scalar versus vector quantization: worst case analysis. *IEEE Transactions on Information Theory*, 48(6):1393–1409, 2002.

[71] Andrea Gabriellini, Matteo Naccari, Marta Mrak, and David Flynn. Spatial transform skip in the emerging high efficiency video coding standard. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 185–188. IEEE, 2012.

[72] M. Mrak, A. Gabriellini, N. Sprljan, and D. Flynn. Transform skip mode. *Document JCTVC-F077*, Torino, Italy, July 2011.

[73] M. Karczewicz, Y. Ye, and I. Chong. Rate distortion optimized quantization. *Document VCEG-AH21*, Antalya, Turkey, January 2008.

[74] Jing He and Fuzheng Yang. High-speed implementation of rate-distortion optimized quantization for h. 264/avc. *Signal, Image and Video Processing*, 9(3):543–551, 2015.

[75] Jia Zhu, Zhenyu Liu, Dongsheng Wang, Qingrui Han, and Yang Song. Hdtv1080p hevc intra encoder with source texture based cu/pu mode pre-decision. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 367–372, 2014.

[76] Yongfei Zhang, Zhe Li, and Bo Li. Gradient-based fast decision for intra prediction in hevc. In *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1–6, 2012.

[77] Vivienne Sze and Madhukar Budagavi. High throughput cabac entropy coding in hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1778–1791, 2012.

[78] Rickard Sjoberg, Ying Chen, Akira Fujibayashi, Miska M Hannuksela, Jonatan Samuelsson, Thiow Keng Tan, Ye-Kui Wang, and Stephan Wenger. Overview of hevc high-level syntax and reference picture management. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1858–1870, 2012.

[79] Liang Zhao, Li Zhang, Siwei Ma, and Debin Zhao. Fast mode decision algorithm for intra prediction in hevc. In *Visual Communications and Image Processing (VCIP), 2011 IEEE*, pages 1–4, 2011.

[80] Wei Jiang, Hanjie Ma, and Yaowu Chen. Gradient based fast mode decision algorithm for intra prediction in hevc. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 1836–1840, 2012.

[81] Thaísa L Da Silva, Luciano V Agostini, and Luis A da Silva Cruz. Fast hevc intra prediction mode decision based on edge direction information. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1214–1218, 2012.

[82] Hao Zhang and Zhan Ma. Fast intra mode decision for high efficiency video coding (hevc). *IEEE Transactions on circuits and systems for video technology*, 24(4):660–668, 2014.

[83] Jiawen Gu, Minhao Tang, Jiangtao Wen, and Yuxing Han. Adaptive intra candidate selection with early depth decision for fast intra prediction in hevc. *IEEE Signal Processing Letters*, 25(2):159–163, 2018.

[84] Abdoli M., Sarikhani H., Ghanbari M., and Brault P. Gaussian mixture model-based contrast enhancement. *IET image processing*, 9(7):569–577, 2015.

[85] H. Yu, R.A. Cohen, K. Rapaka, and J. Xu. Common test conditions for screen content coding. In *ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 26th Meeting: Geneva, CH, 12 – 20, January 2017*, 2015.

[86] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001.

[87] Cheolhong An and Truong Q Nguyen. Adaptive lagrange multiplier selection using classification-maximization and its application to chroma qp offset decision. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(6):783–791, 2011.

[88] Miltiadis Alexios Papadopoulos, Fan Zhang, Dimitris Agrafiotis, and David Bull. An adaptive qp offset determination method for hevc. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 4220–4224, 2016.

[89] M. Schwarz and M. Coban. Description of core experiment 7 (ce 7): Quantization and coefficient coding. *Document JVET-J1027*, Sand Diego, United States, April 2018.

[90] Chih-Ming Fu, Elena Alshina, Alexander Alshin, Yu-Wen Huang, Ching-Yeh Chen, Chia-Yang Tsai, Chih-Wei Hsu, Shaw-Min Lei, Jeong-Hoon Park, and Woo-Jin Han. Sample adaptive offset in the hevc standard. *IEEE Transactions on Circuits and Systems for Video technology*, 22(12):1755–1764, 2012.

[91] Andrey Norkin, Gisle Bjontegaard, Arild Fuldseth, Matthias Narroschke, Masaru Ikeda, Kenneth Andersson, Minhua Zhou, and Geert Van der Auwera. Hevc deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1746–1754, 2012.

[92] Chia-Yang Tsai, Ching-Yeh Chen, Tomoo Yamakage, In Suk Chong, Yu-Wen Huang, Chih-Ming Fu, Takayuki Itoh, Takashi Watanabe, Takeshi Chujoh, Marta Karczewicz, et al. Adaptive loop filtering for video coding. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):934–945, 2013.

[93] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65, 2005.

[94] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[95] Tao Chen, Kai-Kuang Ma, and Li-Hui Chen. Tri-state median filter for image denoising. *IEEE Transactions on Image processing*, 8(12):1834–1838, 1999.

Index