Optimisation du Codec VVC basé sur la Réduction de Complexité et le Traitement Parallèle

Thomas Amestoy

THESE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES RENNES

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par
Thomas Amestoy

VVC Codec Optimization through Complexity Reduction and Parallel Processing

Thèse présentée et soutenue à Rennes, le 12 Mars 2021
Unité de recherche : IETR
Thèse N° : 21ISAR 06 / D21 - 06

Rapporteurs avant soutenance :

CAGNAZZO Marco	Professeur des Universités, Telecom-Paris
WIEN Mathias	Senior Researcher, RWTH Aachen University

Composition du Jury :

Président :	GUILLEMOT Christine	Directrice de Recherche, INRIA Rennes
Examinateurs :	CAGNAZZO Marco	Professeur des Universités, Telecom-Paris
	WIEN Mathias	Senior Researcher, RWTH Aachen University
	ANTONINI Marc	Directeur de Recherche, CNRS I3S Nice
	HAMIDOUCHE Wassim	Maître de conférences, INSA Rennes
Dir. de thèse :	MÉNARD Daniel	Professeur des Universités, INSA Rennes

Invité(s)

Prénom Nom	BERGERON Cyril	Ingénieur R&D , Thales SIX/GTS France

Pour Nonno, Papi, Mamie, Tonton et Tata,
avec amour.

Contents

CHAPTER 1

Introduction

During the last decade, the extensive use of on-line platforms such as video on demand (Netflix, AmazonOCS, MyCanal) or video streaming platforms (Youtube, Dailymotion, Vimeo) has lead to a significant increase in the volume of exchanged video content. The multimedia services have also diversified with the apparition of video applications that offer immersive and more natural viewing experience. These new services require both higher resolution (4K, 8K, 360°) or frame-rates (120fps) and increase neatly the volume of the video content. In parallel, devices conceived to display and distribute these emerging video services are becoming affordable products for the general public thanks to the progress in microelectronics. Televisions, computers or smart phones capable of displaying high-definition resolutions have now entered our daily lives. Therefore, it is now possible to consume video services almost anywhere and at anytime, and the video services are increasingly voluminous. A recent study published by Cisco [1] has predicted that video traffic will increase from 61% of the global IP traffic in 2016 to 82% in 2021. Viewing video content on-line also has a considerable impact on global CO2 emissions. In 2018, the storage, transmission and viewing of on-line video accounted for gas emissions comparable to a country like Spain (1% of global greenhouse gas emissions) [2]. This increasing demand for video contents brings new challenges to compression, mostly to enhance the video coding efficiency and reduce the pollution induced by video storage, transmission and processing. Higher coding efficiency enables the same quality of experience of video services, with lower storage and transmission cost.

Since the late 1980s, academic and industrial researches have been focusing on video coding, and the fundamental video coding recommendations remained unchanged since H.261 standard in 1988. All video codecs are based on the same block-based hybrid coding architecture, combining Inter and Intra predictions with transform coding. Almost every decade, a new standard is released with the objective of 50% bit-rate reduction for equal subjective quality compared with its predecessor. Every standard comes with new encoding and decoding devices that implement the standard. These devices are called codecs. In 2013, the video coding standard named H.265/HEVC, for High Efficiency Video Coding, was released and gradually adopted in many application systems. Compared to its predecessor H.264/Advanced Video Coding (AVC) [3, 4, 5] released in 2003, HEVC enables twice as much video content to be stored on a server or sent through a streaming service. Currently the state-of-the-art in video coding is the nearly completed VVC, for Versatile Video Coding, finalized in July 2020. As for every new generation standard, the primary

objective of VVC was to provide a significant improvement in coding efficiency over the previous existing standard, in this case HEVC. VVC provides close to 40% bit-rate savings over HEVC [6] and it is expected to enable the delivery of Ultra High Definition (4K) services at bit-rates that actually are used to carry HDTV.

The bit-rate savings achieved by each new generation of video codecs are generated by increasingly complex tools, on both encoder and decoder side. The computational complexity of a VVC encoder is estimated to 10 and 27 times HEVC computational complexity in inter and intra coding configuration, respectively [6]. At decoder side, the computational complexity increase of VVC standard compared to HEVC is approximately a factor 2 in both inter and intra coding configurations [6]. This complexity may become a bottleneck for the development of the VVC standard and may interfere with its deployment, especially on embedded platforms with low energy supplies and on live applications that require real-time encoding and decoding.

1.1 Challenges and Objectives

The number of devices connected to IP networks will grow up to 29.3 billion by 2023, which represents more than three devices per world inhabitant in average, according to the Cisco study [1]. The number of devices is in constant increase, and an important share of these network devices are embedded and enable video processing. However, embedded platforms only have limited energy supplies. Drastic energy consumption reduction is therefore crucial for realistic deployment of VVC codecs on embedded platforms such as mobile devices. Even for non-embedded platforms such as video data centers, energy consumption is an important issue. The substantial increase in video codec energy consumption generates a higher budget for electricity, especially for data centers processing millions of video sequences every day. On a more environmental level, reducing video codecs energy consumption on billions of devices is a manner to contain the greenhouse gas emissions generated by video codec processing.

The decrease of VVC codec processing time is also a crucial challenge. For live VVC applications, the optimization of both encoding and decoding processes is mandatory to meet real-time requirements. At encoder side, many live applications require real-time video encoding. Many applications have emerged in the latest years such as individual live streaming (Periscope, Wowza, LiveStream) or conference meeting application (Hangouts, Zoom, Skype), and some were already widespread in the past 30 years such as classical live television streaming. On decoder side, a large majority of applications require real-time decoding where the decoded video sequence is almost always instantaneously consumed to avoid the storage of the uncompressed content on memory.

This thesis aims at reducing efficiently the implementation cost, i.e. energy consumption and processing time, of VVC encoding and decoding processes. The two levers employed are the reduction of computational complexity and the parallel processing of VVC codec. Techniques optimizing computational complexity reduce the overall work of the process, and therefore directly decrease energy consumption and processing time. Parallel processing techniques leverage multi-core architectures in order to distribute optimally the work on several cores. Parallel processing techniques can be used to reduce the processing time through multi-thread speed-up. It is also possible to use them to decrease the processors clock frequency, which decreases the energy consumption by reducing the supply

[1] https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

voltage [7]. Both parallel processing and complexity reduction are conducted while minimizing the coding quality losses, in order to preserve the quality improvement of the new standard VVC.

1.2 Contributions

This document includes one evaluation study and three main contributions, each presented in a distinct Chapter. The first contribution is a preliminary study, providing important information to adequately address the issue of VVC codec optimization. Then are presented three technical contributions aiming at reducing efficiently the implementation cost of VVC encoding and decoding processes, while minimizing the coding quality losses in terms of bitrate and visual quality.

To adequately address the issue of VVC codec optimization, the first stage of this thesis is a quality assessment of coding tools. This assessment identifies the tools likely to induce coding quality losses. In parallel, a complexity profiling of VVC codec highlights the most computationally consuming parts of both encoding and decoding processes. In order to assess the performance of VVC compared to HEVC, both studies (quality assessment and complexity profiling), are conducted in comparison with HEVC standard. The VVC reference software used by researchers and industrial companies to propose and test new coding tools during VVC standardization process is called VVC Test Model (VTM). During HEVC standardization process, the reference software was the HEVC Model (HM). The performance of VTM is compared with the HM, highlighting the main improvements of VVC standard compared to its predecessor. This preliminary work on quality assessment and computational complexity profiling is currently under review for publication in IEEE international journal Transactions on Multimedia (TMM).

The three main contributions have been proposed when VVC standardization process was not yet finalized. Therefore, only few studies on VVC complexity reduction and even less on parallel processing were available in the literature. The contributions presented in this document are among the first applied to VVC standard to be published in their respective fields. They are intended to serve as a basis of comparison for future research work, and are designed to be easily implementable into future professional VVC encoders and decoders. The last contribution of this thesis is particularly representative of this effort, since it has been directly inserted in the open source decoder *openVVC*, which is intended to be distributed to the general public. The three main contributions of this thesis are briefly presented below.

Tunable VVC Block Partitioning based on Lightweight Machine Learning

At the encoder side, the block partitioning scheme selects the appropriate block size according to the local activity of the pixels, achieving significant improvement in coding quality. However, the block partitioning scheme has been identified as the most computationally expensive tool in the VVC standard in our paper [8]. In order to reduce the search space of block partitioning scheme, we propose a lightweight and tunable solution in Random Access (RA) coding configuration. The proposed solution is based on a Random Forest (RF) classifiers [9] to determine for each coding block the most probable partition modes. Classification by RF is a classical method in Machine Learning (ML) that predicts the value of a target variable, named class, from values of several input variables, named features. To minimize the encoding loss induced by misclassification, risk intervals for classifier decisions are introduced in the proposed solution. By varying the size of risk intervals, tunable

trade-off between encoding complexity reduction and coding loss is achieved. The Joint Exploration Model (JEM) is the VVC reference software pre-VTM, used during the first half of standardization process. The proposed solution has been implemented in both VTM-5.0 and JEM-7.0 software models and offers encoding complexity reductions ranging from 30% to 70% in average, for only slight encoding quality loss in RA coding configuration. The overhead induced in the encoding process by RF classifiers is also very light, which is a key point to use this solution in a real-time or embedded framework. This work has lead to the presentation of a poster session in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) in 2019 [10], and has been published in international journal IEEE Transactions on Image Processing (TIP) in 2019 [11].

Efficient Parallel Encoding through Dynamic Tiles and Rectangular-Slices

As mentioned in previous section, in order to achieve real-time and low latency encoding, it is mandatory to rely on parallel processing to distribute optimally the encoding complexity on several actors (cores). The second contribution of this thesis is an effective high-level parallelism solution for VVC encoding process in RA configuration. The VVC standard includes tile grid partitioning that allows to process simultaneously rectangular regions of a frame with independent threads. The tile grid may be further partitioned into a horizontal sub-grid of Rectangular Slices (RSs), increasing the partitioning flexibility. A dynamic Tile and Rectangular Slice (TRS) partitioning solution, i.e. evolving across video sequence time-line, is proposed and benefits from this increased flexibility at the expense of coding loss. The TRS partitioning is carried-out at the frame level, taking into account both spatial texture of the content and encoding times of previously encoded frames. The proposed solution searches the partitioning configuration that minimizes the trade-off between multi-thread encoding time and encoding quality loss. Experiments prove that the proposed solution, compared to uniform TRS partitioning, significantly decreases multi-thread encoding time, with slightly better encoding quality. This work has been presented in the special session *Complexity Reduction and Real Time Implementations of the Versatile Video Coding Standard* of IEEE International Conference on Image Processing (ICIP) in 2020 [12].

Parallel and Real-Time VVC Decoder in AI configuration

The last contribution of this thesis focuses on the decoder side of the new standard. This contribution is implemented in the open source and real-time VVC decoder *openVVC* project, under the All Intra (AI) configuration. The contribution includes the implementation starting from scratch of in-loop filters Adaptive Loop Filter (ALF) and Sample Adaptive Offset (SAO), with special attention to the minimization of filter buffers memory usage. It also includes the consolidation of tile parallelism in order to enable both dynamic and uniform tile partitioning. With in-loop filters and dynamic tile parallelism operational, we reimplemented the dynamic tile partitioning solution proposed in our previous contribution (See A.2), enabling the generation of bitstreams compatible with *openVVC* decoder. The experimental results show that the decoding frame-rates obtained with dynamic tiles are equivalent to the frame-rates obtained with uniform tiles, with improved coding quality. The general performance of the *openVVC* version is also presented. By combining data-level with tile-level or frame-level parallelism, *openVVC* achieves real-time decoding for High Definition (HD) and Ultra High Definition (UHD) content. The main asset of the proposed decoder lies in its very low memory usage, since the sequential decoding of HD and UHD content only requires 20MB and 70MB of memory, respectively.

1.3 Outline

This section briefly summarizes the content of each chapter presented in this document. Chapter 2 presents the fundamental concepts of video compression, from video signal format, through Rate Distorsion Optimization (RDO) process and history of video coding standards, to the recently released video coding standard Versatile Video Coding.

In Chapter 3, a subjective and objective quality assessment of VVC is conducted in order to evaluate the coding quality of this new emerging standard in comparison with HEVC standard. This chapter also provides an analysis of times and complexity repartition for both encoding and decoding processes. This chapter identifies the tools likely to induce quality losses and evaluate the complexity reduction opportunities.

In order to reduce the complexity of the encoding process, Chapter 4 focuses on the partitioning scheme and reduces the number of tested partition configurations in RA coding configuration. All the steps of the proposed solution are detailed, from the creation of the ML classifiers, to the achievement of tunable trade-off between coding quality and execution time.

In Chapter 5 a technique to parallelize efficiently the VVC encoding process in RA coding configuration is presented. The TRS partitioning, briefly presented in previous section, is adjusted at the frame level, taking into account both spatial content and encoding times of previously encoded frames. Experiments prove that the proposed solution decreases significantly multi-thread encoding time, with slightly better encoding quality, compared to other straightforward TRS partitioning techniques.

Chapter 6 presents a real-time VVC decoder based on the open source *openVVC* project, for HD and UHD contents in AI configuration. It presents the contributions of this thesis in the software, as well as the experimental results obtained with these contributions. By combining data-level with threading parallelism, *openVVC* achieves real-time decoding for HD content, with very low memory usage.

Chapter 7 concludes this document and gives several research directions to extend this work for future research.

CHAPTER 2

Video Coding Background

Video coding is a branch of data compression that reduces the amount of data required to represent a video signal, while minimizing the quality degradation compared to the original video signal. This chapter presents the fundamental concepts of video coding required to understand this document, from video signal format to the recently released video coding standard Versatile Video Coding. Section 2.1 describes the properties that characterize a digital video signal. The concept of rate is introduced in Section 2.2, along with the most common metrics used to evaluate the quality of a decoded video signal. Section 2.3 investigates the Rate Distorsion Optimization (RDO) process used at the encoder side to estimate the optimal encoding parameters. In Section 2.4, the history and evolution of video coding standards during the last three decades is presented. The description of recently standardized Versatile Video Coding (VVC) decoder is provided in Section 2.5 as a concrete example of classical hybrid video coding standard. Finally, Section 2.6 introduces the software video codecs used in the three contributions of this thesis.

2.1 Video Signal Format

A video signal is defined as a sequence of frames, also called pictures or images. The characteristics of a frame have been standardized to ensure correct processing and displaying of the video signal. The following section details the characteristics of the video signal.

2.1.1 Spatial Resolution

A frame is represented by a 2 dimensional matrix of pixels. The spatial resolution is defined by the number of pixels in a row and in a column of the frame. These two characteristics are noted W and H for the frame width and height, respectively. Figure 2.1 gives a visual representation of the frame as a 2 dimensional matrix of pixels $p(i, j)$, with $p(i, j)$ the pixel localized in row number i and in column number j of the frame.

For gray-scale video signal, a pixel is composed of a single component called luminance. The luminance represents the visual sensitivity of luminosity of a surface. In the rest of this document, pixel components are also called channels. To represent colors in a color video signal, three channels are required. Among the most widespread color representations, the RGB color space associate each channel to a primary color: red, green and blue. For coding purposes, the YCbCr representation of a pixel is more common in digital imaging, with

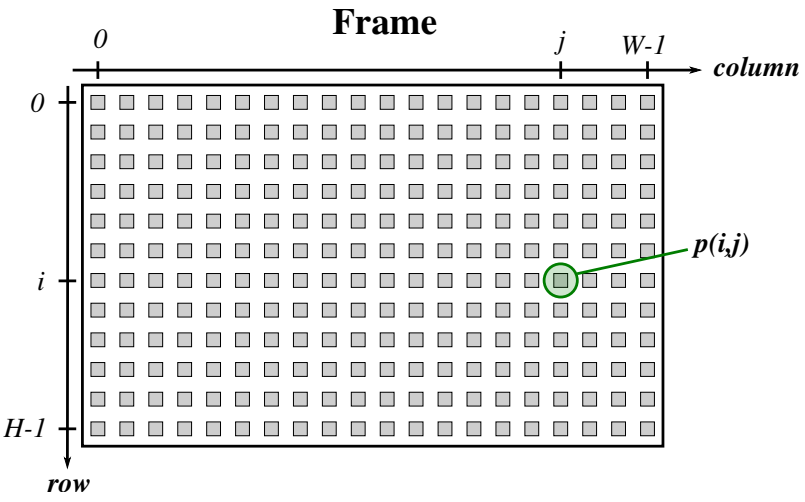


Figure 2.1 – *Video signal with spatial resolution of $W \times H$ pixels.*

Y the luminance, Cb and Cr the blue and red chrominance channels, respectively. The chrominance represents the color intensity of a surface, and is also called chroma.

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{U} \\ \mathrm{V} \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ -0,147\,13 & -0,288\,86 & 0,436 \\ 0,615 & -0,514\,98 & -0,100\,01 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} \tag{2.1}$$

Equation 2.1 can be used to derive Y, U and V from the components R, G and B. Historically, the notation YCbCr is used for digital color video signals and the notation YUV refers to analogue signals (TV sets / cathode ray tube monitors), but both refer to the same color representation. In this document, YCbCr and YUV are both used equivalently.

2.1.2 Temporal Resolution

The temporal resolution of a video signal is characterized by its frame-rate. The frame-rate, usually expressed in Frames Per Second (fps), corresponds to the frequency at which the frames change in the video. In Figure 2.2, a video signal with a temporal resolution of f fps is presented. In this case, the displayed frame is refreshed every $\frac{1}{f}$ seconds.

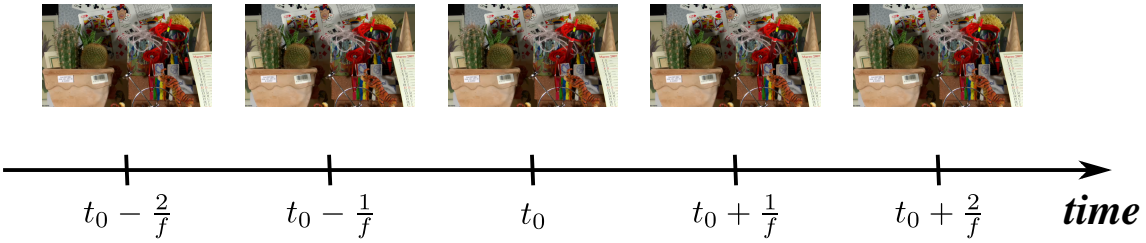


Figure 2.2 – *Video signal with temporal resolution of f fps.*

2.1.3 Bitdepth and Color Sampling

A pixel is composed of three channels, each carrying a part of the pixel visual information. The channel content is generally represented either by an integer or a floating point. The bitdepth of the video signal is the fixed number of bits required to store each channel

content. The most common bitdepth values range from 8-bit to 16-bit formats, resulting in a total of 24 to 48 bits per pixel for a color video.

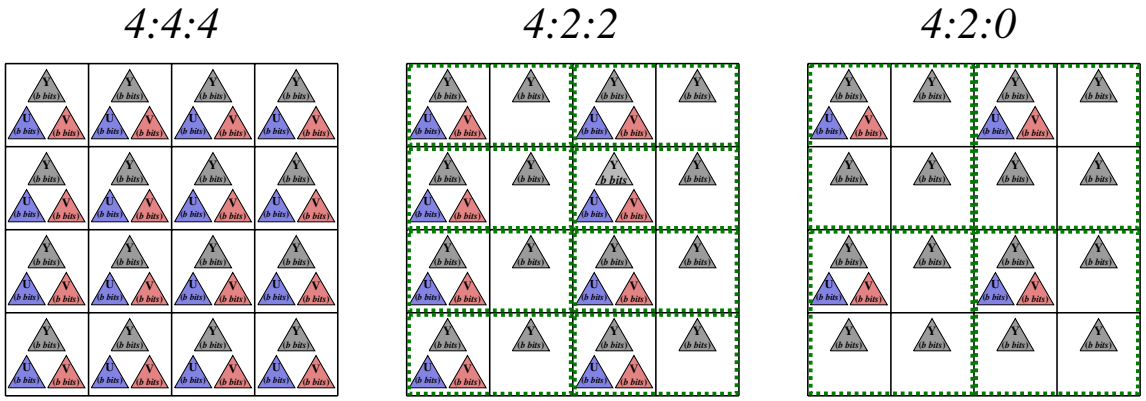


Figure 2.3 – *Video signal with temporal resolution of f fps.*

In order to reduce the total bits per pixel without reducing the bitdepth of the video signal, color sampling is frequently applied. A very common color sampling applied in YUV representation is the chrominance down-sampling, i.e. reducing the chrominance resolution while leaving the luminance resolution unchanged. Indeed, the Human Visual System (HVS) is more sensitive to luminance channel compared to the chrominance channels in YUV representation. It is therefore possible to discard a part of the chrominance information without significantly reducing the perceived quality of the video. This is the main interest of YUV representation and the reason for its extensive use in the video coding field.

In the following, the bitdepth of a video signal is noted b and the total number of bits per pixel is noted B. Figure 2.3 shows the 3 typical color sampling formats used in video coding: 4:4:4, 4:2:2 and 4:2:0. The 4:4:4 color sampling format is the original YUV representation with no chrominance down-sampling. In this case $B = 3b$. With the 4:2:2 color sampling format, a chroma down-sampling by 2 in the horizontal direction is applied. As shown in Figure 2.3, in this case $2B = 4b$, which implies $B = 2b$. With the 4:2:0 color sampling format, a chroma down-sampling by 2 in both horizontal and vertical direction is applied. As shown in Figure 2.3, $4B = 6b$, which implies $B = \frac{3}{2}b$. The chrominance down-sampling is therefore effective to reduce the total bits per pixel.

2.2 Quality Evaluation of a Decoded Video

In this section we first introduce the notions of video quality and rate, and then we present the most common metrics used to evaluate the quality of a decoded video signal.

2.2.1 Video Quality and Rate

Figure 2.4 shows the coding process of a video signal. A raw video is acquired by a recording device, and converted by the encoder into a bitstream composed of binary symbols (bits). The size of the bitstream for a given time interval is called rate, or bit-rate. The rate is often expressed in bits per seconds, i.e. the amount of transmitted numerical data per second. The bitstream is further processed by the decoder, and the decoded video signal is displayed on the user screen. The quality of the decoded video is assessed either by a distortion metric, a rate-distortion metric or by a subjective quality metric. The distortion measures

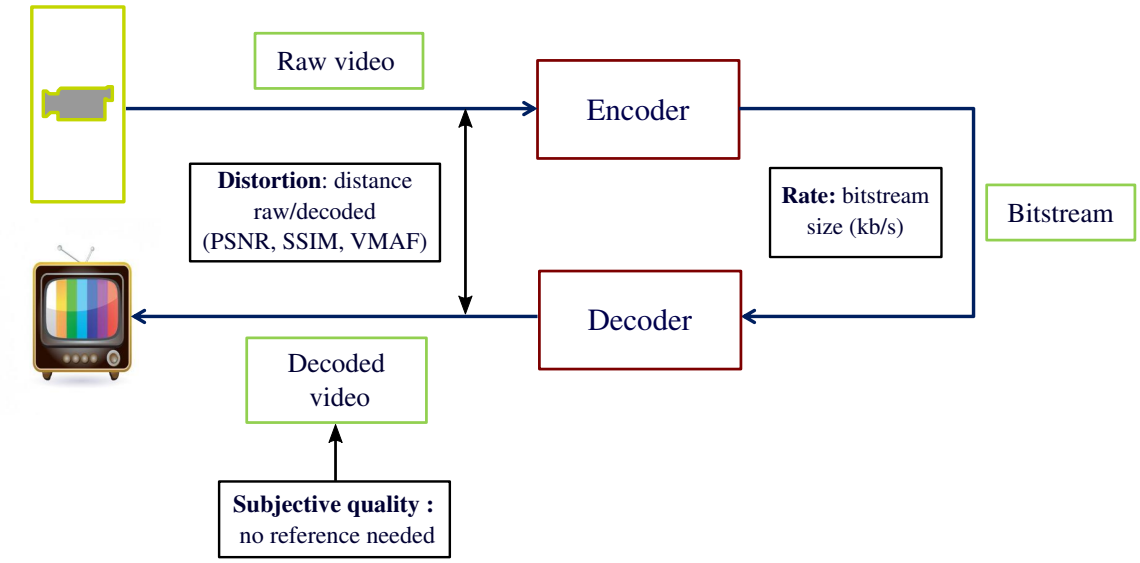


Figure 2.4 – *Video rate and quality in the video coding cycle.*

the distance between the raw and the decoded video signal, and therefore requires the raw video signal as reference to be computed. The most commonly used objective metrics are presented in the following section: Peak Signal to Noise Ratio (PSNR), Structural SIMilarity (SSIM) and Video Multi-Method Assessment Fusion (VMAF). The Bjøntegaard metrics [13] compare two video coding solutions taking into account both rate and distortion performance. The subjective metrics evaluate the quality of the decoded video signal relying on scores from the HVS.

2.2.2 Distortion Metrics

2.2.2.1 PSNR Metric

The PSNR metric is a purely "signal" measure. The PSNR is a pixel-to-pixel treatment regardless of the 2D aspect of the image or video. The PSNR is based on the Mean Squared Error (MSE), to evaluate the error between a degraded image and its reference version. It is defined by:

$$MSE = \frac{1}{W\,H} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} (Im_r(i,j) - Im_e(i,j))^2, \tag{2.2}$$

where W and H are the dimensions of the image, Im_r is the reference image and Im_e is the image to be evaluated.

The PSNR of a signal of amplitude A is defined by Equation 2.3.

$$PSNR = 10 \log_{10} \left(\frac{A^2}{MSE} \right), \tag{2.3}$$

with $A = 2^\gamma - 1$ and γ is the video bitdepth.

For video color sequences, which ordinarily consist of three channels (y, u, v), a weighted PSNR ($wPSNR$) version can be used. The weighted PSNR is computed by weighting three channels Y, U and V as defined bellow.

$$wPSNR = \frac{6\,PSNR_Y + PSNR_U + PSNR_V}{8}. \tag{2.4}$$

PSNR is the most used objective metric in video domain. It is easy and fast to implement. However, it simply computes the pixel-wise distortion without considering the relationships among the pixels.

2.2.2.2 SSIM Metric

To substantially remedy this disadvantage, Wang *et al.* [14] proposed a metric named SSIM based on the similarities between the structures in the image. Thus, it makes it possible to extract three different criteria, for each "window" of the image, judged important for human observers, namely luminance, contrast and structure. The metric is defined as follows for two windows belonging to two images Im_r and Im_e:

$$SSIM(Im_r, Im_e) = \frac{(2\,\mu_{Im_r}\mu_{Im_e} + C_1)(2\,\sigma_{ImIm_e} + C_2)}{(\mu_{Im_r}^2 + \mu_{Im_e}^2 + C_1)(2\,\sigma_{Im_r}^2 + \sigma_{Im_e}^2 + C_2)}, \tag{2.5}$$

where μ_{Im_r} is the local luminance mean in image Im_r, σ_{Im_r} is the local standard deviation, Im_r, $\sigma_{Im_rIm_e}$ is the covariance between Im_r and Im_e. C_1 and C_2 are two constants that depend on the dynamics of the image.

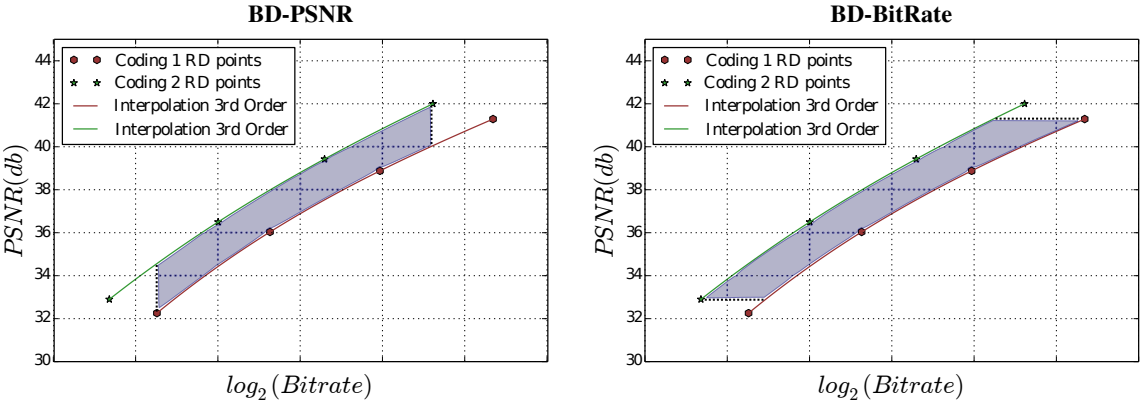
The values of this metric are then obtained for all calculation windows. They are between 0 and 1: a value close to 1 indicates that both images (videos) have a maximum fidelity while values close to 0 indicate a too degraded image (video). Moreover, SSIM has been recently exploited [15] in the High Efficiency Video Coding (HEVC) rate-distortion optimization stage to enhance the perceived video quality under a rate control process.

2.2.2.3 VMAF Metric

VMAF is a perceptual video quality assessment algorithm developed by Netflix with collaboration of Laboratory of Image and Video Engineering [16]. VMAF is a full-reference, perceptual video quality metric focused on quality degradation due to compression and rescaling. It was specifically formulated to approximate human perception of video quality and then to correlate strongly with subjective Mean Opinion Scores (MOSs). Using machine learning techniques, a large sample of MOS scores were used as ground truth to train a quality estimation model. VMAF estimates the perceived quality score by computing scores from multiple quality assessment algorithms and fusing them using a support vector machine. The features taken as input by the support vector machine include:

- Detail loss metric [17]: details loss and visual defects that distract viewer attention.
- Visual information fidelity: information loss at four different spatial scales.
- Mean co-located pixel difference: luminance difference between consecutive frames.
- Anti-noise signal-to-noise ratio.

VMAF has been shown to outperform other image and video quality metrics, such as SSIM and PSNR on several datasets in terms of prediction accuracy, when compared to subjective ratings.



(a) *Bjøntegaard Delta PSNR* **(b)** *Bjøntegaard Delta BitRate*

Figure 2.5 – *Computation of Bjøntegaard metrics between two R-D curves.*

2.2.3 Rate-Distortion Metrics: Bjøntegaard Metrics

The Bjøntegaard metrics [13] compare two video coding solutions taking into account both rate and distortion performance. This characteristic makes the Bjøntegaard metrics the most widely used metrics for the comparison of two video coding solutions. As shown in Figure 2.5, the Bjøntegaard metrics measure the average difference between two Rate Distorsion (RD) curves, i.e. rate/distortion points obtained by coding a raw video with various encoding rates. The green and red RD curves are interpolated through at least four points, using a third-order logarithmic polynomial fitting.

As illustrated in Figure 2.5b, the Bjøntegaard Delta BitRate (BD-BR) measures the average bit-rate difference (in percent) of the two RD curves at the same PSNR quality. This average bit-rate difference is measured on the overlapping PSNR range of the two RD curves. Likewise, the Bjøntegaard Delta PSNR (BD-PSNR) measures the average PSNR difference in decibels (dB) for the two coding solutions at the same bit-rate. In this document, the quality of all the coding solutions proposed are compared to the anchor encoder in term of BD-BR metric.

2.2.4 Subjective Quality Metrics

Unlike signal-based approaches for objective quality measurements, subjective quality assessment is the process of employing human viewers for grading video quality based on individual perception [5]. The global environment, viewing conditions and the implementation process for subjective quality assessments are specified in various International Telecommunication Union (ITU)recommendations. Two specific methodologies are widely used in the video quality assessments: ITU-T Rec. P.910 [18] for multimedia applications and ITU-R Rec. BT.500 [19], for television pictures.

Subjective tests rely directly on the HVS, but they are costly in time and money. For this reason, metrics such as VMAF or SSIM seek objective criteria correlated with HVS.

2.3 Rate-Distortion Optimization

During the encoding process, the encoder estimates the coding parameters Π that minimizes the distortion under bit-rate constraint. This optimization problem is called RDO, and expressed is in Equation 2.6. E_{Π} represents the set of allowed coding parameters and Π^*

the coding parameters that minimize the distortion $D(\Pi)$. $R(\Pi)$ is the bit-rate obtained with the coding parameters Π and R_{max} the bit-rate constraint.

$$\begin{aligned} \Pi^* &= \underset{\Pi \in E_\Pi}{argmin}(D(\Pi)), \\ &\textbf{under constraint, } R(\Pi) < R_{max}. \end{aligned} \tag{2.6}$$

This constrained problem has been modeled by Everett [20] as an unconstrained problem, introducing a Lagrangian multiplier λ. This unconstrained problem is defined by Equation 2.7, with $J(\Pi, \lambda)$ the RD-cost of coding parameters Π and Lagrangian multiplier λ.

$$\begin{aligned} \Pi^*(\lambda) &= \underset{\Pi \in E_\Pi}{argmin}(J(\Pi, \lambda)), \\ &\textbf{with } J(\Pi, \lambda) = D(\Pi) + \lambda R(\Pi). \end{aligned} \tag{2.7}$$

In Equation 2.7, the value of λ controls the trade off between distortion $D(\Pi)$ and bit-rate $R(\Pi)$, and affects the choice of the optimal coding the parameters $\Pi^*(\lambda)$. Since the λ values used in the video coding field have been experimentally fixed for H.263 standard (see Section 2.4 by Wiegand *et al.* [21]), λ is no longer considered as a variable. It is therefore possible to change the notation for the coding parameters that minimize RD-cost from $\Pi^*(\lambda)$ to Π^* in Equation 2.7.

Since no mathematical model exists to define the relation between coding parameters Π and corresponding RD-cost J, a common RDO strategy is to exhaustively test the coding parameters within set of allowed coding parameters E_Π. The optimal set of coding parameters Π^* found after RDO is further used to code the video signal. The set E_Π bounds the RDO search space and is usually determined by the encoding configuration, the encoding limitations or the specificities of the encoder algorithms. In most cases, larger search space leads to better coding efficiency, at the cost of increased computational complexity.

2.4 Video Coding Standards

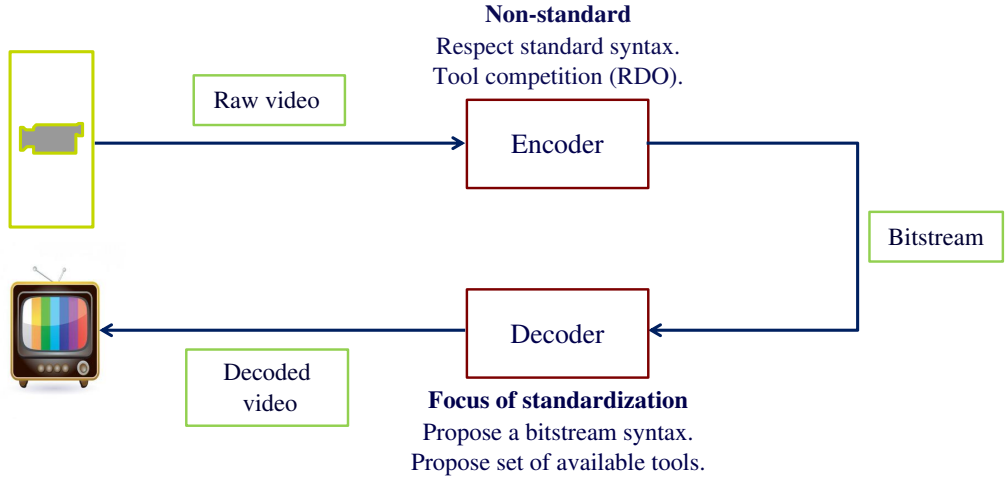


Figure 2.6 – *Roles of the encoder and decoder in the standardization process.*

A video coding standard specifies the bitstream syntax and the decoding process that allows to convert the bitstream in a displayable video sequence.

Each standard introduces specific coding tools, selected based on their maximum coding efficiency. On the other side, the encoding process generates a bitstream compliant with the standard syntax, using the tools specified in the standard. The roles of the encoder and decoder in the standardization process is depicted in Figure 2.6. As mentioned in Section 2.3, at the encoder side the RDO exploits the coding tools available in the standard in order to optimize coding efficiency. Thus, two distinct competitions exist in the video coding field. The competition among standards, whose aim is to provide the best set of coding tools. And the competition among encoders compliant with the syntax of a standard, whose aim is to optimize the use of the coding tools through proprietary RDO algorithms. In this section we give an overview of the competition among existing video coding standards. In the last three decades, the contributors for the most used video coding standards have been the two standard organizations International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) and the ITU-T. Currently, the major alternative is the industry-oriented Alliance for Open Media (AOM).

2.4.1 ISO/IEC and the ITU-T

In Figure 2.7, the history standards is presented chronologically. The left part corresponds to the standards proposed by the ISO/IEC and the ITU-T, which are the most widespread in the last thirty years. In 1984, the first video coding standard known as H.120 [22] is released by the International Telegraph and Telephone Consultative Committee (CCITT). The CCITT and the ITU have further been unified to create the ITU-T group. Inside ITU-T, the group in charge of the video coding standardization, named Video Coding Experts Group (VCEG), proposed in 1988 the first widespread video encoder under the form of recommendation H.261 [23]. The classical hybrid video coding scheme, combining both spatial/temporal prediction and transform coding, has been used for the first time in H.261 standard. Since current standards still rely on this hybrid video coding scheme, it is described in detail in Section 2.5.

Also in 1988, the ISO and the IEC collaborated to create their own standardization group called Moving Picture Experts Group (MPEG), in parallel with VCEG activities. As the name suggests, the initial aim of MPEG was proposing standards for the representation of audio and moving pictures. The first standard developed by MPEG called MPEG-1 [24] was released in 1993. After these first effort from both VCEG and MPEG, the two standardization groups joined their forces and proposed the most widely used standards of the last decades with the H.262/MPEG-2 (1994) [25], H.264/MPEG-4 Advanced Video Coding (AVC) (2003) [26] and H.265/HEVC (2013) [27] standards. The latest video coding standard named H.266/VVC has been developed by the Joint Video Exploration Team (JVET), which is also the result of the collaboration between VCEG and MPEG.

2.4.2 Principal Alternatives

In the right part of Figure 2.7, the standards proposed as alternatives to the ISO/IEC and the ITU-T are displayed. Between 1995 and 1997, the company On2 Technology developed the TrueMotion S, TrueMotion RT and TrueMotion 2 video codecs for 3D-rendering, i.e. process of converting 3D models into 2D images on a computer. The same company proposed VP3 in 2000, focusing this time on natural scenes images. In 2001, the Xiph organization proposed the first open-source Theora [28] standard in 2001, based on the previously released VP3 codec. Another alternative to H.264/AVC has been proposed by

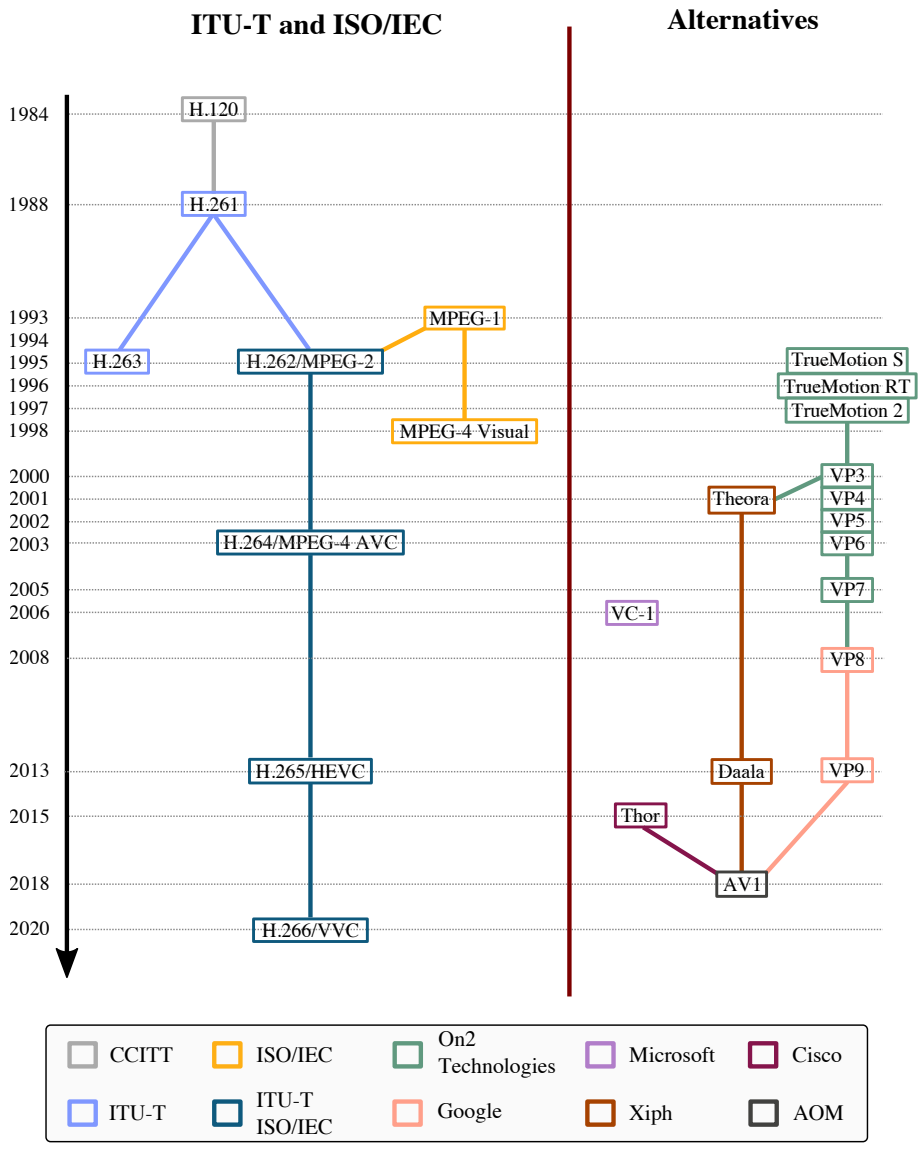


Figure 2.7 – *History of video coding standards.*

Microsoft in 2006 under the name of VC-1. This latler is used in both HD-DVD and Blu-ray discs.

In early 2010, several alternatives to H.265/HEVC have been released. The Xiph organization released Daala [29] in 2013 and Cisco Systems developed Thor [30]. Both video codec being royalty-free and open-source alternatives to VP9 developed by Google. In 2018, Amazon, Apple, ARM, Cisco, Facebook, Google, IBM, Intel Corporation, Microsoft, Mozilla, Netflix and Nvidia forming the industry consortium AOM presented the AV1 standard [31]. This emerging standard competes with the H.265/HEVC in term of efficiency of the available coding tools, and does not induce limitations regarding financial patent cost or legal usage terms.

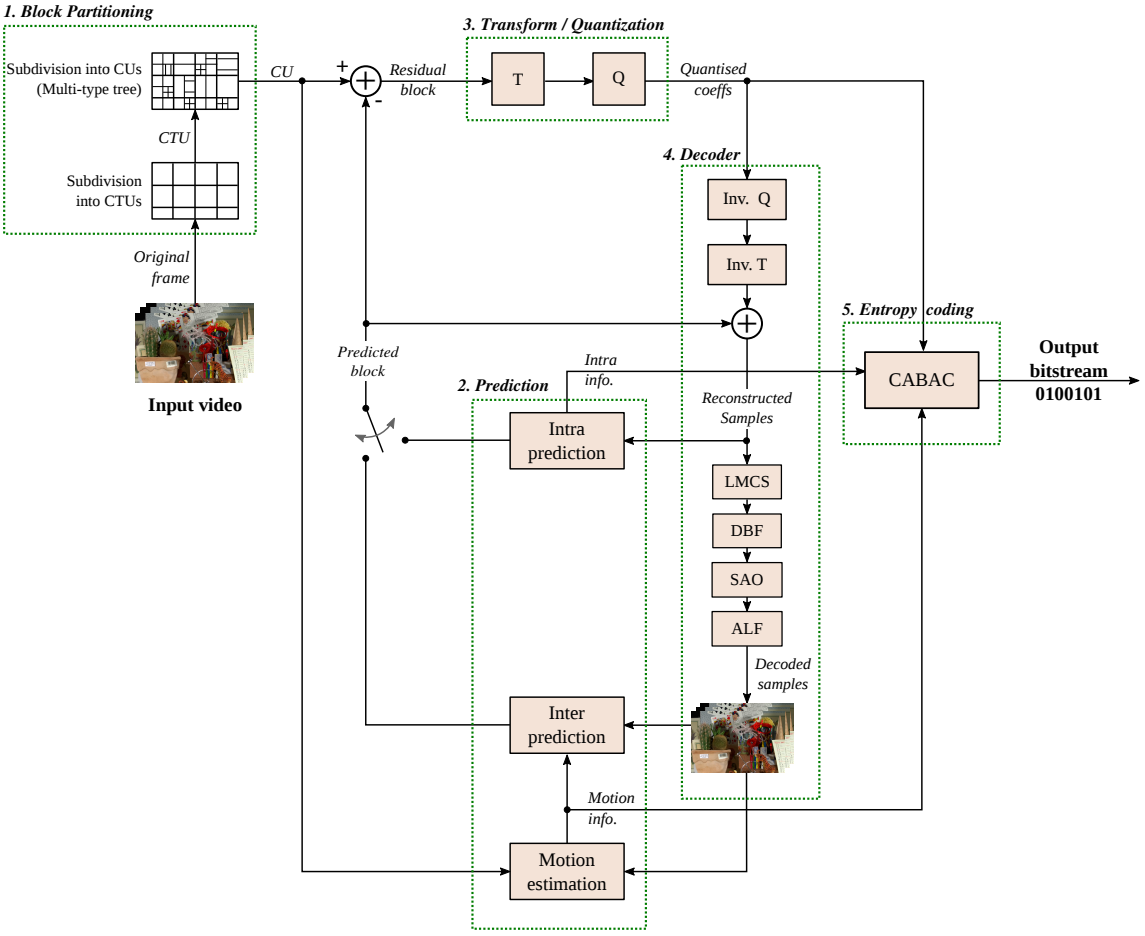


Figure 2.8 – *VVC encoder block diagram.*

2.5 VVC: Classical Hybrid Video Coding Standard

In this section, the classical hybrid coding scheme used in many video coding standards is presented, taking as example the new state of the art video coding standard VVC. Since VVC is the focus of this thesis, a description of the main coding tools available in the standard is further detailed.

2.5.1 Overview of VVC Encoding Process

The encoding process of the vast majority of encoders, including VVC, is divided into 5 main steps. Figure 2.8 represents a classical VVC encoder block diagram. The 5 main steps described below are highlighted in green and identified with their corresponding numerical numbers.

1. Block Partitioning: the input frame is divided into rectangular group of samples, called Coding Units (CUs), on which the other main steps are applied.

2. Prediction: the samples of the CUs are predicted using the reference samples stored in memory. Intra prediction exploits spatial redundancy within the same frame, whereas inter prediction exploits temporal redundancy between frames. The predicted block is an approximation of the original CU. It is then subtracted to the original block, resulting in a residual block.

3. Transform and Quantization: the residual block is transformed into the frequency domain. The transform enables to decorrelate the residual block and pack its energy in a few coefficients. Then lossy coding methods are applied to the transform coefficients in order to remove information less relevant for the HVS and provide quantised coefficients.

4. Internal Decoder: inverse operations are applied to generate the reference samples and store them in the memory. The reconstructed samples correspond to the predicted samples, summed with inverse quantization and inverse transform of quantised coefficients. The reconstructed samples are used by intra prediction. The in-loop filters (further described in Section 2.5.7) are then applied on the reconstructed samples to generate decoded samples. The decoded samples are used as reference for inter prediction.

5. Entropy Coding: all the syntax elements (including chosen coding parameters and quantised coefficients) are further encoded by a lossless entropy coder to reduce statistical redundancies and generate the output bitstream. In VVC, the Context Adaptive Binary Arithmetic Coding (CABAC) [32] is used as entropy coder which is further described in Section 2.5.8.

2.5.2 Block Partitioning Scheme in VVC

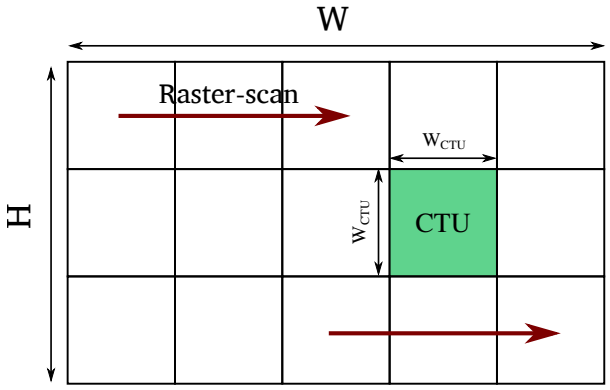


Figure 2.9 – *VVC encoder block diagram.*

The first step of the block partitioning scheme splits the frame into large blocks of equal sizes, named Coding Tree Unit (CTU), as shown in Figure 2.9. The CTU width is noted W_{ctu}, the maximum CTU size is 128×128 pixels in VVC contrary to HEVC standard where the maximum enabled size for a CTU is 64×64.

The CTUs are processed in raster scan order from top left to bottom right, as indicated by the arrows in Figure 2.9. In order to adapt block size for prediction to the local activity of the pixels, each CTU is then recursively split into smaller rectangular CUs, following a Multi-Type Tree (MTT) partition scheme. The MTT partitioning scheme is an extension of the Quad Tree (QT) partitioning scheme adopted in HEVC. We first describe the QT partitioning scheme in HEVC and then present the novelties brought by MTT partition scheme in VVC.

The authorized recursive partition modes for QT partitioning in HEVC are shown in Figure 2.10: the CU is either not partitioned ($4N \times 4N$), or partitioned into 4 equal squares ($2N \times 2N$). A typical QT partitioning of a CTU in HEVC is illustrated by Figure 2.11a.

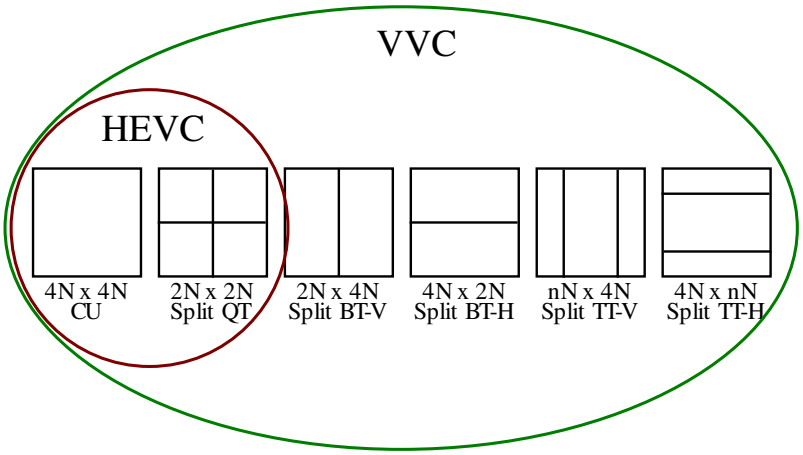
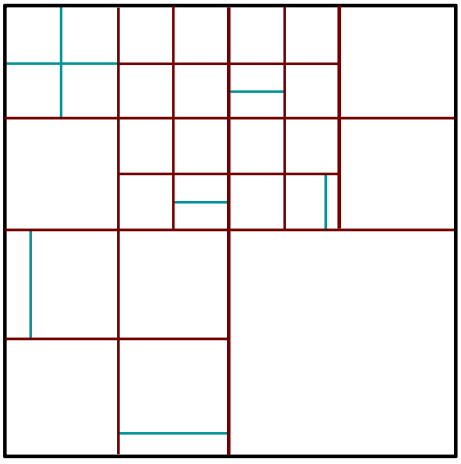
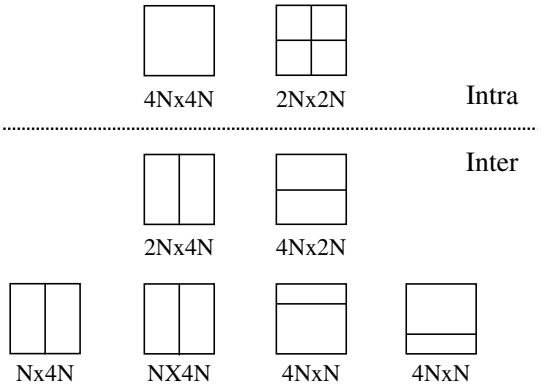


Figure 2.10 – *Available partition modes in HEVC and in VVC for a $4N \times 4N$ square CU.*



(a) *QT partition of a CTU in HEVC.*



(b) *Allowed PU modes in HEVC.*

Figure 2.11 – *QT partition scheme of a CTU in HEVC. QT partition modes in red and further PU modes in blue.*

The red lines correspond to the recursive QT partitioning, while the blue lines correspond to the subdivision of the CUs into Prediction Units (PUs).

In order to perform prediction in HEVC, the square CUs can be divided into PUs of smaller size following one of the eight PU modes illustrated in Figure 2.11b. All the PU modes are available for inter prediction, while only the $4N \times 4N$ and $2N \times 2N$ PU modes are available for intra prediction.

Moreover, after performing prediction, residual blocks can be further split recursively with a second QT partition scheme into Transform Units (TUs), the block on which transform is performed. The block partitioning scheme in HEVC suffers from the following limitations:

- CUs can only be square, with no flexible shape to cover all block characteristics.
- Luma and Chroma components of the encoded sequence have the same QT splitting which is rarely optimal for chroma.
- Residual can only be split into TUs with square shapes, reducing the potential impact of transformation.

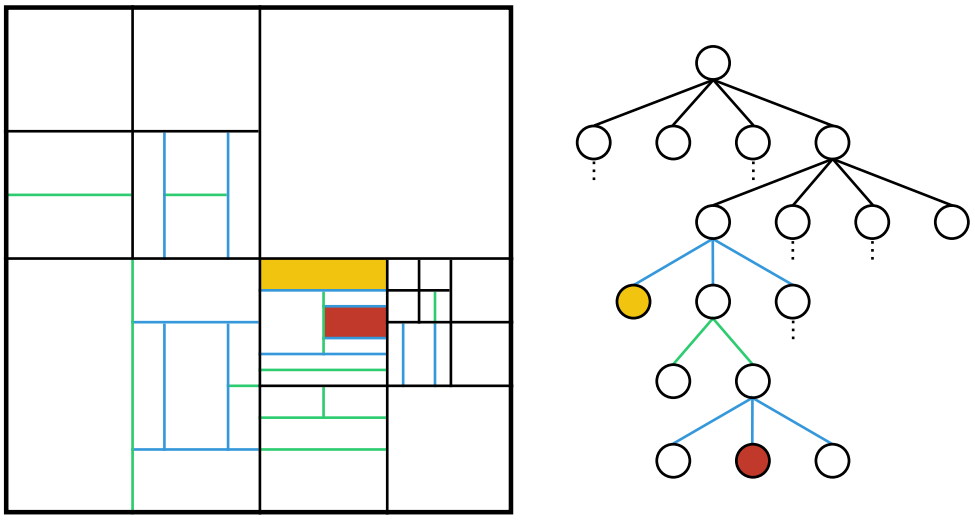


Figure 2.12 – *MTT partitioning scheme of a CTU in VVC, with a part of its corresponding tree representation.*

To overcome these limitations, VVC integrates a nested recursive MTT partitioning, i.e. Binary Tree (BT) and Ternary Tree (TT) partition modes, in addition to the recursive QT partitioning. Figure 2.10 illustrates all available split in VVC for a $4N \times 4N$ CU. The BT partition modes consists of symmetric horizontal splitting (BT-H) and symmetric vertical splitting (BT-V). The TT partition modes enables horizontal ternary splitting (TT-H) and vertical ternary splitting (TT-V). In this case, the CU is split in three blocks with the middle block being half the size of the CU. Once BT or TT split is performed on a CU, QT split is not allowed on its sub-CUs. Figure 2.12 presents on the left an example of a CTU split with MTT partition modes, and on the right a part of the corresponding tree representation. The black, green and blue branches correspond to QT, BT and TT partition modes, respectively. The leafs in the tree filled in yellow and red correspond to the yellow and red CUs in the left part of Figure 2.12.

In order to overcome the restriction of square residual TUs in HEVC, rectangular transforms have been adopted in VVC. The transform is thus directly applied on the CUs independently of the CU shape, without any further splitting of residual blocks [33]. Also, the flexibility brought by MTT partition modes covers the PU modes in Figure 2.11b. In VVC, the CUs are directly used to perform prediction and no subdivision in PUs is needed for prediction. Consequently, the TU and PU partitioning is discarded in VVC, and both prediction and transform are performed directly on the CU.

Moreover, different block partitioning trees can be applied on Luma and Chroma channels for intra predicted frames in VVC. For inter predicted frames, the same block partitioning is used for both of Luma and Chroma and a triangle partition for merge mode has been adopted [34].

2.5.3 Intra Prediction

The Intra prediction exploits spatial redundancy within the same frame, predicting the samples of a CU using the neighboring samples of CUs already encoded. The neighboring CUs already encoded include the left and above CUs. Figure 2.13 shows the reference samples used for intra prediction of a CU of size $N \times M$. The left neighboring CU and the upper neighboring CU provide 2M (in blue) and 2N (in red) reference samples for intra

prediction, respectively. A total of from $2M+2N+1$ reference samples are used, including 1 sample of the upper-left block (in pink).

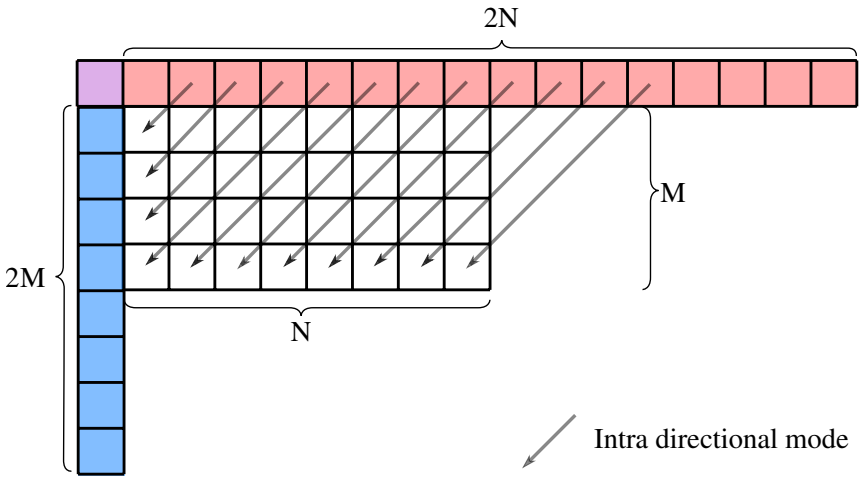


Figure 2.13 – *Reference samples used for intra prediction of a $N \times M$ CU*

In VVC, 65 intra directional prediction modes are used, instead of 33 in HEVC, in order to capture the arbitrary edge directions presented in a natural video. Each intra directional prediction mode corresponds to the propagation with a given angle of the neighboring reference samples. In Figure 2.13 for instance, the selected intra directional mode corresponds to the diagonal down-left mode. To this 65 intra modes, 2 additional modes already used in previous standards are available: Planar-mode and DC-mode. The DC-mode computes the average value of reference pixels to predict current CU samples. The Planar-mode is designed to preserve continuities across CU boundaries using bilinear interpolation. The angular intra prediction directions in HEVC range from 45 degrees to -135 degrees in clockwise direction. They were designed for square CUs. For rectangular blocks, Wide Angular Intra Prediction (WAIP) has been added in VVC to enable intra prediction directions beyond the range of conventional intra prediction directions.

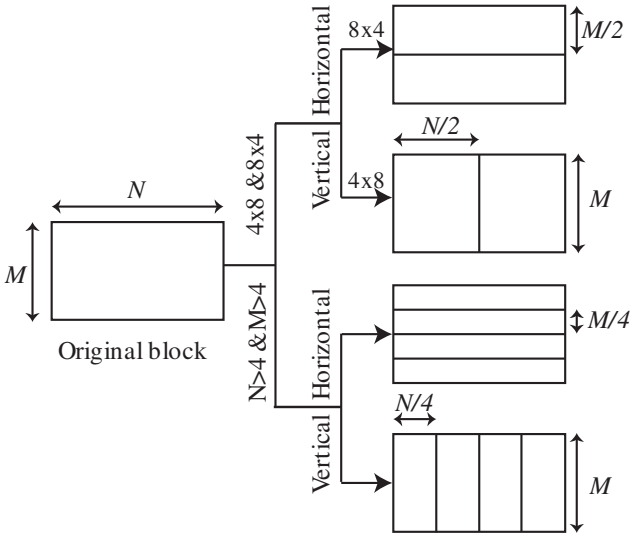


Figure 2.14 – *Intra sub-partitioning configurations*

In VVC, Matrix-based Intra Prediction (MIP) [35] is an alternative for intra prediction. A matrix is selected based on the intra prediction mode, and a convolution is carried out with a row and a column of reference samples presented in Figure 2.13. The available matrices are constructed through off-line training and are inserted in the specification of the standard. Moreover, for Intra predicted CUs an additional partitioning tool as been adopted in VVC. The Intra Sub-Partitionning (ISP) tool may perform additional partitioning of the CU vertically or horizontally into 2 or 4 sub-partitions depending on the block size. Figure 2.14 illustrates the possible ISP partitioning configurations depending on the size of the original block.

2.5.4 Inter Prediction

Inter prediction exploits temporal redundancy in the video sequence. It predicts the samples of a CU from samples of previously coded reference pictures. The process of finding the most similar block in the reference pictures is called Motion Estimation (ME). The ME relies on block matching and is one of most computationally complex operations in the encoding process. It outputs one or several Motion Vectors (MVs), each composed of a vertical and an horizontal component, representing the translation of the CU from the reference frame to current frame.

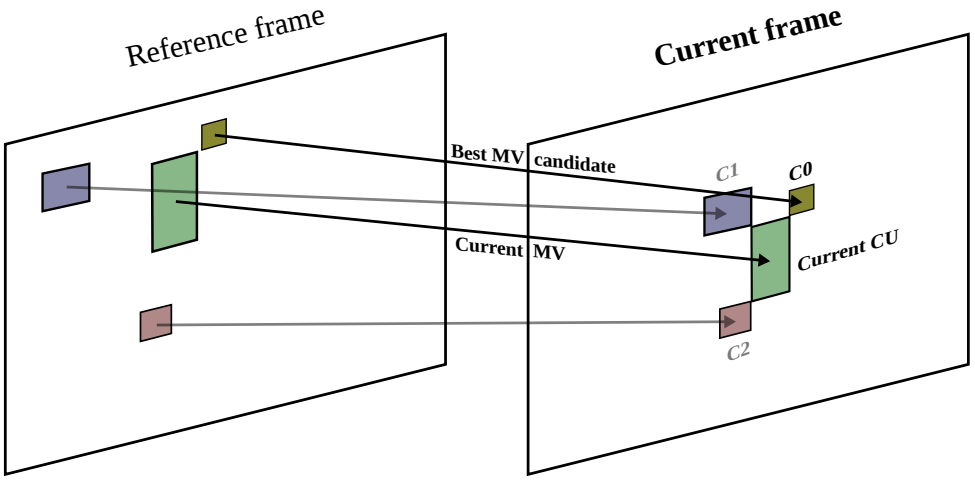


Figure 2.15 – *Simplified example of Advanced Motion Vector Prediction.*

After the ME is processed, other coding tools are applied to reduce the transmission cost of MVs in the bitstream. An effective tool, introduced in HEVC and extended in VVC, is called Advanced Motion Vector Prediction (AMVP) [36]. The AMVP exploits both spatial and temporal homogeneity of the motion inside the video sequence. In Figure 2.15, a simplified example of AMVP is provided considering only spatial neighboring CUs as candidates. For the current green CU, a candidates list containing the previously coded neighboring CUs $C0$, $C1$ and $C2$ is proposed. The CU with the more similar MV compared to current MV is selected. In the case of Figure 2.15, the MV of CU $C0$ is selected as best candidate. The encoder only transmits the index in the list of the best candidate and the MV difference between current MV and best candidate MV. This difference requires less bits to be coded and transmitted compared to the original MV obtained after ME. In VVC, an improved version of AMVP is available compared to HEVC. This new version inherits more information from reference and combines temporal and spatial prediction.

The Skip/Merge Modes [37] are other efficient tools to reduce the transmission size of MVs, with similar operating principle as AMVP.

For Inter Prediction, VVC standard includes several new and refined inter prediction coding tools compared to its predecessors including among others Subblock-based Temporal Motion Vector Prediction (SbTMVP), Bi-Directional Optical Flow (BDOF) and an improved version of AMVP previously mentioned. The Decoder-side Motion Vector Refinement (DMVR), also included in VVC, reduces the bit-rate induced by inter prediction by refining the motion vectors at the decoder side. Less information is transmitted in the bitstream, but this information must be retrieved with extra calculations at decoder side. The reader is referred to the document [38] for detailed description of these new tools. The benefits in term of coding efficiency as well as the computational complexity of these novel inter prediction tools is discussed in Chapter 3.

2.5.5 Transform

The Multiple Transform Selection (MTS) concept in VVC defines three separable trigonometrical transform types including Discrete Cosine Transform (DCT)-II, VIII and Discrete Sine Transform (DST)-VII. As illustrated in Figure 2.16, the MTS concept selects, for Luma blocks of size lower than 64, the set of transforms that minimizes the rate distortion cost among five transform sets and the skip configuration. However, only DCT-II is considered for chroma channels and Luma blocks of size 64. The MTS solution brings a significant coding gain of respectively 0.84% and 0.33% in All Intra (AI) and Random Access (RA) coding configurations [39] compared to HEVC.

The *sps_mts_enabled_flag* flag defined at the Sequence Parameter Set (SPS) enables to activate the MTS concept at the encoder. Two other flags are defined at the SPS level to signal whether implicit or explicit MTS signaling is used for Intra and Inter coded blocks, respectively. For the explicit signaling, used by default in the Common Test Conditions (CTC), the *tu_mts_idx* syntax element signals the selected horizontal and vertical transforms, as specified in Table 2.1. This flag is coded with Truncated Rice code with rice parameter $p = 0$ and $cMax = 4$ (TRp).

Table 2.1 – *Signalling of the MTS in the explicit mode*

tu_mts_idx	Transform Direction	
	Horizontal Transform	Vertical Transform
0	DCT-II	DCT-II
1	DST-VII	DST-VII
2	DCT-VIII	DST-VII
3	DST-VII	DCT-VIII
4	DCT-VIII	DCT-VIII

The Low-Frequency Non-Separable Transform (LFNST) has been adopted in the VTM-5. The LFNST relies on matrix multiplication applied between the forward primary transform and the quantization at the encoder side

$$\vec{Y} = T \cdot \vec{X}, \tag{2.8}$$

where the vector $\vec{X}$ includes the coefficients of the block rearranged in a vector and the matrix T contains the coefficients transform kernel. The LFNST is enabled only when DCT-II is used as a primary transform.

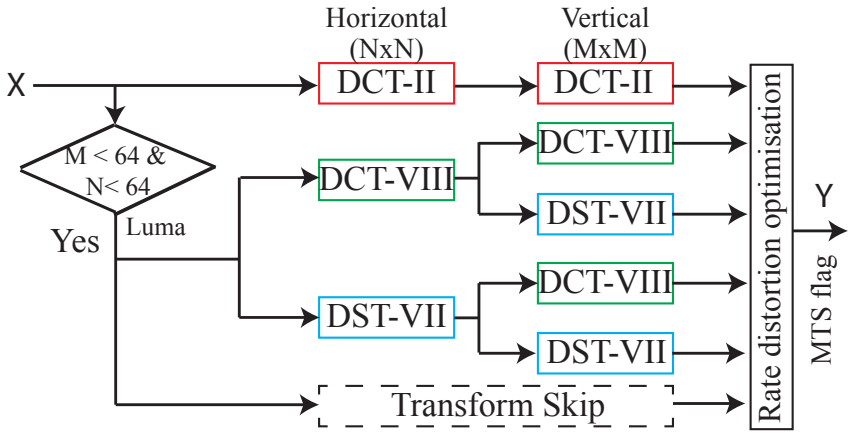


Figure 2.16 – *The concept of 2D separable transforms selection in VVC. X is the input block of residuals, Y is the output transformed block and MTS flag is the index of the selected set of transforms.*

Four sets of two LFNST kernels of sizes 16x16 and64×64 are applied on 16 coefficients of small block (min (width, height) $<$ 8) and 64 coefficients of larger (min (width, height) $>$ 4) blocks, respectively.

Four sets of two LFNST kernels of sizes 16×16 and 64×64 are applied on 16 coefficients of small block (min (width, height) $<$ 8) and 64 coefficients of larger (min (width, height) $>$ 4) blocks, respectively. The VVC specification defines four different transform sets selected depending on the Intra prediction mode and each set defines two transform kernels. The used kernel within a set is signaled in the bitstream. To reduce the complexity in number of operations and memory required to store the transform coefficients, the 64×64 inverse transform is reduced to 48×16. Therefore, only 16 basis of the transform kernel is used and the number of input coefficients is reduced to 48 by excluding the bottom right 4×4 block (i.e. include only coefficients of the top-left, top-right and bottom-left 4×4 blocks).

2.5.6 Quantization

Quantization is a lossy compression method that defines a single quantum value to map a range of values. In the classical hybrid video coding scheme, quantization is applied on the transformed residual coefficients, under the form of discrete mapping of the coefficients into integers. Since the HVS is less sensitive to high frequencies compared to low frequencies, it is possible to discard a part of the high frequency information with little reduction on the perceived quality of the video. For this reason, the quantization mapping is non-linear with larger steps for high transform coefficients values.

The Quantization Parameter (QP) is the coding parameter handling the size of quantization step. Its values range from 0 to 63 in VVC standard, when the maximum value in HEVC was set to 51. The average quantization step increases by approximatively 12% for a QP increase of 1, inducing more information loss in the transformed coefficients. Therefore a higher QP leads to a lower bit-rate, generally at the cost of higher distortion.

2.5.7 In-Loop Filters

Four in-loop filters are performed on the reconstructed pixels in order to reduce the visual artifacts of previous coding tools. They include the Luma Mapping with Chroma Scaling

(LMCS), the Deblocking Filter (DBF), the Sample Adaptive Offset (SAO) and the Adaptive Loop Filter (ALF). The LMCS is a novel tool introduced in VVC [40]. It modifies the predicted values of inter predicted blocks by reshaping (i.e. redistributing) the samples across the entire possible value range. After reconstruction, the inverse reshaping is performed before the DBF stage.

The DBF [41] is applied on block boundaries, reducing the blocking artifacts introduced among others by quantization. First, the vertical filtering for horizontal edges is performed, followed by horizontal filtering for vertical edges. Subsequent to the DBF, the SAO is applied [42]. The SAO classifies the reconstructed samples into different categories and reduces the distortion sample by sample. For each category, an offset value has been retrieved by the entropy decoder, and is added during SAO process to each sample of the category.. The SAO is particularly efficient to filter the ringing artifacts.

The last in-loop filter the ALF [43], which allows block-based filter adaption. For the chroma component, the bitstream may contain up to 8 different 5×5 diamond shape filters. Each chroma CTU uses one of the provided filter coefficients. The filters for luma component are 7×7 diamond shape filters. The filter is applied in each 4×4 pixels block, based on the direction and activity of local gradients. The filter is selected among 16 predefined filters and up to 25 transmitted sets of filter coefficients. The ALF can be considered as an adaptive filter that collects and reduces the remaining coding artifact and further enhances the video quality.

2.5.8 Entropy Coding

The CABAC [32], first introduced in AVC standard, is the entropy engine used in VVC. The CABAC compacts all the syntax elements obtained after coding process, such as intra and inter information, quantised coefficients or in-loop filtering parameters. It is lossless operation and thus does not introduces additions distortion to the reconstructed frames. Its operating principle is divided in the three following steps: binarization, context modeling and arithmetic coding. The syntax elements are converted into binary symbols and associated with a predefined context. One by one, the binary symbols are converted into a probability following a probability model specific to their context. Finally, each binary symbol is arithmetically coded to a number of bits that is function of its probability.

In HEVC the transform coefficients of a coding block are coded using non-overlapped coefficient groups, each of them containing the coefficients of a 4×4 sub-block of a CU. In VVC, various coefficient groups are allowed: 1×16, 2×8, 8×2, 2×4, 4×2 and 16×1. In addition, the core of the CABAC engine includes some important changes in VVC compared to the design in HEVC. The CABAC engine in HEVC uses a table-based probability transition process between 64 different representative probability states. In VVC, a decode decision uses a 2-state model with variable probability updating window sizes [44].

2.5.9 Temporal Prediction Structure

Previous sections gives an overview of the processing of a single frame in VVC classical hybrid video coding scheme. This Section describes the temporal prediction structure of the video sequence into Group of Picturess (GOPs). Each GOP contains a fixed number of frames and is encoded following a certain coding configuration. The most commonly used coding configuration in the video coding field are presented in this section, along with their advantages and drawbacks in concrete use cases.

2.5.9.1 Slice/Frame Types

A frame may be divided into one or several slices. Initially, slices have been introduced to divide a bit-stream into smaller independently decodable parts for transmission. The network characteristics generally define the maximum size of the slices in the bit-stream. Three distinct types of slices are used in the GOP structures further presented, each with distinct prediction properties. To facilitate understanding, we will consider in this section the case of a single slice per frame. The terminology I-slice, P-slice and B-slice is therefore changed in favor of I-frame, P-frame and B-frame.

- **I-frame**: intra coded frame, only the intra prediction is allowed in these frames. Since intra prediction only relies on the spatial content of current frame, I-frames are not dependent of the encoding or decoding of other frames to be processed. Intra prediction usually results in lower coding performance compared to inter prediction. For this reason, the I-frames are usually less used compared to the type of frames presented below.
- **P-frame**: predictive coded frame, uses both intra and inter prediction. For Motion Compensation (MC) in inter prediction, only a single reference frame may be used. P-frames introduce a delay compared to I-frames since they require the previous encoding or decoding of another frame. The coding performance is however better since additional information is used for inter prediction.
- **B-frame**: bi-directional coded frame, same as P-frame but may use several reference frames for MC, improving coding efficiency compared to P-frame. B-frames use both temporally backward (past) and forward (future) frames as reference. The processing of B-frames is the most computationally complex since the ME algorithms must explore several frames.

2.5.9.2 All Intra Coding Configuration

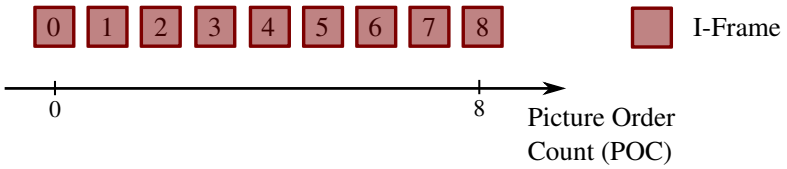


Figure 2.17 – *AI coding configuration.*

In AI coding configuration, the video sequence is encoded only with I-frames. Figure shows the 9 first frames of a sequence in AI coding configuration. The Picture Order Count (POC) in abscissa corresponds to the display order of the frames, the numbers inside the boxes represent the encoding order of the frames. The I-frames are encoded in the POC order and can be independently processed, introducing no latency in the coding process. The bit-rate of AI coded video sequences is usually prohibitive for streaming purposes.

2.5.9.3 Low Delay Coding Configuration

The Low-Delay (LD) coding configuration includes both intra and inter coded frames. It uses smaller GOP sizes and less complex MC dependencies compared to RA coding configuration (see Section 2.5.9.4). Two classical configurations are Low-Delay P (LDP),

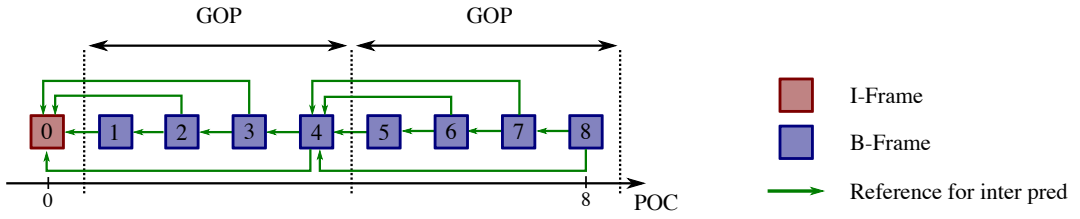


Figure 2.18 – *Low-Delay B (LDB) coding configuration.*

containing only I-frames and P-frames, and LDB, that also includes B-frames. An example of LDB coding configuration is given in Figure 2.18. The GOP size is 4, composed of B-frames and the coding order is the same as the POC. LD coding configurations are widely used in video conferencing for its better coding efficiency compared to AI coding configuration and for its low delay compared to RA coding configuration.

2.5.9.4 Ramdom Access Coding Configuration

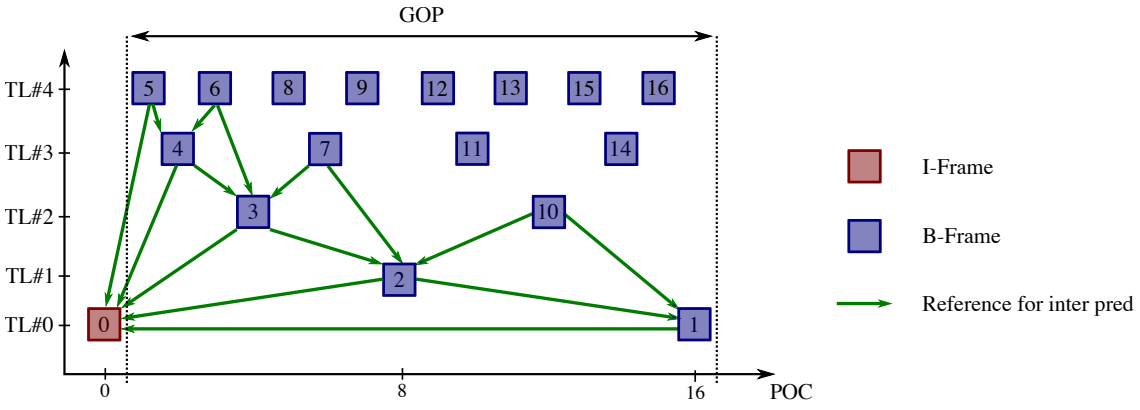


Figure 2.19 – *RA coding configuration.*

The RA coding configuration includes I-frames and B-frames. This coding configuration includes random access points in the bitstream. Random access points are frames that the decoder may process without requiring other decoded frames. The random access points correspond to I-frames distributed along the sequence, localized at the start of several GOPs. Figure 2.19 shows 16 frames localized after a random access point (I-frame) in a typical example of RA coding configuration. The GOP structure size in this example is 16, composed of 16 B-frames. The frames at the same level of ordinates belong to the same Temporal Layer (TL). The frames belonging to a TL only use as reference for MC frames belonging to lower TLs, as shown in Figure 2.19.

The frames with lower TL are processed first since their coded content is required for MC of higher TL frames. The RA coding configurations is widely used in over-the-top and broadcast services since it achieves the highest coding efficiency. This coding efficiency comes at the cost of computational complexity increase and additional delay. For this reason, the contributions in Chapters 4 and 5 focus on the complexity reduction and the parallel processing of VVC encoding process in RA configuration, respectively.

2.6 Baseline Software video codecs

In order to investigate new coding tools and prove the coding performance of a standard, each standard comes with a software that implements a reference encoder and decoder.

As mentioned in Section 2.4, VVC has been developed by the JVET, which is the result of the collaboration between VCEG and MPEG. After every standardization meeting, which takes place approximately every three months, a new version of the reference software is released with the new tools approved by the JVET. The first reference software used by the JVET to investigate the first coding tools for VVC standard is called the Joint Exploration Model (JEM) [45][46]. The reference software used in a more advanced phase of VVC standardization process by researchers and industrials is called VVC Test Model (VTM). During HEVC standardization process, the reference software was called HEVC Model (HM). Since the purpose of the reference software is to demonstrate the capabilities of the standard, their implementation achieve high quality encoding through an almost exhaustive RDO process (see Section 2.3). They are therefore well suited for research, but not usable in use-cases that require low complexity and memory consumption such as over-the-top media services or broadcast.

This thesis has been conducted during the standardisation of VVC, therefore different software projects are used for the three proposed contributions. The software models used in the various contributions of this thesis are shown in Table 3.2, along with their corresponding versions. The contribution presented in Chapter 4 has originally been designed to reduce the encoding complexity in the JEM-7.0 encoder and as further been adapted to the VTM-5.0 encoder. The HEVC reference software HM-16.20 and the VVC reference software VTM-5.0 are compared in term of coding efficiency and complexity repartition in Chapter 3, on both encoder and decoder side. The VTM-6.2 encoder is parallelized in Chapter 5 in order to reduce its execution time. The in-loop filters implementation and the parallelism technique proposed in Chapter 6 have been included to *openVVC* decoder. The *openVVC* decoder is developed by the VAADER team of IETR laboratory [1] in C programming language, and is integrated as a dynamic library inside FFmpeg library [47].

Table 2.2 – *Softwares used in the various contributions of this thesis.*

Software	Chapter 3	Chapter 4	Chapter 5	Chapter 6
HM-16.20	✔			
JEM-7.0		✔		
VTM-5.0	✔	✔		
VTM-6.2			✔	
OpenVVC				✔

2.7 Conclusion

In this chapter we have presented the fundamental concepts of video coding, from video signal format to the recently released video coding standard Versatile Video Coding. Numerous coding concepts, evaluation metrics and tools necessary to understand the rest of the document have been introduced. They include, among others, the quality metrics of

[1] https://www.ietr.fr/spip.php?article1604

a coded video signal, the RDO process that estimates the optimal coding parameters, the block partitioning scheme in VVC and the software projects used during this thesis. Next chapter provides a quality assessment of VVC coding tools as well as a complexity profiling of VVC reference software VTM-5.0 over HEVC reference software HM-16.20.

CHAPTER 3

Video Quality Assessment and Coding Complexity of VVC Standard

Finalized in July 2020, the primary objective of Versatile Video Coding (VVC) is to provide 40% bit-rate savings in terms of Bjøntegaard Delta BitRate (BD-BR) (see Section 2.2.3) over High Efficiency Video Coding (HEVC) standard with reasonable complexity increase at both encoder and decoder. It is important to evaluate the gains in term of couple quality/rate of the new standard to demonstrate its efficiency. In this Chapter we present a subjective and objective quality assessment study in order to evaluate the performance of this new emerging standard in comparison with HEVC standard. The VVC reference software version 5 (VVC Test Model (VTM)-5.0) is compared with the HEVC reference software HEVC Model (HM)-16.20 in Random Access (RA) coding configuration. Several video contents, at different bit-rates, and two spatial resolutions, High Definition (HD) and Ultra High Definition (UHD) have been used. In VVC, most of the coding efficiency gains are provided by a set of tools, given in "Table 3.1" in term of objective metric BD-BR. To obtain these results the authors measured the individual coding efficiency loss when the tools are disabled in the VTM-5.0. In this Chapter we will measure the gain in term of objective and subjective metrics achieved when the tools are enabled altogether.

This Chapter also provides an analysis of times and complexity repartitions, for both encoding and decoding processes. In order to highlight the principal evolutions from one reference software to another, the VTM-5.0 and HM-16.20 profiling are presented simultaneously. In [48], authors have assessed the impact on HEVC complexity of several new tools by separately disabling them and measuring the encoding/decoding times. Many other tools are mandatory in the main profile and cannot be disabled at the encoder side, meaning their complexity cannot be measured with this approach. Moreover, disabling one tool would result in different coding decisions compared to the reference configuration and changes the complexity repartition. The profiling lead in this Chapter has two main benefits compared to previously mentioned paper: it allows assessing the complexity of all encoding/decoding tools, and the complexity is measured in the original encoding/decoding process with all tools enabled. Table 3.2 summarizes the related works proposing a comparison between VTM and HM in the literature. The work presented in this Chapter assesses most points of comparison between VTM and HM including 4 quality metrics and the complexity profiling of both encoding and decoding processes.

The remainder of this Chapter is organized as follows. In Section 3.1, we describe the subjective quality assessments. The results obtained in term of objective and subjective

quality metrics are provided in Section 3.2. Section 3.3 provides an analysis of times and complexity repartitions, for both encoding and decoding processes.

Table 3.1 – *Performance of the main tools included in the VTM-5.0 software [39].*

Module	Tool description	BD-BR
Block partitioning	Multi Type Tree	≈6.0%
	Triangular partition mode	0.35%
	Chroma separate tree	0.14%
	Sub-block transform	0.41%
Transforms	Multiple Transform Selection	0.33%
	Low-frequency non-separable tran.	0.79%
Inter Prediction	Decoder-side Motion Vector Refinement	0.82%
	Subblock-based temp. merging cand.	0.43%
	Affine motion model	2.53%
	Merge with MVD	0.58%
	Bi-directional optical flow	0.78%
	Temporal motion vector predictor	1.19%
Intra Prediction	Multi-reference line prediction	0.20%
	Intra sub-partitioning	0.13%
	Matrix based intra prediction	0.27%
In-Loop Filt.	Adaptive Loop Filter	4.91%
Quantization	Dependent Quantization	1.71%

Table 3.2 – *Related works proposing a comparison between VTM and HM.*

	Codec		**Quality**				**Complexity**		**Configuration**			**Sequence**	
Reference	VTM	HM	PSNR	SSIM	VMAF	Subj.	Enc.	Dec.	AI	RA	LD	Total	$\geq 2160p$
Topiwala *et al.* [49]	5.0	16.18	✔							✔		9	2
Zhang *et al.* [50]	4.1	16.18	✔	✔	✔	✔				✔		9	9
Pakdaman *et al.* [51]	8.0	16.18					✔			✔	✔	8	2
Siqueira *et al.* [52]	6.0	16					✔	✔	✔	✔	✔	6	1
Laude *et al.* [53]	5.0	16.9	✔				✔			✔		19	0
Cerveira *et al.* [54]	4.0	16.19	✔				✔		✔	✔		28	8
Bossen *et al.* [55]	9.0	16.20	✔				✔	✔	✔	✔	✔	22	6
Proposed	5.0	16.20	✔	✔	✔	✔	✔	✔		✔		14	7

3.1 Subjective Quality Assessments

As mentioned in Section 2.2, subjective quality assessment is the process of employing human viewers for grading video quality based on individual perception [5]. This section describes the subjective quality assessment setup and illustrate the test material, environment and used methodologies.

3.1.1 Experimental environment

The subjective study (see Section 2.2 for definition) has been conducted in the INSA/I-ETR PAVIM Lab, which is a platform for video quality monitoring actively involved in the emerging video contents. This platform includes a psycho-visual testing room, complying with the ITU-R BT.500-13 Recommendation. A display screen UHD of 55 inches Loewe Bild 7.55 was used to visualize the video sequences. 44 observers, 30 men and 14 women aged from 20 to 55 years, have participated in this experiment. All the subjects were screened for color blindness and visual acuity using Ishihara and Snellen charts, respectively, and have a visual acuity of 10/10 in both eyes with or without correction. Finally, all participants have been gratified. Viewers are placed at distances of 1.5 and 3 times the height of the screen for UHD and HD resolutions, respectively. In this section, we provide information regarding the used video sequences, test material, test settings, and evaluation procedures.

3.1.2 Test Video Sequences

In this experiment, a set of higher resolution video sequences, from various categories (music, sport, gaming, etc.) has been selected from several datasets (Huawei, SVT, b<>com) as well as 4EVER[1] database. The target resolutions for this test are HD (i.e. 1920x1080) and UHD (3840x2160), in a Standard Dynamic Range (SDR). Firstly, 13 videos sequences in UHD resolution have been selected and down-sampled using the same down sampling filters used in Scalable High Efficiency Video Coding (SHVC). The choice of these video is mainly based on the video encoding complexity in terms of color, movement, texture and homogeneous content.

Table 3.3 – *Configuration parameters for HM-16.20 and VTM-5.0 main profiles, in RA configuration.*

Parameter	HM-16.20 main	VTM-5.0 main
CTU size	64	128
QT max depth	4	4
MTT max depth	0	3
Transform Types	DCT-II, DST-VII	DCT-II, DST-VII, DCT-VIII
Max TU size	32	64
Loop Filters	DBF, SAO	LMCS, DBF, SAO, ALF
Search Type	TZ	TZ
Search Range	384	384
N. Ref. Pictures	5	5
Entropy Coding	CABAC	CABAC
Internal Bit Depth	10	10

All these videos were encoded using both HEVC (HM-16.20) and VVC (VTM-5.0) main profiles, at different bit-rates, in HD and UHD format in RA configurations. The main profiles configuration parameters are summarized in TABLE 3.3. This first selection is done in order to retain the sequences representing a good balance of content variety and video coding artefacts. After this initial selection, only seven scenes have been retained

[1]For Enhanced Video ExpeRience 2 project, www.4ever-2.com

for the experiment , as given in "Table 3.4". In total 168 video sequences are used in this study: 7 scenes × 5 bit-rates × 2 codecs × 2 resolutions + 28 original scenes (2 references per scene in two resolution). An example snapshots of used video sequences is shown in "Fig. 3.2".

Table 3.4 – *Test video sequences*

Sequence	HD	UHD	Fps	Bit Depth
AerialCrowd2	1920 × 1080	3840 × 2160	30	10
CatRobot1	1920 × 1080	3840 × 2160	60	10
CrowdRun	1920 × 1080	3840 × 2160	50	8
DaylightRoad	1920 × 1080	3840 × 2160	60	10
Drums2	1920 × 1080	3840 × 2160	50	10
HorseJumping	1920 × 1080	3840 × 2160	50	10
Sedof	1920 × 1080	3840 × 2160	60	8

In order to measure video contents diversity across the selected test sequences, Spatial Information (SI) and Temporal Information (TI) metrics are used [56]. The SI is increasing with the amount of spatial details, whereas the TI increases with the quantity of motion in the sequence. In Figure 4.5, the 14 test sequences are represented under the SI TI coordinates. The green crosses and red stars correspond to UHD and HD test sequences, respectively. Figure 4.5 shows that the down-sampled HD sequences have equivalent TI coordinates and significantly higher SI coordinates compared to the original UHD sequences. This is due to the increased quantity of spatial details contained in the down-sampled HD frames compared to the original UHD frames, since the same visual information is located in a smaller frame.

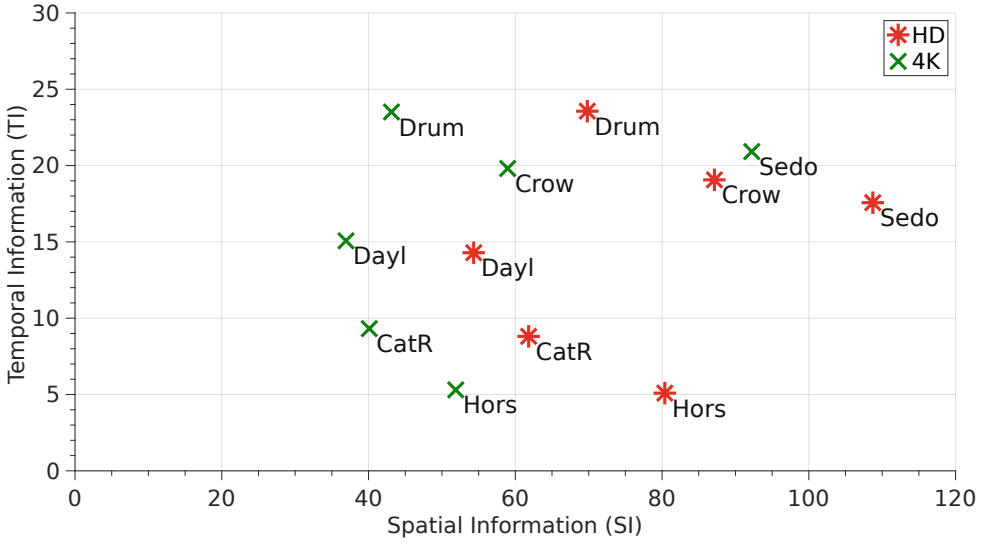


Figure 3.1 – *SI and TI of the test sequences, according to the resolution.*

3.1.3 Evaluation Procedure

In this quality assessment experiment, the Subjective Assessment Methodology for Video Quality (SAMVIQ) method was used [57]. This method has specifically been designed for

multimedia content. It takes into account a range of codec types, image formats, bit-rates, temporal resolutions, etc. It has been recommended by International Telecommunication Union (ITU)-R 6Q in 2004 [58]. Each scene (video sequence) is presented with the following conditions: an explicit reference, a hidden reference and 10 processed video sequences (PVSs). 4 categories of PVS are tested (HEVC-HD, VVC-HD, HEVC-UHD and VVC-UHD). The button with label REF clearly identifies the explicit reference sequence. The hidden reference is identical to the explicit reference but it is not readily accessible to the subject and it is "hidden" among other stimuli. For each scene, participants were asked to evaluate the processed video sequences, given by buttons with letter labels A to K (including the hidden reference), as indicated by the protocol SAMVIQ [59]. The conducted experiment is divided into two parts: HD and UHD. For an optimal visual comfort, these two parts have been done separately but using the same participants. Moreover, to prevent from visual fatigue, each part (HD and UHD) of the test is carried-out in two sessions. Before each experiment, participants receive clear and deep explanations about the evaluation procedures and the used interface. Finally, all viewers scores have been collected using a dedicated graphical user interface, developed in compliance with the SAMVIQ recommendation.

3.2 Quality Evaluation Results

3.2.1 Objective Evaluation Results

Several objective metrics of different categories are used in this Chapter: Mathematical approaches (Peak Signal to Noise Ratio (PSNR), BD-BR, Bjøntegaard Delta PSNR (BD-PSNR)), weighting approaches (Structural SIMilarity (SSIM)) and by modeling the Human Visual System (HVS) (Video Multi-Method Assessment Fusion (VMAF)). All the mentioned metrics have been described in Section 2.2. The objective quality results, for the whole test sequences, are shown in Fig. 3.3 in the form of PSNR (top), SSIM (middle) and VMAF (bottom) versus bit rate plots. The solid and dotted lines represent HEVC and VVC codec curves, respectively. It is worth noting here that the objective (i.e. PSNR, SSIM, VMAF) and subjective (i.e., MOS) results are independently analyzed, and thus no direct correlation between them is demonstrated, since we do not develop here a new objective quality metric. Also, the PSNR results presented here are weighted through the three components (y, u, v), as shown in the equation (2.4). Figs. 3.3 (top) illustrates the average performance in terms of PSNR metric of the whole used dataset. According to these figures, PSNR values increase significantly when using VTM-5.0 coding tools, and consequently VTM enables higher video quality than the HM codec. In addition, it can be observed that VVC codec achieves the same objective quality as HEVC while typically requiring substantially lower bit rates. In these figures, VVC codecs enable a bit-rate saving up to 40% for some sequences as *CatRobot* and *DaylightRoad* in the two used formats HD (left) and UHD (right).

The obtained results using SSIM metric, Figs. 3.3 (middle), have shown also that VVC codecs enables a bit-rate saving up to 50% for some test sequences, in HD (left) and UHD (right) formats, compared to the HEVC standard. In these figures, we can noticed that, beside the *CrowdRun* sequence in low bit-rate, all the used test sequences have a good quality measures: higher than 0.96 in HD and 0.97 in UHD formats. For some sequences, as *AerialCrowd2* in HD and *DaylightRoad* in UHD formats, a considerable bit rate gains is obtained by VVC codec while keeping the same quality measures. In other words, for the

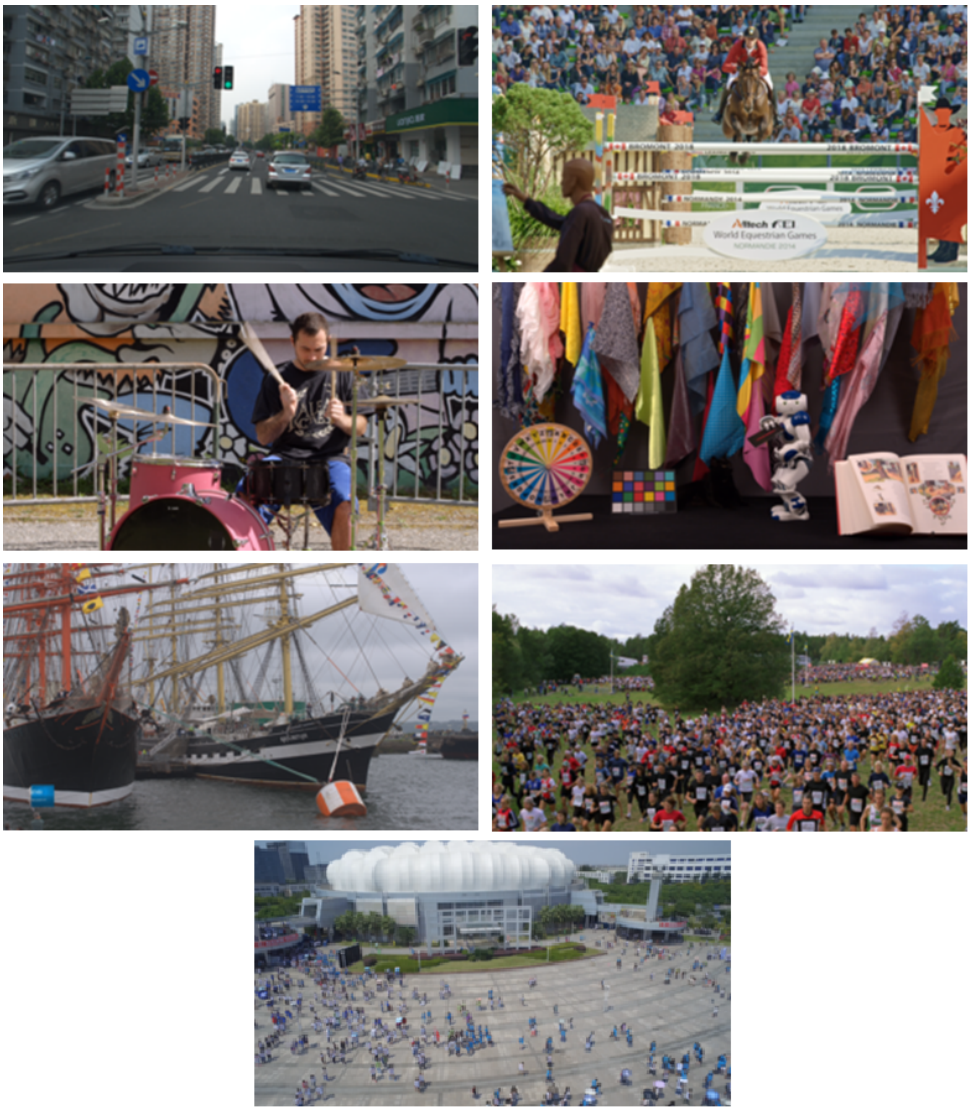


Figure 3.2 – *An example frames of the used video sequences (HD & UHD).*

same quality at high bit-rate (> 0.99), a bit-rate gain from $6Mbps$ to $8Mbps$ is obtained. This gain is ranking from $13Mbps$ to $21Mbps$ in UHD format.

Finally, Figs. 3.3 (bottom) present the objective measurements using VMAF metric, in HD (left) and UHD (right) contents. Similar to the other metrics, these figures indicate that VVC codecs enable a bit-rate saving up to 49% compared to HEVC standard. It is noticed here that we have used these metrics, while presenting the similar behaviors, in order to cover a large number of quality measurement categories, for a possible future comparisons as well as a multiple ground truth datasets.

Table 3.5 summarizes the Bjøntegaard measurement (BD-BR) for HD and UHD contents, using PSNR metric. On average, the VTM-5.0 codec enables, in average, a bit rate savings of about 31% and 34% for HD and UHD video sequences, respectively. Moreover, using VMAF and SSIM, the same behaviors are noticed and a significant bit-rate gains is obtained. These gains are summarized in Table 3.7.

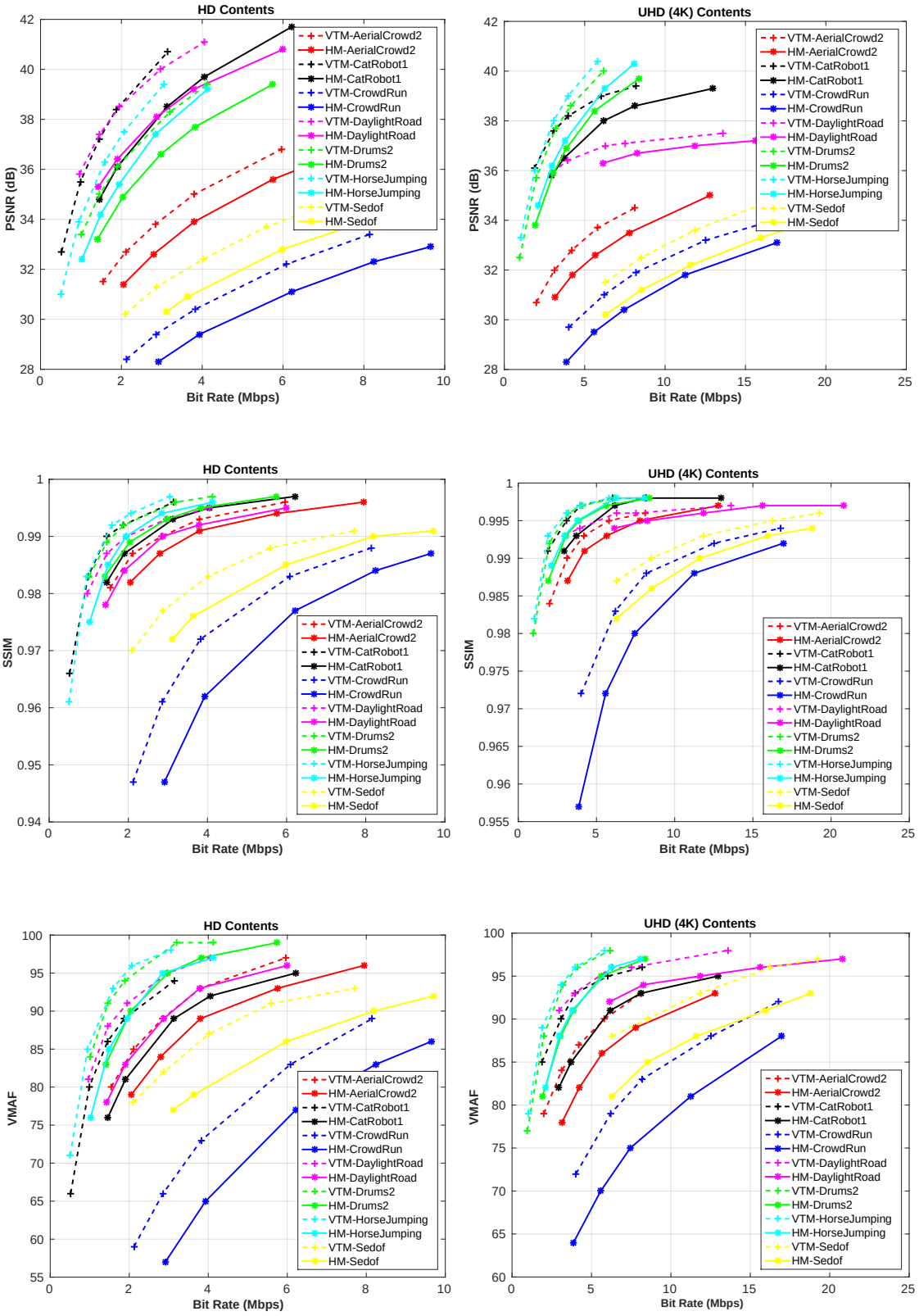


Figure 3.3 – *Objective-based comparison, using three metrics: PSNR (top), SSIM (middle) and VMAF (bottom), for the whole used datasets in HD and UHD (2160p) formats.*

Table 3.5 – *BD-BR (PSNR) of VTM-5 compared to the anchor HM video codec.*

Sequence	BD-BR (HD)	BD-BR (UHD)
AerialCrowd2	-24,5%	-29,9%
CatRobot1	-39,8%	-41,8%
CrowdRun	-27,2%	-30,3%
DaylightRoad	-38,9%	-42,4%
Drums2	-29,5%	-32%
HorseJumping	-31,1%	-32,5%
Sedof	-27.7%	-31,9%
Average	**-31.24%**	**-34.42%**

3.2.2 Subjective Assessment Results

The first step in the results analysis is to calculate the Mean Opinion Score (MOS) for each video used in the experience. This average is given by the equation 3.1.

$$MOS_{jk} = \frac{1}{N}\sum_{i=1}^{N} s_{ijk}, \tag{3.1}$$

where s_{ijk} is the score of participant i for the test video j of the sequence k and N is the number of observers. In order to better evaluate the reliability of the obtained results, it is advisable to associate for each MOS score a confidence interval, usually at 95%. This is given by the equation 3.2. Scores respecting the experiment conditions must be contained in the interval $[MOS_{jk} - IC_{jk}, MOS_{jk} + IC_{jk}]$.

$$IC_{jk} = 1.95\frac{\delta_{jk}}{\sqrt{N}}, \quad \delta_{jk} = \sqrt{\sum_{i=1}^{N}\frac{(s_{ijk} - MOS_{jk})}{N}}. \tag{3.2}$$

Before data analysis, we conducted a verification of the distribution of individual participant scores. In fact, some data may parasites the results. Thus, a filtering procedure was applied to obtained results based on the annex 2 of the ITU-R BT.500 recommendation [58]. In our outlier detection verification, a correlation index greater (Minimum between Pearson and Spearman correlations) than or equal to 0.75 is considered as valid for the acceptance of the viewer's scores; otherwise, the viewer is considered as an outlier. Following this screening process, 8 subjects (6 in HD and 2 in UHD test) were discarded. Consequently, only 36 viewers scores are retained.

Table 3.6 summarizes the BD-Rate gains obtained by VVC relative to HEVC, considering Mean Opinion Score (MOS). As shown in this table, VTM-5.0 outperforms the HEVC for the whole used sequences. A bit rate savings of about 37% and 40% are obtained for HD and UHD video sequences, respectively

The subjective evaluation results, for the whole test sequences are shown in Figs. 3.4 & 3.5 in the form of MOS versus bit rate plots, for HD and UHD formats, respectively. The solid and dotted lines represent HEVC and VVC codec curves, respectively. The associated confidence intervals are displayed for each MOS test point. In these figures, VTM-5.0 codec enables a higher MOS score and consequently a higher video quality, compared to HEVC reference software. As we can see, a considerable gains in terms of bit-rate savings can be noticed with the same perceived quality. For HD format , a bit-rate saving of -50% and -40% is obtained in the "Excellent" and "Good" quality area, respectively. For UHD

format, a bite-rate savings of about -50% is obtained, with the same perceived quality, for the sequence "CatRobot1". Reciprocally, for the same bit-rate, a considerable quality enhancement is obtained by VVC compared to HEVC standard. In fact, at low bit-rate, the sequence "CrowdRun" is judged as "Fair" quality using VVC while it is judged as "Poor" quality using HEVC standard. In addition, this quality is enhanced from "Fair" to "Good" quality area using VVC. The same behavior can be noticed for the other video sequences, depending on the considered bit-rates and the used video format.

Table 3.6 – *BD-BR (MOS) of VTM-5.0 compared to the anchor HM-16.20 video codec.*

Sequence	BD-BR (HD)	BD-BR (UHD)
AerialCrowd2	-33%	-41%
CatRobot1	-48%	-47%
CrowdRun	-35%	-35%
DaylightRoad	-49%	-48%
Drums2	-35%	-39%
HorseJumping	-38%	-45%
Sedof	-19%	-21%
Average	**-37%**	**-40%**

As a conclusion, for HD resolution, a bit-rate saving of 31% and 35% can been achieved with the VTM-5.0 in terms of PSNR and VMAF metrics, respectively. This gain exceed 40% for the UHD resolution, using VMAF metric. For the subjective comparison, the obtained gains is ranging between 37% and 40% for HD and UHD resolutions, respectively, as summarised in Table 3.7. For the same bit rate range, the highest bit-rate saving for HD contents is obtained by MOS-based BD-rate (-37%) while this gains for UHD contents is obtained by VMAF-based BD-rate (-40.44%).

3.2.3 Statistical Analysis

A statistical analysis was performed using the Analyse of Variance (ANOVA) approach [60]. Bit rate, Content, Resolution and Codec are used as independent variables and MOS is used as dependent variable. The null hypothesis in this case would be that theVVC test points have the same quality as the HEVC test point while the alternate hypothesis is that the VVC test points do not have the same perceived quality as the HEVC test point [60]. Results have shown that only codec parameter (VTM, HM) has a significant influence on the subjects scores, with $p\text{-}value < 0.0001$ [2]. Subjectively speaking, VVC coded enables a significant visual quality improvement regardless the used bit-rate, resolution and video content.

Table 3.7 – *Bit-rate Savings of VTM-5.0 over HEVC standard.*

Resolution	PSNR	VMAF	SSIM	MOS
HD	-31.24 %	-35.18 %	-33%	**-37%**
UHD	-34.42%	**-40.44%**	-38%	-40%

[2] a factor is considered influencing if $p\text{-}value < 0.05$

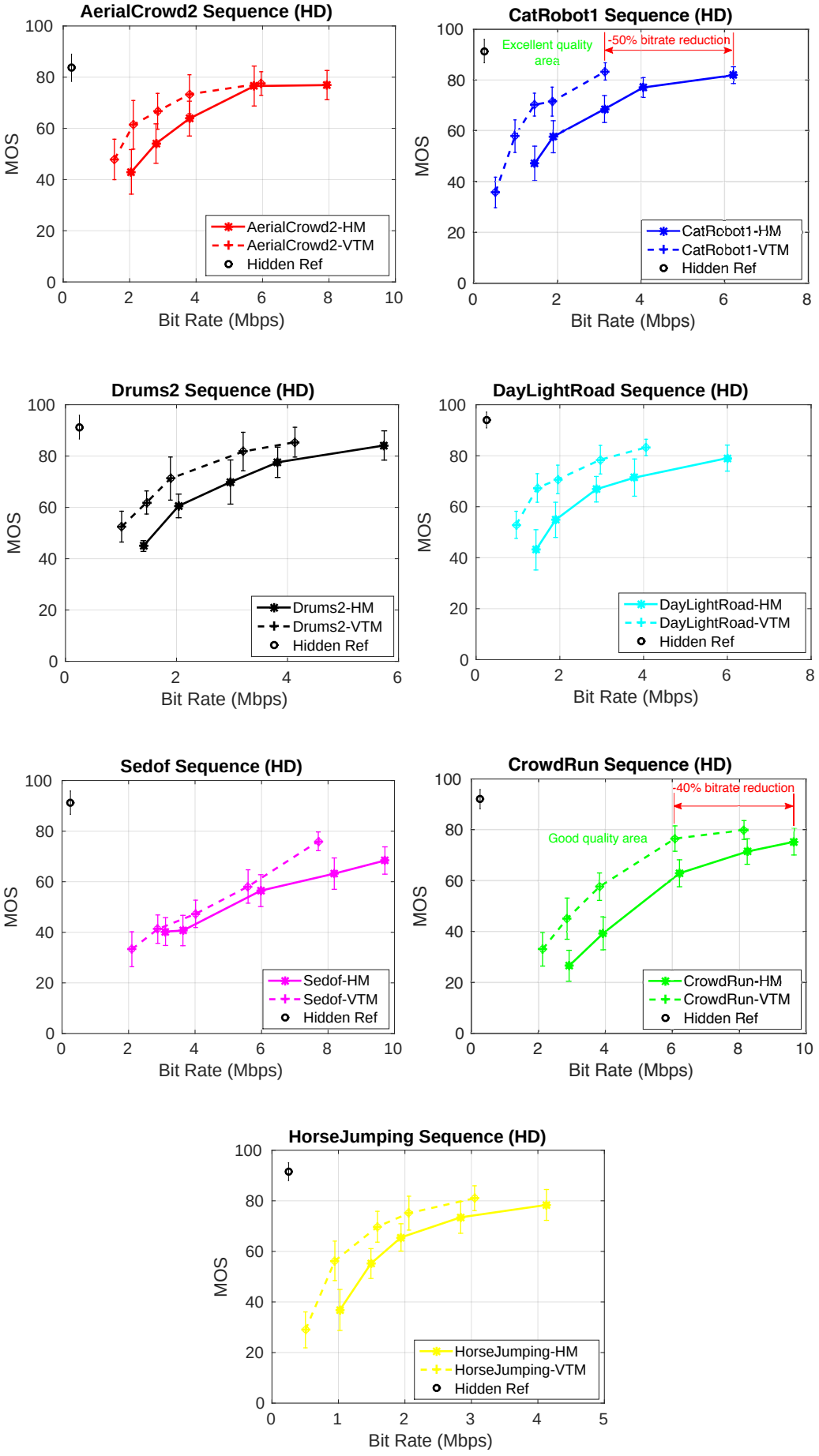


Figure 3.4 – *MOS-based comparison, with associated 95% confidence intervals, in HD format.*

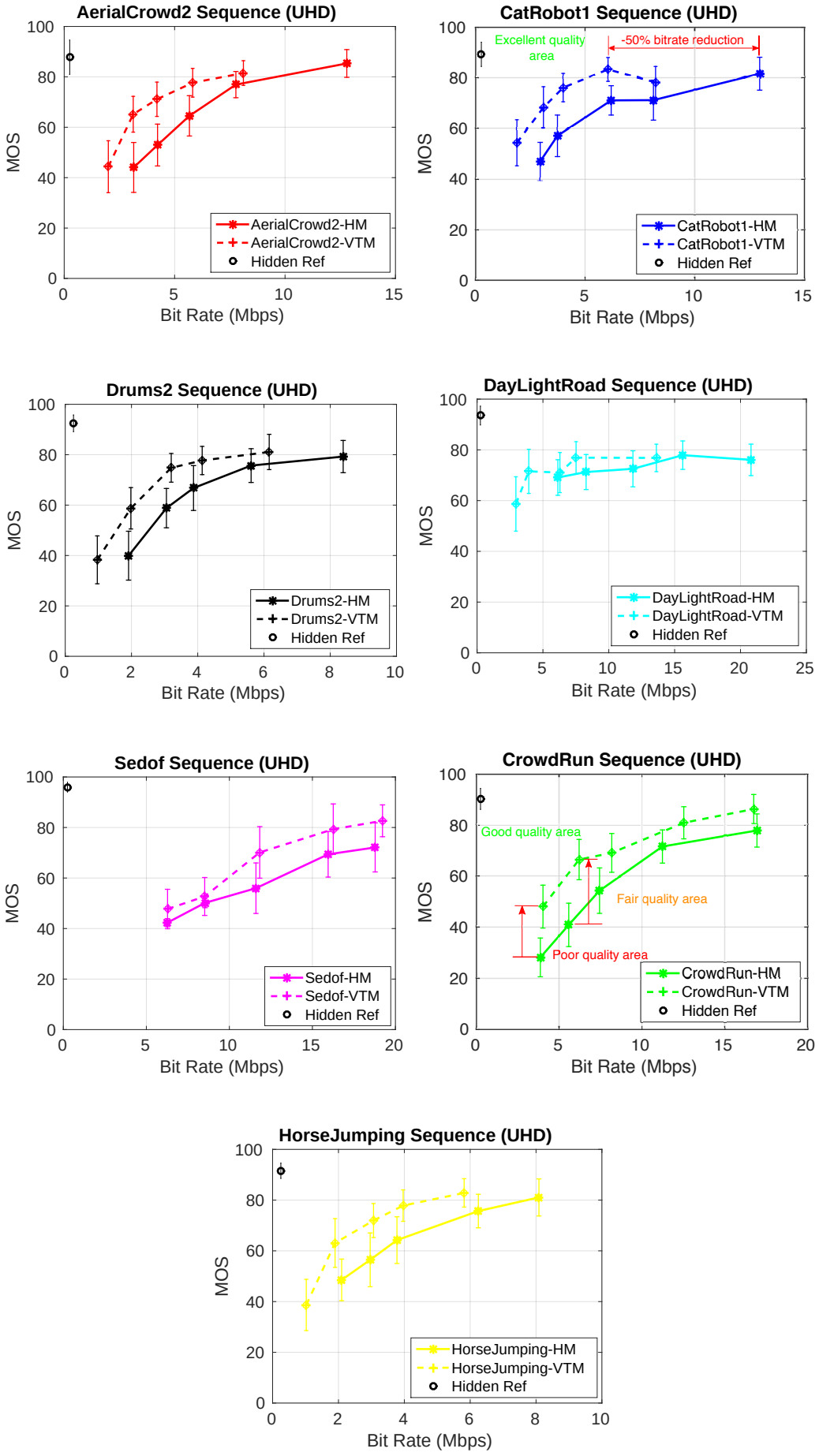


Figure 3.5 – *MOS-based comparison, with associated 95% confidence intervals, UHD format.*

3.3 VVC Coding Time and Complexity Repartition

This section provides an analysis of times and complexity repartitions, for both encoding and decoding processes. In order to highlight the principal evolutions from one reference software to another, the VTM-5.0 and HM-16.20 results are presented simultaneously.

3.3.1 Platform and Configuration Parameters

The experiments are carried-out on the same 14 sequences described in Section 3.1.2. In order to evaluate the impact of Quantization Parameter (QP) on encoding and decoding times, 2 QP values per resolution are selected. Encodings of HD sequences with QP=37 generate bit-rates considered too small to be representative of real use-cases, and bit-rates generated by UHD sequences encodings with QP=27 are considered too large. For this reason, the aforementioned QP values are discarded in the following experiments. Finally, four QP values are retained in the following experiments: QP=27 and QP=32, for HD sequences, and QP=32 and QP=37, for UHD sequences.

Table 3.8 – *Platform setup for complexity analysis*

CPU	Intel Xeon CPU E5-2603 v4
Clock Rate	1.70 GHz
Memory	8 Gb
Compiler	gcc 5.4.0
Operating System	Linux 4.4.0-127-generic

TABLE B.1 details the platform setup for complexity analysis. The experiments on codecs time and complexity repartition are carried-out on an Intel(R) Xeon(R) CPU E5-2603 v4, a Central Processing Unit (CPU) clocked at 1.70 GHz. The HM-16.20 and VTM-5.0 are both built with gcc compiler version 5.4.0, under Linux version 4.4.0-127-generic as distributed in Ubuntu-16.04.

By default, the VTM-5.0 features low-level optimizations, such as Single Instruction on Multiple Data (SIMD) optimizations, at both encoder and decoder sides. SIMD describes computers that perform the same operation on multiple data simultaneously, with a single instruction. To assess the time consumption of the VTM-5.0 without low-level optimizations, the SIMD optimizations at both encoder and decoder sides are disabled in the following. This also enable a fair comparison with the HM-16.20 software that does not include any SIMD optimization. The speed-up offered by SIMD optimizations is nonetheless measured and further discussed.

The repartition of encoding and decoding complexities, presented in upcoming Sections 3.3.2.2 and 3.3.3.2, are obtained by running the executables with Callgrind software [61]. Callgrind is the Valgrind profiling tool that records the call history of program functions as a call graph. By default, the data collected includes the number of instructions executed, the calling/callee relation between functions and the number of calls. Contrary to execution time, that depends among others on memory accesses or CPU frequency, the insight of the complexity repartition given by Callgrind is nearly constant regardless of the execution platform.

3.3.2 Encoders Analysis

3.3.2.1 Encoding Time

Table 3.9 – *Factor between encoding time and real time (×1000), for both VTM-5.0 and HM-16.20 in RA configuration, according to the test sequence and QP value.*

	HD				UHD			
Encoder	**HM-16.20**		**VTM-5.0**		**HM-16.20**		**VTM-5.0**	
QP	**27**	**32**	**27**	**32**	**32**	**37**	**32**	**37**
AerialCrowd2	1.1	0.9	22	15	3.9	3.4	64	41
CatRobot1	2.0	1.8	33	26	7.2	6.7	102	68
CrowdRun	2.4	2.1	49	39	8.6	7.6	179	124
DaylightRoad	2.3	2.0	44	32	8.1	7.3	132	84
Drums2	2.1	1.9	42	32	7.4	6.7	107	72
HorseJumping	1.4	1.3	23	17	5.4	5.0	57	38
Sedof	2.2	1.9	40	28	7.9	7.1	129	85
Mean	**1.9**	**1.7**	**36.2**	**26.9**	**6.9**	**6.3**	**110.3**	**73.2**
Std Dev	**0.4**	**0.4**	**9.8**	**7.9**	**1.5**	**1.4**	**38.9**	**27.3**

TABLE 3.9 shows an average factor of 1,700 between real-time encoding and the HM-16.20 encoding time of HD video content at QP=32. This average factor is 16 time greater (27,000) with VTM-5.0. The VTM-5.0 main profile is therefore 16 times more complex in average compared to the HM-16.20 main profile for the encoding of HD video content. This encoding complexity increase between HM-16.20 and VTM-5.0 is mainly caused by the Multi-Type Tree (MTT) partitioning scheme which allows 4 additional partition modes compared to Quad Tree (QT) partitioning in HM-16.20 (see Section 4.2.1). Indeed, our experiments show that when these additional partition modes are disabled in VTM-5.0, obtaining a partitioning scheme close to QT partitioning scheme, the complexity of the VTM-5.0 encoding process is in average divided by 5. The additional transform types and intra modes enabled in VTM-5.0 are also a non negligible cause for the observed complexity increase between HM-16.20 and VTM-5.0.

The results in TABLE 3.9 also highlight the impact of QP value on encoding complexity. Indeed, in order to avoid exhaustive Rate Distorsion Optimization (RDO) process and to decrease encoding time for researchers experimentations, the reference encoders include early termination techniques. In VTM for instance, a dozen early termination techniques speed-up the encoding process by a factor 8 [62] compared to exhaustive RDO process. The efficiency of these techniques vary with the encoding parameters, in particular with QP value. This explains why in TABLE 3.9, for VTM-5.0, encoding times carried out with lower QP values are 44% higher compared to encodings with higher QP values. The impact of QP decreases to 11% for HM-16.20, due to QT partitioning that offers less early termination opportunities compared to MTT partitioning in VTM-5.0.

Regardless the QP value and resolution, encoding times standard deviations are around 20% and 30% of average encoding times for HM-16.20 and VTM-5.0, respectively. These standard deviations are induced among others by the diversity of frame-rates among test sequences. For instance, the sequence *AerialCrowd2* with the lowest frame-rate (30 frames/sec) has the lowest encoding time in all columns of TABLE 3.9. The standard deviations are also a consequence of the diversity of spatial textures and movements among test sequences. Let us consider the sequences *HorseJumping* and *CrowdRun*, which have the same frame-

rate of 50 frames/sec. Figure 4.5 shows that *CrowdRun* has considerably higher SI and TI compared to *HorseJumping*. Given that previously mentioned early termination techniques are more efficient on sequences with lower spatial textures and slower motion, *CrowdRun* encoding time exceeds at least by a factor 1.4 *HorseJumping* encoding time, in all columns of TABLE 3.9.

As mentioned in Section 3.3.1, SIMD optimizations have been disabled in the VTM-5.0 encoder in this work. Enabling the SIMD optimizations speeds up the VTM-5.0 encoder by 1.9 times in average. This speed-up is very far from the 110,000 speed-up needed in average to achieve UHD real time encoding. Complexity reduction of HEVC encoder has been widely investigated [63, 64] in the past years, and recently more efforts have been dedicated to the VVC encoder. In fact, several techniques already proposed to reduce VTM encoding complexity by predicting a reduced set of likely intra modes [65, 66], or by testing a reduced number of partition configurations [67, 10]. The reduction of the VTM encoding complexity is the scope of next Chapter and we believe will be an active research field over the next years.

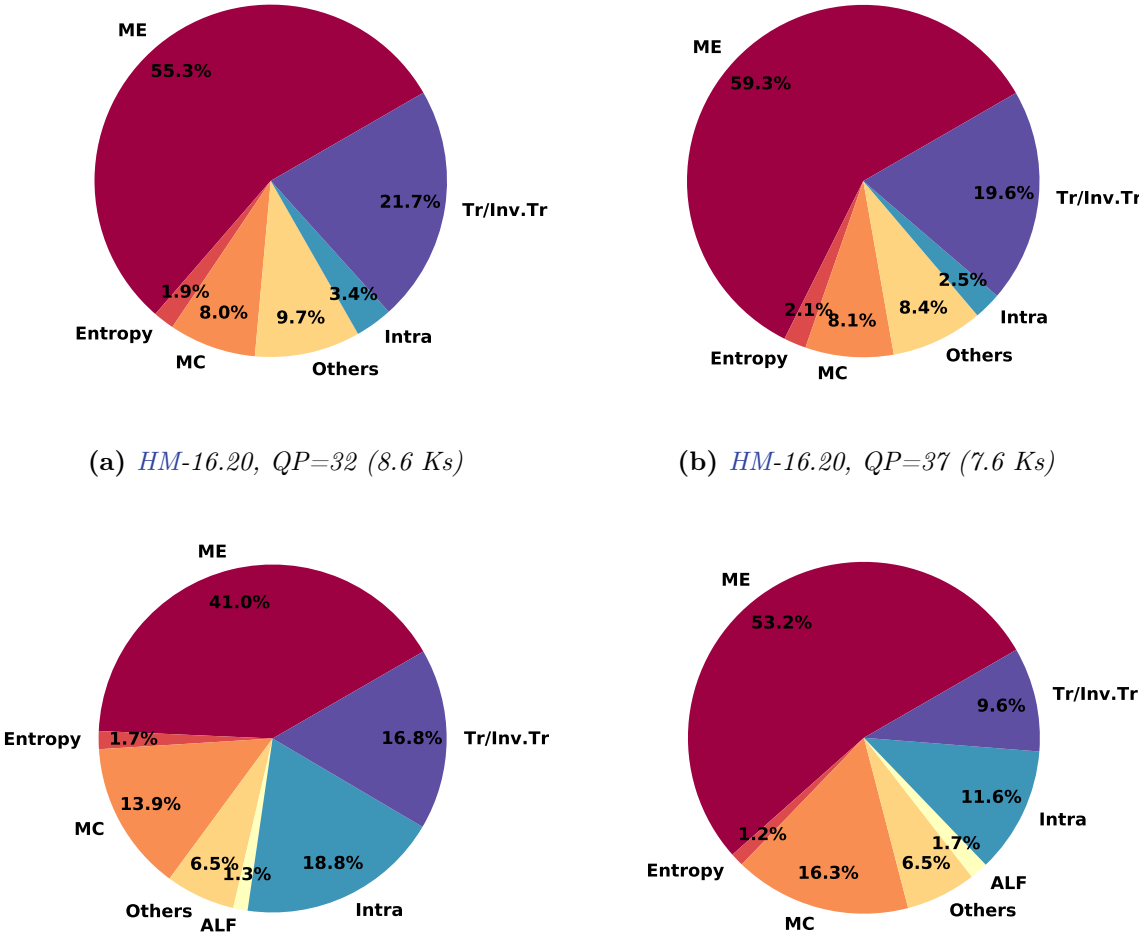
3.3.2.2 Encoding Complexity Repartition

As mentioned in Section 3.3.1, the encoding complexity repartition is obtained with Callgrind profiling tool. The encoders of both HM16.20 and VTM-5.0 have a significant portion of its complexity located in code cycles, since functions call each other in a recursive manner during the partitioning. However, the graphical user interface for Callgrind data visualization, named KCachegrind, contains a cycle detection module that allows an easy profiling of encoding complexity.

Figure 3.6 displays under the form of pie charts the encoding complexity repartition (in %), for UHD sequence CrowdRun, according to the reference software and QP value. The inter prediction stage is divided into 2 sub stages: Motion Estimation (ME) and Motion Compensation (MC). ME establishes the list of Motion Vector (MV) predictor candidates using block matching techniques and then infers the best MV predictor from this list. MC produces the prediction residuals by subtracting the inter predictions from the original Coding Units (CUs) and applying interpolation filters. The *Tr/Inv.Tr* stage sums the complexities induced by both Transform and Inverse Transform stages. The stages representing less than 1% of encoding complexity are gathered in the “Other" category.

In RA configurations, Intra prediction and Inter prediction are in competition for Prediction Units (PUs) of P and B frames. As expected, Figure 3.6 shows that Inter prediction is heavily predominant compared to Intra prediction for both HM-16.20 and VTM-5.0. The percentage is even more unbalanced between Intra and Inter prediction when QP value increases. Indeed, Intra prediction percentage decreases from 3.4% and 18.8%, at QP=32, to 2.5% and 11.6%, at QP=37, while ME percentage raises from 55% and 41%, at QP=32, to 59% and 53%, at QP=37, for HM-16.20 and VTM-5.0, respectively. Therefore Figure 3.6 highlights that the higher the QP value, the more predominantly the encoder elects Inter prediction compared to Intra prediction.

In Figure 3.6, substantial differences are observable between HM-16.20 and VTM-5.0 complexity repartitions, especially for Intra Prediction and MC stages. The Intra Prediction percentage is increased by almost a factor 5 in VTM-5.0 compared to HM-16.20. This increase is caused by the Intra prediction tools added in VTM-5.0 and mentioned in Section 2.5.3, such as Multi-reference line prediction, Intra sub-partitioning and Matrix based intra prediction. The Bi-directional optical flow, Decoder-side Motion Vector Refinement (DMVR) and Affine motion model tools, also mentioned in Section 2.5.4, have been introduced in the VTM-5.0 and are applied during the MC stage. These new tools



(a) *HM-16.20, QP=32 (8.6 Ks)*

(b) *HM-16.20, QP=37 (7.6 Ks)*

(c) *VTM-5.0, QP=32 (179 Ks)*

(d) *VTM-5.0, QP=37 (124 Ks)*

Figure 3.6 – *Encoding complexity repartition (in %), for UHD sequence CrowdRun, according to the reference software and QP value.*

are responsible for 1.0%, 1.4% and 2.9% of total VTM-5.0 complexity, respectively. Therefore they partly explain the increase in MC stage percentage in VTM-5.0 (around 15%) compared to HM-16.20 (around 8%). Finally, it is interesting to notice that the Sample Adaptive Offset (SAO) and deblocking filters do not appear in Figure 3.6 as their complexities are bellow 0.2% for both HM-16.20 and VTM-5.0. However, Adaptive Loop Filter (ALF) process must be taken into account in VTM-5.0, since its complexity represents 1.3% and 1.7% of total encoding complexity at QP=32 and QP=37, respectively.

3.3.3 Decoders Analysis

3.3.3.1 Decoding Time

The decoding process interprets the encoded symbols of a bitstream compliant with the standard specification. It is therefore not burdened with the complex RDO performed during encoding process that includes among others partitioning, intra mode decision or

Table 3.10 – *Factor between decoding time and real time, for both VTM-5.0 and HM-16.20 in RA configuration, according to the test sequence and QP value.*

	HD				UHD			
Decoder	**HM-16.20**		**VTM-5.0**		**HM-16.20**		**VTM-5.0**	
QP	**27**	**32**	**27**	**32**	**32**	**37**	**32**	**37**
AerialCrowd2	2.8	2.3	8.8	7.8	8.4	7.3	27.9	25.7
CatRobot1	4.1	3.6	13.1	11.1	13.1	12.1	44.4	35.5
CrowdRun	5.4	4.4	15.4	13.1	14.2	12.1	49.0	39.6
DaylightRoad	4.5	4.0	15.1	13.8	13.5	12.6	54.2	42.3
Drums2	4.3	3.7	12.1	10.8	12.0	11.3	37.1	31.0
HorseJumping	3.1	2.7	10.8	8.0	10.8	10.0	32.5	28.4
Sedof	4.9	4.1	15.7	14.7	13.9	12.2	52.9	50.7
Mean	**4.1**	**3.5**	**13.0**	**11.3**	**12.3**	**11.1**	**41.4**	**36.1**
Std Dev	**0.9**	**0.7**	**2.5**	**2.4**	**1.9**	**1.7**	**8.4**	**8.0**

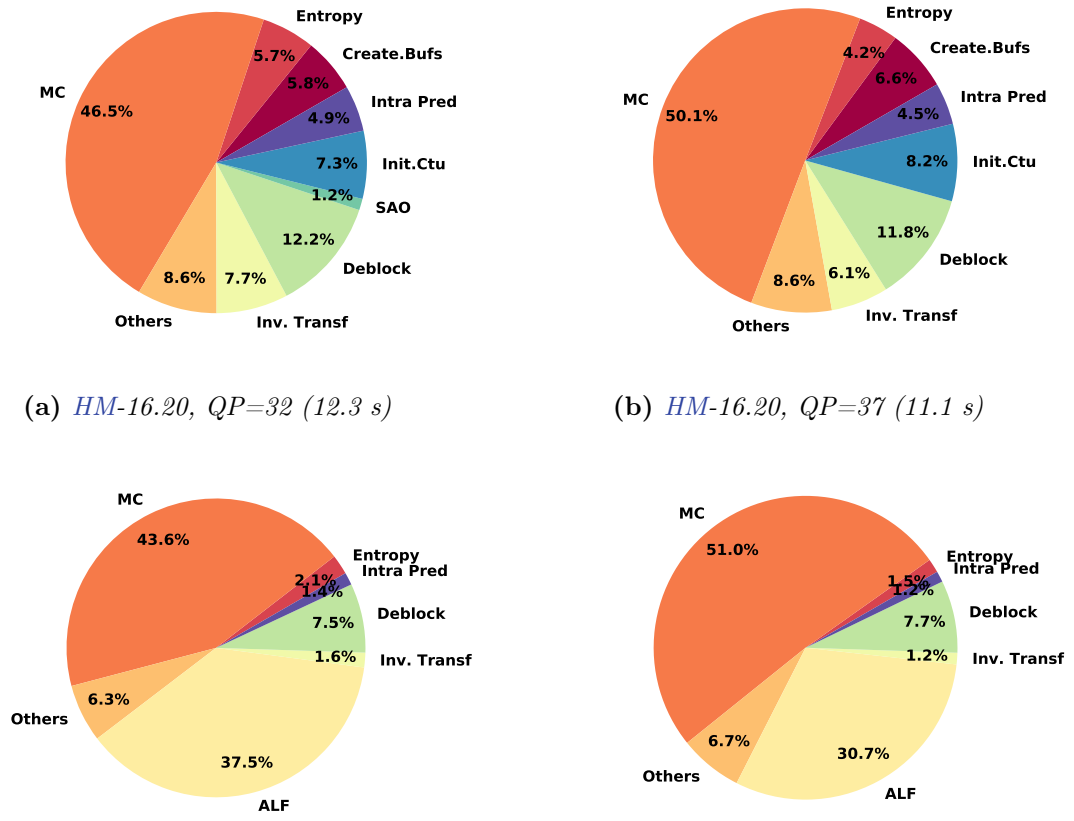
motion estimation. For this reason, comparatively to encoding time values presented in TABLE 3.9, the decoding times shown in TABLE 3.10 are in average 500 and 2000 times lower compared to encoding times for HM-16.20 and VTM-5.0, respectively.

It is interesting to note that the decoding time with VTM-5.0 is almost 3 times greater compared to the decoding time with HM-16.20. This decoding complexity increase is partly imputable to the ALF filter introduced in VTM-5.0, which complexity will be further discussed in Section 3.3.3.2. Moreover, regardless of the resolution or test sequence, the decoding time decreases with the QP value in TABLE 3.10 for both HM-16.20 and VTM-5.0. Indeed, for a given sequence, a lower QP value implies a larger bitstream outputted by the encoder, and therefore more symbols for the decoder to interpret especially at the level of by the CABAC engine.

As mentioned in Section 3.3.1, SIMD optimizations have been disabled at VTM-5.0 decoder side. Enabling the SIMD optimizations speeds up the VTM-5.0 decoder by 2.0 times in average for both HD and UHD sequences. Therefore, by enabling SIMD optimizations, the VTM-5.0 decoding time is only 1.5 times greater than HM-16.20 in average. For HEVC standard, real-time decoding has been reached by adding hardware optimization, implementing assembly written functions and/or enabling high level parallelism. For instance, authors in [68] describe a real-time SHVC decoder based on the OpenHEVC open source decoder, and paper [69] presents a real-time decoder for HEVC standard relying on SIMD optimizations and frame-level parallelism. In order to achieve real-time decoding for VVC standard, similar efforts must be extended and improved in future years, since we have shown that decoding process is 3 times more complex for VVC standard compared to HEVC standard.

3.3.3.2 Decoding Complexity Repartition

As for the encoder in Section 3.3.2.2, the decoding complexity repartition is obtained by running the decoder with Callgrind [61]. Figure 3.7 displays the decoding complexity repartition (in %), averaged across UHD test sequences, according to the reference software and QP value. The Create Bufs. stage consists in the creation of global buffers at picture



(a) *HM-16.20, QP=32 (12.3 s)* **(b)** *HM-16.20, QP=37 (11.1 s)*

(c) *VTM-5.0, QP=32 (41.4 s)* **(d)** *VTM-5.0, QP=37 (36.1 s)*

Figure 3.7 – *Decoding complexity repartition (in %), averaged across UHD test sequences, according to the reference software and QP value.*

level, while Init Ctu creates internal buffers and sets initial values before encoding the Coding Tree Unit (CTU). The sum of these stages is greater than 13% in HM-16.20, but is negligible in VTM-5.0

The first noticeable fact among the presented pie charts is the predominance of MC, ranging from 43.6% to 51.0% in both HM-16.20 and VTM-5.0. The predominance of Inter predicted frames in RA configuration explains these numbers, and also explains the limited portion of Intra prediction in the pie charts. In VTM-5.0, an important share of MC complexity lies in the DMVR, that refines the motion vectors at decoder side, as mentioned in Section 2.5.4. In fact, the share of DMVR in VTM-5.0 global decoding complexity is in average 23% with QP=32 and 32% with QP=37, which represents almost 60% of MC stage complexity. However, the DMVR is not significant burden as the previous percentages suggest for the decoding process. Indeed, when it is disabled, it is replaced to a large extent by usual MV Inter prediction, which decoding complexity approaches DMVR complexity. The contribution [39] confirms this behaviour, since it announces that VTM-5.0 decoding complexity is only 4% lower when DMVR is disabled compared to original VTM-5.0 in RA configurations. The deblocking filtering stage covers a consistent portion of the total decoding complexity: around 7.5% for VTM-5.0 and around 12% for HM-16.20. In VTM-5.0, the in-loop filtering percentage is heavily increased by additional ALF tool, which is

responsible for around a third of total decoding complexity. As mentioned in Section 2.5.7, the ALF provides a significant improvement in encoding efficiency. It does not overload the VTM-5.0 encoding process by more than 2% of total complexity (see Figure 3.6). However, in counterpart for the aforementioned benefits, ALF represents a considerable burden for the decoding process. It is worth mentioning that ALF percentage decreases to less than 20% when the SIMD optimizations are enabled.

3.4 Conclusion

In this Chapter, both quality enhancement and computational complexity of VVC have been investigated. We studied bit-rate savings between VTM-5.0 and the HM-16.20 softwares as well as their respective encoding/decoding complexity repartitions over a typical classification of hybrid decoding tools. As results, the VTM coding tools enable a significant quality improvement compared to the HM reference software, for different videos sequences used in the experiment. Using PSNR, SSIM and VMAF, as quality metrics, the obtained gain is ranging between 31% to 40% depending on video format. Subjectively, the VTM codec outperforms, in a significant manner, the HEVC reference software, especially for the low bit-rate. Moreover, for various sequences, we notice that VVC codec enables the same perceived visual quality as HEVC with a bit-rate reduction of 50% is obtained.

Nevertheless, a consequent computational complexity increase between HM and VTM is observed. The complexity increase of the VTM process is not consistent, since it depends among others on QP value, frame-rate, spatial texture and movement. At the encoder side, the simultaneous profiling of both HM and VTM has identified the additional intra modes and the block partitioning scheme as mainly responsible for the complexity increase in VTM. Based on these observations, in next Chapter we propose a complexity reduction solution for VVC encoding process that reduces the search space of block partitioning scheme in RA coding configuration. At the decoder side, the ALF and DMVR represent the main additional burdens in VTM compared to HEVC decoding processes. For this reason, in the last Chapter of this thesis, special attention is given to ALF implementation in a real-time VVC decoder.

CHAPTER 4

Tunable VVC Block Partitioning based on Lightweight Machine Learning

4.1 Introduction

The complexity increase of Versatile Video Coding (VVC) encoding process may interfere with its deployment especially on embedded platforms and live applications. In previous Chapter 3.3.2.1, we identified the additional intra modes and the new block partitioning scheme as mainly responsible for the complexity increase of VVC. To reduce the computational complexity of High Efficiency Video Coding (HEVC) and VVC, several techniques propose to reduce the tested intra mode candidates. These techniques use features such as gradients of luminance samples [70, 71, 65] or Machine Learning (ML) techniques [66] to predict a reduced set of likely intra modes. Many other techniques reduce the complexity of the encoding process by focusing on the partitioning scheme and testing a reduced number of block partitioning configurations. This is the approach chosen in this Chapter and a detailed review of related works is provided in Section 5.2.

The contribution presented in this Chapter has been proposed in the early standardization phase of VVC. As mentioned in Section 2.6, at that time the reference software used by the Joint Video Exploration Team (JVET) to investigate new coding tools was called the Joint Exploration Model (JEM) [45][46]. For block partitioning, the JEM featured the Quad Tree Binary Tree (QTBT) partitioning scheme, that is a simplified version of the Multi-Type Tree (MTT) partitioning scheme presented in Section 4.2.1. The contribution presented in this Chapter has originally been designed to reduce the encoding complexity of the QTBT partitioning scheme in the JEM and as further been adapted to the MTT partitioning scheme in the VVC Test Model (VTM).

In this Chapter, we reduce the computational complexity of JEM encoding process in Random Access (RA) coding configuration. A reduced number of QTBT partitioning configurations is tested, using a tunable ML solution based on Random Forest (RF) classifiers. The proposed solution explores a novel approach to solve a 4 classes classification problem inherent to QTBT partition scheme, which has not been fully studied in related works. The decision of QTBT partition mode for the Coding Unit (CU) is modeled as three distinct binary classification problems. Thus, three binary RF classifiers are trained independently off-line, with separate training for each CU size. The goal of the classifiers is to skip expensive exploration of the partition modes classified as unlikely. Furthermore, the classifiers take as input only features from the current CU, making the solution parallel-

friendly. To limit the Rate Distorsion (RD) loss induced by misclassification, risk intervals are introduced to control the classifier decisions. When the classifier decision falls into the risk interval, all possible partitioning modes are processed. The risk intervals are set by the encoder based on the encoding of a reference frame, adapting the RD loss induced by misclassification to the encoded content. By varying the size of risk intervals,tunable complexity reduction is achieved.

To the best of our knowledge, the proposed solution is the first tunable complexity reduction solution offering tunable cababilities applied on an encoder post HEVC. It includes various Complexity Reduction Configurations (CRCs), each offering a new trade-off between complexity reduction and Bjøntegaard Delta BitRate (BD-BR) increase. In JEM-7.0 software, encoding complexity reductions vary from 30% to 70% in average at the expense of only 0.7% to 3.0% BD-BR increase. The proposed solution based on RF classifiers is also efficient to reduce the complexity of the MTT partitioning scheme in the VTM-5.0 software, with complexity reductions varying from 25% to 61% in average for limited BD-BR increase of 0.4% to 2.2%. Moreover, the proposed solution induces a very low overhead between 0.2% and 1.8% of the encoding time according to the video content, which is a key point to adopt this solution in a real-time and embedded framework.

The rest of the Chapter is organized as follows. Section 5.2 describes the frame partitioning decision in JEM, and then reviews the related works. Section 4.3 goes through background of RF classifiers and presents the proposed classification problem. Section 4.4 depicts the training dataset. The training process to build RF classifiers is described in Section 4.5. Section 4.6 details how tunable encoding complexity reduction is achieved using various configurations of the risk intervals. Experimental results are presented and analyzed for both JEM-7.0 and VTM-5.0 in Section 5.4. Finally, Section 5.5 concludes this Chapter.

4.2 Related Works

4.2.1 Overview of QTBT Partitioning Scheme JEM

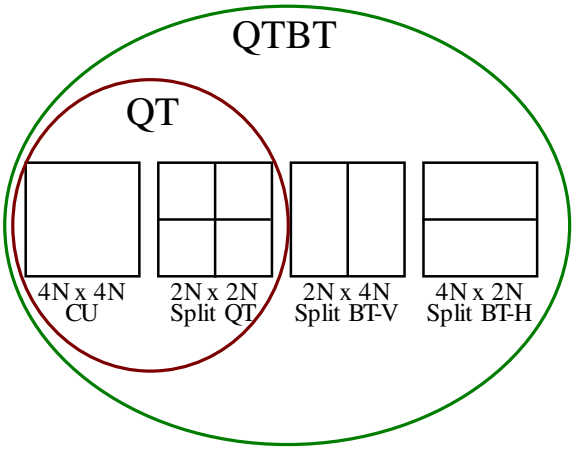
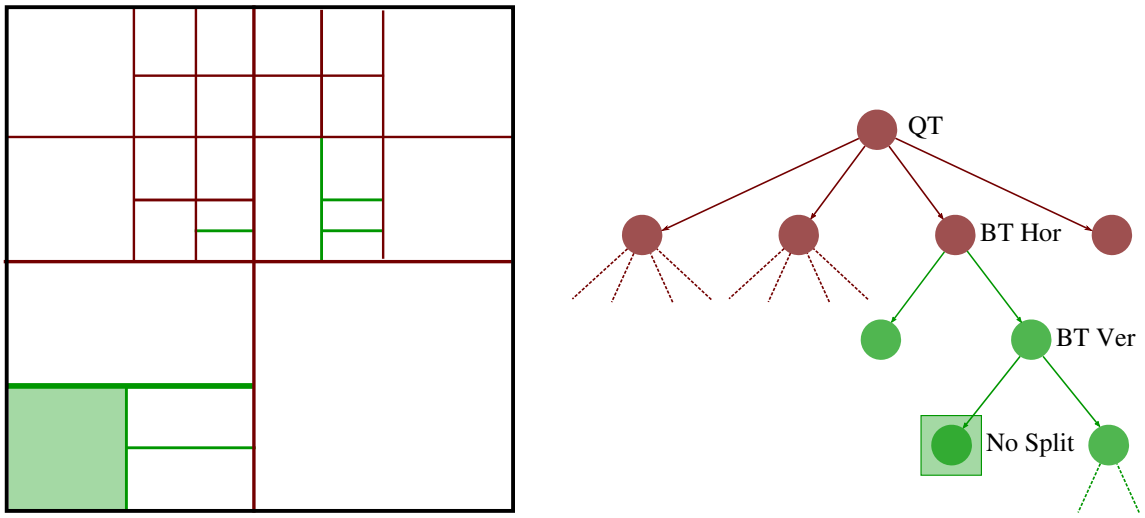


Figure 4.1 – *Available partition modes in QTBT partition scheme for a* $4N \times 4N$ *CU.*

To overcome the limitations of block partitioning scheme in HEVC (see Section 4.2.1), a QTBT partitioning scheme has been proposed in the JEM [72]. QTBT is an extension to the QT partitioning that enables symmetric binary partition modes CUs in both horizontal (Binary Tree Horizontal (BTH)) and vertical (Binary Tree Vertical (BTV)) directions, as



(a) *QTBT partition of a Coding Tree Unit (CTU).*

(b) *QTBT partition tree. Leaf boxed in green corresponds to CU filled with green in Figure 4.2a.*

Figure 4.2 – *QTBT partition scheme in JEM. In red Quad Tree (QT) partition mode and in green Binary Tree (BT) partition modes.*

shown in Figure 4.1. When BT partition mode is used in a CU, QT partition mode is no longer allowed. In the JVET Common Test Conditions (CTC) [73], BT_{depth} parameter is set to 3, enabling only 3 successive BT partitions. Figure 4.2b shows the tree representation of QTBT partition, where the leaf surrounded by a green square corresponds to the green CU of Figure 4.2a.

Unlike in HEVC, the Transform Unit (TU) and Prediction Unit (PU) partitioning is discarded in the JEM, and both prediction and transform are performed directly on the CU. Moreover, different partition trees can be applied on Luma and Chroma components for intra predicted slices. For inter predicted slices, the partition trees of Luma are used for Chroma.

4.2.2 Complexity Reduction of Block Partitioning

QTBT partitioning scheme has not yet been fully studied from the perspective of complexity reduction. Reason why in this section, the techniques proposed to speed up the QT partitioning scheme in HEVC are first described. Subsequently, papers that have investigated the complexity reduction of QTBT partitioning scheme are presented.

4.2.2.1 Partitioning Scheme in HEVC

The techniques proposed to speed up the QT partitioning scheme in HEVC are divided into four categories, whether they involve: intermediate encoding information, texture characteristics, motion divergence, and ML.

Intermediate encoding information. Techniques of the first category are based on intermediate information computed during the encoding process such as depths of previously encoded blocks, encoding flags or RD cost of PU modes. *Pan et al.* [74] use the motion estimation and all-zero block detection informations of $M \times M$ PU mode (Figure 2.11a), coupled with merge mode informations of previous depths to determine if early merge can

save computational time. A threshold based on Sum of Absolute Transform Differences (SATD) of $M \times M$ PU mode has been proposed in [75]. The technique ignores the unlikely remaining PU modes or early terminating QT partitioning mode. In [76], *Correa et al.* figure out that some QT depths tend to be used in co-located areas of adjacent frames and exploit this correlation to target a computational complexity reduction. Same authors adapt this principle to speedup the coding of motion sequences in [77] by using a motion compensated area as reference in previously encoded frame (instead of co-located area) and by using depths of spatial neighboring CTUs. In [78], a thresholding process applied on RD cost of $M \times M$ merge mode determines if the skip mode shall be used for the PU. These techniques [75] [77] [78] reduce the encoding complexity between 30% and 50% with a BD-BR increase in the range 0.4% to 1.4%.

Texture characteristics. The second category covers techniques exploiting the correlation between spatial information of a sequence, also called texture, and the QT partitioning scheme. Usually, areas having complex textures are split into small blocks in order to fit better its local variations. Authors in [79] apply thresholds on local and global edge filters of luminance samples in 4 directions $(0°, 45°, 90°, 135°)$ to determine if a CU shall be split, non-split or undetermined. A similar principle is applied in [80], where the authors compute adaptive thresholds based on texture homogeneity for early determination of QT depth of CTUs. *Mercat et al.* propose in [81] a technique that determines at each step if the 4 smaller CUs shall be merged into one CU. This bottom-up approach exploits correlation between the QT partitioning scheme and texture variance.

Motion divergence. The techniques of the third category explore the correlation between motion divergence in a frame and the QT partitioning scheme. Areas of the frame with continuous motion tend to be split into larger blocks. Reason why in [82], authors detect motion continuity by applying Sobel operators to a pre-computed optical flow and use this information to predict block size and save computational time. In [83], a score called Pyramid Motion Divergence (PMD) based on variance of Motion Divergence Field is assigned to every CU. The authors show that CUs with similar PMD tend to be encoded with the same partitioning. *Blasi et al.* [84] propose a bottom-up approach to ignore certain CU partition modes based on a motion vector variance distance, computed on the four QT sub-CUs.

Machine Learning. The techniques of last category use ML approaches to reduce encoding complexity. They either take as input features extracted from one or more of previous categories, or rely on Deep Learning (DL) approaches such as Convolution Neural Networks (CNNs). *Liu et al.* [85] propose a technique based on CNN for All Intra (AI) configuration that determine if a CU must be early terminated or early split. A CNN is separately trained for every QT CU size from 32x32 to 8x8 pixels. In [86], *Duanmu et al.* focus on Screen Coding Content (SCC), an extension of HEVC that targets typical screen contents. Separate CNN classifiers are separately trained for different Quantization Parameter (QP) values and CU sizes to output a variable between 0 and 1, representing the probability that a CU is early terminated. In [87], to reduce RA encoding complexity in HEVC, Support Vector Machines (SVM) classifiers are separately trained for every CU size in order to determine if a CU should be early terminated. The SVM classifier takes as an input features such as pixel gradients, sub-CUs Motion Vectors (MVs) and intermediate encoding information (encoding flags, CU depth of neighboring blocks and the RD cost). The technique proposed by *Correa et al.* [84] is based on decision trees, each relying on

spatial features and intermediate RA configuration encoding information. The decision trees are trained separately for every CU size from 64x64 to 8x8. In [88], *Shen et al.* propose a Bayesian rule classifier employing features of $M \times M$ PU mode such as SATD, RD cost and MV to determine the PU mode of a CU. Finally, *Mercat et al.* [89] compares features used in state of the art techniques to predict HEVC partition, considering both information gain and computational complexity.

4.2.2.2 QTBT Partitioning Scheme

The QTBT partitioning scheme in the JEM allows two more partition modes compared to QT partitioning scheme, considerably increasing the encoding complexity. Furthermore, intermediate PU splitting modes, that provides useful information for early termination or early skip decision in HEVC, have been removed in the JEM. For these reasons, most of previously mentioned techniques can not be directly used to speedup the JEM encoder.

In [90], *Yamamoto* proposes to reduce the encoding complexity in RA configuration by setting high value of BT_{depth} on frames with low Temporal ID, whereas frames with high Temporal ID use smaller value of BT_{depth}. In the QTBT partition scheme the same CU can be generated by different block partition choices. *Huang et al.* [91] reduce the encoding complexity by re-using the encoder decisions of the same CU explored in previous partition choices. The technique proposed by *Lin et al.* [92] skips the BT Rate Distorsion Optimization (RDO) process of the second sub-CU, when the RD cost of the parent CU and the first sub CU fulfil certain constraints. Authors in [93] and [94], use CNNs to predict a depth description of QTBT partition of the CTUs. In [93], the CNN takes as an input the 32×32 pixels blocks of the frame, as well as QP value, and outputs a class from 0 to 5 describing QTBT partition depth for AI configuration. In [94], the false prediction risk of CNN is controlled based on temporal correlation for RA configuration. *Wang et al* [95] use a combination between Motion Divergence Field and gradient of luminance samples to model the RD cost of a CU. A probabilistic model is then proposed to determine unlikely partition modes of the QTBT partitioning scheme. Complexity reduction techniques [90], [94] and [95] can reduce encoding complexity in RA configuration by 17%, 32% and 52% for 0.5%, 0.5% and 1.4% of BD-BR increase in average, respectively. These results depend on the encoder version of the used software, encoding parameters and hardware configuration. They nevertheless provide an order of magnitude of the techniques efficiencies.

Even though RF is a classical method in ML that offers high classification performance with slight overhead and is widely used in many applications such as image classification [96] and 3D pose estimation [97], none of the previously mentioned QT or QTBT fast partitioning techniques rely on RF. Furthermore, previously mentioned techniques speeding up the QTBT partition scheme, the proposed solution is tunable and offers various complexity reduction opportunities, from 30% to 70% in average. Moreover, this solution can be considered as lightweight since it relies on RF classifiers, inducing much smaller overhead compared to CNN based techniques [93] [94].

4.3 Random Forests for Partition Decision Classification Problem

This section introduces RF classifiers and presents the three binary classification problems proposed in this work to reduce the number of processed partition modes.

4.3.1 Background for Random Forests

Classification by RF [9] is a classical method in ML. RF classifiers predict the value of a target variable, named class, from values of several input variables, named features. They bag many single little-correlated decision trees and gather the results from all the trees to make the final decision.

4.3.1.1 Build Decision Trees

Decision trees are constructed by a recursive partitioning of the data set into subsets called nodes. At each node, a threshold that achieves optimal separation of the classes is selected among the input features. Each child-node corresponds to a set of values of the selected input features, so that the totality of child-nodes cover all possible values of this input features.

The criterion used in this work to select the best split for a node in decision trees is Mutual Information (MI). The differential entropy $\mathbb{H}$ of a continuous random variable X is computed as follows:

$$\mathbb{H}(X) = -\int_{\mathbb{R}} f_X(x) \log_2 \left(f_X(x)\right) \mathrm{d}\, x, \tag{4.1}$$

where $f_X : \mathbb{R} \to [0, +\infty]$ is the Probability Density Function (Pdf) of X. The entropy $\mathbb{H}$ measures the quantity of information delivered by the knowledge of X. MI of class C and feature F, noted $\mathbb{I}(F, C)$, is defined as the entropy decreasing of C when F is known [98]. The value of $\mathbb{I}(F, C)$ is comprised between 0 and $\mathbb{H}(C)$, and is expressed by Equation (4.2)

$$\mathbb{I}(F, C) = \mathbb{H}(C) - \mathbb{H}(C|F). \tag{4.2}$$

In other words, $\mathbb{I}(F, C)$ measures the information shared by C and F. Therefore, the higher $\mathbb{I}(F, C)$, the more the feature F is relevant to estimate class C. When MI is used as a criterion, the optimal threshold of a feature F to split a decision tree node is the threshold that maximizes $\mathbb{I}(F, C)$ on both subsets of child-nodes.

4.3.1.2 Benefits of Random Forests

Let the error rate be the percentage of wrong classification on the training dataset. By de-correlating trees, RF classifiers achieve a better trade-off between error rate and training data over-fitting compared to a single decision tree classifier.

In order to de-correlate the decision trees of the RF, a random subset of the training dataset is selected to build each decision tree. The decision trees in the RF take as an input all the features. However, the splitting threshold of each node in the decision tree is selected among a random subset of the input features. This random selection decreases the probability that two decision trees in the RF select the same set of features in the same order, and therefore is crucial to de-correlate one decision tree from another. The following example illustrates the interest of RF classifiers in term of error rate compared to single decision tree classifiers. Assume the RF classifier is composed of 10 perfectly de-correlated decision trees, each with error rate ϵ_i of 0.3: $\epsilon_i = \epsilon = 0.3, \forall i \in \{1, \ldots, 10\}$.
If a RF classifier makes a wrong prediction when more than half of the base decision tree classifiers are wrong, the error rate ϵ_{RF} of the RF classifier is computed by Equation (4.3)

$$\epsilon_{RF} = \sum_{j=6}^{10} \binom{10}{j} \epsilon^j (1-\epsilon)^{10-j} \approx 0.05. \tag{4.3}$$

It is not possible to de-correlate perfectly the decision trees of the RF as they are trained on the same data. Nonetheless, the more the decision trees are de-correlated, the closer the error rate of the RF classifier is to ϵ_{RF}.

4.3.2 Classification Problem

To find the CU partition mode that achieves the best RD performance, the encoder recursively explores all possible partition modes. This process is called full or exhaustive RDO search. For each CU, the encoder computes the RD cost of the whole CU, QT and BT partition modes, BT partition modes being composed of BTH and BTV partition modes. The encoder selects the partition mode that minimizes the RD cost J, expressed as a trade-off between distortion D and rate R, with λ the Lagrangian multiplier:

$$J = D + \lambda R. \tag{4.4}$$

The aim of the proposed solution is to predict for every encountered CU the partition mode that minimizes the RD cost. Several partition modes are ignored reducing the number of processed partition modes. As shown in Figure 4.3, the problem to solve is a four classes classification problem including the following partition modes: NoSplit, QT, BTH and BTV.

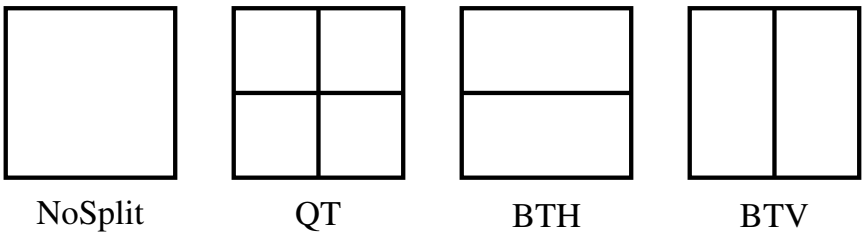


Figure 4.3 – *Four classes of the CU classification problem.*

In this work, instead of creating a RF classifier that solves directly the four classes classification problem, the partition decision is divided into three successive binary classification problems. This division adds flexibility into the decision structure and allows a separate training of the classifiers on specific features, improving global classification performance.

The three binary classifiers are named *S-NS*, *QT-BT* and *BH-BV*. As shown in Figure 4.4, each classifier takes as input a different set of features and the classifiers are used in the following order:

1. **Classifier *S-NS***: The two output classes of classifier *S-NS* are either NoSplit partition mode or Split partition modes, where Split partition modes include QT, BTH and BTV partition modes. When the Classifier *S-NS* outputs class NoSplit, NoSplit partition mode is processed and Split partition modes are ignored. Otherwise, NoSplit partition mode is ignored, and the second classifier *QT-BT* is requested.

2. **Classifier *QT-BT***: The output classes of classifier *QT-BT* are either QT partition mode or BT partition modes, where BT partition modes include BTH and BTV partition modes. When the classifier *QT-BT* outputs QT partition mode, QT partition mode is processed and BT partition modes are ignored. Otherwise, QT partition mode is ignored, and the third classifier *BH-BV* is requested.

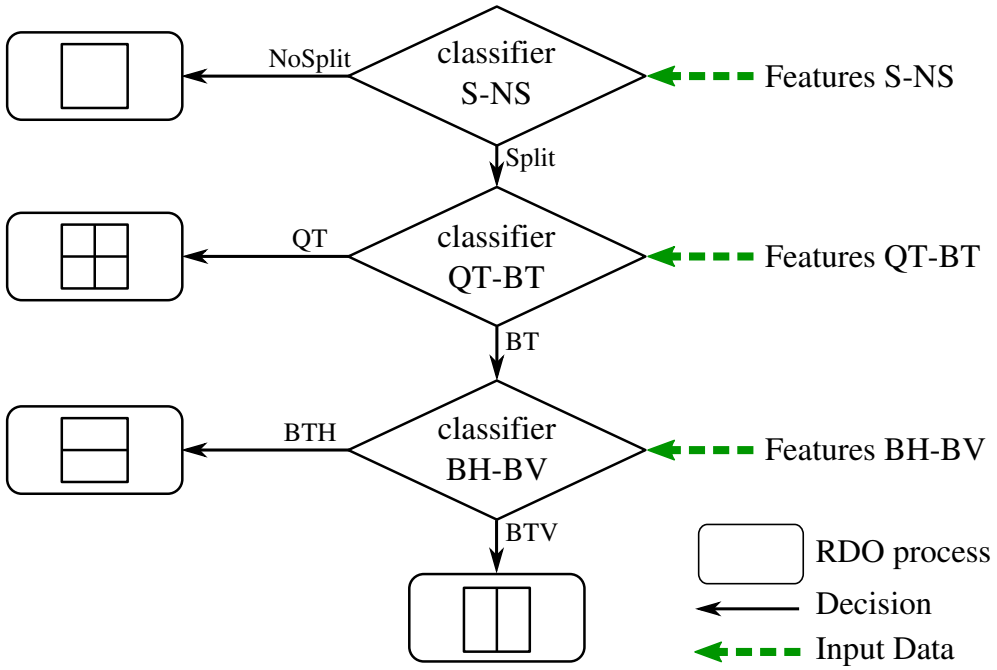


Figure 4.4 – *Convert Four Classes Problem into Three Binary Problems*

3. **Classifier *BH-BV***: The output classes of classifier *BH-BV* are either BTH partition mode or BTV partition mode. When the classifier *BH-BV* outputs BTH partition mode, BTH partition mode is processed and BTV partition mode is ignored. Otherwise, BTV partition mode is processed and BTH partition mode is ignored.

4.4 Dataset Creation

Let us define a training instance as the entity composed of the chosen set of input features and the associated output class. This section details the creation of the training dataset, i.e. the set that contains all the training instances used to train the RF classifiers.

4.4.1 Training Setup

The effectiveness of ML is highly linked to the diversity and relevance of the training dataset. To characterize a video content, Spatial Information (SI) and Temporal Information (TI) metrics are used [56]. The SI estimates the amount of spatial details whereas the TI measures the quantity of motion in the sequence. In Figure 4.5, 25 sequences extracted from JVET CTC [73] are represented under the SI TI coordinates.

To cover a wide range of these two content types, the training dataset is extracted from 10 training sequences spanning a large range of SI and TI space and distributed across 6 classes (A1, A2, B, C, D, E). The training sequences are circled in black in Figure 4.5 including: *DaylightRoad* and *CatRobot1* (class A1), *Traffic* (class A2), *BasketballDrive* and *BQTerrace* (class B), *Flowervase* and *BQMall* (class C), *BQSquare* and *Keiba* (class D), *Johnny* (class E). The training instances are extracted from encodings carried-out with the JEM-7.0 in RA configuration across the 4 QP values used in CTC: 22, 27, 32 and 37. The corresponding output class of a CU is defined as the optimal partition mode selected after a exhaustive RDO process.

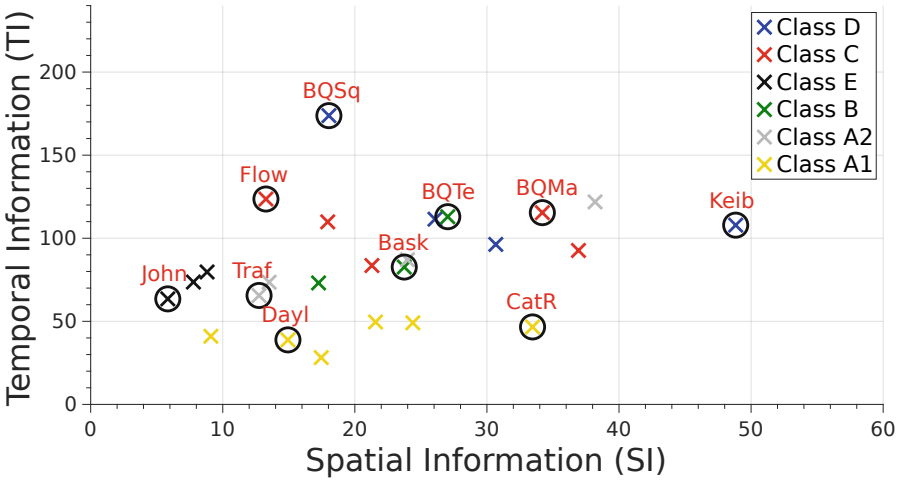


Figure 4.5 – *SI and TI of the CTC sequences according the classes.*

Let the size category of a CU depend only on the number of pixels in the CU. TABLE 4.1 gives the dimensions in width and height of the CUs composing the 7 size categories. The category S_0 corresponds to the largest CUs and the category S_6 to the smallest CUs. For each CU size category, a separate training dataset is created and a separate RF classifier is trained. With this separation, the features are computed on the same number of pixels in each training dataset.

Table 4.1 – *Width and Height of the CUs composing the size categories.*

S_0	S_1	S_2	S_3	S_4	S_5	S_6
128x128	128x64	64x64	64x32	32x32	32x16	16x16
	64x128	128x32	32x64	64x16	16x32	32x8
		32x128	128x16	16x64	64x8	8x32
			16x128	128x8	8x64	
				8x128		

Sequences with high resolution and high frame rate provide more CTUs to the training datasets compared to the low resolution and low frame rate sequences. To avoid being biased by these particular training sequences, the datasets used for training are composed of a fixed number of CTUs by training sequence. To ensure a fixed number of CTUs by training sequence, the number of frames used for training in a sequence differs according to the sequence class: 7 frames for class A1, 13 frames for class A2, 25 frames for class B, 55 frames for class E, 125 frames for class C and 500 frames for class D. To avoid temporal bias, the CTUs come from frames evenly distributed in their sequence time-line. Furthermore, to reduce the problem of imbalanced data, the training datasets are composed randomly with the same amount of training instances classified into each output class.

4.4.2 Feature Evaluation and Selection

Let a feature be a property of the CU used to determine which output class shall be selected by the RF classifiers. In related works, features are extracted among others from intermediate encoding information [88], texture of pixel luminance samples [86] or mo-

tion divergence [95], as mentioned in Section 5.2. Features based on intermediate encoding information have been shown to be effective for the *S-NS* classification problem in HEVC [24][28]. In this work, we choose to build a parallel friendly set of features as an input of the RF classifiers, since intense parallelization will be compulsory to achieve real time encodings for VVC standard. The features must not add dependencies between regions of the frame, in order to preserve the opportunities of high level parallelism and CU level parallelism. Therefore, the selected features are only designed based on current CU data, such as texture of pixel luminance samples and motion divergence. This choice forces to neglect features based on intermediate encoding informations. This section first introduces Motion Divergence Field (MDF), then explains the feature evaluation and selection.

4.4.2.1 Motion Divergence Field

In the following, motion divergence in a frame is considered through the MDF. The MDF is the array of MVs of every 4x4 pixels block of the frame. The MVs point to the closest reference frame in term of temporal distance. In this work, a separate motion search process is needed to compute the MDF. Without optimization and parallelization, the motion search process of the MDF induces an average 0.8% overhead of the encoding complexity. However, many real-time encoders x265 [99] already use look-ahead techniques. A look-ahead technique consists in a pre-analysis of the video sequence, generally including a motion search on small blocks of the frame. For encoders using look-ahead techniques, the overhead to compute the MDF is null.

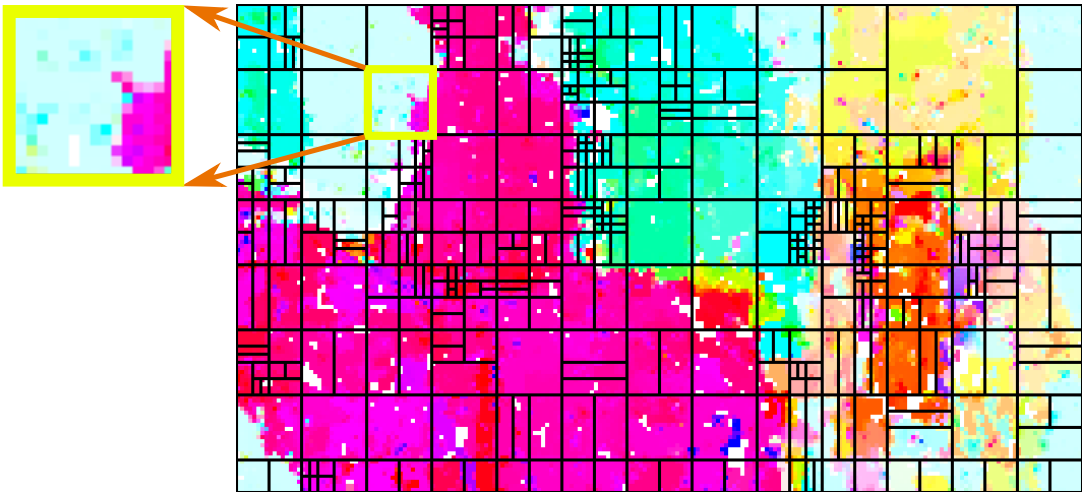
Figure 4.6a displays the original frame #9 of sequence *RaceHorses*, while Figure 4.6b gives a visual representation of its MDF. MVs with different motion directions are displayed with different colors, separating visually regions of the frame with different movement. The optimal QTBT partition selected by JEM-7.0 encoder after a exhaustive RDO process is also displayed. Areas with similar colors tend to be merged together in a CU, showing the correlation between frame's optimal QTBT partition and the MDF. Some edge-examples CUs containing distant colors are recognized, and one of them is highlighted with a yellow square. Indeed, this CU is not further split and the blue zone is not separated from the purple zone. The goal of the feature selection is to determine which features extracted from the MDF and from pixel luminance samples are the most relevant to determine the best partition mode of a CU.

4.4.2.2 Evaluated Features

Evaluated features are divided into 3 categories: features computed on **whole CU**, features computed on **sub-quarters of the CU** and features based on **inconsistency among CU sub-quarters**.
Features computed on the **whole CU** are the following:

- *QP*: Quantization parameter used to encode CU slice.
- *VarPix*: Variance of luminance samples.
- *Grad*: Gradients in horizontal ($grad_x$) and vertical ($grad_y$) directions of the luminance samples (2 features).
- *RatioGrad*: ratio of gradients $\frac{grad_x}{grad_y}$.
- *VarMv*: $\sigma^2_{\text{MVx}} + \sigma^2_{\text{MVy}}$ with σ^2_{MVx} and σ^2_{MVy} respectively variances of horizontal and vertical MVs of MDF.

(a) *Original frame.*



(b) *Visual representation of the MDF. The MVs with different motion directions are displayed with different colors*

Figure 4.6 – *Correlation between MDF and frame QTBT partition, frame #9* RaceHorses*, at QP=32.*

- *MaxDiffMv*: maximum 1-norm distance between MVs of MDF, noted mv, and their mean, noted $\overline{mv}$, as in Equation (4.5).

$$\begin{aligned} MaxDiffMv &= \max_{mv \in \text{MDF}} (||mv - \overline{mv}||_1), \\ &= \max_{mv \in \text{MDF}} (|mv_x - \overline{mv}_x| + |mv_y - \overline{mv}_y|). \end{aligned} \tag{4.5}$$

Features based on **sub-quarters of the CU** are the following:

- *QuarterVarPix*: VarPix on 4 sub-quarters (4 features).
- *QuarterVarMv*: VarMv on 4 sub-quarters (4 features).
- *QuarterMaxDiffMv*: MaxDiffMv on 4 sub-quarters (4 features).

For any feature f, f_1 is the feature computed on top-left sub-quarter, f_2 on top-right, f_3 on bottom-left and f_4 on bottom-right. Let $\delta H(f)$ and $\delta V(f)$ be Horizontal Inconsistency

(HI) and Vertical Inconsistency (VI) as defined by Equation (4.6)

$$\begin{aligned} \delta H(f) &= |f_1 - f_2| + |f_3 - f_4|, \\ \delta V(f) &= |f_1 - f_3| + |f_2 - f_4|. \end{aligned} \tag{4.6}$$

The aim of HI and VI is to highlight which rectangular parts of the CU have the highest differences. Features based on **inconsistency among sub-quarters** of the CU are the following:

- *InconsPix*: HI and VI of mean, variance and gradients-ratio of luminance samples (6 features).
- *InconsMv*: HI and VI of mean and variance of MDF (4 features).
- *DiffInconsPix*: difference between HI and VI for luminance based features (3 features).
- *DiffInconsMv*: difference between HI and VI for MDF based features (2 features).

4.4.2.3 Feature Selection

As decision trees node splitting relies on MI (see Section 4.3.1), the feature evaluation is conducted with MI as metric. Figure 4.7 gives the MI of all evaluated features according to the classifier and CU size. Only MI for CU size categories S_0, S_2, S_4 and S_6 are displayed to avoid overloading the figure as these values are representative of MI of other CU sizes.

Figure 4.7a shows that for classifier *S-NS*, the larger the CU, the higher the MI, independently of the evaluated feature. Therefore, the larger the CU, the more relevant are the evaluated features to determine whether the optimal partition mode of a CU is NoSplit or one of the Split partition modes.

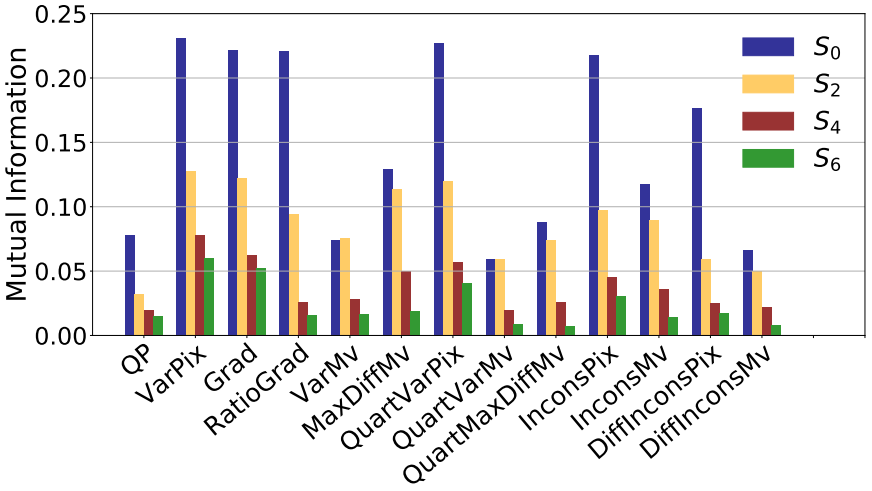
For *QT-BT* and *BH-BV* classifiers, Figure 4.7b and Figure 4.7c respectively show that features based on texture have higher MI than features based on the MDF, independently of CU size. In other words, features based on texture are more relevant than features based on the MDF to estimate the partition modes to process, independently of CU size. It can also be noted that the MI of features are lower for classifiers *QT-BT* and *BH-BV* than for classifier *S-NS*. The maximum MI reaches 0.22 for classifier *S-NS* whereas the maximum MI is 0.07 and 0.14 for classifiers *QT-BT* and *BH-BV*, respectively.

For each classifier, only one set of features is selected to create the training datasets of the various CU size categories. Let the classification rate be the percentage of correct classification given by the 4-fold cross-validation on the training dataset. The selected features are those providing the highest MI and improving the classification rate when added to the set of features.

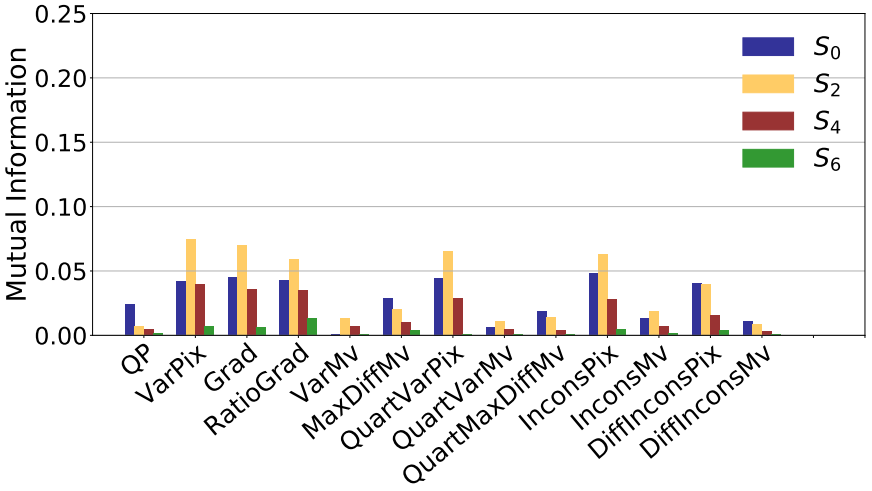
The set of features for classifier *S-NS* is composed of the 24 features: *QP*, *VarPix*, *Grad* (2 features), *RatioGrad*, *VarMv*, *MaxDiffMv*, *QuarterVarPix* (4 features), *InconsPix* (6 features), *InconsMv* (4 features), *DiffInconsPix* (3 features).

The set of features for classifier *QT-BT* is composed of the 19 features: *QP*, *VarPix*, *Grad* (2 features), *RatioGrad*, *MaxDiffMv*, *QuarterVarPix* (4 features), *InconsPix* (6 features) and *DiffInconsPix* (3 features).

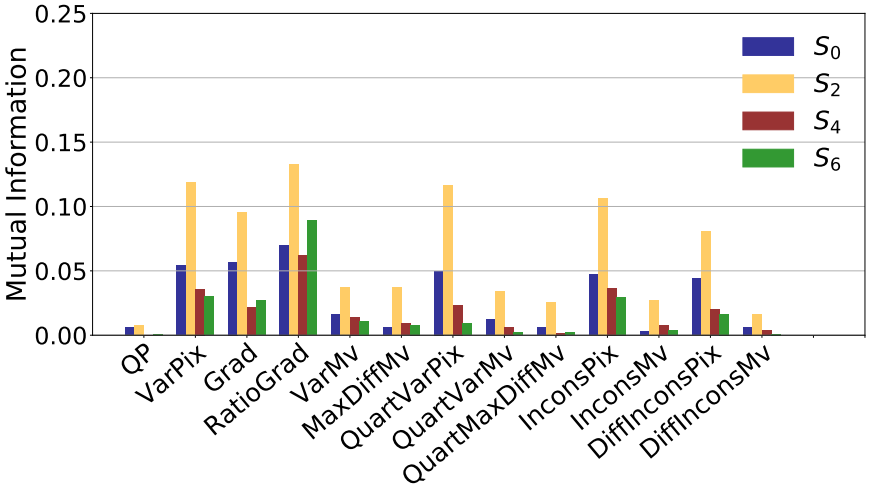
For classifier *BH-BV*, the set of features is composed of the 21 features including *VarPix*, *Grad* (2 features), *RatioGrad*, *QuarterVarPix* (4 features), *QuarterVarMv* (4 features), *InconsPix* (6 features) and *DiffInconsPix* (3 features).



(a) S-NS *Classifier*



(b) QT-BT *Classifier*



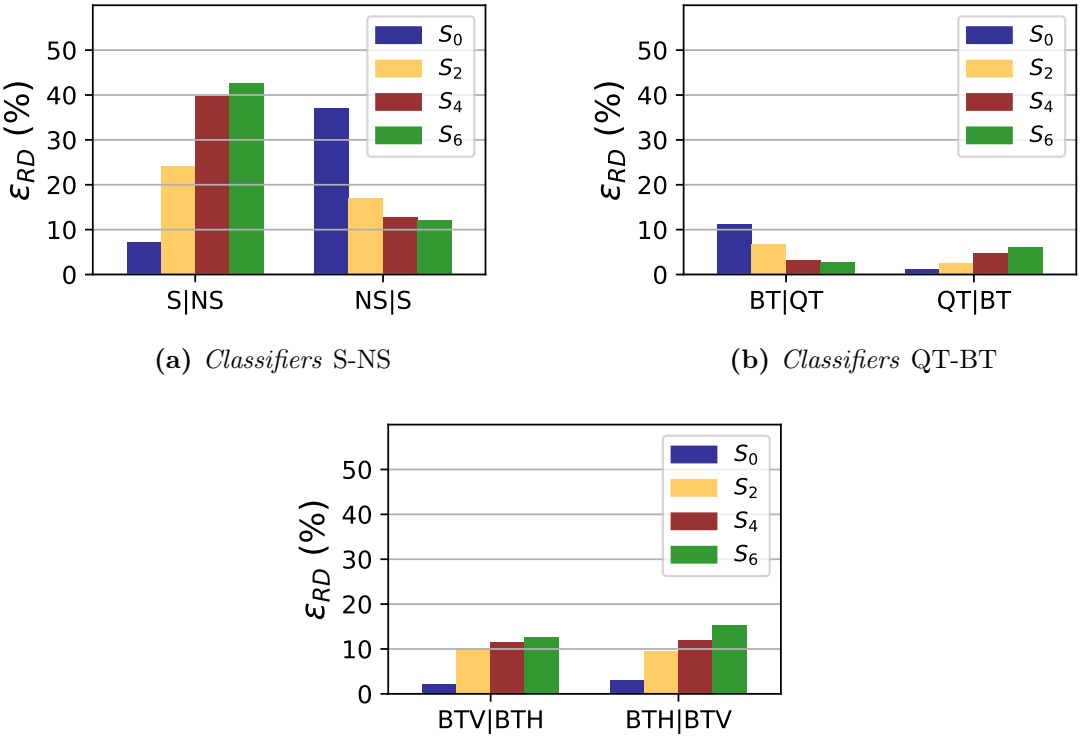
(c) BH-BV *Classifier*

Figure 4.7 – *Mutual Information of evaluated features according to classifier and CU sizes.*

4.5 Classifiers Training Process

The training process consists in building the classifier through maximizing classification rate on the training dataset. In addition to classification rates, the losses of RD performance induced by misclassification are considered in the training process.

4.5.1 Impact of Misclassification on RD-cost Errors



(a) *Classifiers* S-NS

(b) *Classifiers* QT-BT

(c) *Classifiers* BH-BV

Figure 4.8 – *Average RD misclassification error according to the classifier and CU size category. Data from 4 Sequences (*BasketballDrive, BQMall, Flowervase, Johnny*), encoded with* $QP = 22, 27, 32, 37$

To assess the impact of misclassification on the encoding efficiency, the RD error ε_{RD} caused by a misclassification is introduced. In the following, the misclassification $A|B$ is when the partition mode A is chosen by the RF classifier, whereas B is the optimal partition mode selected by the encoder after exhaustive RDO process. RD error of misclassification $A|B$ is defined by Equation (4.7)

$$\varepsilon_{RD}(A|B) = \frac{J_A - J_B}{J_B} \tag{4.7}$$

J_A and J_B are the RD costs resulting from RDO process for partition modes A and B, respectively. In our case,

$$\begin{aligned}&(A,B) \in \{(NS,S),(S,NS),(QT,BT),(BT,QT),\\&(BTV,BTH),(BTH,BTV)\},\\&J_S = min(J_{QT},J_{BTH},J_{BTV}) \text{ and}\\&J_{BT} = min(J_{BTH},J_{BTV}).\end{aligned}$$

Figure 4.8 shows the average ε_{RD} according to the classifier and the CU size. Results are averaged across 4 sequences (*BasketballDrive*, *BQMall*, *Flowervase*, *Johnny*) and 4 QP values.

A statistical study in Figure 4.8a shows that ε_{RD} caused by classifier *S-NS* is in average 3.4 times higher compared to ε_{RD} caused by classifiers *QT-BT* and *BH-BV*. In other words, the partition modes S and NS are in average more divergent in terms of RD cost, compared to partition modes QT and BT and partition modes BTH and BTV. Therefore *S-NS* classification problem is easier to solve compared to classification problems *QT-BT* and *BH-BV*, explaining why the MI of selected features in Section 4.4.2 are lower for classifiers *QT-BT* and *BH-BV* compared to classifier *QT-BT*.

Concerning the first classifier *S-NS*, Figure 4.8a shows that the smaller the CU, the higher $\varepsilon_{RD}(S|NS)$. When the classifier selects Split partition modes instead of the correct NoSplit partition mode selected by exhaustive RDO process, the misclassification has a higher RD impact for small CUs.
On the other hand, larger CU generate higher values of $\varepsilon_{RD}(NS|S)$. This is due to the fact that the larger the CU, the greater the number of partitioning possibilities. Therefore, for large CUs, the NoSplit partition mode is more divergent in average from the optimal partitioning after exhaustive RDO process, compared to small CUs.

For the second classifier *QT-BT*, Figure 4.8b shows that $\varepsilon_{RD}(BT|QT)$ is higher for large CUs (128×128 and 64×64) than small CUs (32×32 and 16×16). When the classifier selects BT partition modes instead of the correct QT partition mode, selected by exhaustive RDO process, the misclassification has stronger impact on RD cost on the large CUs than small CUs in average. Indeed, when BT partition mode is selected on large CUs, QT partition mode is no longer available, as detailed in Section 5.2. Combined with the limit of 3 successive BT partitions, fine grain partitioning is no longer achievable. On the other hand, $\varepsilon_{RD}(QT|BT)$ is higher for small CUs than for large CUs. This is due to the fact that rectangular BT partition modes offer more partitioning shapes than square QT partition mode on small areas in the frame.

Concerning the third classifier *BH-BV*, Figure 4.8c shows that ε_{RD} are symmetric for misclassification $BTH|BTV$ and $BTV|BTH$. Moreover, misclassification has very small impact in average on RD cost losses (below 2%) for S_0 category CUs and quite higher impact (around 10%) for S_2, S_4 and S_6 category CUs.

4.5.2 Weighting of Training Dataset

Previous section shows that the impact of misclassification on RD cost losses depends highly on the classifier and the CU size. From this observation, all the training instances for classifier *A-B* with $(A,B) \in \{(S,NS),(QT,BT),(BH,BV)\}$ are assigned a weight $w(A,B)$. The value of $w(A,B)$ is computed by Eq (4.8).

$$\begin{aligned}&\forall (A,B) \in \{(S,NS),(QT,BT),(BH,BV)\},\\&w(A,B) = max(\varepsilon_{RD}(A|B),\varepsilon_{RD}(B|A)).\end{aligned} \tag{4.8}$$

By assigning a weight to the training instances, the RF classifiers are built in order to minimize the sum of misclassified weights $w(A, B)$, instead of minimizing the classification error rate. Therefore the training process has more probability to classify well training instances with high weights, i.e training instances that induce high RD losses. Note that the weights are needed only during the training process, and not when the trained model is used to reduce the complexity of encoding process.

4.5.3 Classification Rates

As mentioned in Section 4.4.2, the classification rate is the percentage of correct classification given by the 4-fold cross-validation on the training dataset, carried out with the weighting of training instances described in Section 4.5.2. In the following, the number of decision trees of the RFs has been set to 40 which represents a good trade-off between high classification rate and low inference time. TABLE 4.2 gives the average classification rates of the three classifiers according to the CU size category, with 40 decision trees by RF.

Table 4.2 – *Classification rate (in %) according to the classifier and the CU size category.*

	S_0	S_1	S_2	S_3	S_4	S_5	S_6	Average
S-NS	83	82	78	76	73	72	69	76
QT-BT	69	-	70	-	67	-	60	67
BH-BV	62	64	70	67	69	68	67	67

In TABLE 4.2, the classification rates of classifier *S-NS* are between 69% and 83%. The larger the CU, the higher the classification rate for classifier *S-NS*. In the literature, classification rates of techniques using ML to reduce the complexity of the QT partitioning in HEVC are close to 80% [89, 100]. The classifier *S-NS* has therefore classification rate performance comparable to previous works on HEVC.

Section 4.5.1 shows that the *S-NS* classification problem is easier to solve compared to classification problems *QT-BT* and *BH-BV*. For this reason the classification rates of classifiers *QT-BT* and *BH-BV* are in average 67%, which is in 9% lower than the average classification rate of classifier *S-NS*.

4.6 Tunable Complexity Reduction

In order to control RD losses induced by misclassification, risk intervals of classification are introduced for each binary classifier. In the risk interval of the binary classifier, both output partition modes are processed, limiting RD efficiency losses at the expense of complexity reduction. By varying the size of risk intervals, tunable complexity reduction is achieved.

4.6.1 Definition of Risk Interval

A score value, deduced from the votes of individual decision trees, is used to determine the risk interval of a given classifier. The associated score $Score(A)$ corresponds to the percentage of decision trees of the RF classifier that predicts the class A which is defined by Equation (4.9)

$$Score(A) = \frac{N_{votes}(A)}{N_{trees}} \tag{4.9}$$

where $N_{votes}(A)$ is the number of trees voting for class A and N_{trees} is the total number of trees constituting the RF classifier. $Score(A)$ takes values between 0 and 1 and the value of $Score(A)$ quantization step is $1/N_{trees}$. The closer $Score(A)$ is to 1, the more predominantly the RF classifier selects class A.

In our specific case, all the classification problems are between two classes A and B, with $(A, B) \in \{(NS, S), (QT, BT), (BTV, BTH)\}$. For binary classification, as $N_{votes}(A) + N_{votes}(B) = N_{trees}$, then $Score(A) + Score(B) = 1$. Using this relation, the classification decision of the binary RF classifier is A if $Score(A) > 0.5$ and B otherwise (see Section 4.3.1).

An example of risk interval is illustrated in red color in Figure 4.9. The risk interval is the range $[0.5 - dS(A), 0.5 + dS(B)]$ of $Score(A)$, with $dS(A)$ and $dS(B)$ the risk interval boundaries dS for decisions A and B, respectively. The risk interval boundary dS are included in the range $[0, 0.5]$. When $Score(A)$ is inside the risk interval, the classifier makes no decision and both output partition modes A and B are processed.

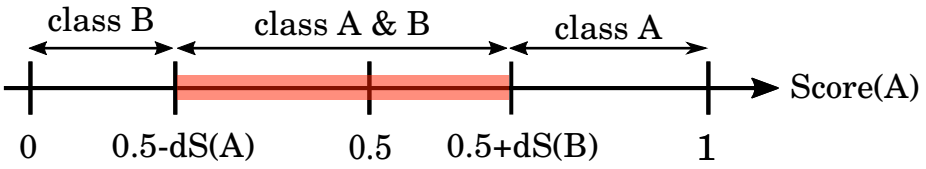


Figure 4.9 – *Risk interval for binary classification.*

4.6.2 Computation of Risk Interval Boundaries

The values of the risk interval boundaries dS for every classifier and CU size category are computed at encoding time. Every 32 frames, 1 reference frame is encoded with the exhaustive RDO process of JEM-7.0, enabling the RF classifiers only to gather misclassification RD statistics. No complexity reduction is achieved on the encoding of the reference frames. The complexity of the reference frames is included in the final results and is compensated by the complexity reduction achieved on frames constrained by the RF classifiers.

In the reference frames, the RD costs of all partition modes are computed, making it possible to compute the sum of ε_{RD} error induced by misclassification, further called cumulative ε_{RD}. The risk interval boundaries dS are computed in order to limit cumulative ε_{RD} on the reference frame. The computation of the the dS values at encoding time adjusts the RD efficiency losses to video content variations, across different sequences and scenes.

In Figure 4.10, the blue and green histograms correspond to a concrete example of cumulative $\varepsilon_{RD}(QT|BT)$ and cumulative $\varepsilon_{RD}(BT|QT)$, respectively, in function of $Score(QT)$. The cumulative ε_{RD} are computed from the RD costs of S_6 category CUs, extracted from the RDO process of sequence *BasketballDrill* reference frame. The maximum cumulative ε_{RD} error tolerated on the reference frame is further noted L and represented by the red line in Figure 4.10. Depending on the L value, the boundaries of the risk interval $dS(QT)$ and $dS(BT)$ are determined such as both cumulative $\varepsilon_{RD}(QT|BT)$ and cumulative $\varepsilon_{RD}(BT|QT)$ are below L. Note that $dS(QT)$ is greater than $dS(BT)$ in this example, since the errors $\varepsilon_{RD}(BT|QT)$ are greater than the errors $\varepsilon_{RD}(QT|BT)$.

4.6.3 Individual Performance of Classifiers

As explained in Section 4.5.1, average RD losses induced by misclassification depend highly on the classifier. Reason why the possibility is left for the user to select a different threshold

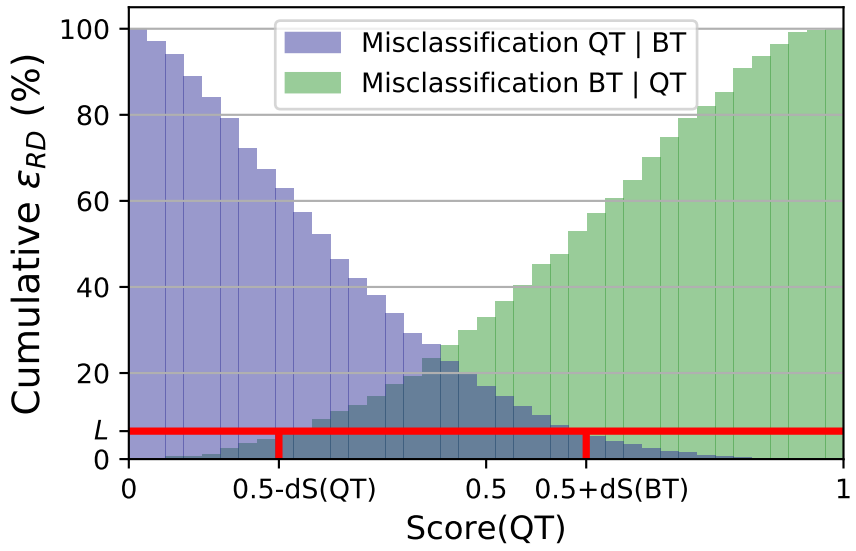


Figure 4.10 – *Cumulative ε_{RD} of QT|BT and BT|QT misclassifications in function of Score(QT). Data from S_6 category CUs of first frame of sequence* BasketballDrill, $QP = 22$.

L by classifier, further noted $L_{S\text{-}NS}$, $L_{QT\text{-}BT}$, $L_{BH\text{-}BV}$. For a given classifier, the same threshold is applied on all CU size categories.

The performance of a complexity reduction solution is evaluated by measuring the trade-off between RD efficiency using the BD-BR increase [13] and encoding complexity reduction ΔT, defined by Equation (4.10)

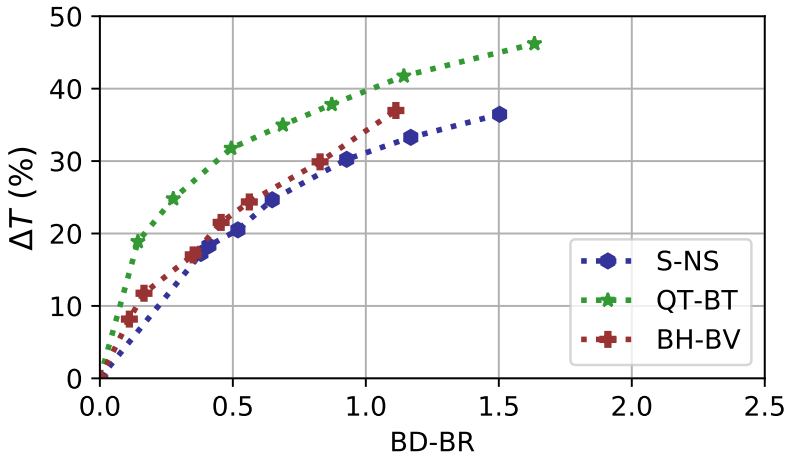
$$\Delta T = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} \frac{T_A(QP_i) - T_R(QP_i)}{T_A(QP_i)}, \tag{4.10}$$

where $T_A(QP_i)$ and $T_R(QP_i)$ are the anchor (encoded with exhaustive RDO process) and reduced time required to encode the video with $QP = QP_i$, respectively.

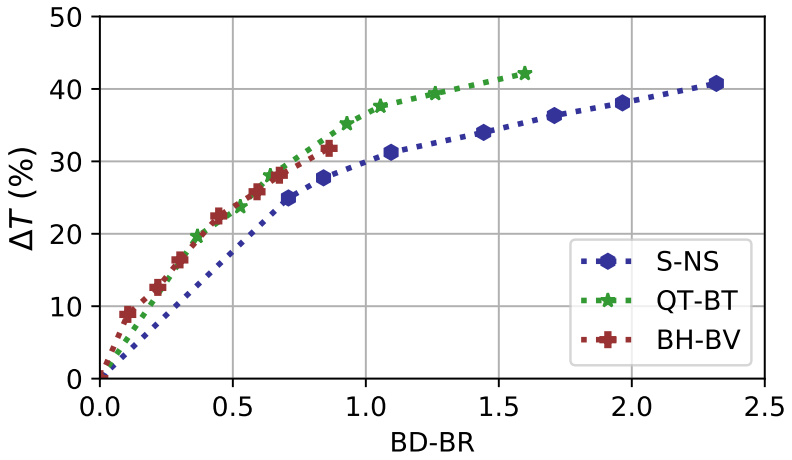
In order to evaluate the performance of the classifiers individually, encodings are run activating only one classifier at a time with different values of L: 0.0%, 0.01%, 0.02%, 0.05%, 0.10%, 0.15%, 0.20%, 0.30%. The value $L = 0.0\%$ means that the classifier is disabled. The performance is gathered across the encodings of the 32 first frames of 10 training sequences over 4 QP values.

Figure 4.11 shows the average ΔT versus the average BD-BR, according to the classifier for BT_{depth} equals to 1, 2 and 3. As explained in Section 4.2.1, BT_{depth} is the encoding parameter that specifies the number of successive allowed BT partitions. In the CTC [73], BT_{depth} value is set to 3, reason why the conducted experiences only consider BT_{depth} values lower than 3. The blue, green and red curves correspond to performance obtained with the individual activation of classifiers *S-NS*, *QT-BT* and *BH-BV*, respectively. The points of the curves are obtained from left to right for the following values of L: 0.0%, 0.01%, 0.02%, 0.05%, 0.10%, 0.15%, 0.20%, 0.30%. In Figure 4.11, the higher left the curve, the better the classifier performance, as it minimizes BD-BR while maximizing ΔT.

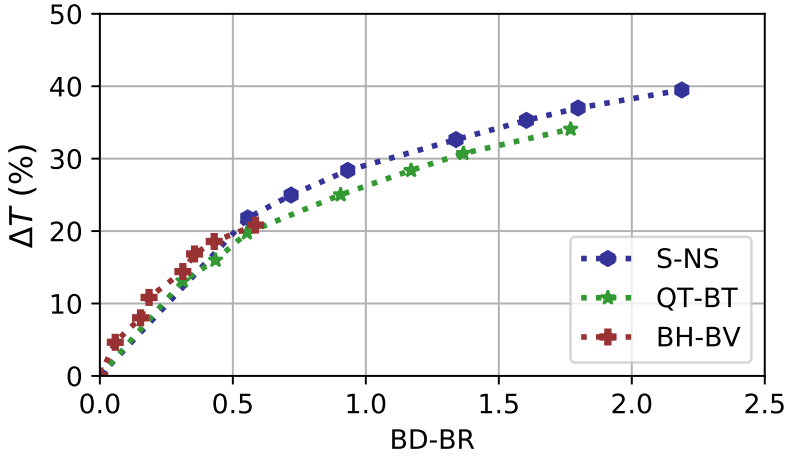
Figure 4.11a shows that for $BT_{depth} = 3$, when BD-BR is lower than 1.2%, the curve of classifier *BH-BV* is below the curve of classifier *QT-BT* and above the curve of classifier *S-NS*. Classifier *QT-BT* has therefore the best performance. Even though classifier *S-NS* has the best successful classification rates (see Section 4.5.3), it has the worst performance in term of trade-off between ΔT and BD-BR. This is due to the fact that misclassification



(a) $BT_{depth} = 3$



(b) $BT_{depth} = 2$



(c) $BT_{depth} = 1$

Figure 4.11 – *Average* ΔT *and BD-BR, according to the* BT_{depth} *and classifier. Average computed across the encodings of 32 first frames of 10 training sequences with 4 QP values. Points correspond to different values of* L*:* 0.0%, 0.01%, 0.02%, 0.05%, 0.10%, 0.15%, 0.20%, 0.30%.

induces much higher RD errors in average for classifier *S-NS* compared to other classifiers, as explained in Section 4.5.1.

For $BT_{depth} = 2$, Figure 4.11b shows that classifiers *QT-BT* and *BH-BV* have equivalent performance as their curves overlap from 0% to 30% ΔT. Classifier *S-NS* has the lowest performance as its curve is the rightmost from 0% to 40% ΔT.

Finally, when $BT_{depth} = 1$, BT partition modes are available for less CUs compared to $BT_{depth} = 3$ and $BT_{depth} = 2$. Classifiers *QT-BT* and *BH-BV* have therefore less CUs to address, meaning less complexity reduction opportunities compared to classifier *S-NS*. This explains why in Figure 4.11c, the curve of classifier *QT-BT* is lower compared to the curve of classifier *S-NS* when BD-BR is between 0% and 1.7%. This also enlighten why ΔT for classifier *BH-BV* only reaches 20%, whereas it reaches 30% for $BT_{depth} = 3$ and $BT_{depth} = 2$.

4.6.4 Optimal Selection of Complexity Reduction Configurations

In order to activate the three classifiers simultaneously at encoding time, three values of L are required. The triplet ($L_{S\text{-}NS}$, $L_{QT\text{-}BT}$, $L_{BH\text{-}BV}$) is further called Complexity Reduction Configuration (CRC).

For a given CRC, knowing the individual complexity reductions of classifiers $\Delta T(\mathbb{L}_{S\text{-}NS})$, $\Delta T(\mathbb{L}_{QT\text{-}BT})$ and $\Delta T(\mathbb{L}_{BH\text{-}BV})$ presented in Section 4.6.3, this section first explains how to estimate the expected complexity reduction (called ΔT_{crc}) when the three classifiers are used simultaneously. In the following, ΔT_{crc} is computed considering an example where $\Delta T(\mathbb{L}_{S\text{-}NS}) = 15\%$, $\Delta T(\mathbb{L}_{QT\text{-}BT}) = 25\%$ and $\Delta T(\mathbb{L}_{BH\text{-}BV}) = 20\%$. Intermediate values ΔT_S, ΔT_Q and ΔT_B are introduced by Equation (4.11)

$$\begin{aligned} \Delta T_S &= \Delta T(\mathbb{L}_{S\text{-}NS}) = 15\%, \\ \Delta T_Q &= (1.0 - \Delta T_S) \cdot \Delta T(\mathbb{L}_{QT\text{-}BT}) \\ &= 0.85 \cdot 25\% = 21\%, \\ \Delta T_B &= (1.0 - \Delta T_S - \Delta T_Q) \cdot \Delta T(\mathbb{L}_{BH\text{-}BV}) \\ &= 0.64 \cdot 20\% = 13\%. \end{aligned} \tag{4.11}$$

The expected complexity reduction ΔT_{crc} is given by Equation (4.12)

$$\Delta T_{crc} = \Delta T_S + \Delta T_Q + \Delta T_B = 49\%. \tag{4.12}$$

Over all CRCs achieving an expected complexity reduction ΔT_{crc}, a CRC is considered optimal when it obtains the lowest sum of BD-BR after exhaustive search. The optimal CRCs, noted from C_0 to C_4, are given in TABLE 4.3 each corresponding to a target ΔT_{crc}, according to the BT_{depth}.

For both $BT_{depth} = 3$ and $BT_{depth} = 2$, in all optimal CRCs the value of $L_{S\text{-}NS}$ is 0.0, meaning classifier *S-NS* is not used to reduce encoding complexity. This is explained by the results of Section 4.6.3, where classifier *S-NS* has lower performance in term of trade-off between BD-BR and ΔT compared to classifiers *QT-BT* and *BH-BV*, for $BT_{depth} = 3$ and $BT_{depth} = 2$,.

When $BT_{depth} = 1$, the value of $L_{QT\text{-}BT}$ is 0.0 for all optimal CRCs. It is therefore more efficient in term of BD-BR to use only classifiers *S-NS* and *BH-BV* when $BT_{depth} = 1$.

Table 4.3 – *Optimal CRCs and associated expected ΔT_{crc}, according to BT_{depth}.*

CRC Name	ΔT_{crc}	$BT_{depth} = 3$			$BT_{depth} = 2$			$BT_{depth} = 1$		
		$L_{S\text{-}NS}$	$L_{QT\text{-}BT}$	$L_{BH\text{-}BV}$	$L_{S\text{-}NS}$	$L_{QT\text{-}BT}$	$L_{BH\text{-}BV}$	$L_{S\text{-}NS}$	$L_{QT\text{-}BT}$	$L_{BH\text{-}BV}$
C_0	30%	0.0	0.01	0.02	0.0	0.01	0.02	0.01	0.0	0.05
C_1	35%	0.0	0.05	0.02	0.0	0.02	0.05	0.01	0.0	0.15
C_2	40%	0.0	0.10	0.05	0.0	0.05	0.10	0.02	0.0	0.20
C_3	45%	0.0	0.10	0.15	0.0	0.10	0.15	0.05	0.0	0.20
C_4	50%	0.0	0.15	0.15	0.0	0.15	0.15	0.10	0.0	0.20

4.7 Experimental Results

This section gives the experimental setup and the results obtained for the proposed tunable complexity reduction solution. Sections 4.7.2 and 4.7.3 present the results obtained on JEM-7.0, while Section 4.7.4 present the results obtained on VTM-5.0.

4.7.1 Experimental Setup

The selected set test sequences is composed of 18 video sequences different from training set sequences (see Section 4.4.1), selecting 3 sequences by class: *Campfire*, *ParkRunning3*, *ToddlerFountain*, *PeopleOnStreet*, *SteamLocomotiveTrain*, *NebutaFestival*, *Cactus*, *RitualDance*, *Kimono*, *RaceHorsesC*, *PartyScene*, *BasketballDrill*, *ParkScene*, *KristenAndSara*, *FourPeople*, *BlowingBubbles*, *RaceHorsesD* and *BQSquare*.

The experiments are carried-out under the CTC [73] in RA coding configuration at four QP values: 22, 27, 32 and 37. The performance of the proposed complexity reduction solution is evaluated by measuring the trade-off between BD-BR increase and encoding complexity reduction ΔT, defined in Section 4.6.3. In the following, the complexity overhead induced by the RF inference during partition scheme is noted θ. The proposed complexity reduction solution is implemented in both JEM-7.0 and VTM-5.0. In order to limit the encoding time, JEM-7.0 encoder compares the RD cost of the whole current CU with those of the BTH and BTV partition modes to prune the QT partition mode. As our solution does not compute all the RD costs of the BT partition mode, this condition is removed in the experiments.

4.7.2 Performance Evaluation of the Proposed Solution in the JEM-7.0.

In order to set the upper bound in term of complexity reduction of the proposed solution, the maximum complexity reduction opportunity ΔT_{max} for QTBT partitioning scheme in JEM-7.0 is computed. ΔT_{max} is the value of ΔT achieved when the only tested QTBT partition is the optimal partition. The average ΔT_{max} value across the 18 test sequences in RA configuration is:

$$\Delta T_{max} = 90\%.$$

TABLE 4.4 shows the average BD-BR and ΔT values across the 18 test sequences, encoded with exhaustive RDO process according to the BT_{depth} value. Reducing BT_{depth} value and allowing exhaustive RDO process is a straightforward technique to reduce complexity of QTBT partition scheme. For a fixed value of BD-BR, if the average ΔT value

Table 4.4 – *BD-BR(%) and ΔT(%) for JEM-7.0 exhaustive RDO process encodings with different BT_{depth} values. Results averaged across 18 test sequences.*

JEM-7.0 Exhaustive RDO							
$BT_{depth} = 3$		$BT_{depth} = 2$		$BT_{depth} = 1$		$BT_{depth} = 0$	
BD-BR	ΔT	BD-BR	ΔT	BD-BR	ΔT	BD-BR	ΔT
+0.0%	0%	+0.6%	20%	+2.0%	50%	+5.1%	79%

of the proposed solution is lower than the average ΔT value obtained simply by allowing exhaustive RDO process with a reduced value of BT_{depth}, the CRC is considered as non-efficient for this value of BD-BR.

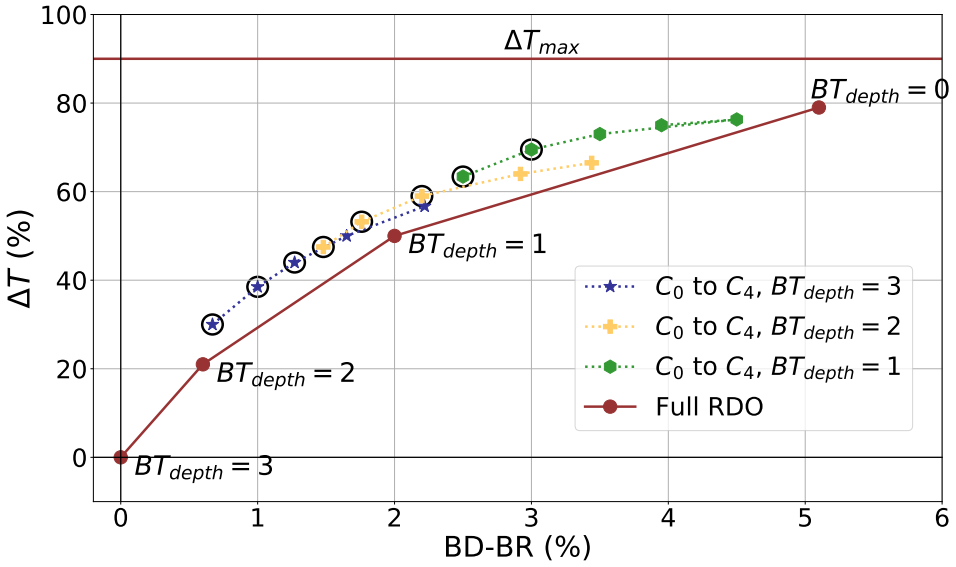


Figure 4.12 – *Average BD-BR and ΔT for optimal CRCs and exhaustive RDO with different BT_{depth}. CRCs adopted in the proposed solution are circled in black. All results averaged across 18 test sequences and 4 QP values.*

The performance of the CRCs in term of BD-BR and ΔT is displayed in Figure 4.12. The upper bound ΔT_{max}, as well as BD-BR and ΔT for exhaustive RDO process with different BT_{depth} values, are also displayed in red. In Figure 4.12, the blue stars correspond to CRCs with $BT_{depth} = 3$, as in the CTC. With $BT_{depth} = 3$, the optimal CRCs offer an average ΔT value between 30% and 57% for an average BD-BR increase comprised between 0.67% and 2.22%. The yellow crosses and green hexagons correspond to CRCs with reduced values $BT_{depth} = 2$ and $BT_{depth} = 1$, respectively. With $BT_{depth} = 2$, the ΔT of the optimal CRCs are in average comprised between 48% and 66% with BD-BR increase between 1.48% and 3.40%. With $BT_{depth} = 1$, the optimal CRCs offer an average ΔT value between 63% and 78% for BD-BR increase between 2.45% and 5.21%.

The 8 CRCs adopted in the proposed solution are circled in black in Figure 4.12. They are located on the pareto front, i.e. the CRCs with lower BD-BR for a given value of ΔT. The CRC inducing a BD-BR increase superior than 3% are not adopted in the proposed solution since they offer a trade-off between ΔT and BD-BR not considered good enough. The adopted CRCs include: $C_0(BT_3)$, $C_1(BT_3)$, $C_2(BT_3)$, $C_0(BT_2)$, $C_1(BT_2)$, $C_2(BT_2)$,

Table 4.5 – *Average BD-BR, ΔT and complexity overhead θ of the CRCs adopted in the proposed solution.*

Proposed Solution in the JEM-7.0				
BT_{depth}	Adopted CRC	BD-BR (%)	ΔT (%)	θ (%)
$BT_{depth} = 3$	C_0	0.7	30.2	1.4
	C_1	1.0	37.3	1.2
	C_2	1.3	44.1	1.1
$BT_{depth} = 2$	C_0	1.6	48.2	0.7
	C_1	1.9	54.3	0.7
	C_2	2.3	59.4	0.6
$BT_{depth} = 1$	C_0	2.5	63.2	0.5
	C_1	3.0	70.0	0.5

$C_0(BT_1)$ and $C_1(BT_1)$. TABLE 4.5 summarizes the average BD-BR, ΔT and θ of the CRCs adopted in the proposed solution. TABLE 4.5 shows that in order to achieve complexity reductions higher than 43% in average, it is more efficient to apply our solution with a value of $BT_{depth} < 3$, compared to applying our solution with $BT_{depth} = 3$. With these adopted CRCs, the proposed tunable solution offers a range of average ΔT between 30% and 70% for an average BD-BR increase between 0.7% and 3.0%.

Table 4.6 – *BD-BR, ΔT and complexity overhead θ of CRCs $C_0(BT_3)$ and $C_0(BT_1)$ of the proposed solution with respect to the performance announced by Wang et al. in papers [94] and [95]. The results are shown by test sequence.*

		Proposed Solution CRC $C_0(BT_3)$			Related Work Wang CNN [94]			Proposed Solution CRC $C_0(BT_1)$			Related Work Wang Proba [95]	
Class	Sequence name	BD-BR (in %)	ΔT (in %)	θ (in %)	BD-BR (in %)	ΔT (in %)	θ (in %)	BD-BR (in %)	ΔT (in %)	θ (in %)	BD-BR (in %)	ΔT (in %)
A1	*Campfire*	0.36	35.1	0.5	0.66	40.6	2.4	3.4	68.8	0.2	1.7	46.7
A1	*ParkRunning3*	0.49	21.1	1.0	-	-	-	2.0	57.4	0.4	-	-
A1	*ToddlerFountain*	0.58	18.2	0.8	-	-	-	1.8	59.5	0.3	1.2	48.4
A2	*PeopleOnStreet*	0.22	21.8	1.5	-	-	-	2.2	58.0	0.5	-	-
A2	*SteamLocomotiveTrain*	0.57	24.6	1.0	-	-	-	2.7	63.5	0.3	-	-
A2	*NebutaFestival*	1.37	35.1	0.9	-	-	-	2.2	68.0	0.3	-	-
B	*Cactus*	1.18	34.1	1.2	-	-	-	2.8	64.1	0.4	1.5	48.2
B	*RitualDance*	0.90	32.0	0.7	0.55	32.6	4.2	3.9	64.7	0.2	-	-
B	*Kimono*	0.79	23.1	1.0	-	-	-	4.0	65.3	0.3	1.4	54.7
B	*ParkScene*	0.63	30.8	1.6	-	-	-	2.5	68.1	0.5	1.2	45.7
C	*RaceHorsesC*	0.42	32.8	0.9	0.47	30.7	1.7	2.1	63.1	0.3	1.6	60.5
C	*PartyScene*	0.45	29.2	1.5	0.54	34.6	2.2	1.6	62.8	0.5	1.7	62.3
C	*BasketballDrill*	0.61	30.2	1.2	-	-	-	2.4	63.5	0.4	1.5	62.3
D	*RaceHorsesD*	0.41	28.8	1.2	0.51	36.3	2.7	2.3	61.5	0.6	1.3	56.5
D	*BQSquare*	0.64	25.1	1.8	0.44	26.0	2.1	1.5	57.5	0.8	1.3	51.8
D	*BlowingBubbles*	0.46	31.6	1.6	0.60	35.7	1.9	1.9	60.0	0.8	1.2	50.3
E	*FourPeople*	0.76	40.8	1.8	0.32	28.9	4.1	2.5	68.3	0.5	1.3	47.7
E	*KristenAndSara*	1.17	46.6	1.2	0.38	33.8	3.6	2.7	70.2	0.4	1.0	45.6
	Same Sequences Average	**0.62**	**33.5**	**1.4**	**0.50**	**32.8**	**2.8**	**2.42**	**64.0**	**0.5**	**1.38**	**52.3**
	Global Average	**0.67**	**30.2**	**1.3**				**2.45**	**63.4**	**0.5**		

4.7.3 Comparison with Related Works in JEM-7.0.

The proposed solution is evaluated and compared to previous techniques on QTBT partition scheme in RA configuration [90], [92], [94] and [95]. Previous techniques [90] and [92] offer an average encoding complexity reduction of 17% and 10% for an average BD-BR

increase of 0.5% and 0.2%, respectively. The encoding complexity reductions are much lower compared to the encoding complexity reductions proposed in our solution, which is in minimum equals to 30% for $C_0(BT_3)$.

For a fairer comparison with the two most recent techniques [94] and [95], TABLE 4.6 details the performance of 2 CRCs of the proposed solution - $C_0(BT_3)$ and $C_0(BT_1)$ - with respect to the performance announced by Wang et al. in papers [94] and [95]. The performance in TABLE 4.6 is shown by test sequence, in terms of BD-BR increase, encoding complexity reduction ΔT and θ induced by the respective techniques.

TABLE 4.6 shows that configuration $C_0(BT_3)$ applied to the same sequences as technique [94], offers in average the same encoding complexity reduction ($\Delta T \approx 33\%$) for a BD-BR increase 0.12% higher in average. However, technique [94] is based on CNNs to reduce the encoding complexity without specifying his implementation, whereas CNNs are known to have high computational overhead. For $C_0(BT_3)$, θ has values between 0.7% and 1.8%, whereas θ for technique [94] has values between 1.7% and 4.2% according to the sequence. θ is included in the encoding complexity reductions of of the proposed solution. These overhead performance confirms the lightweight of our approach and highlights that RF classifiers consume few computing resources, which is a key point to use this solution in a real-time or embedded framework.

TABLE 4.6 also shows that configuration $C_0(BT_1)$ achieves higher encoding complexity reductions for all tested sequences compared to considered previous techniques, and achieves in average 12% higher encoding complexity reduction compared to technique [95], with tolerable BD-BR increase of 2.45% in average. Moreover, the fact that the proposed solution is tunable offers more flexibility for concrete use-cases compared to previous techniques that aim to reduce the complexity of QTBT partition scheme.

4.7.4 Performance Evaluation of the Proposed Solution in the VTM-5.0

The VTM-5.0 is the latest reference software for VVC standardization. Several new coding tools have been added compared to the JEM-7.0 reference software. For instance, the VTM-5.0 includes the MTT partitioning scheme, more complex than QTBT in JEM-7.0. Thus, the proposed solution has also been implemented in the VTM-5.0 in order to verify its performance. The following section first introduces the MTT partitioning scheme and second presents the results of the proposed solution integrated in the VTM-5.0.

The MTT partitioning scheme is an extension to QTBT that enables Ternary Tree (TT) partition modes, including Ternary Tree Horizontal (TTH) partition mode and Ternary Tree Vertical (TTV) partition mode. When TT partition modes are used, the CU is divided either horizontally or vertically into three blocks and the size of the middle block is half the size of the CU, as shown in Figure 4.13. The MT_{depth} parameter defines the maximum number of successive BT or TT partitions allowed for the encoding of a CTU.

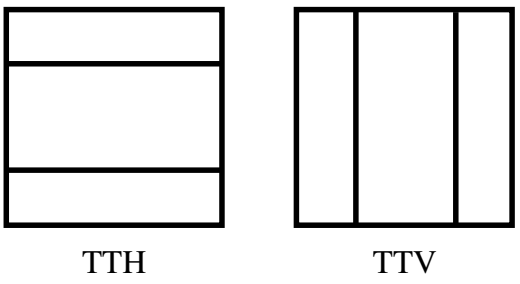


Figure 4.13 – *Additional partition modes in MTT partition scheme.*

The proposed solution has originally been designed for complexity reduction of QTBT partitioning scheme. MTT partitioning scheme is more complex than QTBT partitioning scheme, as it enables two additional partition modes. In order to adapt the proposed solution to MTT partitioning scheme, horizontal partition modes including TTH and BTH, and vertical partition modes including TTV and BTV, are both grouped as outputs of the *BH-BV* classifier, as shown in Figure 4.14. The same classifier *BH-BV* is used to classify both BT and TT partition modes in the VTM-5.0. This choice is supported by the fact that partition modes TTH and BTH, as well as partition modes TTV and BTV, generate partitions with the same directions.

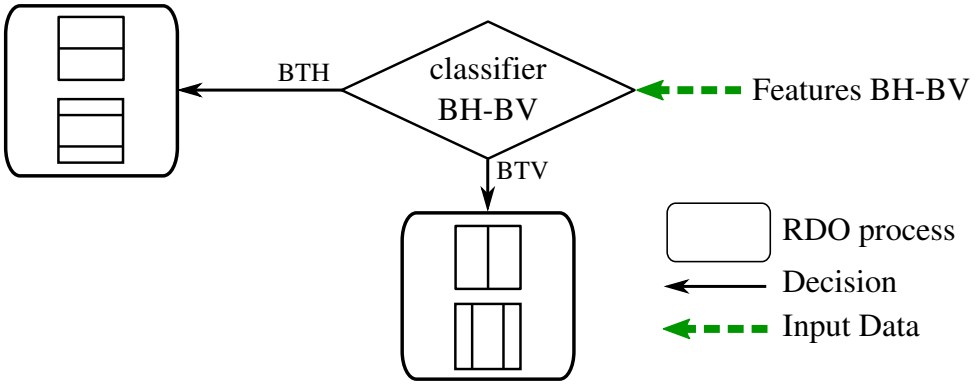


Figure 4.14 – *Outputs modification of* BH-BV *classifier in VTM-5.0.*

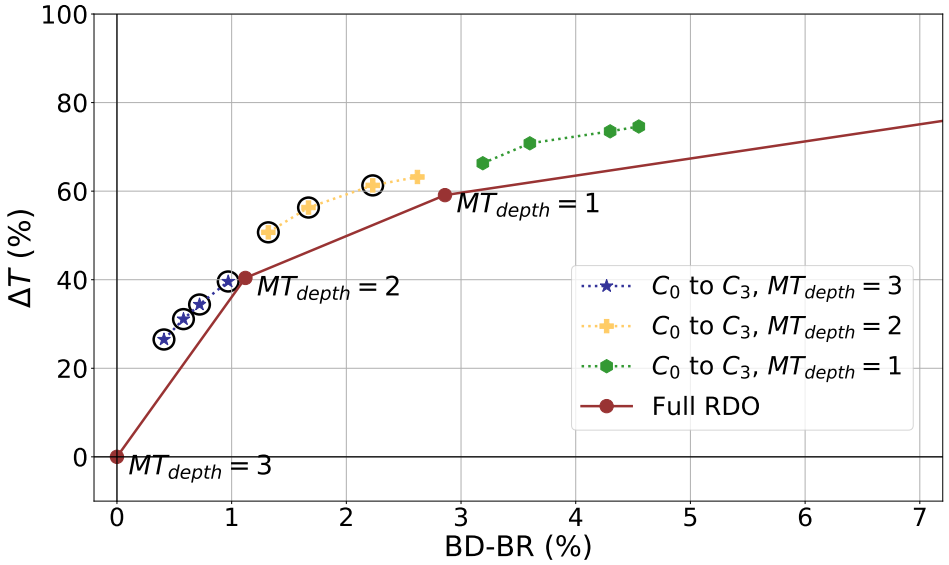


Figure 4.15 – *Average BD-BR and* ΔT *for optimal CRCs and exhaustive RDO with different* BT_{depth}*. CRCs adopted in proposed solution are circled in black. All results averaged across 18 test sequences and 4 QP values.*

The same features selected in Section 4.4.2 are used as an input of the RF classifiers. Moreover, CRC assessed under the VTM-5.0 are selected with the same process described in Section 4.6.4.

Instead of applying the CRCs with different values of BT_{depth} as in JEM-7.0, the CRCs are applied with different values of MT_{depth} in the VTM-5.0. The performance

of the CRCs in term of BD-BR and ΔT is illustrated in Figure 4.15. The BD-BR and ΔT for exhaustive RDO process with different MT_{depth} values, are also displayed in red. In Figure 4.15, the blue stars correspond to CRCs with $MT_{depth} = 3$, while the yellow crosses and green hexagons correspond to CRCs with reduced values $MT_{depth} = 2$ and $MT_{depth} = 1$, respectively.

Figure 4.15 shows that for a similar ΔT complexity reduction, the proposed solution offers a BD-BR value 1.0% lower compared to exhaustive RDO process with $MT_{depth} = 1$. This result confirms that our approach is relevant compared to the most straightforward complexity reduction technique, that allows exhaustive RDO process with reduced values of MT_{depth}.

The 7 CRCs circled in black in Figure 4.12 are adopted in the proposed solution. As for the JEM-7.0 in Section 4.7.2, the CRC inducing a BD-BR increase superior than 3% are not adopted in the proposed solution since they do not offer a relevant trade-off between ΔT and BD-BR. TABLE 4.7 summarizes the average BD-BR, ΔT and θ of the CRCs adopted in the proposed solution. TABLE 4.7 shows that the complexity reductions vary from 25% to 61% in average for 0.4% to 2.2% BD-BR increase. Moreover, the RF inference overhead θ is below 0.7% for all the CRCs, which confirms the lightweight overhead of our approach.

Table 4.7 – *Average BD-BR, ΔT and complexity overhead θ of the CRCs adopted in the proposed solution for VTM-5.0.*

Proposed Solution in the VTM-5.0

	Adopted CRC	BD-BR (%)	ΔT (%)	θ (%)
$MT_{depth} = 3$	C_0	0.43	25.5	0.6
	C_1	0.61	30.1	0.6
	C_2	0.75	33.4	0.6
	C_3	0.97	38.6	0.5
$MT_{depth} = 2$	C_0	1.32	50.7	0.3
	C_1	1.67	56.3	0.3
	C_2	2.22	61.5	0.3

TABLE 4.8 shows the BD-BR, ΔT and θ of CRCs $C_1(MT_3)$ and $C_0(MT_2)$ of the proposed solution under the VTM-5.0, according to the test sequence. We can notice that the scores given in TABLE 4.8 differ slightly according to the test sequence. For instance, the lowest resolution sequences (class D), including *RaceHorsesD*, *BQSquare* and *BlowingBubbles*, achieves in average 54.6% ΔT for $C_2(MT_2)$ configuration. The ΔT values of class D sequences are in average 7% lower compared to the average ΔT value of all test sequences for $C_2(MT_2)$. Indeed, the lowest resolution sequences tend to have a finer grained partitioning, which offers less complexity reduction opportunities compared to higher resolution sequences.

In conclusion, this section has shown that it is possible to apply the proposed solution in the VTM-5.0. The performance under the VTM-5.0 prove that the proposed solution is scalable to two different encoders and does not over-fit the JEM-7.0 encoding characteristics. The good performance achieved under VTM-5.0, the latest reference software of VVC standard, also attests the reliability of the proposed solution for future encoders, compliant with VVC standard. Finally, to the best of our knowledge, this contribution is the first to

Table 4.8 – *BD-BR, ΔT and complexity overhead θ of CRCs $C_1(MT_3)$ and $C_2(MT_2)$ of the proposed solution in the VTM-5.0, according to the test sequence.*

	Proposed Solution CRC $C_1(MT_3)$			**Proposed Solution CRC $C_0(MT_2)$**		
Sequence name	BD-BR (in %)	ΔT (in %)	θ (in %)	BD-BR (in %)	ΔT (in %)	θ (in %)
Campfire	0.95	29.3	0.4	3.06	65.1	0.2
ParkRunning3	0.46	30.9	0.4	1.91	62.9	0.2
ToddlerFountain	0.74	23.2	0.4	1.95	64.2	0.2
PeopleOnStreet	0.72	25.6	0.6	3.13	62.2	0.2
SteamLocomotive	0.92	44.9	0.4	1.71	69.5	0.2
NebutaFestival	0.25	36.3	0.5	0.83	75.8	0.2
Cactus	1.00	36.7	0.5	2.45	64.5	0.2
RitualDance	0.93	27.5	0.5	3.26	61.5	0.2
Kimono	1.16	33.7	0.5	2.74	64.7	0.2
ParkScene	0.29	30.3	0.6	1.93	59.2	0.3
RaceHorsesC	0.35	26.9	0.6	2.77	64.1	0.2
PartyScene	0.42	19.6	0.8	2.05	58.0	0.3
BasketballDrill	1.00	29.9	0.7	2.96	62.8	0.3
RaceHorsesD	0.17	26.6	0.7	2.19	57.1	0.4
BQSquare	0.16	26.2	1.0	1.55	49.1	0.6
BlowingBubbles	0.54	31.3	0.8	1.79	57.7	0.5
FourPeople	0.46	31.6	0.7	1.90	55.6	0.5
KristenAndSara	0.53	30.5	0.6	1.74	54.3	0.4
Average	**0.61**	**30.1**	**0.6**	**2.22**	**61.5**	**0.3**

propose a complexity reduction technique for the VTM reference software in Inter coding configuration.

4.8 Conclusion

In this Chapter, a tunable ML solution based on RF classifiers to speed up the QTBT partitioning scheme in RA configuration is proposed. Three binary RF classifiers are trained off-line in order to ignore expensive exploration of the partition modes classified as unlikely. By varying the size of risk intervals for classification decision, tunable complexity reduction is achieved, offering an average encoding complexity reduction varying from 30% and 70% for an average BD-BR increase between 0.7% and 3.0% in the JEM-7.0, with very low overhead.

The proposed solution as also been implemented in the JVET software post JEM, named VTM. To this end, the proposed solution as been extended to the new TT partition modes included in the VTM partition scheme. In VTM-5.0 software, encoding complexity reductions vary from 25% to 61% in average for only 0.4% to 2.2% BD-BR increase. Tunable encoding complexity reduction being the first step for encoding time control,

future works will investigate the possible modification in the proposed solution in order to achieve encoding time control.

CHAPTER 5

Efficient Parallel Encoding through Dynamic Tiles and Rectangular-Slices

5.1 Introduction

In order to reduce the energy consumption and processing time of Versatile Video Coding (VVC) encoder, the second lever employed in this thesis is the parallel processing. Parallel processing techniques exploit multi-core architectures in order to distribute the complexity to several actors. Techniques for parallel video coding essentially operate at three levels of parallelism: data level, frame level and high-level. The data level parallelism techniques are applied on elementary operations, and no encoding quality is lost compared to sequential encoding, i.e. encoding with a single thread. They include among other techniques relying on Single Instruction on Multiple Data (SIMD) architectures [69]. Frame level and high-level parallelism operate at thread level. The frame level techniques encode a group of frames in parallel where each frame is entirely processed by a single thread [68]. The encoding time of a single frame is not reduced with frame level techniques, i.e. the latency is not reduced. In high-level parallelism techniques, several threads operate on continuous regions of the same frame. These techniques improve equally both speed-up and latency, and are the focus of this Chapter.

For high-level parallelism, both High Efficiency Video Coding (HEVC) and VVC include high-level partitioning of the frame into tiles or slices [101] that allow to process simultaneously large regions of a frame with independent threads. Tiles are rectangular regions of the frame independently encodable, following a grid shaped partition. The standards allow to further partition the tile grid into an horizontal sub-grid of Rectangular Slices (RSs), increasing the partitioning flexibility. The partitioning combining tiles and RSs is called Tile and Rectangular Slice (TRS) partitioning in this Chapter and described with details in Section 5.2.1. The partitioning of a frame into TRSs raises two distinct optimization issues: on one side the multi-thread encoding time minimization (or speedup maximization), on the other side the minimization of encoding quality loss caused by the partitioning. This encoding quality loss is the consequence of prediction dependencies disabling across RSs boundaries and of the entropy coding state reinitialization for each RS In the literature, both issues have been addressed for HEVC tile partitioning. However, the related works on HEVC tile partitioning only address independently minimization of encoding time and encoding quality loss.

In this work, we take advantage of the increased flexibility offered by the RSs in VVC, in order to propose a dynamic TRS partitioning solution under VVC Test Model (VTM)-6.2 software. The TRS partitioning is adjusted at the frame level, taking into account both spatial content and encoding times of previously encoded frames. The proposed solution addresses as a trade-off the multi-thread encoding time and encoding quality loss minimization. Experiments prove that the proposed solution decreases significantly multi-thread encoding time, with slightly better encoding quality, compared to uniform TRS partitioning. Moreover, to the best of our knowledge, this is the first work that implements a multi-thread VVC reference encoder, generating baseline results for future related works. Since Random Access (RA) coding configurations is the most computational complex coding configuration for the encoder (see Section 2.5.9.4), it is also the configuration studied in this work.

The rest of this Chapter is organized as follows. The Section 5.2 describes the TRS partitioning in VVC standard and provides an overview of the related works on HEVC tile partitioning. Section 5.3 describes the proposed solution, which establishes the trade-off between encoding time and encoding quality. Section 5.4 presents the experimental results on VTM-6.2 in RA configuration. Finally, Section 5.5 concludes this Chapter.

5.2 State of the Art

This Section first describes the operating principle of TRS partitioning in VVC standard. It further provides an overview of the related works. These works only provide results on HEVC tile grid partitioning, since this is the first work that implements a multi-thread VVC reference encoder.

5.2.1 Overview of Tiles and Slices in VVC standard

As mentioned in Section 5.1, in high-level parallelism techniques several threads operate on continuous regions of the same frame. They include among others tile and slice partitioning tools enabled in VVC and presented in this section.

5.2.1.1 Tiles

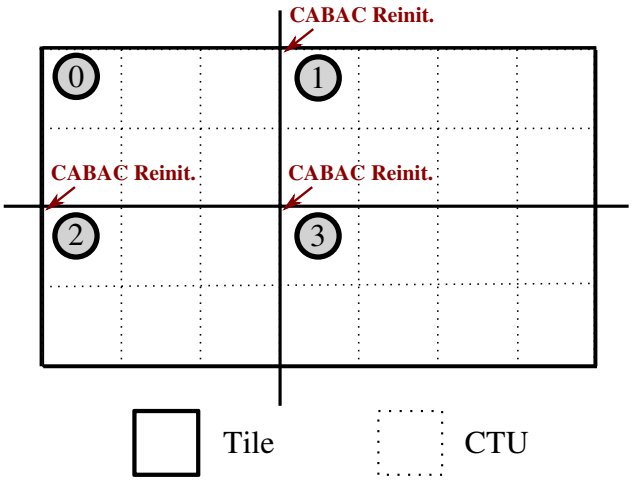


Figure 5.1 – *Illustration of tile partitioning: grid of 4 tiles labeled from 0 to 3.*

Tiles are rectangular regions of the frame, independently encodable and decodable, allowing several threads to process simultaneously the same frame. By enabling independent processing of frame regions, prediction dependencies across boundaries are broken and entropy encoding state is reinitialized for each region. These restrictions lead to an encoding quality loss compared to the encoding of the non-partitioned sequence. The encoding quality decreases with the number of independent regions of the frame, as has been measured in HEVC by *Chin et al.* [102]. In HEVC and VVC standard, only grid shaped tile partitioning is allowed, as shown by Figure 5.1. The tiles are delimited by the continuous black lines and the dashed lines correspond to the Coding Tree Unit (CTU) delimitation. The tile partitioning forms a 2x2 grid and tiles are labeled from 0 to 3.

5.2.1.2 Slices

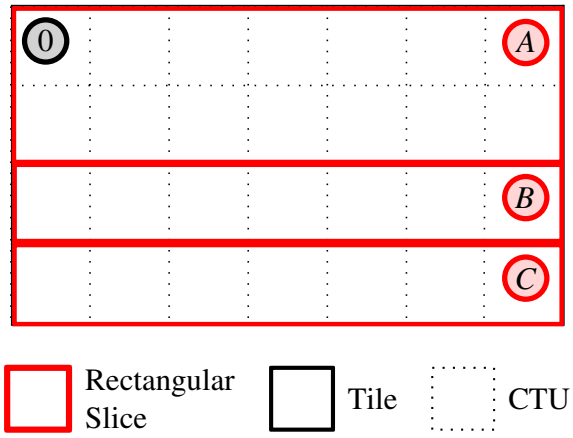
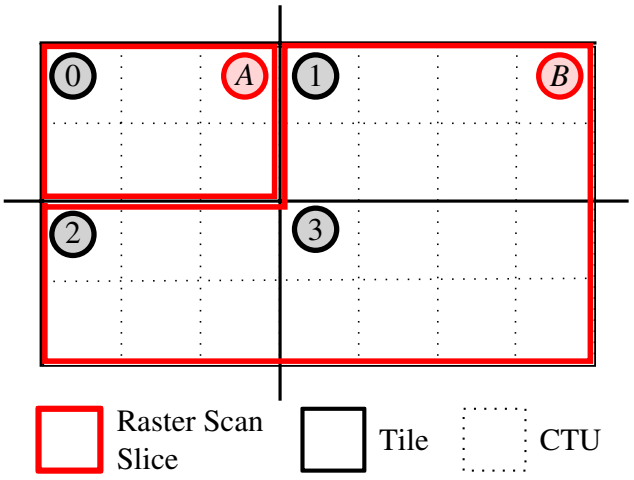


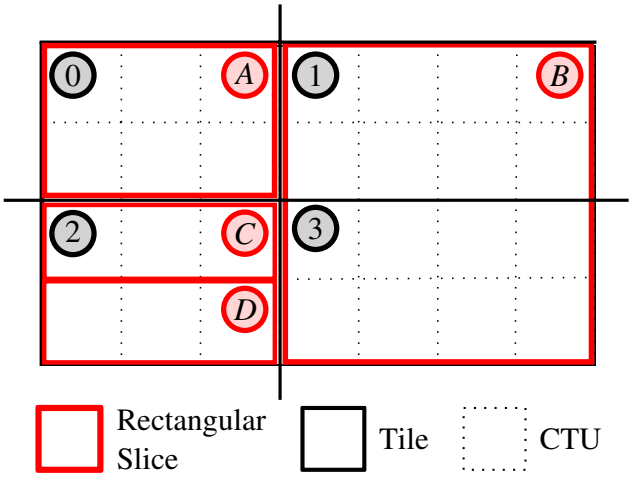
Figure 5.2 – *Particular case of RS partitioning with only 1 Tile.*

Initially, slices have been introduced to divide a bit-stream into smaller independently decodable parts for transmission. The network characteristics generally define the maximum size of the slice bit-stream. Among other criteria, the maximum transmission unit size of the considered network is often selected as a upper bound for the size of slice bit-stream. Slices are not only useful for packetization of the bitstream, it is possible to use them for parallelization of the encoding process. Indeed, the slices are also independently encodable, allowing several threads to process simultaneously the same frame. In the case presented in Figure 5.2 where the tile partitioning is composed of a single tile, the slices allow to partition the frame into an horizontal grid. In the rest of this work, the parallelism offered by the combination of several tiles and slices is studied and is presented in Figure 5.3.

Figure 5.3 shows the various combinations of tiles and slices enabled in VVC, based on the same 2x2 tile grid presented in Figure 5.1. The slices are delimited by the continuous red lines and are labeled with letters. Two modes of slices are supported, namely the raster-scan slice mode and the Rectangular Slice (RS) mode. Both slice modes are presented in Figure 5.3. In the raster-scan slice mode, a slice contains a sequence of complete consecutive tiles in a Raster-scan order of the picture. The slice B in Figure 5.3a for instance, contains consecutive tiles in Raster-scan order #1, #2 and #3. In the RS mode, the slice must form a rectangular region of the frame as the name suggests. The slices A and B in Figure 5.3b are an example of RSs containing one or several complete tiles. Moreover, as shown in



(a) *Raster-scan slices labeled from A to B.*



(b) *Rectangular Slices labeled from A to D.*

Figure 5.3 – *Difference between Rectangular and Raster-scan slices in VVC.*

the examples C and D, a RS may be a rectangular sub-region of the tile, composed of a number of complete and consecutive CTU rows of a tile. In this latter case, the new concept of RS allows to further partition the tile grid into a horizontal sub-grid.

5.2.1.3 Combination of Tiles and Rectangular Slices

In this Chapter, the partitioning combining tiles and RSs is called TRS partitioning. It is illustrated by the Figure 5.3b previously described. As mentioned, the RSs C and D are rectangular sub-regions of the tile #2. In this case, the RSs allow to further partition the tile grid into a horizontal sub-grid of RSs, increasing greatly the TRS partitioning flexibility compared to tile grid partitioning. In the literature, the possibility to further partition the tile grid is a sub-grid of slices has not yet been studied. This work aims to take advantage of the TRS partitioning flexibility, in order to propose a high-level parallelism solution adjustable to sequences spatial and temporal content.

5.2.2 Related Works on HEVC tile partitioning

The partitioning of a frame into tiles and RSs raises two distinct optimization issues: on one side the multi-thread encoding time minimization (or speedup maximization), on the other side the minimization of encoding quality loss caused by the partitioning. In the literature, both issues have been addressed for HEVC tile partitioning.

5.2.2.1 Encoding Time Minimization

The multi-thread encoding time minimization is investigated by *Storch et al* [103] and *Koziri et al.* [104]. They observe that the encoding time does not vary significantly from a CTU to the co-located CTU in the closest temporal frame. Considering this temporal stability, the authors use the encoding times of previous frames to determine the tile partitioning that minimizes the multi-thread encoding time. In [105], the time estimator for each CTU is computed based on previously encoded frame CTU statistics (number of Skip, Inter, Intra blocks for instance). *Papdopoulos et al* [106] propose a per-frame balancing of core workload given a number of tiles T and a number of available cores C. A thread scheduling problem is solved when $T > C$. However, they do not consider the possibility to decrease T depending on available cores C.

5.2.2.2 Encoding Quality Loss Minimization

Authors in [107] minimize the encoding quality loss induced by the tile partitioning. This work considers On-Chip memory limitations to establish a constraint on tile dimensions. Under the established constraint on tile dimensions, the tile boundaries are determined dynamically for each frame in order to minimize the sum of square errors of luminance samples within the tiles. The technique proposed in [108] focuses on the particular case of variable number of available cores. The encoding loss is lowered in some cases by setting a number of tiles inferior to the number of available cores. Authors in [109] minimize the encoding quality loss induced by the tile partitioning of Intra frames by analyzing the CTU variance map of the encoded frame. The proposed tile partitioning results in unbalanced work-load for a single frame. Authors observe that in All Intra (AI) configuration, this problem is solved by properly scheduling tiles between distinct frames. In [110], the maximum tile size that enables real time HEVC encoding in Intra configuration is determined, relying on sequence resolution, sequence frame rate, Quantization Parameter (QP) value and CPU frequency. The tile boundaries are then determined based on the sequence spatial characteristics, in order to reduce quality loss. The tiles encoding are finally distributed to the minimal number of cores, in order to reduce encoding power consumption. However, the related works on HEVC tile partitioning only address independently minimization of encoding time and encoding quality loss, which is not the case of the proposed solution.

5.3 Proposed Dynamic TRS Partitioning for VVC Standard

This section describes the proposed TRS partitioning solution for VVC encoder. The proposed solution addresses as a trade-off the minimization multi-thread encoding time and encoding quality loss.

5.3.1 Encoding Time Minimization

Let P be the partitioning of current frame into n RSs: $P = \{s_0, ..., s_{n-1}\}$. In the following, $T(P)$ is the encoding time of current frame partitioned with P, and simultaneously

processed by N threads in parallel (each thread entirely dedicated to encode a single RS). In this case, $T(P)$ is equal to the time required by the slowest thread to encode his RS. Eq. 5.1 formally establishes $T(P)$, with $T(c_i)$ the encoding time of CTU c_i belonging to the RS s_j, and $T(s_j)$ the encoding time of s_j.

$$\begin{aligned} T(s_j) &= \sum_{c_i \in s_j} T(c_i), \\ T(P) &= \max_{s_j \in P}(T(s_j)). \end{aligned} \tag{5.1}$$

Eq. 5.1 shows that $T(P)$ is directly determined by the CTU encoding times $T(c_i)$. However, during the TRS partitioning stage, these values are not available, since the TRS partitioning stage takes place before the encoding of current frame. In order to overcome this lack of information, the values $T(c_i)$ are replaced during the TRS partitioning stage by estimated valuess noted $\tilde{T}(c_i)$. The Rate Distorsion Optimization (RDO) process presented in Section 2.3 and implemented in the VTM generates considerable disparities in CTU encoding times. For instance, a factor 5 between the minimum and maximum CTU encoding times in a frame is observed in the 32 first frames of High Definition (HD) sequence *BasketballDrive* QP27. This disparity makes the search for a time estimator both crucial and challenging. Several related works [103, 104] define $\tilde{T}(c_i)$ as the encoding time of the co-located CTU (located at the same spatial coordinates) in the closest temporal frame previously encoded. This choice is motivated by the temporal continuity of the video sequences content.

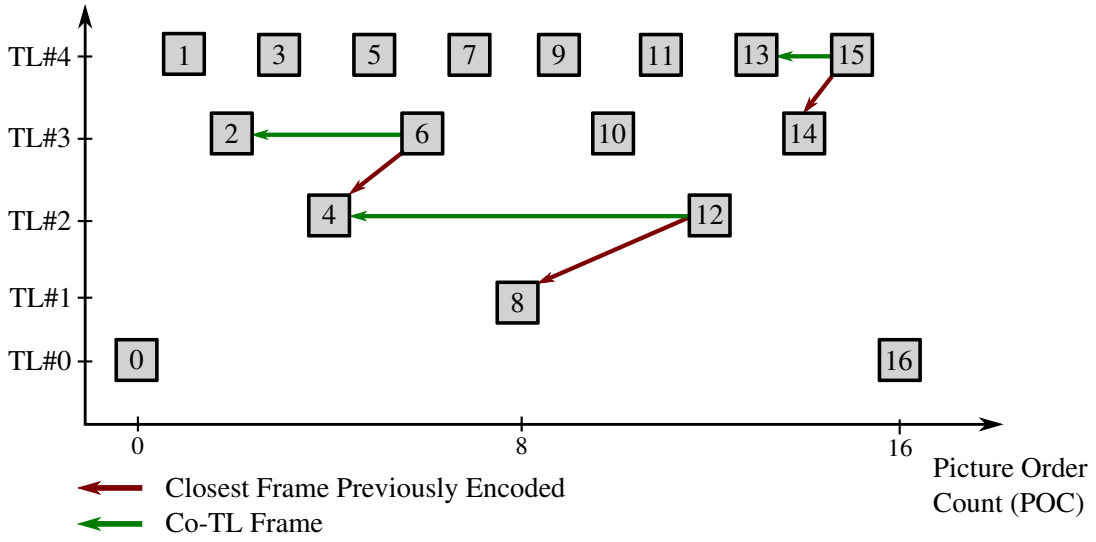


Figure 5.4 – *Difference between closest frame previously encoded and co-TL frame.*

Let the co-Temporal Layer (TL) frame refer to the previously encoded frame belonging to same temporal layer. Figure 5.4 illustrates the difference between the closest frame previously encoded and the co-TL frame, in the classical group of picture structure defined in the Common Test Conditions (CTC) [73]. For instance, during the encoding of frame #6, the CTU times are available for both frames #4 and #2. Frame #4 is temporally closer to frame #6 compared to frame #2. However due to the shared coding parameters of frames at similar TL, authors in [111] have shown that $T(c_i)$ is more correlated with the times of the co-located CTU in co-TL frame (frame #2), compared to the co-located CTU of the closest temporal frame (frame #6).

The following experiment corroborates the observations of paper [111]. Let us define for a given CTU c_i the error $\epsilon_{\tilde{T}}$ of a time estimator as:

$$\epsilon_{\tilde{T}}(c_i) = \frac{|T(c_i) - \tilde{T}(c_i)|}{T(c_i)} \tag{5.2}$$

with $T(c_i)$ the real execution time and $\tilde{T}(c_i)$ the estimated time. The average value for $\epsilon_{\tilde{T}}$ is computed across the 32 first frames of HD sequence *BasketballDrive* at QP27. The results are shown in TABLE 5.1 according to the time estimator and the TL. The error $\epsilon_{\tilde{T}}$ is in average 6% lower for Co-TL time estimator. This result confirms that Co-TL time estimator is more efficient compared to Closest Frame time estimator. TABLE 5.1 also shows that the higher the TL, the lower $\epsilon_{\tilde{T}}$ for both time estimators. This is explained by a lower temporal distance between frames belonging to higher TL, as shown by Figure 5.4. For instance, the estimator results for TL#1 are particularly high in TABLE 5.1. Since TL#1 only includes frame #8 in the Group of Pictures (GOP), the closest temporal frame previously encoded from frame #8 is frame #0. The CTUs content is very likely to have changed between these frames, inducing a high value $\epsilon_{\tilde{T}} = 69.3\%$. The closest Co-TL frame from frame #8 is the frame #8 in previous GOP. For the Co-TL time estimator also, this high temporal distance is the cause for the value $\epsilon_{\tilde{T}} = 63.1\%$.

Table 5.1 – *Average $\epsilon_{\tilde{T}}$ value (in %), computed on 32 first frames of HD sequence* BasketballDrive*, according to the time estimator and the TL*

	Time Estimator $\tilde{T}$	
Temporal Layer	**Closest Frame** $\epsilon_{\tilde{T}}$ (%)	**Co-TL Frame** $\epsilon_{\tilde{T}}$ (%)
TL #0	31.9	31.9
TL #1	69.3	63.1
TL #2	39.0	32.4
TL #3	36.6	27.3
TL #4	24.7	19.1
Average	**40.3**	**34.6**

Following these results, the selected estimator $\tilde{T}(c_i)$ is defined as the encoding time of the co-located CTU in the co-TL frame. The encoding time minimization technique consists in the search of a TRS partitioning P that minimizes the estimated $\tilde{T}(P)$, computed with $\tilde{T}(c_i)$ values as an input.

5.3.2 Limitation of Encoding Quality Losses

As mentioned in Section 5.2, prediction dependencies across RSs boundaries are disabled and entropy coding state is reinitialized at each RS. In order to limit the encoding quality loss induced by these restrictions, the optimal TRS partitioning P^* gathers similar spatial information inside the same RSs. This corresponds to a clustering problem [112] of the spatial information into the RSs, further called RS clustering. Since RSs with similar spatial information tend to have low luminance variances, the RS clustering searches the TRS partitioning P^* that minimizes the sum of luminance variance on all RSs. Eq. 5.3 computes the partitioning P^* where p_i is the value of luminance samples, and μ_j is the

Figure 5.5 – *TRS partitioning of* BQTerrace *frame #4, computed with slice clustering.*

mean of RS s_j luminance samples.

$$P^* = \underset{P}{argmin} \left[\sum_{s_j \in P} \sum_{p_i \in s_j} (p_i - \mu_j)^2 \right]. \tag{5.3}$$

Figure 5.5 shows the 8 RSs partitioning, obtained by solving Eq. 5.3 for frame #4 of sequence *BQTerrace*. In Figure 5.5, regions of the frame with similar spatial information tend to be clustered into the same RSs. The dark water of the river is almost entirely contained in RSs 6 and 7, and the light homogeneous regions of the frame are mainly included in RSs 0, 3 and 5. On the other hand, the RSs 1, 2 and 4 contain the regions with more complex spatial information.

5.3.3 Two Steps Slice Partitioning Search

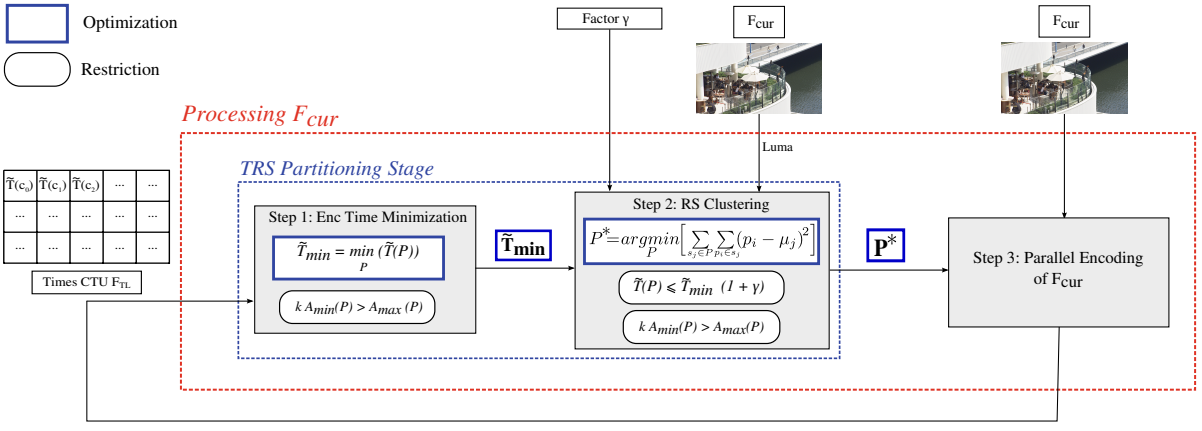


Figure 5.6 – *Proposed solution flowchart.*

The TRS partitioning in Figure 5.5 gathers similar spatial information inside the same RSs, but is far from optimal regarding the encoding time minimization. For instance,

the encoding at $QP = 27$ of RS #1 is 12 times slower compared to the encoding of RS #3, due among others to the greater area and spatial complexity of RS #1 compared to RS #3. The encoding time of the considered frame is therefore sub-optimal due to the high encoding time of RS #1. In order to reduce such imbalances between RSs encoding times, the proposed solution combines the RS clustering (Section 5.3.2) with the encoding time minimization technique (Section 5.3.1).

The proposed solution is represented as a flowchart in Figure 5.6. The TRS partitioning stage, enclosed in the blue dashed box, is applied prior to the parallel encoding of current frame F_{cur}, enclosed in the red dashed box. The TRS partitioning stage is divided into 2 distinct steps. The first step is called encoding time minimization step. This step computes the minimum estimated encoding time, defined by Equation 5.4 and noted $\tilde{T}_{min}$.

$$\tilde{T}_{min} = \min_{P}(\tilde{T}(P)) \tag{5.4}$$

The encoding time minimization step takes the CTU times of the co-TL frame F_{TL} as input.

The second step of the TRS partitioning stage computes the RS clustering of F_{cur}, under encoding time constraint. This step takes as inputs $\tilde{T}_{min}$ estimated during previous step, the luminance samples of F_{cur}, and a relaxation factor γ that manages the trade-off between encoding time and encoding quality. The possible values for $\tilde{T}(P)$ are determined by Eq. 5.5.

$$\begin{aligned} P^* = \underset{P}{argmin} \left[\sum_{s_j \in P} \sum_{p_i \in s_j} (p_i - \mu_j)^2 \right], \\ \tilde{T}(P) \leq (1+\gamma)\tilde{T}_{min}. \end{aligned} \tag{5.5}$$

When $\gamma = 0$, only the partitioning P that minimizes the estimated time is considered, since $\tilde{T}(P) = \tilde{T}_{min}$. When γ increases, more partitioning opportunities are offered to the RS clustering, and therefore higher weight is given to encoding quality compared to encoding time minimization. The parameter γ is therefore a means for the encoder to manage the trade-off, according to the requirement.

The aim of this Chapter is to show the relevance of a solution combining the 2 complementary steps previously presented. For this reason, a near exhaustive search is conducted to compute both $\tilde{T}_{min}$ and RS clustering. As shown in Figure 5.6, the only constraint given to the search algorithm:

$$kA_{min}(P) > A_{max}(P),$$

with A_{min} and A_{max} the area of the smallest and the largest RSs, respectively. The constant k is set to 4 in this work in order to contain search complexity. The choice of less complex heuristics for the TRS partitioning stage is a distinct issue, that will be part of future works. The global complexity overhead induced by the TRS partitioning stage is nonetheless measured and discussed further in this Chapter.

5.4 Experimental Results

This section presents the experimental setup, as well as the performance of the proposed TRS partitioning solution.

5.4.1 Experimental Setup

The following experiments are conducted under VTM-6.2 software, built with gcc compiler version 7.4.0, under Linux version 4.15.0-74-generic as distributed in Ubuntu-18.04.1. The platform setup is composed of Central Processing Units (CPUs) Intel(R) Xeon(R) E5-2690 v3 clocked at 2.60 GHz, each of them disposing of 12 cores. The cores have each 768KB L1 cache, 3MB L2 cache and 30MB L3 cache.

The high-level parallelism structures included in VVC standard allow to tackle complexity increase on multi-core processors. This complexity increase raises a critical issue mainly for high resolution video sequences. For this reason, the test sequences selected in this work contain 4 Ultra High Definition (UHD) and 5 HD sequences included in the CTC [73]: *CatRobot1*, *DaylightRoad2*, *FoodMarket4*, *Tango2* (UHD), and *BQTerrace*, *Cactus*, *MarketPlace*, *RitualDance* (HD). The test sequences are encoded under RA configuration at four QP values: 22, 27, 32, 37. The performance of our TRS partitioning solution is assessed by measuring the trade-off between the encoding quality using the Bjøntegaard Delta BitRate (BD-BR) [13] and the multi-thread speed-up ξ, defined by Eq. 5.6.

$$\xi = \frac{1}{4} \sum_{QP_i \in \{22,27,32,37\}} \frac{T_A(QP_i)}{T_R(QP_i)} \tag{5.6}$$

$T_A(QP_i)$ and $T_R(QP_i)$ are the anchor time (encoded with 1 RS and 1 single thread) and reduced time (encoded with N RSs and N threads) spent to encode the video sequence with QP_i, respectively. The overhead induced by TRS partitioning stage is further noted θ and measured in percentage of T_R.

5.4.2 Performance of the Proposed Solution

The theoretical upper bound in terms of speed-up, noted ξ_{max}, for the proposed solution is computed with the Amdahl law [113]. Let s be the sequential part (in %) of an application. The upper bound ξ_{max} obtainable with n threads is expressed by Eq. 5.7.

$$\xi_{max}(n) = \frac{1}{s + \frac{1-s}{n}} \tag{5.7}$$

In our case, the sequential portion of VTM-6.2 encoder contains the data initialization, entropy, in-loop filter and bitstream writing stages. All together, these stages represent 4% of the encoding time in average across test sequences and QP values. Therefore, Eq. 5.7 provides the following upper bounds: $\boldsymbol{\xi_{max}(4) = 3.57}$, $\boldsymbol{\xi_{max}(8) = 6.25}$ and $\boldsymbol{\xi_{max}(12) = 8.33}$.

Figure 5.7 shows the BD-BR and Speed-up offered by the proposed solution, according to the number of threads and sequence resolution. In order to evaluate the performance of the proposed solution, the results of the uniform TRS partitioning are also displayed and surrounded by the black boxes. The uniform TRS partitioning is an usual and straightforward technique that partitions the frame in a grid of the same RS dimension. As mentioned in Section 5.3.3, the relaxation factor γ manages the trade-off between encoding quality and encoding time minimization induced by the proposed TRS partitioning. Four values of γ are tested and displayed in Figure 5.7. From left to right on the subplots: $\gamma = \infty$, $\gamma = 0.3$, $\gamma = 0.1$ and $\gamma = 0$. Finally, the upper bounds ξ_{max} provided by Amdahl law are represented by the red vertical lines. In Figure 5.7, the points further down to the right offer the best performance, since they maximize the speed-up ξ while minimizing the BD-BR increase.

The proposed solution offers various trade-offs depending on the γ value, for all resolutions and number of threads. The curves of the proposed solution are all growing, i.e. a

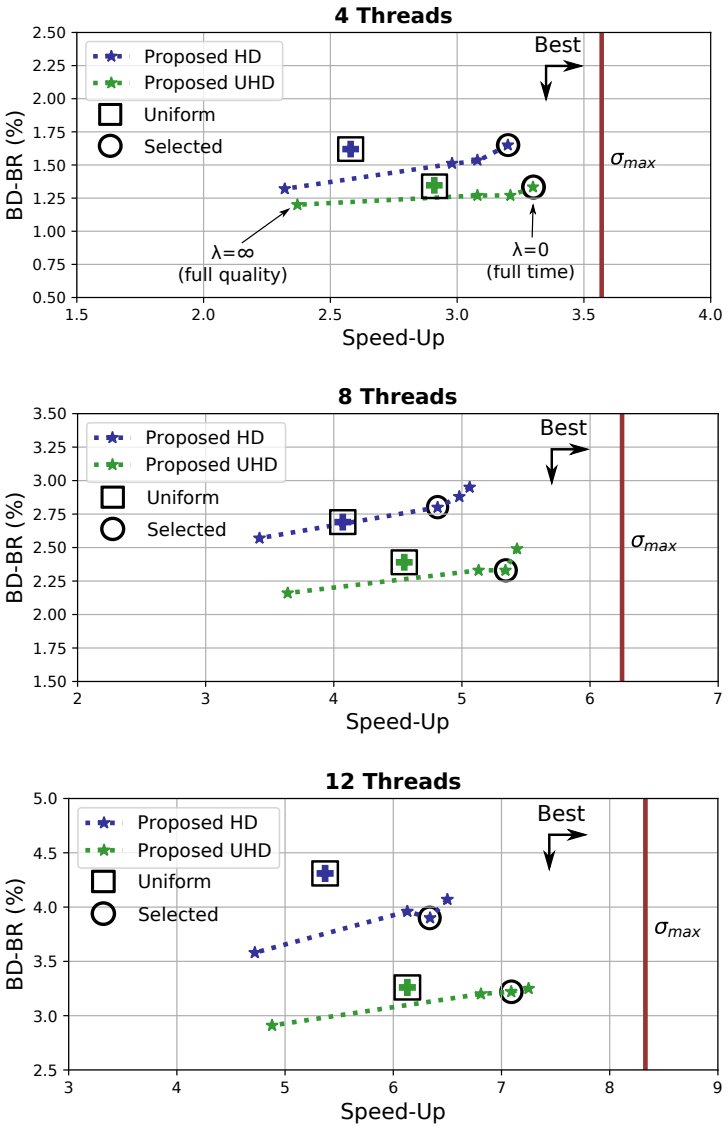


Figure 5.7 – *BD-BR and Speed-up offered by uniform and proposed solution, according to the number of threads. From left to right on each subplot:* $\gamma = \infty$, $\gamma = 0.3$, $\gamma = 0.1$ *and* $\gamma = 0$.

higher ξ is coupled with a higher BD-BR. This behavior is coherent with the explanations of Section 5.3.3. When γ decreases, more importance is given to the speed-up with regard to RS clustering (i.e encoding quality). Two exceptions are however noticeable on the curves of UHD content with 4 threads and of HD content with 12 threads, respectively. These exceptions show that in some corner cases a higher flexibility given to the RS clustering leads to a higher BD-BR, which is not the expected behavior. In conclusion, the RS clustering presented in Section 5.3.2 is highly but not perfectly correlated with the BD-BR.

The results obtained by the uniform TRS partitioning are surrounded by the black boxes in Figure 5.7. The proposed TRS partitioning solution applied with $\gamma \in \{0, 0.1, 0.3\}$ reduces significantly the distance to the upper bounds ξ_{max}, compared to uniform TRS partitioning. In order to provide a more accurate comparison with uniform TRS partitioning, the points offering the best trade-off between ξ and BD-BR are surrounded with a black circle in Figure 5.7, according to the resolution and number of threads. TABLE 5.2 presents the

Table 5.2 – *Average speed-up ξ, BD-BR and overhead θ obtained by both uniform and proposed TRS partitioning, according to the resolution and number of threads n.*

		HD		UHD	
		Unif	**Proposed**	**Unif**	**Proposed**
$n = 4$	BD-BR (%)	1.62	$\gamma = 0$ **1.57**	1.31	$\gamma = 0$ **1.27**
	Speed-up ξ	2.68	**3.10**	2.91	**3.27**
	θ(%)		0.0		0.0
$n = 8$	BD-BR (%)	2.69	$\gamma = 0.3$ 2.80	2.39	$\gamma = 0.1$ **2.33**
	Speed-up ξ	4.27	**5.07**	4.55	**5.34**
	θ(%)		0.01		0.08
$n = 12$	BD-BR (%)	4.31	$\gamma = 0.1$ **3.90**	3.26	$\gamma = 0.1$ **3.20**
	Speed-up ξ	5.57	**6.44**	6.13	**7.09**
	θ(%)		0.54		1.84

average results obtained with these selected points compared to uniform TRS partitioning, according to the resolution and number of threads n.

TABLE 5.2 shows that the proposed TRS partitioning solution enables better results compared to uniform TRS partitioning in term of ξ, regardless the resolution and number of threads n. The TRS partitioning solution obtains ξ values 0.36 and 0.94 higher compared to uniform partitioning, for UHD content with $n = 4$ and $n = 12$, respectively. This significant ξ increase proves the efficiency of the encoding time minimization step, presented in Section 5.3.1. It is important to note that the encoding time of every frame is reduced. Therefore both speed-up and latency are improved equally by the proposed solution.

In term of BD-BR, the results of the proposed solution with the selected γ values are slightly better (around -0.05%) compared to uniform TRS partitioning. Two exceptions are however noticeable. The BD-BR decrease is substantial (-0.41%) for HD content with $n = 12$, and the only case for which the BD-BR is slightly higher is for HD content with $n = 8$ ($+0.11\%$). The related works in HEVC minimizing the BD-BR reported 0.16% [109] and 0.10% [111] average BD-BR decrease with 8 threads on HD and UHD content. Our results in term of BD-BR are therefore close to the results of previously mentioned works, even though these works minimize the BD-BR without taking into consideration the speed-up optimization.

The conclusion of TABLE 5.2 is that the proposed solution is highly effective to improve the speed-up offered by the TRS partitioning in VVC compared to uniform RS partitioning. Moreover, the BD-BR results are slightly better than uniform RS partitioning. The variation of γ value is however not sufficient to decrease significantly the BD-BR, except for HD content with $n = 12$. Regarding the overhead θ, the values are half induced by the encoding time minimization step, and half by the encoding quality loss limitation step. The values are negligible when $n = 4$ and $n = 8$. For $n = 12$, θ is greater than 0.5% due to the almost exhaustive search implemented (see Section 5.3.3). This overhead is relatively high considering the high complexity of the VTM. Nevertheless it allows to obtain optimal performance and prove the viability of our approach. For professional or real time application, we are confident that the investigation of simple heuristics in future works will reduce

greatly θ, without degrading the results presented in TABLE 5.2.

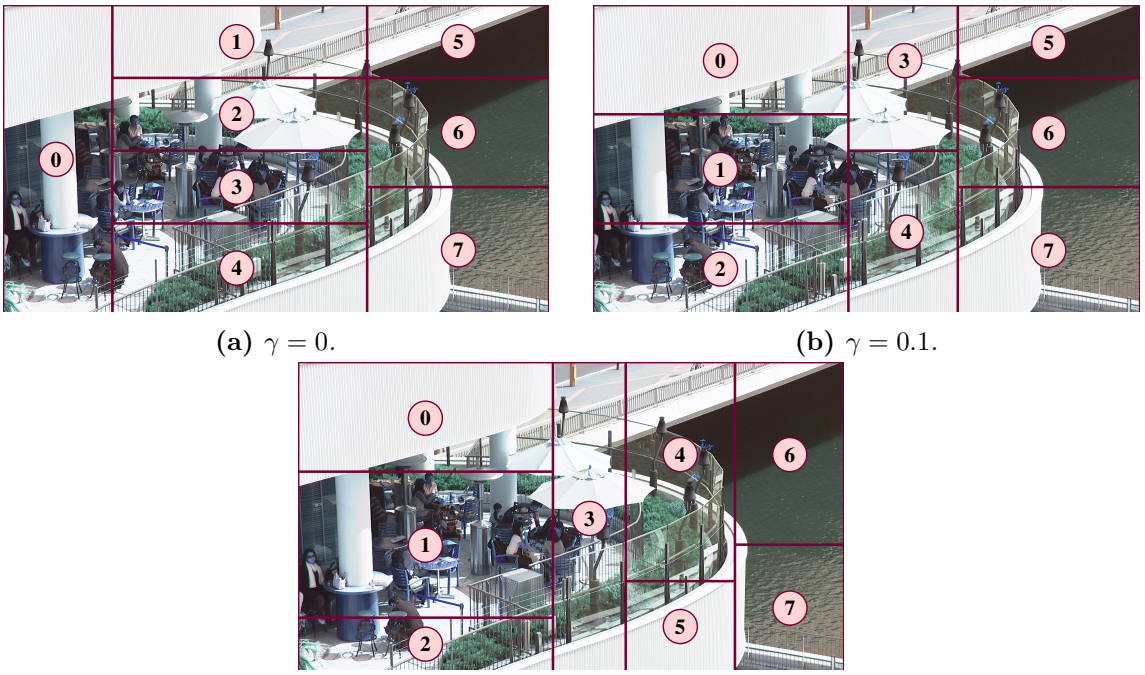
5.4.3 Impact of relaxation factor γ

Table 5.3 – *Proposed solution with $\gamma = 0$ and $\gamma = 0.1$, encoded with 8 threads, according to sequence.*

8 Threads				
	Proposed Solution $\gamma = 0$		**Proposed Solution $\gamma = 0.1$**	
Sequence	BD-BR (in %)	ξ	BD-BR (in %)	ξ
CatRobot1	1.38	5.24	1.14	5.19
DaylightRoad	1.82	5.79	1.70	5.70
FoodMarket	4.09	5.16	3.85	5.10
Tango2	2.67	5.54	2.61	5.40
BasketballDrive	3.27	5.13	3.23	5.08
BQTerrace	2.29	4.95	2.17	4.82
Cactus	2.33	4.87	2.17	4.85
MarketPlace	3.52	5.13	3.48	5.06
RitualDance	3.25	4.76	3.28	4.75
Average	**2.74**	**5.17**	**2.62**	**5.10**

TABLE 5.3 shows the performance of the proposed solution with $\gamma = 0$ and $\gamma = 0.1$ running with 8 threads, according to the test sequence. As explained in Section 5.3.3, the higher γ, the more importance is given to encoding quality with regard to the speed-up. The results of TABLE 5.3 are coherent with this explanation. Indeed, for almost every sequence the proposed solution with $\gamma = 0.1$ enables better BD-BR but lower ξ compared to the proposed solution with $\gamma = 0$. The only exception is sequence *RitualDance*, for which the BD-BR is not improved with $\gamma = 0.1$. In average, the BD-BR is 0.12% better when selecting $\gamma = 0.1$, without degrading significantly ξ (-0.07). The results are particularly noticeable for sequence *FoodMarket*. For this sequence, the BD-BR is 0.24% better and ξ only decreases by 0.06% when selecting $\gamma = 0.1$, compared to the proposed solution with $\gamma = 0$.

Figure 5.8 provides 3 examples of the proposed TRS partitioning solution applied to *BQTerrace* frame #9, according to γ value ($\gamma = 0$, $\gamma = 0.1$ and $\gamma = \infty$). With $\gamma = 0$, the TRS partitioning in Figure 5.8a only focuses on minimizing the encoding time. To that end, the areas of the RSs are almost evenly balanced. The RSs #2 and #3 (detailed regions) are slightly smaller compared to RSs #0, #4 or #7 (homogeneous regions), since more detailed regions induce higher encoding time. However, the encoding quality loss limitation is not taken into account in Figure 5.8a. The regions of the frame with similar spatial information are not necessarily clustered into the same RSs. For instance, the RSs #6 and #7 contain simultaneously the dark water of the river and the edge of the terrace. The RSs #0 is composed part of the homogeneous white roof, and part of the people on the terrace.



(a) $\gamma = 0$. **(b)** $\gamma = 0.1$.

(c) $\gamma = \infty$.

Figure 5.8 – *Proposed TRS partitioning of* BQTerrace *frame #9, according to the* γ *value.*

With $\gamma = \infty$, the TRS partitioning in Figure 5.8c limits encoding quality loss by gathering similar spatial information inside the same RSs, as explained in Section 5.3.2. However, the TRS partitioning is far from optimal regarding the encoding time minimization. The encoding time of RS #1 is considerably higher compared to RSs #3 and #7, due among others to the greater area and spatial details in RS #1.

The TRS partitioning in Figure 5.8b, obtained with $\gamma = 0.1$, offers a trade-off between $\gamma = 0$ and $\gamma = \infty$. In order to limit encoding quality loss, the white roof is almost entirely contained in RS #0 and the details of the terrace are mainly assigned to RS #1. Moreover, in order to minimize the encoding time, the areas of the regions assigned to the RSs are way more balanced compared to Figure 5.8c. Indeed, the tile with the highest encoding time (RS #1) is 1.5 times smaller in Figure 5.8b compared to Figure 5.8c. The encoding time will therefore be lower with $\gamma = 0.1$.

5.5 Conclusion

In this Chapter, a dynamic TRS partitioning technique is proposed for next generation video standard VVC. The proposed solution combines two techniques to minimize multi-thread encoding time and encoding quality loss, respectively. A relaxation factor γ is applied, allowing to select a trade-off between encoding time and encoding quality. The experiments show that the proposed solution decreases significantly multi-thread encoding time, with slightly better encoding quality, compared to uniform RS partitioning. Future works will focus among other points on the improvement of the CTU time estimator, used in the encoding time minimization step. Instead of simply relying on the co-located CTU times of the co-TL frame, future solutions will rely on CTU deduced by motion information. The investigation of lightweight heuristics for the TRS partitioning stage will also be part

of future works. We are confident they will reduce drastically the overhead, especially for 12 threads encodings of UHD content.

CHAPTER 6

Parallel and Real-Time VVC Decoder in AI configuration

6.1 Introduction

In order to reach the bit-rate savings promised by the Versatile Video Coding (VVC) standard over High Efficiency Video Coding (HEVC), computationally expensive tools have been added also at decoder side, as explained in Chapter 3. At the decoder side, the computational complexity increase of VVC standard is approximately 2 times compared to HEVC in All Intra (AI) coding configurations [6]. As presented in Section 6.3, the energy consumption and the processing time of the decoder are generally reduced through parallel processing techniques. For the decoder proposed in this Chapter, different levels of parallelism have been investigated to reduce the implementation cost.

In this Chapter, we propose a real-time and parallel VVC decoder in AI configuration based on the open source *openVVC* project. The *openVVC* decoder is developed by the VAADER team of IETR laboratory [1] in C programming language, and is integrated as a dynamic library inside FFmpeg player [47]. The used version of the decoder is compliant with VVC Test Model (VTM)-9.0 bitstreams, that are very close from the finalized VVC standard. The decoder relies on data-level parallelism techniques (Single Instruction on Multiple Data (SIMD) optimization) to reduce the decoding time of the most computationally complex operations. The multiprocessor architectures are exploited through tile-level or frame-level parallelism. In this chapter, the decoding frame-rate is evaluated through the number of decoded frames per second, noted fps. It measures the speed in term of decoding time of *openVVC*. By combining data-level parallelism with 8 threads parallelism, *openVVC* in AI configuration achieves 100 fps and 30 fps frame-rates for High Definition (HD) and Ultra High Definition (UHD) content, respectively. The main asset of the proposed decoder lies in its very low memory usage. For instance, the sequential decoding of HD and UHD content only requires 20MB and 70MB of memory, respectively.

The contributions of this work to *openVVC* software are the following. The implementation from scratch of Adaptive Loop Filter (ALF) and Sample Adaptive Offset (SAO) filters. The design of filter buffers that minimize the memory usage of in-loop filtering process. The consolidation of tile parallelism so that the software supports both dynamic and uniform tile partitioning. With these pre-requisites, we reimplemented in VTM-9.0 the dynamic tile partitioning solution proposed in Chapter 5 for VTM-6.2, in order to generate bitstreams compatible with *openVVC*. The experimental results show that the decoding

[1] https://www.ietr.fr/spip.php?article1604

frame-rates obtained with dynamic tiles are equivalent to the frame-rates obtained with uniform tiles. It proves that the dynamic tile partitioning proposed in Chapter 5 is an effective asset at encoder side, without being a burden for the decoding process.

The rest of the Chapter is organized as follows. Section 6.2 presents the general block diagram of a VVC decoder and describes the main decoding stages. In Section 6.3, the principal techniques used for the decoder parallelization are described along with the related works relying on these techniques. The proposed *openVVC* decoder is presented in Section 6.4. The general code structure, the buffer sizes, the in-loop filtering process and the parallelization are described. Section 6.5 presents the experimental setup, as well as the performance in term of memory usage and frame-rate of the proposed *openVVC* decoder in AI configuration. The results are presented first enabling frame-level parallelism and second enabling tile-level parallelism. Finally, Section 6.6 concludes this chapter.

6.2 Overview of VVC Decoder

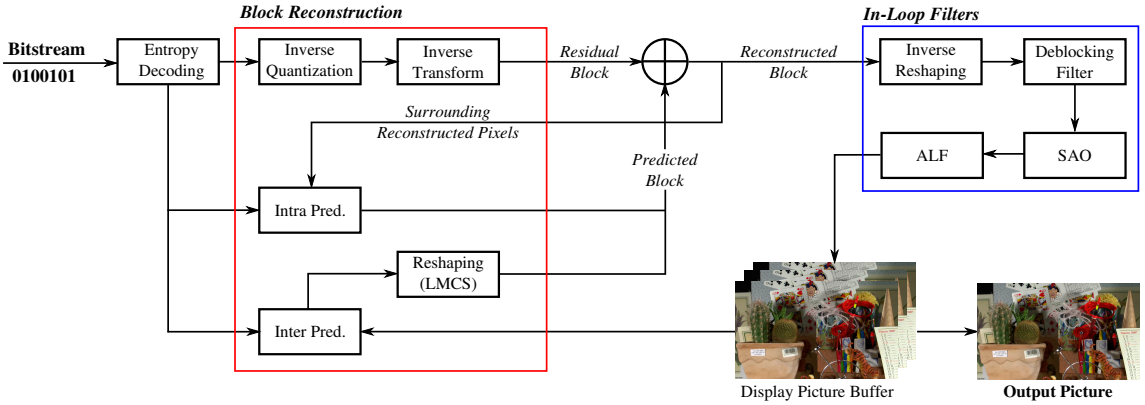


Figure 6.1 – *VVC decoder block diagram.*

Figure 6.1 presents the general block diagram of a VVC decoder. The decoding process is based on the classical hybrid scheme, taking advantage of both intra/inter prediction and transform coding. The decoder converts an input bitstream composed of binary symbols (bits) into a decoded pictures. Each block in the diagram corresponds to one of the main decoding stages. These decodig blocks are briefly decribed in this Section.

Entropy decoding The first decoding stage is the entropy decoding of the bitstream. In VVC, the Context Adaptive Binary Arithmetic Coding (CABAC) is used as an entropy engine [114]. During the entropy decoding stage, the CABAC reads the bitstream bit by bit and converts the binary symbols into non-binary symbols. When a bit is read, the state of the corresponding CABAC context is updated, allowing to read correctly the following bit of this CABAC context. The entropy decoding provides the input data for all the other main decoding stages.

Block Reconstruction At encoder side, the block partitioning scheme divides the frame into appropriate block sizes according to the local activity of the pixels. At decoder side, the block sizes are retrieved by the entropy decoder. Each block is reconstructed applying the stages inside the red box in Figure 6.1 : prediction, reshaping, inverse residual quantization and transform. The block is either intra or inter predicted (or both Intra and inter coded).

When the block is intra predicted, the prediction requires the intra prediction mode, as well as the previously reconstructed pixels of surrounding blocks. Motion Compensation (MC) is performed by the inter prediction block for blocks coded in inter mode. It uses the motion vectors information, as well as the content of the previously decoded pictures, contained in the Decoded Picture Buffer (DPB). In both intra and inter prediction cases, a predicted block is reconstructed. The inverse quantization and inverse transform decoding stages compute the residual block from the residual transform coefficients. The residual block reconstructed by inverse quantization and inverse transform is then added to the predicted block of the prediction stages, forming the reconstructed block. The reconstructed blocks are then processed by the in-loop filters, which aim to remove encoding artifacts and further enhance the visual quality. In VVC, a new stage called the Luma Mapping with Chroma Scaling (LMCS) has been introduced [40]. The LMCS modifies the predicted values of inter predicted blocks by reshaping (i.e. redistributing) the samples across the entire possible value range. After reconstruction, the inverse reshaping is performed before the Deblocking Filter (DBF) stage.

In-loop Filters Four in-loop filters are performed on the reconstructed pixels, including the inverse LMCS, DBF, the SAO and the ALF. The in-loop filters have been described with more detail in Section 2.5.7. Once the in-loop filters are performed on the entire picture, the decoded picture is inserted in the DPB. It will remain in the DPB until it is any more required as reference for MC of the next decoded frames. The decoded picture with Picture Order Count (POC) number p is displayed when the previous picture in display order with POC number $p-1$ is entirely decoded and displayed.

6.3 Sate of the Art for Decoding Parallelism

As explained in Section 5.1, techniques for parallel processing essentially operate at three levels of parallelism: data-level, high-level and frame level. In this Section, the main techniques for the decoder parallelization are described along with the related works relying on these techniques. Since only few works on VVC decoders are currently available, most of the presented related works focus on the parallelism opportunities of HEVC decoding process. It is nonetheless possible to adapt the presented related works to VVC decoders with little extra work, considering that the operating procedure of HEVC decoders is very similar to VVC decoders.

6.3.1 Single Instruction on Multiple Data

The techniques relying on SIMD architectures are part of data-level parallelism techniques. With a single instruction, SIMD architectures apply simultaneously an operation on a vector of data, producing a vector of results. This abbreviation is used as opposed to Single Instruction on Single Data (SISD), the usual architectures, and Multiple Instructions on Multiple Data (MIMD) the architectures using several processors with independent memories. Figure 6.2 provides an example of SIMD extensions SSE [115] operating on 128 bits registers. The result vector is obtained with a single SIMD operation, reducing the execution time compared to multiple operations on SISD architectures. In order to apply these operations, the data must be properly located in memory. If not, the data must be displaced using *pack* and *unpack* calls, adding additional overhead. Typically, the calculations that benefit from SIMD architectures are those including elementary operations on already created vectors and matrices. In the decoding process, these calculations include

among others the application of the diamond shape filters of the ALF, the process of the MC interpolation filter, the derivation of reconstructed samples in intra prediction and the Inverse Transform applied on the residual transform coefficients. On the other hand, the entropy coding stage does not include significant data-level parallelism which makes it difficult to use SIMD methods.

Related works widely rely on SIMD architectures to speed up the decoding process. The paper [116] provides a good summary of the possible SIMD accelerations in an HEVC decoder. The authors discuss the challenges of the SIMD implementation for many of the aforementioned calculations, and provides experimental results on 14 different platforms. Yan *et al.* [117] also rely on intensive SIMD acceleration to reduce the HEVC decoding time. In the particular case of the scalable extension of HEVC standard, named Scalable High Efficiency Video Coding (SHVC), Hamidouche *et al.* [68] optimize among other the upsampling of the base layer frame with SIMD methods.

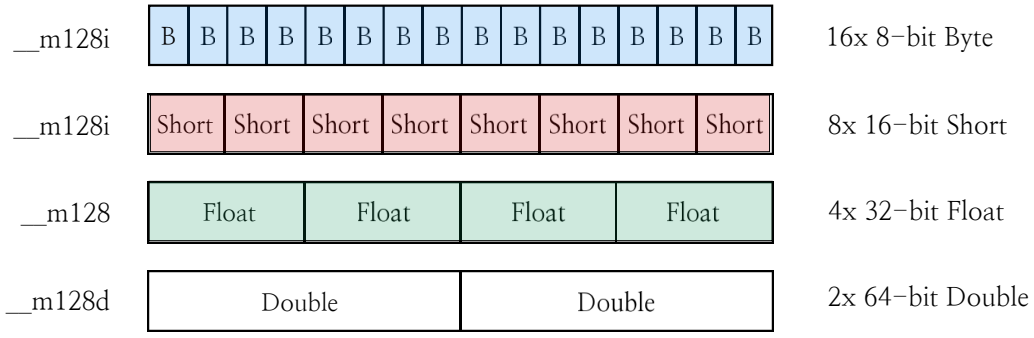


Figure 6.2 – *Example of SIMD extensions SSE [115] operating on 128 bits registers.*

6.3.2 Constrained Bitstream

This subsection presents tiles and Wavefront Parallel Processing (WPP) high level parallelism techniques. They have been standardized in both HEVC and VVC, to facilitate the use of parallel processing architectures for encoding and decoding. These tools must be enabled at the encoder side, and additional information is transmitted in the bitstream in order to decode the bitstream in parallel. Therefore, bitstreams containing tiles and WPP are considered as constrained.

6.3.2.1 Tiles

Tile partitioning in VVC has been describe in Section 5.2.1.1. Prediction dependencies across tile boundaries are broken and entropy encoding state is reinitialized for each tile. These prerequisites ensure that tiles are independently decodable, allowing several threads to decode simultaneously the same frame. The in-loop filtering stages across tile boundaries must however be performed when the reconstructed pixels of both tiles are available.

In paper [118], the tile partitioning is adapted at the encoder side in order to minimize the decoding time. The decoding load imbalance between tiles is reduced based on the relation between the decoding time and the number of coded bits of a given Coding Tree Unit (CTU). On computing systems with asymmetric processors, authors in [119] take advantage of tile partitioning flexibility in order to optimize HEVC decoding time on these specific platforms. Asymmetric tile work load is delivered at encoder side by varying tile sizes. At decoder side, the bigger and smaller tiles are further allocated to faster and slower threads, respectively.

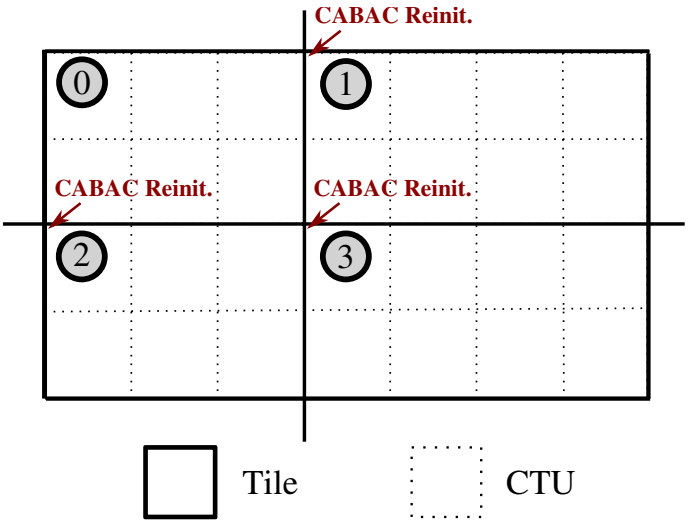


Figure 6.3 – *Illustration of tile partitioning: grid of 4 tiles labeled from 0 to 3.*

6.3.2.2 Wavefront Parallel Processing

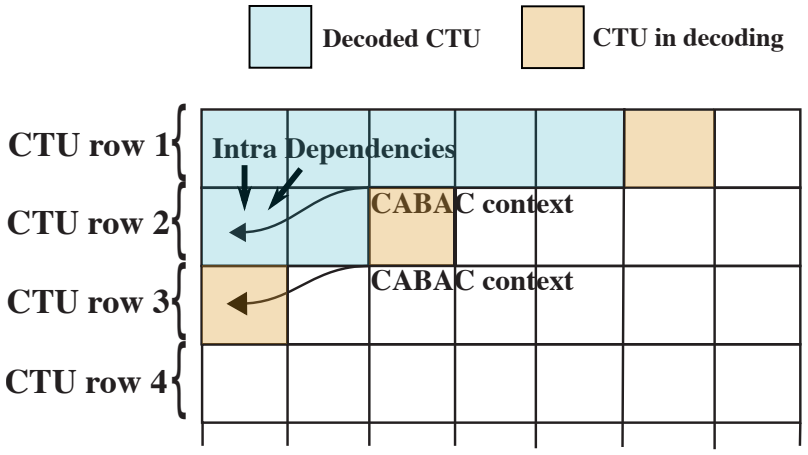


Figure 6.4 – *Principle of the WPP approach in the decoding process.*

The WPP tool, enabled at encoder side, divides the frame into CTU rows [120]. As shown in Figure 6.4, the CABAC context is reinitialized at the start of each CTU row with the CABAC context of the third CTU in the preceding row. The decoding of a row may therefore begin when the two first CTUs in the preceding row are reconstructed, since it ensures that the decisions needed for prediction and CABAC reinitialization are made in the preceding row. These constraints allow several processing threads to decode the frame in parallel, with a delay of two CTUs between adjacent rows. The in-loop filtering of current CTU requires the reconstructed pixels of the following row, also inducing a delay in the WPP process. The propagation of these delays across the frame rows induces parallelism inefficiencies, especially when the number of processing threads increases. For this reason, many works including [121] and [122] combine WPP tool with frame-level parallelism in their solutions. Authors in [122] show that the decoder combining frame-level parallelism and WPP enables much higher frame-rates compared to the decoder using WPP alone, as soon as the number of threads is superior to 5 for high resolutions video.

6.3.3 Unconstrained Bitstream

In opposition with decoding parallelism techniques that require specific processing at encoder side, the parallelism techniques presented in this section are suitable to decode any input bitstream.

6.3.3.1 Frame-level Parallelism

When frame-level parallelism is activated, the decoder processes a group of frames in parallel, with each frame entirely decoded by a single thread. The frames are simultaneously decoded, under the restriction that the MC dependencies are satisfied. Frame-level parallelism is particularly efficient in AI coding configuration, since there are no MC dependencies. However, the frame-level parallelism suffers from various limitations. The decoding latency is not reduced since a single thread is assigned to the decoding of a frame. The performance of the frame-level parallelism also highly depends on the motion activity in the sequence and on the ranges of Motion Vectors (MVs) used for MC. Based on this observation, Chi *et al.* [121] restrict the downwards component of movement vectors to 1/4 of image height. This restriction reduces the MC dependencies for the decoding of consecutive frames. Finally, the decoding of multiple frames induces a large memory overhead since the decoder must store simultaneously multiple frame buffers. For systems with strong memory constraints such as mobile devices, this memory overhead is a serious limitation. For instance in the context of mobile and hand-held devices, authors in [123] rely on high-level parallelism tools rather than frame-level parallelism to accelerate the HEVC decoding process.

6.3.3.2 Parallelism at Stage Level

In this work, stage-level techniques refer to techniques in which several threads process simultaneously the same decoding stage, exploiting its specific parallel opportunities. They are part of high-level techniques, but have not been standardized and do not require constrained bitstreams. Entropy decoding process is the most difficult stage to parallelize in the decoder, due to its sequential nature. In order to process the CABAC in parallel, CABAC reinitialization must be included in the bitstream at encoder side, as presented in tiles or WPP high level parallelism techniques. Habermann *et al.* [124], propose three solutions to improve CABAC processing in WPP for low-delay applications. However, these solutions require constrained bitstreams as an input.

When no constrains are imposed at encoder side, the entropy decoding stage is often performed sequentially as a pre-processing of the frame decoding. It provides the input data for all the decoding stages described in Figure 6.1. This approach, adopted in recent works on VVC [125] and [126], allows to parallelize the decoding at stage-level, but requires the storage of the CABAC output on a frame basis. The work [126] is the first to provide experimental results on a real-time VVC decoder. The authors propose intensive SIMD optimizations, that achieve 70% decoding time reduction, coupled with stage-level parallelization. This parallelization approach does not require constrained bitstreams and obtains interesting speed-up results. However, the memory usage of the decoder is not optimized and not discussed. Another approach consists in retrieving and storing the CABAC output data only for the reconstruction of a single CTU [68]. With this approach, the reconstruction of the CTUs is processed sequentially but the memory consumption to store CABAC output is negligible.

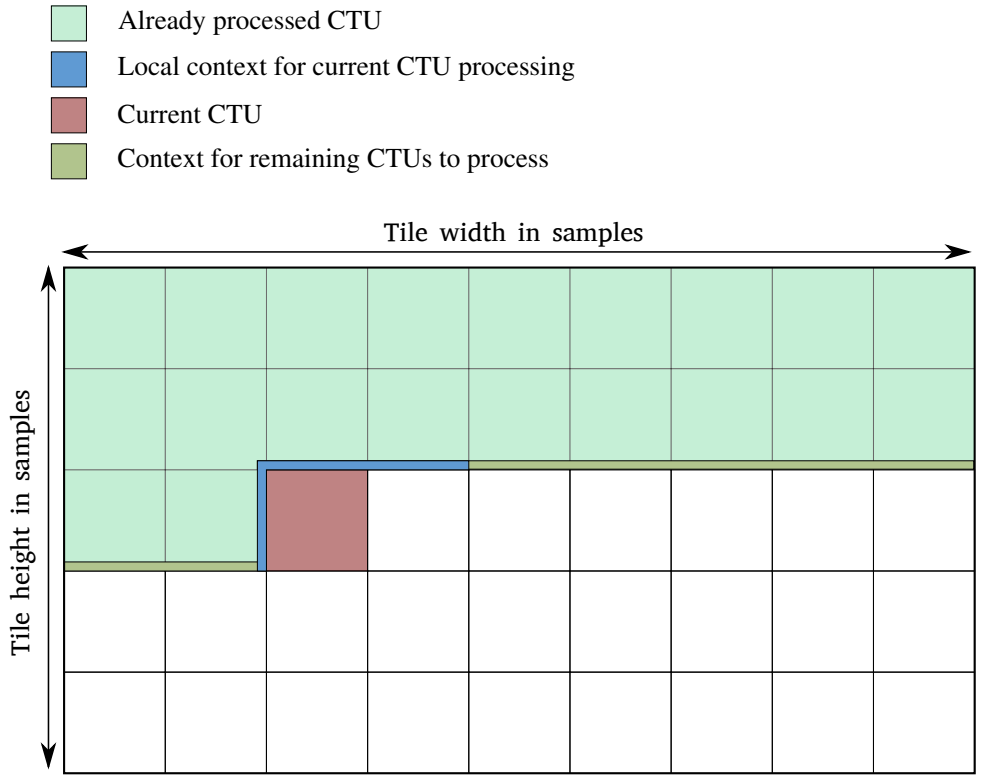


Figure 6.5 – openVVC*: local context for raster scan CTU processing.*

For in-loop filtering, a classical approach consists in processing the in-loop filter in a separate pass when the entire frame is reconstructed. Very little synchronization is needed in this case since the in-loop filter are applied one by one on the entire frame. Kotra *et al.* [127] provide three parallel implementations of the DBF for HEVC decoding, operating on the entire reconstructed frame. The limitation of this approach is that the final pixels needed as reference for MC are available only after reconstruction, DBF and SAO are processed on the entire frame. When in-loop filters are performed on a CTU level, the final pixels are available with a smaller delay. For instance in [128], all the steps necessary to output most of the final pixels of a CTU are finished with a delay of less than 2 CTUs. This approach improves the frame-level parallelism for inter decoding.

6.4 Proposed OpenVVC Decoder

The proposed VVC decoder is based on the open source *openVVC* project. The *openVVC* decoder is developed in C programming language and integrated as a dynamic library inside FFmpeg player [47]. It implements a conforming VVC decoder and currently supports most of the new tools introduced in the VVC standard. The actors developing *openVVC* also developed the open source software decoder OpenHEVC [129] compliant with HEVC standard. OpenHEVC is the state of the art of real time HEVC software decoding [130] and is used in widespread players such as VLC [2] and GPAC [131].

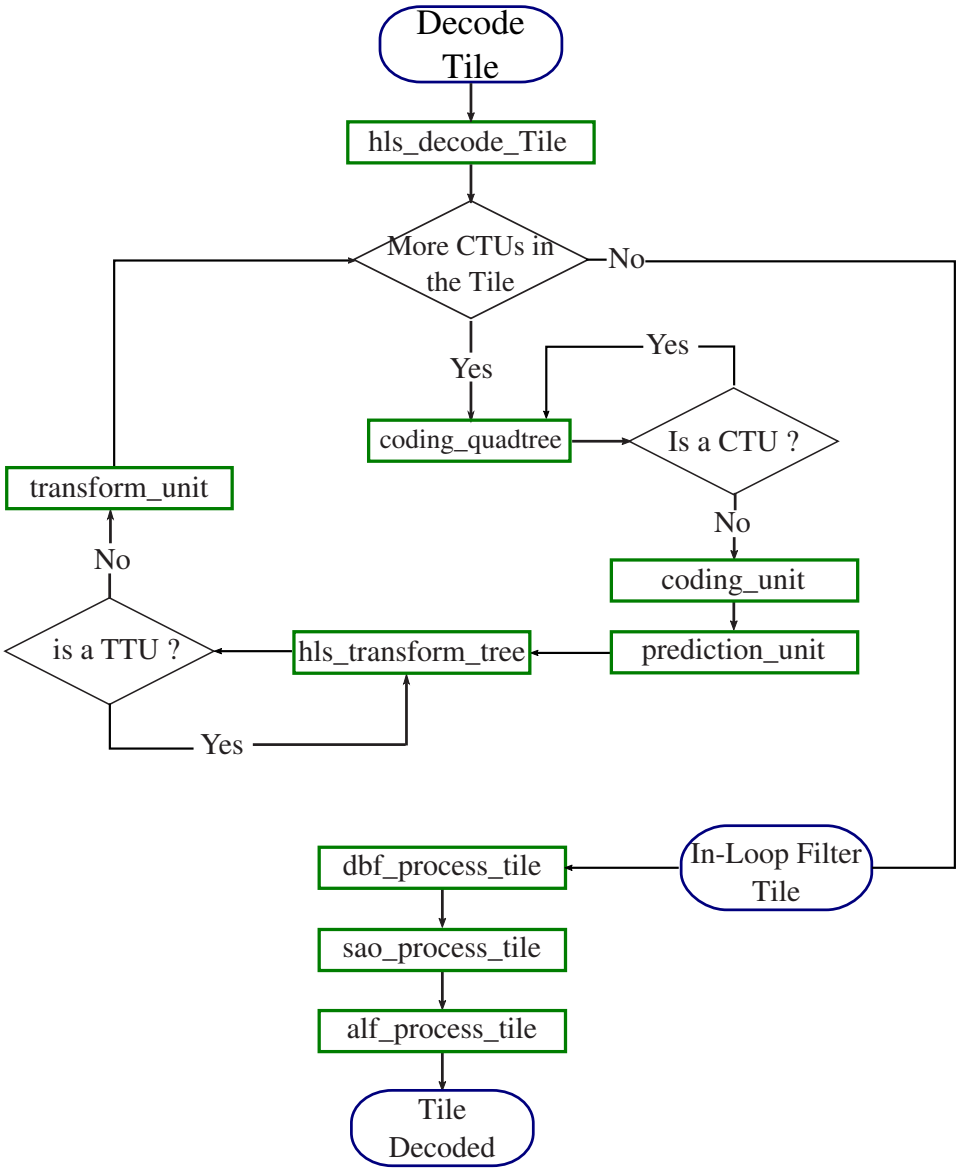


Figure 6.6 – *Block diagram of the* openVVC *decoder architecture.*

6.4.1 Code Structure

The decoding parameters required at the sequence, picture, slice or tile level are retrieved in the first place by parsing global parameter sets such as the Sequence Parameter Set (SPS), Picture Parameter Set (PPS) and slice header. These parameters are stored in the global context structure of the decoder and used to initialize its internal structure. In this work, the frame is composed of a single slice, but may be partitioned into several tiles for parallelism purpose. The decoder browses in raster scan the CTUs within the tile, as displayed in Figure 6.5. All information required to decode current CTU, are stored in a lightweight local context structure. This CTU by CTU processing in *openVVC* decoder is designed to reduce as much as possible the memory usage. As mentioned in Section 6.3.3.2, this approach reduces substantially the memory usage compared to the storage in a frame-basis of the CABAC output.

[2] https://www.videolan.org/index.fr.html

The general block diagram of the *openVVC* decoder is presented in Figure 6.6. The decoder processes each CTU by calling the function *hls_ coding_ tree* that browses the Coding Units (CUs) within CTU in z-scan order. For each CU the *hls_ coding_ unit* function performs inverse prediction and inverse transform operations. The reconstructed pixels of the CUs are stored in a buffer the size of the frame further called the decoding frame buffer. Once all CUs within the CTU are decoded, the decoder processes the upcoming CTU.

When the entire tile is reconstructed, the decoder performs the in-loop filters sequentially on the entire tile. As mentioned in Section 6.3.3.2, applying in-loop filter on the entire tile is not optimal for frame-level parallelism in Inter configuration. The final pixels, needed as reference for MC of next frames, are available only after the reconstruction stages, DBF and SAO are processed on a tile level, and after the ALF is processed on a CTU level. Since this work focuses on AI configuration, the considered approach for in-loop filtering process induces no overhead for frame-level parallelism. In future works that will also focus on Inter configuration, the in-loop filters will be applied as soon as possible on a CTU basis. In order to anticipate this evolution and reduce memory usage, the filter buffers dimensions have been reduced to the minimum required for the application of filters on a CTU basis.

6.4.2 Filtering Process

Figure 6.7 illustrates the dimensions of the filter buffers. In order to filter the current CTU pixels, several not filtered pixels belonging to neighboring CTUs are required. These pixels form a margin around current CTU. The selected margin size M is half the maximum filter size. In our case the filter with maximum size is the ALF luma diamond shaped filter, that is of size 7. Therefore, the margin is selected with value $\boldsymbol{M=3}$

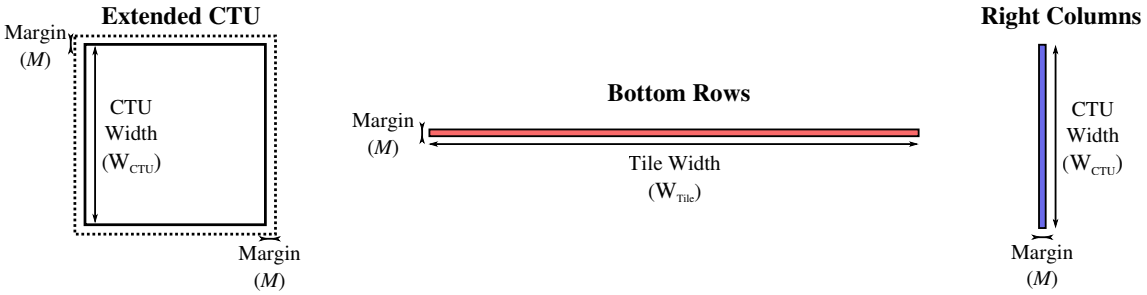
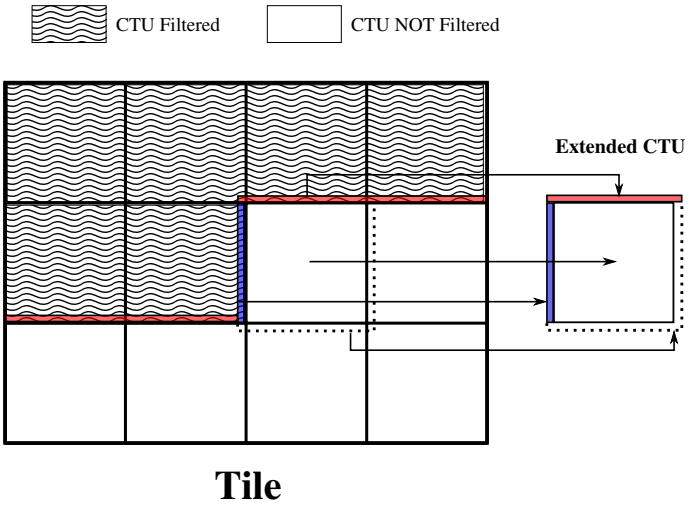


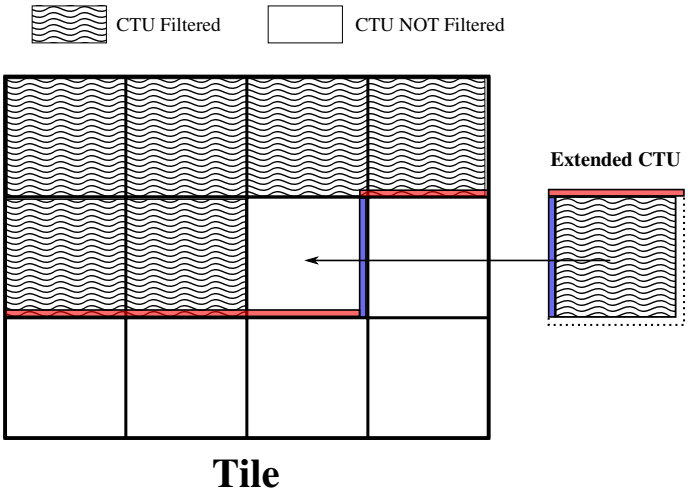
Figure 6.7 – *Dimension of buffers required for the filtering on a CTU basis.*

As shown in Figure 6.7, the extended CTU buffer is a square of size $W_{CTU} + 2 \cdot M$, and contains all not filtered pixels required to filter current CTU. The bottom rows buffer dimensions depends on the tile dimensions. It is a rectangle of dimensions $(M \times W_{Tile})$ used to store the bottom not filtered pixels of the CTU, that will be used for the filtering of the CTU below. The right columns buffer is a rectangle of dimensions $(M \times W_{CTU})$, that will be used for the filtering of the right CTU.

Figure 6.8 illustrates the usage of the filter buffers during the filtering process for a given CTU. First, the extended CTU buffer is filled with not filtered pixels, as shown in Figure 6.8a. The center area, of size $W_{CTU} \times W_{CTU}$, is filled with the pixels of the frame buffer. The bottom and right margins are also filled with the content of frame buffer, since the CTUs to which they belong have not yet been filtered. The left margin is filled with the not filtered right columns of the left CTU, contained in the right columns buffer. The upper margin is filled with the not filtered bottom rows of the upper CTU, stored in the



(a) *Fill extended CTU buffer with not filtered pixels of current CTU and with previously saved content of other filter buffers.*



(b) *Update right columns and bottom lines buffers with not filtered pixels and apply filter on extended CTU buffer.*

Figure 6.8 – *Two steps usage of the filter buffers in the filtering process of current CTU.*

bottom rows buffer. The second step is shown in Figure 6.8b. The right columns and bottom lines buffers are updated with not filtered pixels of current CTU, and the in-loop filter is applied on the center area of the extended CTU buffer. The center area is finally copied inside the frame buffer, and the upcoming CTU is ready to be filtered.

6.4.3 Parallelism Strategy

OpenVVC decoder currently supports frame-level parallelism, as well as slice-level and tile-level parallelism (both dynamic and static) defined in the VVC standard. The WPP tool is not yet supported and therefore is not studied in this work. At a data-level parallelism, several computationally expensive methods are optimized with SIMD instructions.

With frame-level parallelism, several frames are decoded in parallel with a single thread assigned to a frame. In *openVVC*, each thread requires separate main picture buffer, filter buffer, global context and local context structures. The memory usage is therefore expected to increase linearly with the number of threads.

With tile-level parallelism, the decoder calls in parallel the function *hls_decode_tile*. This function processes in raster-scan order the CTUs within a tile, as described is Section 6.4.1. Both dynamic and static tile partitioning of the sequence are supported in *openVVC*. With static tile partitioning, the same tile partitioning is applied on the entire sequence (uniform tile partitioning for instance). In contrast, the tile partitioning is called dynamic when it evolves during the sequence time-line. In order to decode a tile, each thread requires separate filter buffers and local context structure. The main picture buffer and global context structure are however shared between all the threads. The memory overhead induced by tile-level parallelism is expected to be negligible for high resolution sequences, since filter buffers and local context structure sizes are negligible compared to main picture buffer size.

For data-level parallelism, *openVVC* relies on SIMD extensions SSE [115] operating on 128 bits registers. Currently, the methods optimized with SIMD instructions include inverse DCT and DST transform, intra mode application, MC interpolation filtering, residual addition and ALF filtering. A quick profiling pointed out that the application for luma and chroma of ALF diamond shape filters is a bottleneck for *openVVC* decoding process. It is typically a calculation that benefits from SIMD architectures, since it applies repeatedly elementary operations on vectors of samples. The SIMD divides by 2.5 in average the time consumption of the ALF luma and chroma diamond shape filters. In the future, other computationally expensive methods such as SAO filtering or derivation of ALF classification will take advantage of SIMD architectures.

6.5 Experimental Results

This section presents the experimental setup, as well as the performance in term of memory usage and frame-rate of the proposed *openVVC* decoder in AI coding configuration. The results are presented first with enabling frame-level parallelism and then with tile-level parallelism.

6.5.1 Experimental Setup

The following experiments are conducted with the proposed *openVVC* decoder, compliant with the reference VTM-9.0 decoder. These two software decoders are built with gcc compiler version 7.4.0, under Linux OS version 4.15.0-4-generic as distributed in Ubuntu-18.04.1. The platform setup is composed of a 12 cores Central Processing Units (CPUs) Intel(R) Xeon(R) E5-2690 v3 running at 2.60 GHz. The cores have each 768KB L1 cache, 3MB L2 cache and 30MB L3 cache.

Table 6.1 – *Platform setup used for the decoding performance analysis*

CPU	Intel(R) Xeon(R) E5-2690 v4
Clock Rate	2.60 GHz
Memory	64 GB
Cache (L1, L2, L3)	768 KB, 3 MB, 30 MB
Compiler	Linux 4.15.0-74-generic
Encoder Version	VTM-9.0
Decoder Version	*openVVC* (branch *update_vtm9.0*)

The complexity increase of VVC decoding process raises a critical issue mainly for high resolution video sequences. For this reason, the test sequences selected in this work

contain 4 UHD and 5 HD sequences included in the Common Test Conditions (CTC) [73]: *CatRobot1*, *DaylightRoad2*, *FoodMarket4*, *Tango2* (UHD), and *BasketballDrive*, *BQTerrace*, *Cactus*, *MarketPlace*, *RitualDance* (HD). In order to obtain bitstreams of comparable sizes, only the first 300 frames of all test sequences are encoded. The bitstreams are generated with VTM-9.0 encoder under AI configuration. Current version of *openVVC* is compatible with most tools enabled in the CTC, and some tools are still under implementation. The main tools not yet available in current version include the joint coding of chrominance residuals [132], dual chroma tree [133], Matrix-based Intra Prediction (MIP) [35] and in-loop filters DBF and LMCS.

The performance of the *openVVC* decoding is assessed at both high and low bit-rates, obtained with Quantization Parameter (QP) values of 27 and 37, respectively. The memory usage of the software and the output frame-rate are used as performance metrics. The instantaneous memory usage is measured at decoding time with C function *getrusage* included in *sys/resource.h*. It is a crucial information to assess the portability of *openVVC* decoder on platforms with strong memory constraints. The frame-rate is evaluated through the number of decoded frames per second, noted fps. It measures the speed in term of decoding time of *openVVC* with respect to real-time decoding.

Table 6.2 – *Buffer and structure sizes, depending on sequence resolution.*

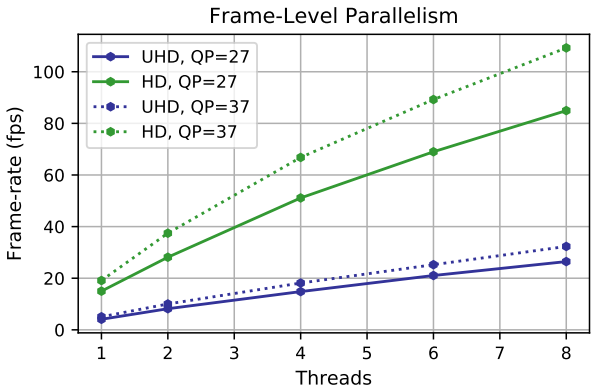
Resolution	Local Context	Global Context	Filter Buffers	Frame Buffer
HD	16 KB	4 KB	60 KB	6.2 MB
UHD	16 KB	5 KB	69 KB	24.8 MB

TABLE 6.2 summarizes the sizes of the *openVVC* buffers and structures, with chroma format 4:2:0, input bit depth 10 and 128×128 CTUs. The local context structure size is constant regardless of the resolution since it contains local information required to decode a CTU. The global context structure contains the frame parameters including among others SAO and ALF flags per CTU. It is therefore slightly larger for UHD resolution compared to HD. For filter buffer sizes, the worst case of 1 tile per frame is considered in TABLE 6.2. The filter buffers include the bottom rows buffer of size $M \times W_{Tile}$ (see Section 6.4.2), that is larger for UHD resolution compared to HD. For this reason the filter buffer size is 60KB for HD and 69KB for UHD. The frame buffer has the dimensions of the frame and is therefore 4 times larger for UHD compared to HD sequences.

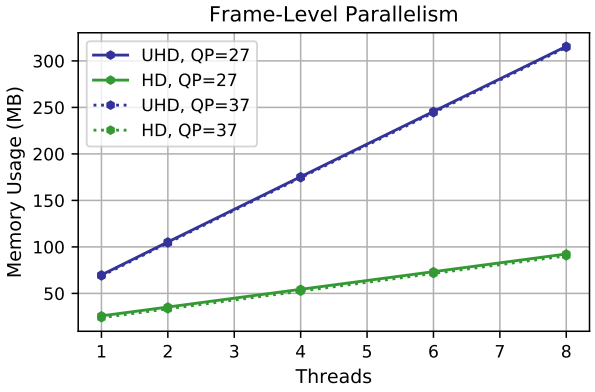
6.5.2 Frame-Level Parallelism

6.5.2.1 Average Performance

Figure 6.9 presents the performance of frame-level parallelism in term of frame-rate (Figure 6.9a) and memory usage (Figure 6.9b) in AI configuration. The performance is averaged across test sequences depending on resolution and QP value. As mentioned in 6.3.3.1, there are no MC dependencies between frames in AI configuration. This explains why the frame-rate increases almost linearly with the number of used threads in Figure 6.9a. Figure 6.9a shows that the frame-rate is in average higher for higher QP value. This behavior is further explained in detail on a per sequence basis. For HD sequential decoding, *openVVC* achieves in average 18 fps and 20 fps at QP=27 and QP=37, respectively. The frame-rate reaches 85 fps and 110 fps with 8 thread at QP=27 and QP=37, respectively. The performance is divided by 4 in average for UHD content. With 8 threads 27 fps and 33 fps is achieved at QP27 and QP37, respectively.



(a) *Average frame-rate depending on number of threads.*



(b) *Average memory usage (in MB) depending on number of threads.*

Figure 6.9 – *Performance of frame-level parallelism in AI configuration.*

Figure 6.9b shows the memory usage (in MB) of frame-level parallelism as function of the number of threads. Figure 6.9b highlights the low memory usage required by *openVVC*. With less than 105MB, *openVVC* is able to decode simultaneously up to 8 frames in HD resolution and 2 frames in UHD resolution. As a comparison, the software used in work [126] consumes 500MB of memory for the sequential decoding of HD content, which is 20 time higher compared to *openVVC*. The software has been developped for academic use [3], and the HD bitstreams *APSALF_A_Qualcomm_2* and *CROP_B_Panasonic_3* [4] have been decoded. As mentioned in Section 6.3.3.2, the memory usage of their decoder is not optimized, reason for this high memory usage. However, it gives an order of magnitude of the very low memory usage required for the proposed decoding approach in *openVVC*.

Figure 6.9b also shows the linear increase of memory usage with the number of threads required for frame-level parallelism. This behaviour is explained by the simultaneous storage of multiple frame buffers. The memory usage reaches 310 MB in average for the decoding of UHD content with 8 threads. This result is low relatively to other softwares, but could become a bottleneck for systems with strong memory constraints. Finally, Figure 6.9b shows that the average memory usage is almost constant in function of the QP value, since the plain and dashed curves are overlapping.

[3] https://github.com/fraunhoferhhi/vvdec

[4] https://www.itu.int/wftp3/av-arch/jvet-site/bitstream_exchange/VVC/under_test/VTM-10.0/

Table 6.3 – *Frame-level parallelism: bitstream size for 300 frames (in MB), frame-rate (in fps) and memory usage (in MB) per test-sequence, for various QP values and number of threads.*

		QP=27					QP=37				
			4 Threads		8 Threads			4 Threads		8 Threads	
	Sequence	Bitstream size (MB)	Fr-rate (fps)	Mem (MB)	Fr-rate (fps)	Mem (MB)	Bitstream size (MB)	Fr-rate (fps)	Mem (MB)	Fr-rate (fps)	Mem (MB)
UHD	*CatRobot1*	63.1	13.8	150	25.1	315	20.5	17.8	174	31.7	314
	DaylightRoad	64.1	14.4	168	25.7	315	18.2	18.2	174	33.0	295
	FoodMarket4	47.7	15.5	175	27.5	314	16.3	18.7	159	32.5	311
	Tango2	30.0	15.4	150	27.3	315	9.6	17.7	173	31.9	307
HD	*BasketballDrive*	20.5	54.9	50	89.5	84	5.4	67.1	51	111.2	85
	BQTerrace	41.1	41.6	48	70.8	91	11.2	62.0	47	100.2	85
	Cactus	35.1	45.2	49	89.5	86	10.1	62.2	51	111.2	90
	MarketPlace	19.6	53.9	54	89.8	91	4.9	69.7	49	112.0	82
	RitualDance	17.4	59.7	48	97.8	91	4.9	72.8	50	119.6	85

6.5.2.2 Per-Sequence Performance

TABLE 6.3 presents the bitstream size (in MB), frame-rate (in fps) and memory usage (in MB) per test-sequence, for various QP values and number of threads. It is possible to compare the sequences with the bitstream size, since the bitstreams contain the same number of frames (300). TABLE 6.3 shows that for all the sequences, the frame-rate increases when the QP value increases. This is coherent with the average values plotted in Figure 6.9b, where the curves for higher QP are higher compared to lower QP. Indeed, for QP37 the bitstream size is in average divided by 3 compared to QP27, the decoder has much less symbols to process (see Section 3.3.3.1). At sequence level, TABLE 6.3 shows that for a given QP value and number of threads, the frame-rate is globally increasing when the bitstream size of the sequence decreases. For example at QP27 with 4 threads, the HD sequence *BasketballDrive* has a bitstream 20MB smaller compared to *BQTerrace* (20MB vs 41MB). The frame-rate obtained is significantly higher for *BasketballDrive* compared to *BQTerrace* (54.9 fps vs 41.6 fps). However, the frame-rate difference is not as significant when the size differences between the bitstreams are smaller. For instance at QP37 with 8 threads, the HD sequence *MarketPlace* has a bitstream 5MB smaller compared to *Cactus* (4.9MB vs 10.1MB). The frame-rate obtained is almost equal for *MarketPlace* compared to *Cactus* (112fps vs 111.2fps). In some cases where the difference between bitstream sizes is very small, the relation is even inversed (QP37, 8 Threads, *DaylightRoad* vs *FoodMarket*). This is explained by several decoding stages, such as in-loop filters, which complexity is independent of the number of symbols to process.

TABLE 6.3 also shows that the variation of memory usage for a given thread number is less than 8% according to the sequence, across same resolution sequences and QP values. The memory usage of *openVVC* is therefore not dependent on the bitstream size, it is dependent only on the sequence resolution and number of threads. These results corroborate the Figure 6.9b shows that the average memory usage is almost constant in function of the QP value, since the plain and dashed curves are overlapping.

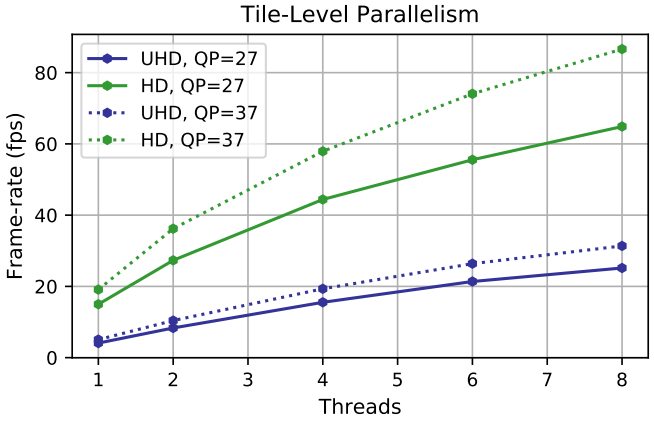
6.5.3 Uniform Tile Partitioning

In this Section, the decoding performance of uniform tile partitioning is assessed. The uniform tile partitioning is an usual and straightforward technique that partitions the frame in a grid of the same tile dimensions. It is a static tile partitioning, since the same tile partitioning is applied on the entire sequence. The constrained bitstreams containing uniform tile partitioning are generated with VTM-9.0. The prediction dependencies across

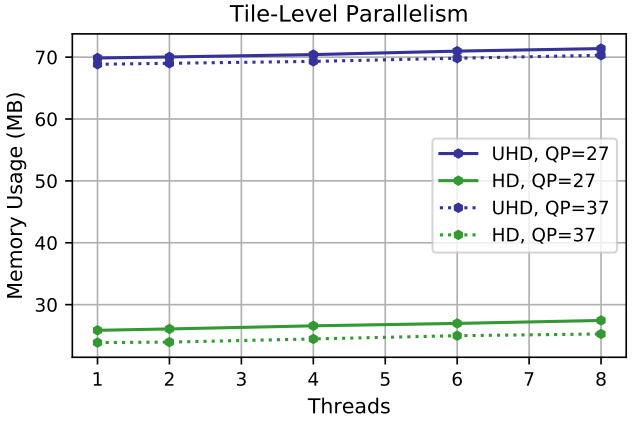
tile boundaries are disabled and entropy coding state is reinitialized at each tile. These restriction induce encoding quality losses, averaged across test sequences in TABLE 6.4.

Table 6.4 – *Tile partitioning BD-BR increase (in %) averaged across test sequences according to the resolution and number of tiles.*

	Tiles 2 BD-BR(%)	**Tiles 4** BD-BR(%)	**Tiles 6** BD-BR(%)	**Tiles 8** BD-BR(%)
HD	1.11	1.62	2.02	2.69
UHD	0.82	1.31	1.78	2.39



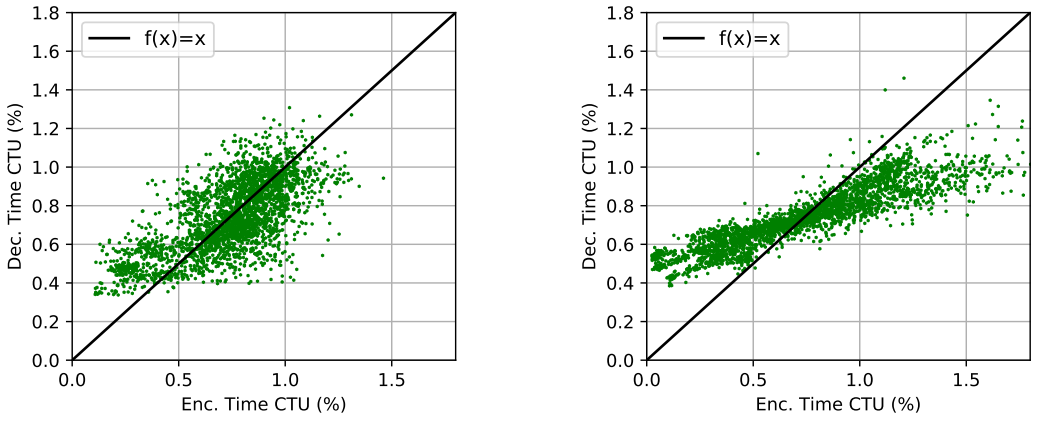
(a) *Average frame-rate depending on number of threads.*



(b) *Average memory usage (in MB) depending on number of threads.*

Figure 6.10 – *Performance of Uniform and Dynamic tile parallelism in AI configuration.*

Figure 6.10 presents the average performance of uniform tile level parallelism in term of memory usage (Figure 6.10b) and frame-rate (Figure 6.10a) in AI configuration. The performance is averaged across test sequences depending on resolution and QP value. Figure 6.9a shows that with 8 threads for HD content, the frame-rate reaches 66 fps and 85 fps for QP27 and QP37, respectively. With 2 and 4 threads, the performance of tile parallelism is equivalent to frame-level parallelism presented in Figure 6.9a. With 6 and 8 threads however the performance is lower. This is due to the decoding time imbalance



(a) *All Intra configuration, QP=27*

(b) *All Intra configuration, QP=37*

Figure 6.11 – *Decoding time of CTU (in % of total frame decoding time), as a function of CTU encoding time (in % of total frame encoding time). Times extracted from the processing of HD sequences* BasketballDrive, BQTerrace *and* Cactus.

between tiles, imbalance that grows with the number of tiles. For UHD content with 8 threads, 25 fps and 31 fps are achieved at QP27 and QP37, respectively. Even with 8 threads, the performance for UHD content of tile level parallelism is very close compared to frame-level parallelism (26 fps and 32 fps). This is explained by the fact that the decoding time imbalance between tiles has less impact on UHD content compared to HD content since frames are larger.

Figure 6.10b shows one of the main advantages of tiles in *openVVC* decoder. The memory usage is almost constant depending on the number of threads since the memory overhead is only induced by lightweight filter buffers and structure local context, as mentioned in section 6.4.3. With less than 72MB, *openVVC* decodes simultaneously 8 tiles for UHD content.

6.5.4 Dynamic Tile Partitioning

In this Section, the dynamic tile partitioning solution initially designed to speed-up the encoder in Chapter 5 is applied to *openVVC* decoding process. Its decoding performance is compared to previous results on uniform tile partitioning.

6.5.4.1 Reminder Proposed Solution

In Chapter 5, we proposed a dynamic Tile and Rectangular Slice partitioning solution for VTM-6.2 encoder (see Section 5.2.1 for definition on Tile and Rectangular Slice partitioning). A Tile and Rectangular Slice (TRS) partitioning is called dynamic when it evolves during the sequence time-line. The solution has been reimplemented into the VTM-9.0 encoder, in order to generate bitstreams compatible with *openVVC*. For the current experiments, the Rectangular Slices are not activated and only the tile grid partitioning is considered. As a reminder, the solution operates as follows: a distinct tile partitioning is computed prior to the encoding of each frame. The times of previously encoded CTUs are

used to balance the encoding time between the selected tiles. This dynamic approach reduces significantly multi-thread encoding time with same encoding quality losses compared to uniform tile partitioning.

6.5.4.2 CTU Decoding Times

To apply this solution in the decoding process, the preliminary study evaluates the relation between CTU encoding time and CTU decoding time. Figure 6.11 plots the *openVVC* decoding time per CTU (in % of total frame decoding time), as a function of VTM-9.0 encoding time of the CTU (in % of total frame encoding time). The times are extracted from the processing of HD sequences *BasketballDrive, BQTerrace* and *Cactus*. For QP27, the points in Figure 6.11a are localized close to the function $f(x) = x$, meaning the decoding time of a CTU in *openVVC* is close to VTM-9.0 encoding time, in percentage of frame processing time. For QP37, the distribution in Figure 6.11b is substantially more distant from $f(x) = x$. This is mostly explained by the filtering time of a CTU that is almost constant regardless of the number of symbols required to code the CTU. It homogenizes the CTU decoding times, especially for high QP values where the CTU reconstruction stages are less complex, because of less symbols to process. The conclusion of this preliminary study is that the dynamic tile partitioning is expected to balance more efficiently the tile decoding times for QP27 compared to QP37.

6.5.4.3 Performance Dynamic vs Uniform Tile Partitioning

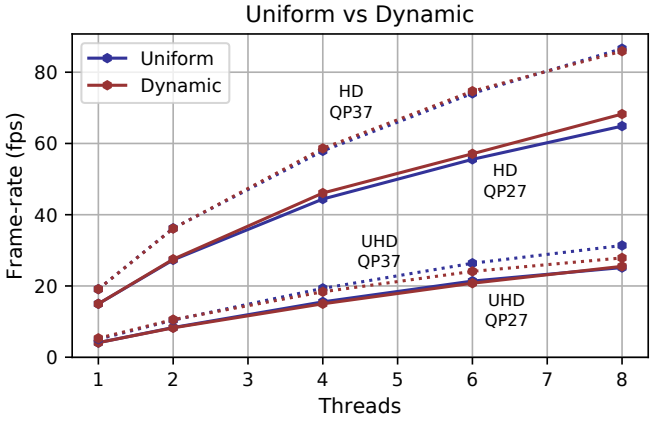


Figure 6.12 – *Performance of Uniform and Dynamic tile parallelism in AI configuration.*

The frame-rates obtained by both Uniform and Dynamic tile partitioning in AI configuration are displayed in Figure 6.10a. The red curves correspond to the dynamic tile partitioning results, while the blue curves correspond to uniform partitioning already plotted in Figure 6.9a. For HD sequences, the red and blue curves are almost overlapping for QP37, and the red curve is slightly higher to the blue curve for QP27. It means that the dynamic tile partitioning has equivalent performance for QP37 and slightly better performance for QP27, compared to uniform tile partitioning. For UHD content, the dynamic tile partitioning has equivalent performance for QP27 and slightly lower performance for QP37, compared to uniform tile partitioning.

The first analysis is that the results in Figure 6.12 are coherent with the conclusion of previous Subsection 6.5.4.2. Indeed for both resolutions, the dynamic tile partitioning

is higher relatively to uniform tile partitioning for QP27 compared to QP37. The second analysis is that the dynamic partitioning solution proposed in Chapter 5, initially designed to speed-up the encoder, has equivalent decoding performance compared to uniform tile partitioning. It is therefore an effective asset at encoder side, without being a burden for the decoding process.

6.6 Conclusion

In this Chapter, we proposed a real-time VVC decoder based on the open source *openVVC* project, for HD and UHD content in AI configuration. By combining data-level parallelism with high-level or frame-level parallelism, *openVVC* achieves real-time decoding for HD content and 30fps for UHD content with very low memory usage. The results presented in this Chapter confirm that the assets of frame-level and tile-level parallelism in the decoding process. Frame-level parallelism induces no encoding quality loss and slightly better speedup compared to tile-level parallelism. Tile-level parallelism improves decoding latency, with almost constant memory in *openVVC*. We have also shown in this Chapter that the dynamic tile partitioning solution proposed in Chapter 5 obtains equivalent results to those obtained with uniform tile partitioning. It proves that the dynamic tile partitioning proposed in Chapter 5 is an effective asset at encoder side, without being a burden for the decoding process.

Future works will focus on several improvements for *openVVC* decoder. The first aspect consists in speeding-up the in-loop filtering process that consumes 40% and 55% of decoding time in average for QP27 and QP37, respectively. The speed-up will be achieved reducing the memory accesses during the in-loop filtering process, and optimizing with SIMD operations other time consuming computations such as SAO filtering or derivation of ALF classification. In future works that will also focus on Inter configuration, the in-loop filtering process will be applied as soon as possible on a CTU basis. As mentioned in Section 6.4.2, this approach will reduce the overhead induced by MC for frame-level parallelism. Finally, future works will combine frame-level with tile-level parallelism. Currently in *openVVC*, the decoder waits until all the tiles are decoded in current frame before decoding the following frame. By combining frame and tile parallelism, a trade-off between the assets of both approaches could be found.

CHAPTER 7

Conclusion

As for every new generation of video coding standard, the primary objective of Versatile Video Coding (VVC) standard is to provide a significant improvement in coding efficiency over the High Efficiency Video Coding (HEVC) standard. VVC provided around 40% bit-rate savings over HEVC for the same quality of experience of video services. This higher coding efficiency will enable lower storage and transmission cost for video signal. However, the bit-rate savings achieved by VVC are generated by increasingly complex tools, on both encoder and decoder sides. This complexity may interfere with the deployment of VVC standard in the future years, especially on embedded platforms with low energy supplies and on live applications that require real-time encoding and decoding.

In this context, this thesis has presented three contributions on various software projects to reduce efficiently the implementation cost, i.e. energy consumption and processing time of VVC encoding and decoding processes. The two levers employed are the reduction of computational complexity and the parallel processing of VVC codec. Both parallel processing and complexity reduction are conducted minimizing the coding quality losses, in order not to affect the quality improvement of the new standard VVC. Both parallel processing and complexity reduction are considered with the aim to minimize the coding quality losses. This approach is crucial to affect the least possible the quality improvement of the new standard VVC.

The contributions presented in this thesis are among the first to be published on complexity reduction and parallel processing of VVC standard. In addition to providing promising results, they are intended to serve as a basis of comparison for future research work, and are designed to be easily implementable into future professional VVC encoders and decoders.

7.1 Reminder of Contributions

To adequately address the issue of VVC codec optimization, the first stage of this thesis is a quality assessment of coding tools. This assessment identifies the tools likely to induce coding quality losses. In parallel, a complexity profiling of VVC codec highlights the most computationally consuming parts of both encoding and decoding processes. In order to assess the performance of VVC compared to HEVC, both studies (quality assessment and complexity profiling), was conducted in comparison with HEVC standard. The three main

contributions proposed to reduce efficiently the implementation cost of VVC standard rely on this preliminary study and are reminded briefly in this section.

7.1.1 Tunable VVC Block Partitioning based on Lightweight Machine Learning

The block partitioning scheme has been identified in Chapter 3 as the most computationally expensive tool in the VVC standard. In order to reduce the search space of block partitioning scheme, we proposed a lightweight and tunable solution in Random Access (RA) coding configuration. The proposed solution is based on a Random Forest (RF) classifiers [9] to determine for each coding block the most probable partition modes. To minimize the encoding loss induced by misclassification, risk intervals for classifier decisions are introduced in the proposed solution. By varying the size of risk intervals, tunable trade-off between encoding complexity reduction and coding loss is achieved. The proposed solution has been implemented in both VVC Test Model (VTM)-5.0 and Joint Exploration Model (JEM)-7.0 softwares and offers encoding complexity reductions ranging from 30% to 70% in average, for only slight encoding quality loss in RA coding configuration. The overhead induced in the encoding process by RF classifiers is also very light, which is a key point to use this solution in a real-time or embedded framework.

7.1.2 Efficient Parallel Encoding through Dynamic Tiles and Rectangular-Slices

The second contribution of this thesis is an effective high-level parallelism solution for VVC encoding process in RA configuration. A dynamic Tile and Rectangular Slice (TRS) partitioning solution is proposed and benefits from the combination of tiles and Rectangular Slice (RS). The TRS partitioning is dynamic and carried-out at the frame level, taking into account both spatial texture of the content and encoding times of previously encoded frames. The proposed solution searches the best partitioning configuration that minimizes the trade-off between multi-thread encoding time and encoding quality loss. Experiments proved that the proposed solution, compared to uniform tile grid partitioning, significantly decreases multi-thread encoding time, with slightly better encoding quality.

7.1.3 Parallel and Real-Time VVC Decoder in AI configuration

The last contribution of this thesis focuses on the decoder side of the new standard. Our work is implemented in the open source and real-time VVC decoder *openVVC* project, operational in All Intra (AI) configuration. After the implementation of in-loop filters and dynamic tile parallelism in *openVVC*, we integrated the dynamic tile partitioning solution proposed in our previous contribution (See A.2), enabling the generation of bitstreams compatible with *openVVC* decoder. The experimental results showed that the decoding frame-rates obtained with dynamic tiles are equivalent to the frame-rates obtained with uniform tiles. The general performance of *openVVC* is also discussed. By combining data-level with tile-level or frame-level parallelism, *openVVC* achieves real-time decoding for High Definition (HD) content and 30fps for Ultra High Definition (UHD) content. The main asset of the proposed decoder lies in its very low memory usage, since the sequential decoding of HD and UHD content only requires 20MB and 70MB of memory, respectively.

7.2 Future Works

7.2.1 Encoder Complexity Reduction

Short-Term Perspectives

The contribution presented in Chapter 3 has been proposed in the early standardization phase of VVC. The related works that inspired the contribution mainly reduce the complexity of Quad Tree (QT) partitioning scheme in HEVC and Quad Tree Binary Tree (QTBT) partition scheme in JEM software. The Multi-Type Tree (MTT) partitioning scheme in VVC standard allows two more partition modes compared to QTBT partitioning scheme, considerably increasing the encoding complexity. An overview of the numerous complexity reduction techniques for MTT partitioning scheme that have been published afterwards could provide ideas to improve the proposed solution. In the literature for instance, many works rely on Convolution Neural Networks (CNNs) to extract relevant features from a block of pixels [93, 85]. Implementing a low complexity CNN to improve the classification performance could therefore be an interesting lead.

The performance under the VTM-5.0 proves that the proposed solution is scalable to two different encoders and does not over-fit the JEM-7.0 encoding characteristics. The results in VTM-5.0 have been achieved with little extra implementation work. The RF features and training process, as well as the computation technique for the Complexity Reduction Configurations (CRCs) remained unchanged between VTM-5.0 and JEM-7.0. Another interesting future work would be to adapt the proposed solution to a less complex VVC encoder in order to prove its reliability for future professional encoders. The new Fraunhofer Versatile Video Encoder, called VVenC [134], could be an interesting candidate to adapt our solution.

Long-Term Perspectives

The risk intervals introduced to control the classifier decisions are set by the encoder, adapting the solution to the encoded content. This process requires the deactivation of the proposed solution on the encoding of a reference frame every 32 frame. An interesting perspective consists in finding off-line criteria (using statistical models or machine learning techniques for instance) allowing to be independent of this reference frame.

Encoding time control consists in keeping the encoding process complexity as close as possible to a predefined target. This target can be expressed in term of energy (Joule) or encoding time (Seconds) in function of the application. The tunable encoding complexity reduction proposed in Chapter 3 provides several Complexity Reduction Configurations (CRCs), each obtaining a distinct level of complexity reduction. It is therefore the first step for encoding time control. The second step would be to implement a control system that selects in real-time the optimal CRC, i.e. the CRC that generates the encoding complexity closest to the predefined complexity target.

7.2.2 Encoder Parallel Processing

Short-Term Perspectives

The theoretical upper bound in terms of speed-up for the high-level parallelism solution presented in Chapter 5 is computed with the Amdahl law [113]. In our case, the sequential portion of VTM encoder contains the data initialization, entropy, in-loop filters and bitstream writing stages, reaching 4% of the encoding time in average. Implementing the

in-loop filters inside the parallel portion of the encoder would increase the theoretical upper bound and provide a use case more representative of real-life encoders.

The proposed solution benefits from the combination of dynamic tile grid and RS partitioning. The experiments proved that it outperforms the uniform tile grid partitioning. In future works, the performance of the proposed solution with only dynamic tile grid partitioning will also be assessed. It will quantify which share of the good results is generated by the dynamic tile partitioning, and which share is induced by the more flexible RS partitioning.

Long-Term Perspectives

A potential direction for future works could also be to improve the Coding Tree Unit (CTU) time estimator, used in the encoding time minimization step. Instead of simply relying on the co-located CTU times of the co-Temporal Layer (TL) frame, future solutions could rely on CTU times deduced by motion information.

The TRS partitioning search takes place before the encoding of each frame and adapts the optimal TRS partitioning to the frame content. The investigation of lightweight heuristics for the TRS partitioning search will also be part of future works. We are confident they will reduce drastically the overhead, especially for 12 threads encodings of UHD content.

In order to limit the encoding quality loss induced by TRS partitioning, the proposed solution gathers similar spatial information inside the same RSs. However, a more efficient technique to reduce encoding loss consists in lowering the number of RSs in the frame [108]. This possibility has not been considered in this work, since the number of RSs in the frame is a fixed input parameter. An interesting perspective would be to target an estimated encoding speed-up, and search the TRS partitioning with minimum number of RSs that obtains this target speed-up.

7.2.3 Real-time Decoder

Short-Term Perspectives

Future works will focus on several improvements for *openVVC* decoder. The first identified improvement is to speed-up the in-loop filtering process. The latter consumes 40% and 55% of decoding time in average for Quantization Parameter (QP)27 and QP37, respectively. The speed-up will be achieved reducing the memory accesses during the in-loop filtering process, and optimizing with Single Instruction on Multiple Data (SIMD) operations other time consuming computations such as Sample Adaptive Offset (SAO) filtering or derivation of Adaptive Loop Filter (ALF) classification.

Long-Term Perspectives

The contribution presented in Chapter 6 focuses on AI configuration that includes no Motion Compensation (MC) dependencies is frames. The frames are therefore independently decodable. In this case, the application of in-loop filter at the end of the tile reconstruction induces no overhead for frame-rate parallelism. In future works that will also focus on inter coding configuration, the in-loop filtering process will be applied on a CTU basis. The final pixels required as reference for Motion Compensation (MC) will be available with much lower delay, improving frame-rate parallelism.

Finally, future works will combine frame-level with tile-level parallelism. Currently in *openVVC*, the decoder waits until all the tiles are decoded in current frame before decoding

the following frame. By combining frame and tile parallelism, a trade-off between the assets of both approaches could be found.

APPENDIX A

FRENCH SUMMARY

Au cours de la dernière décennie, l'utilisation intensive de plateformes en ligne telles que la vidéo à la demande (Netflix, AmazonOCS, MyCanal) ou les plateformes de partage vidéo (Youtube, Dailymotion, Vimeo) a entraîné une augmentation significative du volume de contenu vidéo échangé. Les services multimédias se sont également diversifiés avec l'apparition d'applications vidéo qui offrent une expérience de visionnage immersive et plus naturelle. Ces nouveaux services nécessitent à la fois une résolution (4K, 8K, 360°) ou une fréquence d'images (120fps) plus élevées et augmentent nettement le volume du contenu vidéo. Parallèlement, les appareils conçus pour afficher et distribuer ces services vidéo émergents deviennent des produits abordables pour le grand public grâce aux progrès de la microélectronique. Les téléviseurs, les ordinateurs ou les téléphones intelligents capables d'afficher des résolutions haute définition sont désormais entrés dans notre vie quotidienne. Il est donc désormais possible de consommer des services vidéo presque partout et à tout moment, et les services vidéo sont de plus en plus volumineux. Une étude récente publiée dans Cisco [1] a prédit que le trafic vidéo passera de 61% du trafic IP mondial en 2016 à 82% en 2021. Le visionnage de contenus vidéo en ligne a également un impact considérable sur les émissions mondiales de CO2. En 2018, le stockage, la transmission et la visualisation de vidéos en ligne représentaient des émissions de gaz à effet de serre comparables à celles d'un pays comme l'Espagne (1% des émissions mondiales) [2]. Cette demande croissante en contenus vidéo pose de nouveaux défis à la compression, principalement pour améliorer l'efficacité de codage et réduire la pollution induite par le stockage, la transmission et le traitement de la vidéo. Une meilleure efficacité de codage permet d'obtenir la même qualité d'expérience des services vidéo, avec des coûts de stockage et de transmission plus faibles.

Depuis la fin des années 1980, les recherches universitaires et industrielles se sont concentrées sur le codage vidéo, et les recommandations fondamentales en matière de codage vidéo sont restées inchangées depuis H.261 en 1988. Tous les codecs vidéo sont basés sur la même architecture de codage hybride par blocs, combinant les prédictions inter et intra et le codage par transformation. Presque tous les 10 ans, une nouvelle norme est publiée avec pour objectif une réduction de 50% du débit pour une qualité subjective égale à celle de son prédécesseur. En 2013, la norme de codage vidéo appelée H.265/HEVC, pour High Efficiency Video Coding, a été publiée et progressivement adoptée dans de nombreux systèmes d'application. Par rapport à son prédécesseur H.264/Advanced Video Coding (AVC) [3, 4, 5] sorti en 2004, HEVC permet de stocker et de transmettre deux fois plus de contenu vidéo pour la même qualité visuelle. Actuellement, la technologie de pointe en

matière de codage vidéo est la norme VVC, pour Versatile Video Coding, finalisée à la fin de 2020. Comme pour toute nouvelle norme, l'objectif premier de VVC est d'améliorer considérablement l'efficacité du codage par rapport à la norme existante précédente, en l'occurrence HEVC. La norme VVC permet d'économiser plus de 40% de débit par rapport à la norme HEVC et devrait permettre de délivrer des services en UHD à des débits qui sont en fait utilisés pour transporter la TV-HD.

Les économies de débit réalisées par chaque nouvelle génération de codecs vidéo sont générées par des outils de plus en plus complexes, tant du côté de l'encodeur que du décodeur. La complexité de calcul d'un encodeur VVC est estimée à 10 et 27 fois la complexité de calcul HEVC en configuration de codage inter et intra, respectivement [6]. Du côté du décodeur, l'augmentation de la complexité de calcul de la norme VVC par rapport à la norme HEVC est approximativement d'un facteur 2 dans les configurations de codage inter et intra [6]. Cette complexité peut devenir un goulot d'étranglement pour le développement de la norme VVC et peut interférer avec son déploiement, en particulier sur les plateformes embarquées à faible consommation d'énergie et sur les applications en direct qui nécessitent un codage et un décodage en temps réel.

A.1 Challenges et Objectifs

Le nombre de dispositifs connectés aux réseaux IP passera à 29,3 milliards d'ici 2023, ce qui représente plus de trois dispositifs par habitant en moyenne, selon Cisco [1]. Leur nombre est en constante augmentation, et une part importante de ces dispositifs de réseau sont embarqués et permettent le traitement vidéo. Les plates-formes embarquées ne disposent que d'un approvisionnement énergétique limité. Une réduction drastique de la consommation d'énergie est donc cruciale pour un déploiement réaliste des codecs VVC sur les plateformes embarquées telles que les appareils mobiles. Même pour les plates-formes non embarquées telles que les centres de données vidéo, la consommation d'énergie est une question importante. L'augmentation substantielle de la consommation d'énergie des codecs vidéo génère un budget plus important pour l'électricité, en particulier pour les centres de données qui traitent des millions de séquences vidéo chaque jour. Sur un plan plus environnemental, la réduction de la consommation d'énergie des codecs vidéo sur des milliards d'appareils est une manière de contenir les émissions de gaz à effet de serre générées par les traitements des codecs vidéo.

La diminution du temps de traitement des codecs VVC est également un défi crucial. Pour les applications video en direct, l'optimisation des processus de codage et de décodage est obligatoire pour répondre aux exigences du temps réel. Du côté des encodeurs, plusieurs applications en direct nécessitent un encodage vidéo en temps réel. Beaucoup ont vu le jour ces dernières années, comme la diffusion individuelle en direct (Periscope, Wowza, LiveStream) ou les applications de visio-conférence (Hangouts, Zoom, Skype). Certaines étaient déjà très répandues ces 30 dernières années, comme la diffusion en direct de programmes à la télévision. Côté décodeur, une grande majorité d'applications nécessite un décodage en temps réel. En effet, la séquence vidéo décodée est presque toujours consommée instantanément pour éviter le stockage du contenu non compressé sur le disque dur.

Cette thèse vise à réduire efficacement le coût de mise en œuvre, c'est-à-dire la consommation d'énergie et le temps de traitement, des processus de codage et de décodage VVC. Les deux leviers utilisés sont la réduction de la complexité de calcul et le traitement

[1] https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

parallèle du codec VVC. Les techniques d'optimisation de la complexité de calcul réduisent le travail global du processus et, par conséquent, diminuent directement la consommation d'énergie et le temps de traitement. Les techniques de traitement parallèle exploitent les architectures multi-cœurs afin de répartir de manière optimale le travail entre plusieurs acteurs. Les techniques de traitement parallèle peuvent être utilisées pour réduire le temps de traitement grâce à l'utilisation de plusieurs threads. Il est également possible de les utiliser pour diminuer la fréquence des processeurs, ce qui réduit quadratiquement la consommation d'énergie [7]. Le traitement parallèle et la réduction de la complexité sont tous deux effectués en minimisant les pertes de qualité du codage, afin de ne pas affecter l'amélioration de la qualité de la nouvelle norme VVC.

A.2 Contributions

Ce document comprend trois contributions principales visant à réduire efficacement le coût de mise en œuvre des processus de codage et de décodage VVC, tout en minimisant les pertes de qualité du codage en termes de débit et de qualité visuelle. Afin d'aborder de manière adéquate la question de l'optimisation du codec VVC, la première étape de cette thèse est une évaluation de la qualité des outils de codage. Cette évaluation permet d'identifier les outils susceptibles d'induire des pertes de qualité de codage. En parallèle, un profilage de la complexité du codec VVC met en évidence les parties les plus gourmandes en calcul de la nouvelle norme. Afin d'évaluer les performances du codec VVC par rapport au codec HEVC, les deux études (évaluation de la qualité et profilage de la complexité) sont menées par rapport à la norme HEVC. Le logiciel de référence VVC utilisé par les chercheurs et les industriels pour proposer et tester de nouveaux outils de codage au cours du processus de normalisation VVC est appelé VTM. Au cours du processus de normalisation HEVC, le logiciel de référence était le HEVC Model (HM). Les performances de lu logiciel VTM sont comparées à celles du HM, mettant en évidence les principales améliorations de la norme VVC par rapport à son prédécesseur. Plusieurs contenus vidéo, à différents débits, et deux résolutions spatiales, HD et UHD, sont encodés et décodés afin d'extraire des statistiques sur la qualité et la complexité. Ce travail préliminaire sur l'évaluation de la qualité et le profilage de la complexité est actuellement en cours d'examen en vue de sa publication dans la revue internationale de l'IEEE Transaction on Multimedia (ToM).

Les trois principales contributions ont été proposées alors que le processus de normalisation VVC n'était pas encore finalisé. C'est pourquoi seules quelques études sur la réduction de la complexité de la norme VVC et encore moins sur le traitement parallèle étaient disponibles dans la littérature. Les contributions présentées dans ce document sont parmi les premières appliquées à la norme VVC à être publiées dans leurs domaines respectifs. Elles sont destinées à servir de base de comparaison pour les futurs travaux de recherche, et sont conçues pour être facilement transposables dans les futurs encodeurs et décodeurs VVC professionnels. La dernière contribution de cette thèse est particulièrement représentative de cet effort, puisqu'elle a été directement insérée dans le décodeur open source *openVVC*, qui est destiné à être distribué au grand public. La section suivante présente brièvement les trois principales contributions de cette thèse.

Partitionnement en Blocs Ajustable pour VVC basé sur de l'Apprentissage Machine

Du côté de l'encodeur, le schéma de partitionnement en blocs sélectionne la taille de bloc appropriée en fonction de l'activité locale des pixels. Il s'agit d'un module essentiel pour

obtenir une amélioration significative de la qualité du codage. Cependant, le schéma de partitionnement des blocs a été identifié comme l'outil le plus coûteux en termes de calcul dans la norme VVC [8]. Afin de réduire l'espace de recherche du schéma de partitionnement par blocs, nous proposons une solution légère et ajustable en configuration de codage RA. La solution proposée est basée sur un classificateur RF [9] qui détermine pour chaque bloc de codage les modes de partitionnement les plus probables. La classification par RF est une méthode classique en Machine Learning (ML) qui prédit la valeur d'une variable cible, appelée classe, à partir des valeurs de plusieurs variables d'entrée, appelées caractéristiques. Pour minimiser la perte de codage induite par une mauvaise classification, des intervalles de risque pour les décisions de classification sont introduits dans la solution proposée. En faisant varier la taille des intervalles de risque, on obtient un compromis entre la réduction de complexité de codage et la perte de qualité. Le JEM est le logiciel de référence VVC pre-VTM, utilisé pendant la première moitié du processus de normalisation. La solution proposée a été mise en œuvre dans les logiciels VTM-5.0 et JEM-7.0 et offre des réductions de la complexité de codage allant de 30 à 70% en moyenne, pour une légère perte de qualité de codage dans la configuration de codage RA. Le surcoût induit dans le processus d'encodage par les classificateurs RF est également très faible, ce qui est un point clé pour utiliser cette solution dans un cadre temps réel ou embarqué. Ces travaux ont conduit à la présentation d'une session de posters lors de la conférence IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) en 2019 [10], et ont été publiés dans la revue internationale IEEE Transactions on Image Processing (TIP) en 2019 [11].

Encodage Parallèle Efficace via des Tiles et des Slices Rectangulaires Dynamiques

Comme mentionné dans la section précédente, afin d'obtenir un encodage en temps réel et à faible latence, il est obligatoire de s'appuyer sur un traitement parallèle pour distribuer de manière optimale la complexité de l'encodage à plusieurs acteurs. La deuxième contribution de cette thèse est une solution efficace de parallélisme de haut niveau pour le processus d'encodage VVC en configuration RA. La norme VVC comprend un partitionnement en grille de tiles qui permet de traiter simultanément les régions rectangulaires d'une image avec des threads indépendants. La grille de tiles peut être subdivisée en une sous-grille horizontale de RSs, ce qui augmente la flexibilité du partitionnement. Une solution de partitionnement dynamique de TRS, c'est-à-dire évoluant au cours de la séquence vidéo, est proposée et bénéficie de cette flexibilité. Le partitionnement TRS est effectué au niveau de l'image, en tenant compte à la fois de la texture spatiale du contenu et des temps d'encodage des images précédemment encodées. La solution proposée recherche la configuration de partitionnement qui minimise le compromis entre le temps d'encodage multi-threads et la perte de qualité de l'encodage. Les expériences prouvent que la solution proposée, par rapport à un partitionnement uniforme TRS, diminue considérablement le temps d'encodage multi-thread, avec une qualité d'encodage légèrement meilleure. Ces travaux ont été présentés lors de la session spéciale *Complexity Reduction and Real Time Implementations of the Versatile Video Coding Standard* de la conférence internationale de l'IEEE sur le traitement de l'image (ICIP) en 2020 [12].

Décodeur VVC parallèle et temps réel en configuration AI

La dernière contribution de cette thèse se concentre sur le côté décodeur de la nouvelle norme. Notre travail est implémenté dans le décodeur VVC open source et temps réel *openVVC*, opérationnel en configuration AI. La contribution comprend la mise en œu-

vre à partir des filtres ALF et SAO, avec une attention particulière à la minimisation de l'utilisation de la mémoire tampon des filtres. Elle comprend également la consolidation du parallélisme de tiles qui permet à la fois un partitionnement dynamique et uniforme des tiles. Une fois les filtres et le parallélisme dynamique des tiles opérationnels, nous avons réimplémenté la solution de partitionnement en tiles dynamique proposée dans notre précédente contribution (voir A.2), permettant la génération de bitstreams compatibles avec le décodeur *openVVC*. Les résultats expérimentaux montrent que la fréquence de décodage obtenue avec les tiles dynamiques est équivalente à celle obtenue avec des tiles uniformes. Les performances générales de *openVVC* sont également présentées. En combinant un parallelisme de données avec un parallélisme par tiles ou par images, la version présentée de *openVVC* permet d'obtenir un décodage en temps réel pour les contenus HD et UHD. Le principal atout du décodeur proposé réside dans sa très faible consommation de mémoire, puisque le décodage séquentiel du contenu HD et UHD ne nécessite que 20 et 70 Mo de mémoire, respectivement.

A.3 Plan Manuscrit

Cette section résume brièvement le contenu de chaque chapitre présenté dans ce document. Le chapitre 2 présente les concepts fondamentaux de la compression de vidéo. Ce chapitre introduit entre autre les caractéristiques d'un signal vidéo, le processus d'encodage Rate Distorsion Optimization (RDO), l'historique de la création des normes de codage vidéo et la norme de codage Versatile Video Coding récemment achevée.

Dans le chapitre 3, une évaluation subjective et objective de la qualité de VVC est réalisée afin d'évaluer la qualité de codage de cette norme émergente par rapport à la norme HEVC. Ce chapitre fournit également une analyse des répartitions de temps et de complexité, tant pour le processus de codage que de décodage. Ce chapitre identifie les outils susceptibles d'induire des pertes de qualité et évalue les possibilités de réduction de la complexité.

Afin de réduire la complexité du processus d'encodage, le chapitre 4 se concentre sur le schéma de partitionnement en blocs et réduit le nombre de partitionnements testés en configuration de codage RA. Toutes les étapes de la solution proposée sont détaillées, de la création des classificateurs ML à la réalisation d'un compromis ajustable entre la qualité de codage et le temps d'exécution.

Le chapitre 5 présente une technique permettant de paralléliser efficacement le processus de codage VVC dans la configuration de codage RA. Le partitionnement TRS, brièvement présenté dans la section précédente, est ajusté pour chaque image, en tenant compte à la fois du contenu spatial et des temps de codage des images précédentes. Les résultats obtenus prouvent que la solution proposée réduit considérablement le temps de codage multi-thread, avec une qualité de codage légèrement meilleure par rapport à d'autres techniques de partitionnement TRS plus directes.

Le Chapitre 6 présente un décodeur VVC temps réel basé sur le projet open source *openVVC* project, pour les contenus HD et UHD en configuration AI. Il présente les contributions de cette thèse dans le logiciel, ainsi que les résultats expérimentaux obtenus avec ces contributions. En combinant un parallelisme de données avec un parallélisme de threads, *openVVC* atteint le décodage en temps réel pour les contenus HD, avec une très faible utilisation de la mémoire.

Le Chapitre 7 conclut ce document et propose plusieurs axes de recherche pour prolonger les travaux présentés dans ce manuscript.

APPENDIX B

Notations

Table B.1 – *Notations used in this Thesis*

Notation	Definition
RGB	Color space associate each channel to a primary color: red, green and blue.
YUV (or YCbCr)	Color space with Y luminance, U (or Cb) and V (or Cr) the blue and red chroma channels.
W	Number of pixels in a row of the frame.
H	Number of pixels in a column of the frame.
$p(i,j)$	Pixel in row number i and in column number j of the frame.
b	Bitdepth of a video signal.
B	Total number of bits per pixel.
Π	Set of coding parameters.
Π^*	Set of coding parameters that minimize RD-cost.
E_Π	Ensemble of allowed coding parameters.
$D(\Pi)$	Distortion obtained with the coding parameters Π.
$R(\Pi)$	Bit-rate obtained with the coding parameters Π.
R_{max}	Bit-rate constraint for Rate Distorsion Optimization (RDO).
λ	Lagrangian multiplier for RDO.
J	RD-cost of coding parameters Π and Lagrangian multiplier λ.
Im_r	Reference image for quality metrics.
Im_e	Image to be evaluated.
A	Amplitude of the signal.
$wPSNR$	Weighted PSNR with Y, U and V channels.
$\mu(X)$	Local mean for a variable X.
σ	Local standard deviation.
$\sigma_{Im_r Im_e}$	covariance between images Im_r and Im_e.
W_{CTU}	CTU width in pixels.
$\mathbb{H}(X)$	Differential entropy of a continuous random variable X.
f_X	Probability Density Function of continuous random variable X.
$\mathbb{I}(F,C)$	Mutual Information : entropy decreasing of C when F is known.
ϵ_{RF}	Error rate of the RF classifier.
S-NS	Binary classifier determining which of the Split modes or Non-Split mode is more likely.
QT-BT	Binary classifier determining which of the QT mode or BT modes is more likely.
QT-BT	Binary classifier determining which of the BTH mode or BTV mode is more likely.
$grad_x$, $grad_y$	Gradients in horizontal ($grad_x$) and vertical ($grad_y$) directions.
σ^2	Local variance of a variable.
$\delta H(f)$	Horizontal Inconsistency of a feature f.
$\delta V(f)$	Vertical Inconsistency of a feature f.
ε_{RD}	Percentage Rate Distorsion (RD) error caused by a misclassification.
$w(A,B)$	Weight assigned to the training instances.

Table B.1 – *Continued*

Notation	Definition
$N_{votes}(A)$	Number of trees voting for class A in the RF classifier.
N_{trees}	Total number of trees constituting the RF classifier.
$Score(A)$	Percentage of decision trees that predicts the class A.
ΔT	Percentage of complexity reduction of a solution.
T_A	Time of the anchor software (encoded with exhaustive RDO).
T_R	Reduced time after applying a complexity reduction solution.
P	Partitioning of a frame into RSs.
P^*	Optimal P after rectangular clustering of the RSs.
c_i	CTU number i in raster-scan order.
s_j	Slice number j in raster-scan order.
$T(X)$	Effective time to encode a region X (frame, slice or CTU).
$\tilde{T}(X)$	Estimated time to encode a region X (frame, slice or CTU).
$\epsilon_{\tilde{T}}$	Error of a time estimator $\tilde{T}$.
A_{min}, A_{max}	Area of the smallest and the largest RSs.
γ	Relaxation factor that manages the trade-off between encoding time and quality.
ξ	Multi-thread speed-up.
ξ_{max}	Maximum obtainable speed-up (Amdhal law).

List of Figures

List of Tables

Personal Publications

[1] Thomas Amestoy, Alexandre Mercat, Wassim Hamidouche, Cyril Bergeron, and Daniel Menard, "Random Forest Oriented Fast QTBT Frame Partitioning," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, May 2019, pp. 1837–1841, IEEE. 4, 42, 118

[2] Alexandre Tissier, Alexandre Mercat, Thomas Amestoy, Wassim Hamidouche, Jarno Vanne, and Daniel Menard, "Complexity Reduction Opportunities in the Future VVC Intra Encoder," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, Kuala Lumpur, Malaysia, Sept. 2019, pp. 1–6, IEEE. 3, 118

[3] Thomas Amestoy, Alexandre Mercat, Wassim Hamidouche, Daniel Menard, and Cyril Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 1313–1328, 2020. 4, 118

[4] Thomas Amestoy, Wassim Hamidouche, Cyril Bergeron, and Daniel Menard, "Quality-Driven Dynamic VVC Frame Partitioning for Efficient Parallel Processing," in *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 3129–3133, IEEE. 4, 118

[5] Thomas Amestoy, Naty Sidaty, Pierrick Philippe, Wassim Hamidouche, and Daniel Menard, "Video Quality Assessment and Coding Complexity of the Versatile Video Coding Standard," *IEEE Transactions on Multimedia, Under Review*, 2020.

Bibliography

[1] CISCO, "Global_2021_forecast_highlights," *https://www.cisco.com/c/dam/m/en_us/solutions/serv provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf*, p. 6, 2016. 1, 115, 113

[2] Maxime Efoui-Hess, "Climate crisis: The unsustainable use of online video. The practical case for digital sobriety.," *Cisco visual networking index: Forecast and trends, 2017-2022*, July 2019. 1, 115, 113

[3] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012. 1, 115, 113

[4] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards including high efficiency video coding (hevc)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, Dec 2012. 1, 115, 113

[5] T. K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J. Ohm, and G. J. Sullivan, "Video quality evaluation methodology and verification testing of hevc compression performance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 76–90, Jan 2016. 1, 12, 30, 115, 113

[6] Frank Bossen, Karsten Suehring, and Xiang Li, "JVET-P0003 : AHG report: Test model software development (AHG3)," 2019. 2, 91, 116, 89, 114

[7] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sebastien Lafond, and Johan Lilius, "Energy efficiency and performance management of parallel dataflow applications," in *Proceedings of the 2014 Conference on Design and Architectures for Signal and Image Processing*, Madrid, Spain, Oct. 2014, pp. 1–8, IEEE. 3, 117, 2, 115

[8] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, "Complexity Reduction Opportunities in the Future VVC Intra Encoder," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, Kuala Lumpur, Malaysia, Sept. 2019, pp. 1–6, IEEE. 3, 118, 116

[9] Leo Breiman, "Random Forests," *Machine learning*, vol. 45, pp. 5–32, 2001. 3, 52, 110, 118, 108, 116

[10] Thomas Amestoy, Alexandre Mercat, Wassim Hamidouche, Cyril Bergeron, and Daniel Menard, "Random Forest Oriented Fast QTBT Frame Partitioning," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, May 2019, pp. 1837–1841, IEEE. 4, 42, 118, 116

[11] Thomas Amestoy, Alexandre Mercat, Wassim Hamidouche, Daniel Menard, and Cyril Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 1313–1328, 2020. 4, 118, 116

[12] Thomas Amestoy, Wassim Hamidouche, Cyril Bergeron, and Daniel Menard, "Quality-Driven Dynamic VVC Frame Partitioning for Efficient Parallel Processing," in *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 3129–3133, IEEE. 4, 118, 116

[13] Gisle Bjontegaard, "Calculation of average PSNR differences between RD-Curves," Apr. 2001, VCEG-M33 ITU-T. 10, 12, 64, 84

[14] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process*, vol. 13, no. 4, pp. 600–612, April 2004. 11

[15] Mingliang Zhou, Xuekai Wei, Shiqi Wang, Sam Kwong, Chi-Keung Fong, Peter H. W. Wong, Wilson Y. F. Yuen, and Wei Gao, "SSIM-Based Global Optimization for CTU-Level Rate Control in HEVC," *IEEE Transactions on Multimedia*, vol. 21, no. 8, pp. 1921–1933, Aug. 2019. 11

[16] Anne Aaron et al Zhi Li, "Toward A Practical Perceptual Video Quality Metric," in *Netflix TechBlog*, Berlin, Germany, June 2016. 11

[17] Songnan Li, Fan Zhang, Lin Ma, and King Ngi Ngan, "Image Quality Assessment by Separately Evaluating Detail Losses and Additive Impairments," *IEEE Transactions on Multimedia*, vol. 13, no. 5, pp. 935–949, Oct. 2011. 11

[18] ITU-T, "ITU-T Rec. P.910: Subjective Video Quality Assessment Methods for Multimedia Applications," Apr. 2008. 12

[19] ITU-R, "ITU-R Rec. BT.500, ITU-R, Methodology for the Subjective Assessment of the Quality of Television Picture," Jan. 2012. 12

[20] H Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research*, pp. 399–417, May 1963. 13

[21] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," in *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, Thessaloniki, Greece, 2001, vol. 2, pp. 542–545, IEEE. 13

[22] CCITT, "Recommendation H.120 : Codecs for Videoconferencing Using Primary Digital Group Transmission," pp. 7–9, 1988. 14

[23] ITU-T, "Recommendation H.261: Video Codec for Audiovisual Services at p64 kbits," 1993. 14

[24] ISO/IEC, "International Standard 11172-2: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbit/s – Part 2: Video," 1993. 14

[25] ITU-T, "Recommendation H.262: Transmission of Non-Telephone Signals," 1994. 14

[26] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003. 14

[27] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012. 14

[28] Xiph.Org Foundation, "Theora Specifications," *https://www.theora.org/doc/Theora.pdf*, 2001. 14

[29] Jean-Marc Valin, Timothy B. Terriberry, Nathan E. Egge, Thomas Daede, Yushin Cho, Christopher Montgomery, and Michael Bebenita, "Daala: Building a next-generation video codec from unconventional technology," in *2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP)*, Montreal, QC, Canada, Sept. 2016, pp. 1–6, IEEE. 15

[30] Gisle Bjontegaard, Thomas Davies, Arild Fuldseth, and Steinar Midtskogen, "The Thor Video Codec," in *2016 Data Compression Conference (DCC)*, Snowbird, UT, USA, Mar. 2016, pp. 476–485, IEEE. 15

[31] Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, Ching-Han Chiang, Yunqing Wang, Paul Wilkins, Jim Bankoski, Luc Trudeau, Nathan Egge, Jean-Marc Valin, Thomas Davies, Steinar Midtskogen, Andrey Norkin, and Peter de Rivaz, "An Overview of Core Coding Tools in the AV1 Video Codec," in *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, June 2018, pp. 41–45, IEEE. 15

[32] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, July 2003. 17, 24

[33] Xin Zhao, Jianle Chen, Marta Karczewicz, Amir Said, and Vadim Seregin, "Joint Separable and Non-Separable Transforms for Next-Generation Video Coding," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2514–2525, May 2018. 19

[34] Liao Ru-Ling and Lim Chong Soon, "CE10.3.1.b: Triangular prediction unit mode," *JVET-L0124*, 2018. 19

[35] Michael Schafer, Bjorn Stallenberger, Jonathan Pfaff, Philipp Helle, Heiko Schwarz, Detlev Marpe, and Thomas Wiegand, "Efficient Fixed-Point Implementation Of Matrix-Based Intra Prediction," in *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 3364–3368, IEEE. 21, 102, 100

[36] Qin Yu, Liang Zhao, and Siwei Ma, "Parallel AMVP candidate list construction for HEVC," in *2012 Visual Communications and Image Processing*, San Diego, CA, USA, Nov. 2012, pp. 1–6, IEEE. 21

[37] Huanbang Chen, Fan Liang, and Sixin Lin, "Affine SKIP and MERGE modes for video coding," in *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, Xiamen, China, Oct. 2015, pp. 1–5, IEEE. 22

[38] Y. Ye J. Chen and S-H. Kim, "Algorithm description for Versatile Video Coding and Test Model 10 (VTM 10)," in *document JVET-N2002, 14th JVET meeting*, Teleconference, June 2020. 22

[39] W.-J. Chien, J. Boyce, Y.-W. Chen, R. Chernyak, K. Choi, R. Hashimoto, Y.-W. Huang, H. Jang, S. Liu, and D. Luo, "JVET AHG report: Tool reporting procedure (AHG13)," in *JVET document O0013 (JVET-O0013), 15th JVET Meeting*, Gothenburg, SE, 2019. 22, 30, 45, 129, 127

[40] Taoran Lu, Fangjun Pu, Peng Yin, Sean McCarthy, Walt Husak, Tao Chen, Edouard Francois, Christophe Chevance, Franck Hiron, Jie Chen, Ru-Ling Liao, Yan Ye, and Jiancong Luo, "Luma Mapping with Chroma Scaling in Versatile Video Coding," in *2020 Data Compression Conference (DCC)*, Snowbird, UT, USA, Mar. 2020, pp. 193–202, IEEE. 24, 93, 91

[41] 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*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012. 24

[42] 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*, vol. 22, no. 12, pp. 1755–1764, Dec. 2012. 24

[43] 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, and Shaw-Min Lei, "Adaptive Loop Filtering for Video Coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 934–945, Dec. 2013. 24

[44] B. Bross, J. Chen, and S. Liu, "Versatile Video Coding (Draft 5)," in *document JVET-N1001, 14th JVET meeting*, Geneva, CH, Mar. 2019. 24

[45] "N17055, Algorithm Description of Joint Exploration Test Model 7 (JEM 7)," vol. https://mpeg.chiariglione.org/standards/exploration/future-video-coding/n17055-algorithm-description-joint-exploration-test-model. 27, 47

[46] "Joint Exploration Model (JEM) reference software 7.0," *https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/branches/HM-16.6-JEM-7.0-dev/*. 27, 47

[47] "FFmpeg: Open Source and Cross-platform Multimedia Library," Oct. 2020. 27, 91, 97, 89, 95

[48] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "Hevc complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, Dec 2012. 29

[49] P. Topiwala, M. Krishnan, and W. Dai, "Performance comparison od vvc, av1 and evc," in *Proc. Applications od Digital Image Processing XLII*, Sep. 2019, pp. 290–301. 30

[50] F. Zhang, A. Katsenou, M. Afonso, G. Dimitov, and D. Bull, "Comparing vvc, hevc and av1 using objective and subjective assessments," in *arXiv:2003.10282*, Mar. 2020. 30

[51] F. Pakdaman, M. Adelimanesh, M. Gabbouj, and M. Hashemi, "Complexity analysis of next-generation vvc encoding and decoding," in *arXiv:2005.10801*, Mar. 2020. 30

[52] I. Siqueira, G. Correa, and M. Grellert, "Rate-distorsion and complexity comparison of hevc and vvc video encoders," in *Proc. 11th Latin Amer. Sympos. on Circuits and Syst. (LASCAS)*, Feb. 2020, pp. 1–4. 30

[53] T. Laude, Y. Adhisantoso, J. Vosges, M. Munderloh, and J. Ostermann, "A comprehensive video codec comparison," in *APSIPA Trans. on Signal and Inform. Process.*, Nov. 2019, vol. 8. 30

[54] A. Cerveira, L. Agostini, B. Zatt, and F. Sampaio, "Memory assessment of versatile video coding," in *arXiv:2005.13331*, June 2020. 30

[55] F. Bossen, X. Li, and K. Suering, "Jvet ahg report: test model software development (ahg3)," in *document JVET-S0003*, June 2020. 30

[56] ITU, "Recommandation ITU-T P.910 : Subjective video quality assessment methods for multimedia applications," 1999. 32, 54

[57] F. Kozamernik, V. Steinman, P. Sunna, and E. Wyckens, "SAMVIQ a New EBU Methodology for Video Quality Evaluations in Multimedia," Amsterdam, IBC 2004, pp. 191–202. 32

[58] ITU-R BT.1788, "Methodology for the Subjective Assessment of Video Quality in Multimedia Applications," 2007. 33, 36

[59] EBU BPN 056, "SAMVIQ a Subjective Assessment Methodology for Video Quality," in *Report by EBU Project Group B/VIM (Video In Multimedia)*, May. 2003. 33

[60] G. Gamst, L. Meyers, and A. Guarino, "Analysis of variance designs: A conceptual and computational approach with SPSS and SAS," in *Cambridge University Press*, New York, USA, 2008. 37

[61] "Callgrind," *http://valgrind.org/docs/manual/cl-manual.html*. 40, 44

[62] Adam Wieckowski, Jackie Ma, Heiko Schwarz, Detlev Marpe, and Thomas Wiegand, "Fast Partitioning Decision Strategies for The Upcoming Versatile Video Coding (VVC) Standard," in *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, Sept. 2019, pp. 4130–4134, IEEE. 41

[63] Sik-Ho Tsang, Yui-Lam Chan, and Wei Kuang, "Mode Skipping for HEVC Screen Content Coding via Random Forest," *IEEE Transactions on Multimedia*, vol. 21, no. 10, pp. 2433–2446, Oct. 2019. 42

[64] Liquan Shen and Guorui Feng, "Content-Based Adaptive SHVC Mode Decision Algorithm," *IEEE Transactions on Multimedia*, vol. 21, no. 11, pp. 2714–2725, Nov. 2019. 42

[65] Tao Zhang, Ming-Ting Sun, Debin Zhao, and Wen Gao, "Fast Intra-Mode and CU Size Decision for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1714–1726, Aug. 2017. 42, 47

[66] Sookyung Ryu and Jewon Kang, "Machine Learning-Based Fast Angular Prediction Mode Decision Technique in Video Coding," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5525–5538, Nov. 2018. 42, 47

[67] Hao Yang, Liquan Shen, Xinchao Dong, Qing Ding, Ping An, and Gangyi Jiang, "Low Complexity CTU Partition Structure Decision and Fast Intra Mode Decision for Versatile Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019. 42

[68] Wassim Hamidouche, Mickael Raulet, and Olivier Deforges, "4K Real-Time and Parallel Software Video Decoder for Multilayer HEVC Extensions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 169–180, Jan. 2016. 44, 75, 94, 96, 92

[69] Benjamin Bross, Mauricio Alvarez-Mesa, Valeri George, Chi Ching Chi, Tobias Mayer, Ben Juurlink, and Thomas Schierl, "HEVC real-time decoding," San Diego, California, United States, Sept. 2013, p. 88561R. 44, 75

[70] Wei Jiang, Hanjie Ma, and Yaowu Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Yichang, China, Apr. 2012, pp. 1836–1840, IEEE. 47

[71] Thasa Silva, Luciano Agostini, and Lus Cruz, "Fast HEVC Intra Prediction Mode Decision Based on Edge Direction Information," p. 5. 47

[72] J. An, Y.-W. Chen, K. Zhang, H. Huang, Y.-W. Huang, and S. Lei, "Block partitioning structure for next generation video coding," vol. https://www.itu.int/md/T13-SG16-C-0966/en, 2015. 48

[73] Jill Boyce, Karsten Suehring, and Xiang Li, "JVET-J1010 : JVET common test conditions and software reference configurations," 2018. 49, 54, 64, 67, 80, 84, 102

[74] Zhaoqing Pan, Sam Kwong, Ming-Ting Sun, and Jianjun Lei, "Early MERGE Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC," *IEEE Transactions on Broadcasting*, vol. 60, no. 2, pp. 405–412, June 2014. 49

[75] Pai-Tse Chiang and Tian Sheuan Chang, "Fast zero block detection and early CU termination for HEVC Video Coding," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, Beijing, May 2013, pp. 1640–1643, IEEE. 50

[76] Guilherme Correa, Pedro Assuncao, Luciano Agostini, and Luis da Silva Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, Nov. 2011. 50

[77] G. Correa, P. Assuncao, L. Agostini, and L. A. D. S. Cruz, "Coding Tree Depth Estimation for Complexity Reduction of HEVC," in *2013 Data Compression Conference*, Snowbird, UT, Mar. 2013, pp. 43–52, IEEE. 50

[78] Hoyoung Lee, Huik Jae Shim, Younghyeon Park, and Byeungwoo Jeon, "Early Skip Mode Decision for HEVC Encoder With Emphasis on Coding Quality," *IEEE Transactions on Broadcasting*, vol. 61, no. 3, pp. 388–397, Sept. 2015. 50

[79] Biao Min and Ray C. C. Cheung, "A Fast CU Size Decision Algorithm for the HEVC Intra Encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 892–896, May 2015. 50

[80] Liquan Shen, Zhaoyang Zhang, and Zhi Liu, "Effective CU Size Decision for HEVC Intracoding," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4232–4241, Oct. 2014. 50

[81] Alexandre Mercat, Florian Arrestier, Maxime Pelcat, Wassim Hamidouche, and Daniel Menard, "Prediction of quad-tree partitioning for budgeted energy HEVC encoding," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Lorient, Oct. 2017, pp. 1–6, IEEE. 50

[82] Liquan Shen, Zhi Liu, Zhaoyang Zhang, and Xuli Shi, "Fast Inter Mode Decision Using Spatial Property of Motion Field," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1208–1214, Oct. 2008. 50

[83] Jian Xiong, Hongliang Li, Qingbo Wu, and Fanman Meng, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 559–564, Feb. 2014. 50

[84] Saverio G. Blasi, Ivan Zupancic, Ebroul Izquierdo, and Eduardo Peixoto, "Fast HEVC coding using reverse CU visiting," in *2015 Picture Coding Symposium (PCS)*, Cairns, Australia, May 2015, pp. 50–54, IEEE. 50

[85] Zhenyu Liu, Xianyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang, "CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5088–5103, Nov. 2016. 50, 111, 109

[86] Fanyi Duanmu, Zhan Ma, and Yao Wang, "Fast CU partition decision using machine learning for screen content compression," in *2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, QC, Canada, Sept. 2015, pp. 4972–4976, IEEE. 50, 55

[87] Xiaolin Shen and Lu Yu, "CU splitting early termination based on weighted SVM," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, Dec. 2013. 50

[88] Xiaolin Shen, Lu Yu, and Jie Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," in *2012 Picture Coding Symposium*, Krakow, May 2012, pp. 453–456, IEEE. 51, 55

[89] Alexandre Mercat, Florian Arrestier, Maxime Pelcat, Wassim Hamidouche, Daniel Menard, INSA Rennes, and Institut Pascal, "Machine Learning Based Choice of Characteristics for the One-Shot Determination of the HEVC Intra Coding Tree," *2018 Picture Coding Symposium (PCS)*, pp. 263–267. 51, 62

[90] Yoshiya Yamamoto, "AHG5 : Fast QTBT encoding configuration," *JVET-D0095*, Oct. 2016. 51, 69

[91] Han Huang, Shan Liu, and Yu-Wen Huang, "Ahg5 : Speed-up for jem-3.1," *JVET-D0077*, Oct. 2016. 51

[92] Po-Han Lin, Chun-Lung Lin, and Yao-Jen Chang, "AHG5 : Enhanced fast algorithm of JVET-E0078," *JVET-F0063*, Apr. 2017. 51, 69

[93] Zhipeng Jin, Ping An, Liquan Shen, and Chao Yang, "CNN oriented fast QTBT partition algorithm for JVET intra coding," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, St. Petersburg, FL, Dec. 2017, pp. 1–4, IEEE. 51, 111, 109

[94] Zhao Wang, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, and Siwei Ma, "Fast QTBT Partitioning Decision for Interframe Coding with Convolution Neural Network," *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 2550–2554, 2018. 51, 69, 70, 129, 127

[95] Zhao Wang, Shiqi Wang, Jian Zhang, Shanshe Wang, and Siwei Ma, "Probabilistic Decision Based Block Partitioning for Future Video Coding," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1475–1486, Mar. 2018. 51, 56, 69, 70, 129, 127

[96] Anna Bosch, Andrew Zisserman, and Xavier Munoz, "Image Classification using Random Forests and Ferns," in *2007 IEEE 11th International Conference on Computer Vision*, Rio de Janeiro, Brazil, 2007, pp. 1–8, IEEE. 51

[97] Gabriele Fanelli, Juergen Gall, and Luc Van Gool, "Real time head pose estimation with random regression forests," in *CVPR 2011*, Colorado Springs, CO, USA, June 2011, pp. 617–624, IEEE. 51

[98] Brian C. Ross, "Mutual Information between Discrete and Continuous Data Sets," *PLoS ONE*, vol. 9, no. 2, pp. e87357, Feb. 2014. 52

[99] "x265," *https://bitbucket.org/multicoreware/x265*. 56

[100] Guilherme Correa, Pedro A. Assuncao, Luciano Volcan Agostini, and Luis A. da Silva Cruz, "Fast HEVC Encoding Decisions Using Data Mining," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 4, pp. 660–673, Apr. 2015. 62

[101] Kiran Misra, Andrew Segall, Michael Horowitz, Shilin Xu, Arild Fuldseth, and Minhua Zhou, "An Overview of Tiles in HEVC," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 969–977, Dec. 2013. 75

[102] Chi Ching Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel Scalability and Efficiency of HEVC Parallelization Approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1827–1838, Dec. 2012. 77

[103] Iago Storch, Daniel Palomino, Bruno Zatt, and Luciano Agostini, "Speedup-aware history-based tiling algorithm for the HEVC standard," in *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA, Sept. 2016, pp. 824–828, IEEE. 79, 80

[104] Maria Koziri, Panos K. Papadopoulos, Nikos Tziritas, Nikos Giachoudis, Thanasis Loukopoulos, Samee U. Khan, and Georgios I. Stamoulis, "Heuristics for tile parallelism in HEVC," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, Aug. 2017, pp. 1514–1518, IEEE. 79, 80

[105] Yong-Jo Ahn, Tae-Jin Hwang, Dong-Gyu Sim, and Woo-Jin Han, "Complexity model based load-balancing algorithm for parallel tools of HEVC," in *2013 Visual Communications and Image Processing (VCIP)*, Kuching, Malaysia, Nov. 2013, pp. 1–5, IEEE. 79

[106] Panos K. Papadopoulos, Maria Koziri, and Thanasis Loukopoulos, "A Fast Heuristic for Tile Partitioning and Processor Assignment in Hevc," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Athens, Oct. 2018, pp. 4143–4147, IEEE. 79

[107] Xin Jin and Qionghai Dai, "Clustering-Based Content Adaptive Tiles Under On-chip Memory Constraints," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2331–2344, Dec. 2016. 79

[108] Giovani Malossi, Daniel Palomino, Claudio Diniz, Altamiro Susin, and Sergio Bampi, "Adjusting video tiling to available resources in a per-frame basis in High Efficiency Video Coding," in *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, Vancouver, BC, Canada, June 2016, pp. 1–4, IEEE. 79, 112, 110

[109] Cauane Blumenberg, Daniel Palomino, Sergio Bampi, and Bruno Zatt, "Adaptive content-based Tile partitioning algorithm for the HEVC standard," in *2013 Picture Coding Symposium (PCS)*, San Jose, CA, USA, Dec. 2013, pp. 185–188, IEEE. 79, 86

[110] Muhammad Shafique, Muhammad Usman Karim Khan, and Jorg Henkel, "Power efficient and workload balanced tiling for parallelized high efficiency video coding," in *2014 IEEE International Conference on Image Processing (ICIP)*, Paris, France, Oct. 2014, pp. 1253–1257, IEEE. 79

[111] Chia-Hsin Chan, Chun-Chuan Tu, and Wen-Jiin Tsai, "Improve load balancing and coding efficiency of tiles in high efficiency video coding by adaptive tile boundary," *Journal of Electronic Imaging*, vol. 26, no. 1, pp. 013006, Jan. 2017. 80, 81, 86

[112] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100, 1979. 81

[113] Mark D Hill and Michael R Marty, "Amdahl's Law in the Multicore Era," p. 6. 84, 111, 109

[114] Vivienne Sze and Madhukar Budagavi, "High Throughput CABAC Entropy Coding in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1778–1791, Dec. 2012. 92, 90

[115] S. Thakkur and T. Huff, "Internet Streaming SIMD Extensions," *Computer*, vol. 32, no. 12, pp. 26–34, Dec. 1999. 93, 94, 101, 127, 91, 92, 99, 125

[116] Chi Ching Chi, Mauricio Alvarez-Mesa, Benjamin Bross, Ben Juurlink, and Thomas Schierl, "SIMD Acceleration for HEVC Decoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 841–855, May 2015. 94, 92

[117] Leju Yan, Yizhou Duan, Jun Sun, and Zongming Guo, "Implementation of HEVC decoder on x86 processors with SIMD optimization," in *2012 Visual Communications and Image Processing*, San Diego, CA, USA, Nov. 2012, pp. 1–6, IEEE. 94, 92

[118] Hyunki Baik and Hwangjun Song, "A complexity-based adaptive tile partitioning algorithm for HEVC decoder parallelization," in *2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, QC, Canada, Sept. 2015, pp. 4298–4302, IEEE. 94, 92

[119] Seehwan Yoo and Eun-Seok Ryu, "Parallel HEVC decoding with asymmetric mobile multicores," *Multimedia Tools and Applications*, vol. 76, no. 16, pp. 17337–17352, Aug. 2017. 94, 92

[120] F. Henry and S. Pateux, "Wavefront Parallel Processing," *Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E196*, Mar. 2011. 95, 93

[121] Chi Ching Chi, Mauricio Alvarez-Mesa, Ben Juurlink, Valeri George, and Thomas Schierl, "Improving the parallelization efficiency of HEVC decoding," in *2012 19th IEEE International Conference on Image Processing*, Orlando, FL, USA, Sept. 2012, pp. 213–216, IEEE. 95, 96, 93, 94

[122] Shaobo Zhang, Xiaoyun Zhang, and Zhiyong Gao, "Implementation and improvement of Wavefront Parallel Processing for HEVC encoding on many-core platform," in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, Chengdu, China, July 2014, pp. 1–6, IEEE. 95, 93

[123] Rafael Rodriguez-Sanchez and Enrique S. Quintana-Orti, "Tiles-and WPP-based HEVC Decoding on Asymmetric Multi-core Processors," in *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, Laguna Hills, CA, USA, Apr. 2017, pp. 299–302, IEEE. 96, 94

[124] P. Habermann, B. Juurlink, C. C. Chi, and M. Alvarez-Mesa, "Efficient Wavefront Parallel Processing for HEVC CABAC Decoding," in *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Västerås, Sweden, Mar. 2020, pp. 339–343, IEEE. 96, 94

[125] Srinivas Gudumasu, Saurav Bandyopadhyay, and Yong He, "Software-based versatile video coding decoder parallelization," in *Proceedings of the 11th ACM Multimedia Systems Conference*, Istanbul Turkey, May 2020, pp. 202–212, ACM. 96, 94

[126] Adam Wieckowski, Gabriel Hege, Christian Bartnik, Christian Lehmann, Christian Stoffers, Benjamin Bross, and Detlev Marpe, "Towards A Live Software Decoder Implementation For The Upcoming Versatile Video Coding (VVC) Codec," in *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 3124–3128, IEEE. 96, 103, 94, 101

[127] Anand Meher Kotra, Mickael Raulet, and Olivier Deforges, "Comparison of different parallel implementations for deblocking filter of HEVC," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 2013, pp. 2721–2725, IEEE. 97, 95

[128] Hyunho Jo, Donggyu Sim, and Byeungwoo Jeon, "Hybrid parallelization for HEVC decoder," in *2013 6th International Congress on Image and Signal Processing (CISP)*, Hangzhou, China, Dec. 2013, pp. 170–175, IEEE. 97, 95

[129] IETR/VAADER, "Open Source HEVC Decoder (OpenHEVC)," *https://github.com/OpenHEVC*, 2016. 97, 95

[130] F. Pescador, J. P. Cano, M. J. Garrido, E. Juarez, and M. Raulet, "A DSP HEVC decoder implementation based on OpenHEVC," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, Jan. 2014, pp. 61–62, IEEE. 97, 95

[131] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac, "GPAC: open source multimedia framework," in *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, Augsburg, Germany, 2007, p. 1009, ACM Press. 97, 95

[132] J Lainema, "CE7 : Joint Coding of Chrominance Residuals (CE7-1)," *document JVET-N0054*, Mar. 2019. 102, 100

[133] Chen Ching-Yeh and Chuang Tzu-Der, "Cel-related: Sepa-rate tree partitioning at 64x64-luma/32x32-chroma unitleve," *11th JVET Meeting: Ljubljana*, 2018. 102, 100

[134] J. Brandenburg, A. Wieckowski, T. Hinz, A. Henkel, V. George, I. Zupancic, C. Stoffers, D. Marpe, and B. Bross, "Towards fast and efficient vvc encoding," in *2020 IEEE 22nd Workshop on Multimedia Signal Processing (MMSP 2020)*, Sep. 2020. 111, 109

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Optimisation du Codec VVC basé sur la Réduction de Complexité et le Traitement Parallèle

Nom Prénom de l'auteur : AMESTOY THOMAS

Membres du jury :
- Madame GUILLEMOT Christine
- Monsieur WIEN Mathias
- Monsieur MENARD Daniel
- Monsieur HAMIDOUCHE Wassim
- Monsieur CAGNAZZO Marco
- Monsieur ANTONINI Marc

Président du jury : Christine GUILLEMOT

Date de la soutenance : 12 Mars 2021

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 12 Mars 2021

Signature du président de jury

Le Directeur,

Abdellatif MIRAOUI

Titre Optimisation du Codec VVC basé sur la Réduction de Complexité et le Traitement Parallèle

Mots clés : Codage vidéo, VVC, Réduction de la complexité, Traitement parallèle, Apprentissage machine

Résumé : Au cours de la dernière décennie, les progrès des technologies numériques ne augmentation importante du volume de contenu vidéo échangé. Fina lisée en juillet 2020, la nouvelle norme de codage vidéo Versatile Video Coding (VVC) développée par l'ITU-T VCEG et l'ISO/IEC MPEG répond au besoin de performances de codage plus élevées. Pour une même qualité vidéo, VVC permet d'économiser 40% de débit par rapport au dernier codec vidéo High Efficiency Video Coding (HEVC). Toutefois, pour obtenir ces économies de débit, des outils complexes ont été ajoutés au niveau de l'encodeur et du décodeur. Ce document présente un ensemble de contributions visant à réduire efficacement la consommation d'énergie et le temps de traitement des codecs VVC, tout en minimisant les pertes de qualité du codage. Tout d'abord, un schéma de partitionnement en bloc léger et adaptable, basé sur une approche d'apprentissage machine, est proposé. La deuxième contribution, du côté de l'encodeur, tire parti des outils de parallélisme de haut niveau inclus dans VVC, tels que le parallélisme en tile et en slice. Du côté du décodeur, l'augmentation de la complexité de calcul de la norme VVC par rapport à la norme HEVC est d'un facteur 2 environ. Un décodeur VVC en temps réel et à faible mémoire, basé sur le projet openVVC open source, est proposé pour la configuration de codage Intra.

Title : VVC Codec Optimization through Complexity Reduction and Parallel Processing

Keywords : Video coding, VVC, Complexity reduction, Parallel processing, Machine learning

Abstract : During the last decade, the progress in digital technologies has led to an important increase in the volume of exchanged video content. Finalized in July 2020, the new video coding standard Versatile Video Coding (VVC) developed by the ITU-T VCEG and ISO/IEC MPEG answers the need for higher coding performance. For the same video quality, VVC provides 40% bit-rate savings over the latest state-of-the-art video codec High Efficiency Video Coding (HEVC). However, in order to obtain these bit-rate savings, computationally expensive tools have been added at both encoder and decoder sides. This document presents a set of contributions aiming at reducing efficiently the energy consumption and processing time of VVC codecs, while minimizing the coding quality losses. First, a lightweight and tunable block partitioning scheme based on a machine learning approach is proposed. The second contribution at the encoder side takes advantage of the high-level parallelism tools included in VVC, such as tile and slice parallelism. At decoder side, the computational complexity increase of VVC standard compared to HEVC is approximately a factor 2. A real-time and low-memory VVC decoder based on the open source openVVC project is proposed for Intra coding configuration.