UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CAUANE BLUMENBERG SILVA

Adaptive Tiling Algorithm Based on Highly Correlated Picture Regions for the HEVC Standard

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Prof. Dr. Sergio Bampi
Advisor

Prof. Dr. Bruno Zatt
Co-Advisor

Porto Alegre, março de 2014.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Blumenberg, Cauane

Adaptive Tiling Algorithm Based on Highly Correlated Picture Regions for the HEVC Standard / Cauane Blumenberg Silva. – 2014.

91 f.:il.

Orientador: Sergio Bampi; Co-orientador: Bruno Zatt.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2014.

1.Codificação de Vídeo Digital. 2.Padrão de Codificação de Vídeos de Alta Eficiência. 3.Ferramentas Orientadas ao Paralelismo. I. Bampi, Sergio. II. Zatt, Bruno. III. *Adaptive Tiling Algorithm Based on Highly Correlated Picture Regions for the HEVC Standard*.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecário-Chefe do Instituto de Informática: Alexsander Borges Ribeiro

AGRADECIMENTOS

Entendo que a família, principalmente os pais, é quem dita como seus filhos vão se portar no início da vida. Depois disso, algumas coisas vão mudando e nos tornamos seres únicos. Embora tenha me tornado uma pessoa diferente, tenho orgulho de afirmar que minhas ideias convergem na mesma direção das ideias dos meus pais, os quais foram pessoas fundamentais para que eu pudesse completar estes dois anos de mestrado. Os conselhos, o apoio psicológico e todo o amor que nos une são incríveis! Obrigado por tudo pai e mãe, amo muito vocês!

Também gostaria de agradecer, e muito, a minha "colega de quarto" e irmã. Depois de mais de oito anos separados por cerca de 270 quilômetros, voltamos a morar juntos e fizemos cair por terra a ideia que todos tinham de que irmãos não podem conviver em harmonia. Estão todos errados! Quem conhece, sabe o quanto te admiro e o quanto gosto da tua parceria mana, te amo para sempre!

Continuando na linha familiar, gostaria de agradecer à minha namorada. Rafa, tu és muito mais que minha companheira, tu és um membro da família. Tenho muito a te agradecer! Obrigado por entender que, quando o amor é forte, uma mera distância não vai fazer nada se apagar. Obrigado pelos conselhos, pelas visitas em Porto Alegre, pelas viagens pelo mundo afora e pelas risadas. Obrigado por fazer parte da terceira "formatura" da minha vida, obrigado por fazer parte da minha vida! Te amo muito, meu amor!

Gostaria também de agradecer aos meus demais familiares, que acompanharam de muito perto minha trajetória no mestrado. Em especial, gostaria de agradecer à minha melhor amiga, minha avó Bety. Certamente, a pessoa que mais se orgulhou de mim. A pessoa que mais se alegrava com um simples telefonema e passeio de carro. A pessoa que sabia viver como ninguém, e que sorria mesmo sem ter por que. Sem sombra de dúvidas, a pessoa que mais vibrou com meu mestrado, mas que infelizmente foi dar risadas em outro lugar, e não teve tempo de me acompanhar até o final. Vó, tu és a pessoa mais especial que já conheci, e isto tudo é para ti! Como te disse pela última vez: "O que fica é a alegria do teu sorriso!".

O mestrado me fez ganhar não somente conhecimento, mas também amigos muito especiais. Gostaria de agradecer a parceria, o apoio e os ensinamentos dos meus colegas de laboratório. Obrigado Daniel, Leo, Felipe, Duda, Mateus, Vizzotto, Cláudio e Kleber. Com certeza vocês tornaram essa caminhada muito mais tranquila. Gostaria também de agradecer aos amigos que conquistei durante estes dois anos em Porto Alegre. Pude conhecer muita gente nova, mas gostaria de agradecer em especial ao pessoal do Futebol de Segunda-Feira. Faça chuva ou faça sol, a parceria está sempre pronta para o embate. Obrigado por fazer com que a segunda-feira fosse o dia mais esperado da semana.

Além destes, também não posso deixar de agradecer aos amigos de infância. A galera do UFE é especial, em minhas constantes visitas a Pelotas sempre havia

programações especiais me esperando. Obrigado UFE, vocês são a família que escolhi, e confesso que escolhi muito bem. Vocês são demais!

Por fim, mas não menos importante, gostaria de agradecer ao meu orientador prof. Sergio Bampi, e ao meu co-orientador prof. Bruno Zatt. Obrigado pelos conselhos, obrigado pelas conversas, obrigado pelos ensinamentos e obrigado por acreditar no meu trabalho. Muito obrigado, vocês são geniais!

Algoritmo de *Tiling* Adaptativo Baseado em Regiões Altamente Correlacionadas de um Quadro para o Padrão de Codificação de Vídeos de Alta Eficiência

RESUMO

Esta dissertação de mestrado propõe um algoritmo adaptativo que é capaz de dinamicamente definir partições *tile* para quadros intra- e inter-preditos com o objetivo de reduzir o impacto na eficiência de codificação. *Tiles* são novas ferramentas orientadas ao paralelismo que integram o padrão de codificação de vídeos de alta eficiência (HEVC – *High Efficiency Video Coding standard*), as quais dividem o quadro em regiões retangulares independentes que podem ser processadas paralelamente. Para viabilizar o paralelismo, os *tiles* quebram as dependências de codificação através de suas bordas, gerando impactos na eficiência de codificação. Este impacto pode ser ainda maior caso os limites dos *tiles* dividam regiões altamente correlacionadas do quadro, porque a maior parte das ferramentas de codificação usam informações de contexto durante o processo de codificação. Assim, o algoritmo proposto agrupa as regiões do quadro que são altamente correlacionadas dentro de um mesmo *tile* para reduzir o impacto na eficiência de codificação que é inerente ao uso de *tiles*. Para localizar as regiões altamente correlacionadas do quadro de uma maneira inteligente, as características da imagem e também as informações de codificação são analisadas, gerando mapas de particionamento que servem como parâmetro de entrada para o algoritmo. Baseado nesses mapas, o algoritmo localiza as quebras naturais de contexto presentes nos quadros do vídeo e define os limites dos *tiles* nessas regiões. Dessa maneira, as quebras de dependência causadas pelas bordas dos *tiles* coincidem com as quebras de contexto naturais do quadro, minimizando as perdas na eficiência de codificação causadas pelo uso dos *tiles*. O algoritmo proposto é capaz de reduzir mais de 0.4% e mais de 0.5% o impacto na eficiência de codificação causado pelos tiles em quadros intra-preditos e inter-preditos, respectivamente, quando comparado com *tiles* uniformes.

Palavras-Chave: Codificação de Vídeo Digital, Padrão de Codificação de Vídeos de Alta Eficiência, Ferramentas Orientadas ao Paralelismo, Partições *Tile*, Eficiência de Codificação.

Adaptive Tiling Algorithm Based on Highly Correlated Picture Regions for the HEVC Standard

ABSTRACT

This Master Thesis proposes an adaptive algorithm that is able to dynamically choose suitable tile partitions for intra- and inter-predicted frames in order to reduce the impact on coding efficiency arising from such partitioning. Tiles are novel parallelism-oriented tools that integrate the High Efficiency Video Coding (HEVC) standard, which divide the frame into independent rectangular regions that can be processed in parallel. To enable the parallelism, tiles break the coding dependencies across their boundaries leading to coding efficiency impacts. These impacts can be even higher if tile boundaries split highly correlated picture regions, because most of the coding tools use context information during the encoding process. Hence, the proposed algorithm clusters the highly correlated picture regions inside the same tile to reduce the inherent coding efficiency impact of using tiles. To wisely locate the highly correlated picture regions, image characteristics and encoding information are analyzed, generating partitioning maps that serve as the algorithm input. Based on these maps, the algorithm locates the natural context break of the picture and defines the tile boundaries on these key regions. This way, the dependency breaks caused by the tile boundaries match the natural context breaks of a picture, then minimizing the coding efficiency losses caused by the use of tiles. The proposed adaptive tiling algorithm, in some cases, provides over 0.4% and over 0.5% of BD-rate savings for intra- and inter-predicted frames respectively, when compared to uniform-spaced tiles, an approach which does not consider the picture context to define the tile partitions.

Keywords: Digital Video Coding, High Efficiency Video Coding Standard, Parallelism-Oriented Tools, Tile partitions, Coding Efficiency.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS AND ACRONYMS

AdapTa	Adaptive Tiling Algorithm
AI	All Intra
AMP	Asymmetric Motion Partitioning
AMVP	Advanced Motion Vector Predictor
AVC	Advanced Video Coding
BD-rate	Bjontegaard Delta Bit-rate
CABAC	Context Adaptive Binary Arithmetic Coding
CB	Coding Block
CDR	Clean Decoded Refresh
CODEC	Encoder-Decoder
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
DBF	Deblocking Filter
DCT	Discrete Cosine Transform
DMV	Differential Motion Vector
DST	Discrete Sine Transform
ED	Euclidean Distance
FCFS	First Come First Serve
fps	frames per second
GB	Gigabyte
Gbps	Gigabits per Second
Chroma	Chrominance
HD	High Definition
HM	HEVC Test Model
HEVC	High-Efficiency Video Coding
JCT-VC	Joint Collaborative Team on Video Coding

LB	Low Delay B
LP	Low Delay P
Luma	Luminance
MC	Motion Compensation
ME	Motion Estimation
MPM	Most Probable Modes
MTU	Maximum Transmission Unit
MV	Motion Vector
PB	Prediction Block
PMV	Predictive Motion Vector
PSNR	Peak Signal-to-Noise Ratio
PU	Prediction Unit
QP	Quantization Parameter
Qstep	Quantization Step
Quadtree	Quadratic Tree
RA	Random Access
RDO	Rate Distortion Optimization
RDOQ	Rate Distortion Optimized Quantization
RGB	Red, Green and Blue
ROI	Regions of Interest
SAO	Sample Adaptive Offset
SMP	Symmetric Motion Partitioning
TU	Transform Unit
TZ Search	Test Zonal Search
UHD	Ultra-High Definition
WPP	Wavefront Parallel Processing
YCbCr	Luminance, Blue Chrominance and Red Chrominance

SUMMARY

1 INTRODUCTION

Recent advances on consumer multimedia devices enabled recording and reproducing high definition (HD) video content. Nowadays, various types of technologies, including mobile devices and video streaming, are able to handle HD720p and HD1080p video resolutions (NETFLIX, 2013). This kind of content involves a huge amount of data, which is only possible to be transmitted or stored due to the video encoding process. The broadly used H.264/Advanced Video Coding (H.264/AVC (WIEGAND, SULLIVAN and LUTHRA, 2003)) standard was a cornerstone to enable the manipulation of HD videos. However, the desire of users for better quality and immersion brings the need for manipulating beyond-HD video resolutions, such as 4K and 8K. Since the H.264/AVC standard is not optimized to deal with ultra-high definition (UHD) content (SULLIVAN, OHM, *et al.*, 2012), the High-Efficiency Video Coding (HEVC) standard (BROSS, HAN, *et al.*, 2013) was developed by the Joint Collaborative Team on Video Coding (JCT-VC) and achieved its first standardized version in mid-2013.

The JCT-VC main goal was to develop a new video coding standard capable of reducing by half the bit-rate, reaching the same visual quality when compared to the previous H.264/AVC standard. Recent studies show that the bit-rate reductions vary between 36% and 70% (OHM, SULLIVAN, *et al.*, 2012). However, this reduced bit-rate is achieved at the cost of two to ten times higher computational effort. To deal with this increased computational effort requirement, the HEVC focus on another key aspect, besides being able to handle UHD video resolutions: the expanded use of parallelism to assist the video encoding and decoding process (POURAZAD, DOUTRE and NASIOPOULOS, 2012).

Supporting parallelism is a very important feature on current multimedia applications, including video encoders-decoders (CODECs), as this kind of application involves a huge amount of data. In addition, the adoption of parallel architectures by modern consumer electronic devices, such as tablets, smartphones and computers, is growing (NVIDIA, 2010). In the H.264/AVC standard the parallelism was exploited by the use of slices, which delimit independent regions that are able to be encoded and decoded in parallel. However, these structures were originally proposed to diminish coding losses related to transmission errors. Thus, using slices to exploit parallelism showed to be very inefficient in terms of coding efficiency, as they require the insertion of additional headers on the final bit stream (JUURLINK, ALVAREZ-MESA, *et al.*, 2012). Considering this, some novel parallelism-oriented tools were proposed during the HEVC development, such as tiles.

Tiles divide the picture in rectangular regions, which are defined by horizontal and vertical boundaries. The number of partitions can be freely defined, and they can adopt uniform or non-uniform shapes. When uniform tiles are considered the encoder splits

the picture on equally sized partitions, whereas when non-uniform tiles are used the size and shape (the form factor can be freely defined but it must be rectangular) of each partition can be freely set. These partitions are considered independent, being able to be processed in parallel only because the coding dependencies are broken across tile boundaries. Still, these breaks reflect in coding losses, as it compromises the process of some coding tools such as entropy coding, intra- and inter-prediction, which are strongly based on context information to fully achieve higher compression rates. However, these are reduced losses when compared to the ones caused by slices as no additional headers are inserted in the final bit stream (CHI, MESA, *et al.*, 2012).

Some works like (MISRA, SEGALL, *et al.*, 2013) and (FULDSETH, HOROWITZ, *et al.*, 2012) compare the coding efficiency between using slices and tiles to exploit parallelism, proving that the tool proposed in the HEVC standard cause less impact on the overall encoding process. In (CHI, MESA, *et al.*, 2012), the authors perform the same comparison as the previous metioned works, although an additional parallelism tool is considered in the analysis. Apart from comparing the coding efficiency of slices and tiles, it is also analyzed the impact of using the Wavefront Parallel Processing (WPP) to exploit parallelism, which is another novel parallelism-oriented tool proposed in the HEVC standard that splits the picture into row partitions which can be processed in parallel. In this case, the WPP achieved better coding efficiency, however the partitions defined by this approach are not considered totally independent as references from portions of the previous encoded row are used to encode the current row. At the same time that this dependency leads to better coding efficiency, it also compromises the parallel processing potential since it requieres an inter communication between the processing units that are encoding the WPP rows.

As can be seen, the above mentioned works only perform a comparison of the video coding parallelization tools. It can be noticed that there is a trade off between coding efficiency and parallelism potential in the analyzed tools. Although, no works consider the possibility of reducing the impact in coding efficiency inherently caused by the parallelism-oriented tools, without abdicating to exploit parallelism in the video encoding/decoding process. Considering this, a chalenge arises: *adopt the novel HEVC parallelism-oriented tools in the video coding process whereas reducing the inherent coding efficiency losses of these tools.*

1.1 Objectives of this Master Thesis

This work proposes an adaptive algorithm that is able to dynamically choose suitable tiling partitions to reduce the impact on coding efficiency caused by the use of tiles. The algorithm analyses several different strategies, such as image characteristics and coding information, to define the tile partitions. The tiling definition is based on non-uniform tiles since its shape and size can be freely defined. This work also proposes an alternative tile scheduling scheme to diminish the issue of defining non-balanced tiling patterns, as tiles with different sizes can compromise the parallelism potential of the algorithm. This scheme wisely distributes the tile workload between the available processing units to increase the algorithm parallelism potential.

1.2 Text Organization

Chapter 2 presents basic concepts of digital videos and video coding concepts. Additionally, it describes the profiles and novel tools incorporated by the HEVC

standard. Chapter 3 discusses the video coding parallelism tools, comparing the novel parallelism-oriented tools to the ones used by previous video coding standards. It also presents an analysis of the efficiency of using tile partitioning and the challenges of exploiting parallelism in the video coding process. Chapter 4 describes the proposed adaptive tiling algorithm focusing on two different coding configurations of the HEVC standard, along with the proposed tile scheduling scheme. Chapter 5 presents the experimental results in terms of coding efficiency and also parallelism potential. Finally, Chapter 6 concludes this work and points future extensions of this work.

2 VIDEO CODING CONCEPTS AND THE HEVC STANDARD

Video compression is essential to enable broadcasting and storing digital video content. Non-compressed digital video sequences involve huge amount of data to be handled. For instance, considering a 10 minutes high definition HD1080p (*1920x1080* pixels) uncompressed video sequence with 30 frames per second and 24 bits per pixel, it would be necessary 1.5 gigabits per second (1.5 Gbps) to transmit this sequence. To store this same 10 minute sequence, it would be necessary 112 gigabytes (112 GB) (AGOSTINI, 2007). The natural increase on video resolution, as nowadays 4K (*3840x2160*) and 8K (*7680x4320*) are already being considered, require even higher and unfeasible bandwidth and storing capacities for uncompressed videos. This way, compressing digital videos is a key factor to reduce the requirements to store and broadcast HD and beyond-HD video content.

In spite of requiring a huge amount of data to represent digital videos, this kind of content presents high data redundancy. The main goal of the video CODECs is to exploit and reduce as much as possible the data redundancies, representing the same digital video sequence with reduced amount of data. The video characteristics and the process of exploiting the data redundancies are described on the following sections.

2.1 Digital Video Characteristics and Color Sampling

Digital videos are composed by a series of static images (entitled *frames* or *pictures*) that, presented on a certain exhibition rate, give the feeling of movement. The exhibition rate, in general, should be between 24 and 30 frames per second to guarantee a smooth motion sense (RICHARDSON, 2010). Although, the dissemination of HD video content, and the desire for better realism and immersion leads to higher frame rates, which are reaching up to 120 frames per second (SAMPAIO, 2013).

The frames are composed of a pixel matrix, which stores one or more samples that represent the color and brightness information at that point. One key factor that involves the pixel representation in digital videos is the amount of bits per sample. The greater the number of bits per sample, the more accurate will be the color representation. Considering that the current video CODECs are designed to deal with HD and UHD video sequences, usually they consider between 8 and 10 bits per sample to achieve a better image quality (BROSS, HAN, *et al.*, 2013). Table 2.1 presents the amount of values (gray-scale tones of color information) that can be represented when 8 to 10 bits per sample are adopted.

Table 2.1 – Bits per sample and the number of values represented.

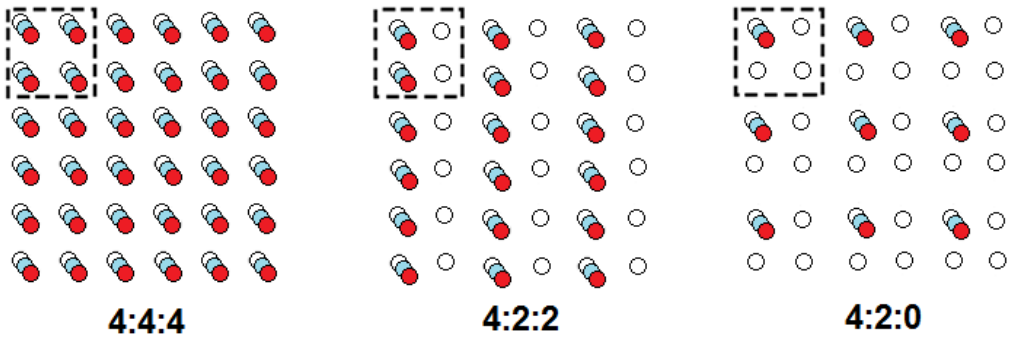
Bits per Sample	Tones/Colors Represented
8 bits	256
9 bits	512
10 bits	1024

The color information can be represented by different color spaces. One of the most popular, used by several devices, is the RGB. The RGB color space stores the information in three channels: ***R**ed*, ***G**reen* and ***B**lue*, hence the acronym RGB. However, this color space is not a good option to be used by video CODECs, since the correlation between its channels is very strong (RICHARDSON, 2010). To address this issue, video CODECs generally adopt the YCbCr color space which distinguishes luminosity information from color information.

Alike the RGB color space, the YCbCr color space is divided into three channels: *luminance* (**Y**), *blue chrominance* (**Cb**), and *red chrominance* (**Cr**). However, there is not such strong correlation between the channels, since the luminance (*luma* channel) channel stores the light intensity information and the other two channels, Cb and Cr (*chroma* channels), store the color tones information. This allows the video CODECs to adaptively exploit specific properties depending on the channel that is being processed.

Taking into consideration that the human visual system is more sensitive to luma components, in general the video CODECs discard some chroma information that do not affect the human visual perception, this process is called *color subsampling*. By applying the subsampling in chroma components, the color samples matrix has its resolution reduced causing negligible losses to the subjective video quality. The video encoding process commonly considers three chroma subsampling formats: *(i)* 4:4:4, *(ii)* 4:2:2, and *(iii)* 4:2:0. In the first format the chroma subsampling is not applied, as for each of the four luma samples there are also four Cb samples and four Cr samples. The 4:2:2 format considers four luma samples and two samples for each chroma channel, whereas in the 4:2:0 format there are four luma samples, one red chroma sample and one blue chroma sample. Figure 2.1 illustrates the three described chrominance subsampling formats.

Figure 2.1 – Chrominance subsampling formats.



Afonso (2012, p. 53).

The chrominance subsampling is one of the first and immediate types of video compression techniques that exploit the context redundancy, since the samples use information of neighboring samples to be represented. For instance when the 4:2:2 subsampling format is adopted, 25% of the video information is discarded without impacting the subjective video quality. Whereas, adopting the 4:2:0 subsampling format leads to a discard of 50% of the video information when compared to the format 4:4:4.

Apart from using the chrominance subsampling as a video compression method, much of the compression gains achieved by the video CODECs are related to context redundancy. Considering high resolution digital videos, it is natural to imagine that there will be certain homogeneous regions on a picture, as the images are represented by a high number of samples. This way, neighboring samples located on these homogeneous regions, such as a background for instance, present similar spatial context information (i.e. data redundancy among spatially neighboring samples). Moreover, digital videos adopt a high frame rate. Taking into consideration that in one second it is needed a minimum of 24 frames to guarantee the motion sense (RICHARDSON, 2010); neighboring frames are likely to have similar information between them (temporal context). Exploiting these context information, which will be described on the following section, is essential to achieve high compression rates.

2.2 Context Redundancy in Digital Videos

The video coding main objective is to reduce the amount of information used to represent each frame by exploiting the context redundancies. Redundancies in video coding are data that do not contribute with new relevant information to represent video's content. Considering this, the data can be reduced or even discarded when the context redundancy information is used. There are three types of data redundancies exploited by the video coding standards: *(i)* spatial, *(ii)* temporal, and *(iii)* entropic.

2.2.1 Spatial Redundancy

Spatial redundancy in digital videos is the correlation between neighboring samples inside the same frame. This redundancy is also known as *intra-frame redundancy* (GHANBARI, 2003) or *inter-pixel redundancy*, and can be noticed both in the spatial and in the frequency domain.

In the spatial domain the sample correlation can be visually recognized by pixels that present similar values. This kind of redundancy is commonly treated by the intra-frame prediction, which is a step usually performed by the video coding standards. Whereas, in the frequency domain, reducing the spatial redundancy is performed by applying a quantization step. To apply the quantization, the video information must be transformed from the spatial to the frequency domain. Then, the quantization discards the remainder generated by applying an integer division on the transformed information. When this information is discarded it makes impossible to retrieve the original video data. The resulting subjective video quality depends on the amount of information discarded, which can be controlled via a parameter during encoding time (OU, ZENG and WANG, 2012).

2.2.2 Temporal Redundancy

The high exhibition frame rate of the current digital videos contributes to the existence of big similarity between temporal neighboring frames. In some cases the information of samples located in different frames is the same, on other the samples

have value variation (WIEGAND, SULLIVAN, *et al.*, 2003). This variation can be caused, for instance, by lightning changes or by moving objects.

These temporal redundancies, also known as *inter-frame redundancy* (GHANBARI, 2003), if efficiently predicted can lead to high compression rates. However, the inter-frame prediction is the most computational intensive step of the video coding process (MONTEIRO, 2012).

2.2.3 Entropic Redundancy

Unlike the spatial and temporal redundancies, the entropic redundancy is not related to the video content, but to how the symbols are represented and to the probability of occurrence of each symbol. Video coding standards consider that the amount of new information transmitted by a symbol diminishes as it becomes more frequent (SZE and BUDAGAVI, 2008). Due to that, the main goal is to transmit the largest possible amount of information per symbol, willing to represent more information with fewer bits.

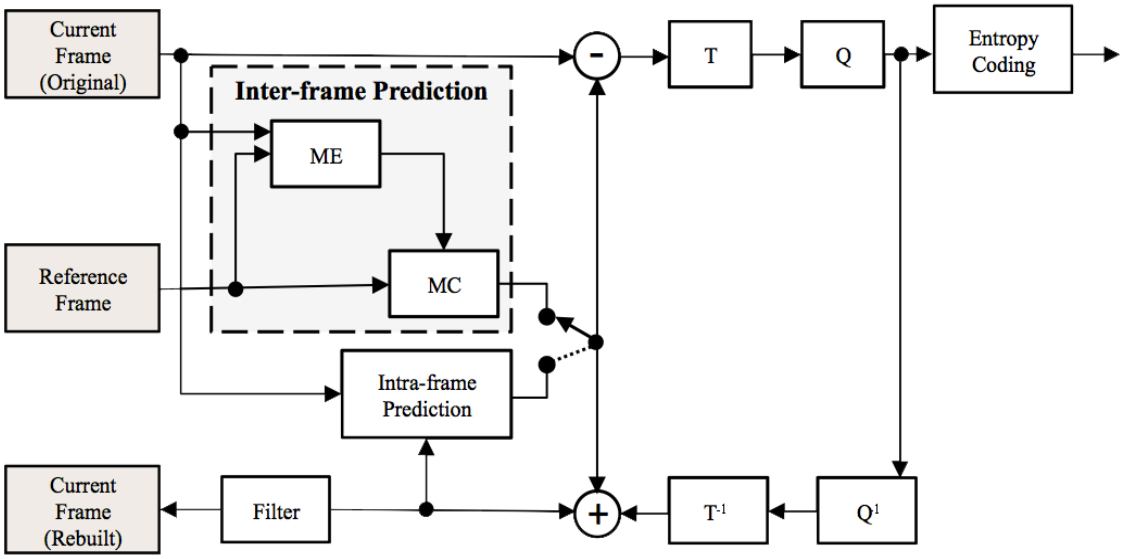
To achieve this goal, several techniques and algorithms are employed to exploit the entropic redundancy. These algorithms are highly based on context, since it uses previous encoded symbols as reference to encode the following symbols. This step does not cause any impact on the video quality as no information is discarded; instead, the information is just reorganized to be transmitted in a compact manner.

2.3 Basic Concepts on Video Coding

As already mentioned, the video coding standards' main goal is to exploit the context redundancy. However, analyzing the spatial, temporal, and entropic redundancies for the entire frame at once would lead to a huge computational effort. On the other hand, doing it sample by sample would lead to inefficient coding efficiency and to a long serial process. Hence, video coding standards split the video frames into blocks, which are composed by a group of samples, and are considered its basic coding unit. The size of each block can vary according to the video coding standard.

The most used video coding standards usually assume a hybrid model, which includes the following main modules: intra- and inter-frame prediction, transforms, quantization and entropy coding. A generic video encoder block diagram is presented in Figure 2.2.

Figure 2.2 – Generic video encoder block diagram.



Adapted from Agostini (2007, p. 173).

The inter-frame prediction consists of two steps: *(i)* motion estimation (ME) and *(ii)* motion compensation (MC). The ME is the most costly video encoder step in terms of computational effort, being responsible for around 80% of the total encoding time (MONTEIRO, 2012), although this costly step also leads to high compression rates (PURI, CHEN and LUTHRA, 2004). The main ME goal is to exploit temporal context between the current frame being encoded and a set of one or more reference frames that were already encoded. For each block, the ME searches for the best match (using a similarity criteria) in the reference frames, generating motion vectors (MVs) that indicate the displacement of the blocks in the current frame, based on the location of these blocks in the reference frame. The generated vectors are used by the motion compensation step to rebuild the predicted frame.

The intra-prediction aims to reduce the spatial context redundancy by analyzing the similarity between neighboring samples of a frame. For a given block being encoded, the encoder can decide which prediction mode (inter or intra) will be employed to achieve the best results in terms of coding efficiency. This decision step is performed after testing all the available possibilities, or a subset of these possibilities, from both prediction modes.

Following the prediction step, the encoder performs a subtraction between the reconstructed and the original frames to get the frame residuals. These residuals must be added to the reconstructed frame in order to become a reference frame to the encoding process of future frames. The magnitude of the residuals indicates if the prediction step was efficiently performed.

The transform step, represented by *T* in the block diagram of Figure 2.2, translates the information from the spatial to the frequency domain. Usually, the encoders use an approximation of the Discrete Cosine Transform (DCT) to transform the information. After transformed, information related to the frequencies that are not easily perceived by the human visual system is discarded (high frequency are discarded more strongly) by the quantization step (represented by *Q* in Figure 2.2). The amount of information to be discarded can be controlled by a quantization parameter (QP), where bigger values

discard more information and achieve higher compression rates. However, bigger values also generate less detailed images.

The last step of the encoding process is the entropy coding. This step is highly based on the probability of occurrence of the encoded symbols to change how they are represented (DEPRÁ, ROSA and BAMPI, 2008). The main entropy coding structure is the Context Adaptive Binary Arithmetic Coding (CABAC), which represents the syntactic elements generated along the encoding process based on three key factors: *(i)* the element to be encoded, *(ii)* the current encoding algorithm step, and *(iii)* the syntactic elements already encoded (context elements) (AGOSTINI, 2007).

Besides the above mentioned steps, it is necessary to apply the inverse transform (T^{-1} in Figure 2.2) and inverse quantization (Q^{-1} in Figure 2.2) to rebuild the current frame, which will be used as reference for future frames. In addition, the residuals must also be added in order to have the same reference frame used both in the encoding and decoding steps.

The described encoding flow is related to a generic video encoder. However, the most used video coding standards, such as the HEVC standard, follow the above mentioned steps to compress videos. Next section details the HEVC standard and its structures as it was used for all evaluations presented in this work.

2.4 The High-Efficiency Video Coding Standard

The Joint Collaborative Team on Video Coding opened the first HEVC call for proposals in January-2010. The first task of the development group was to gather all the proposals to build a CODEC model able to reduce by half the bit-rate, maintaining the same visual quality of the previous H.264/AVC standard. This model resulted in a reference software called HEVC Test Model (HM) (BOSSEN, FLYNN and SÜHRING, 2013). During more than three years, several proposals were analyzed and adopted by the development group, aiming to enhance the performance of the HEVC. In mid-2013 the HEVC reached its first standardized version, however several improvements are still being considered to enable scalable and 3D/multiview video coding, for instance. These improvements are going to be proposed as extensions of the HEVC standard final version.

The HEVC standard, like the previous H.264/AVC, follows the same encoder hybrid model presented in Figure 2.2. Several coding tools were incorporated from the previous standard. However, as the HEVC main objective is to double the compression rate and maintain the same video quality of the previous standard, these inherited tools were upgraded and a set of new coding tools were also added to the HEVC. The main tools are going to be described in the following sections.

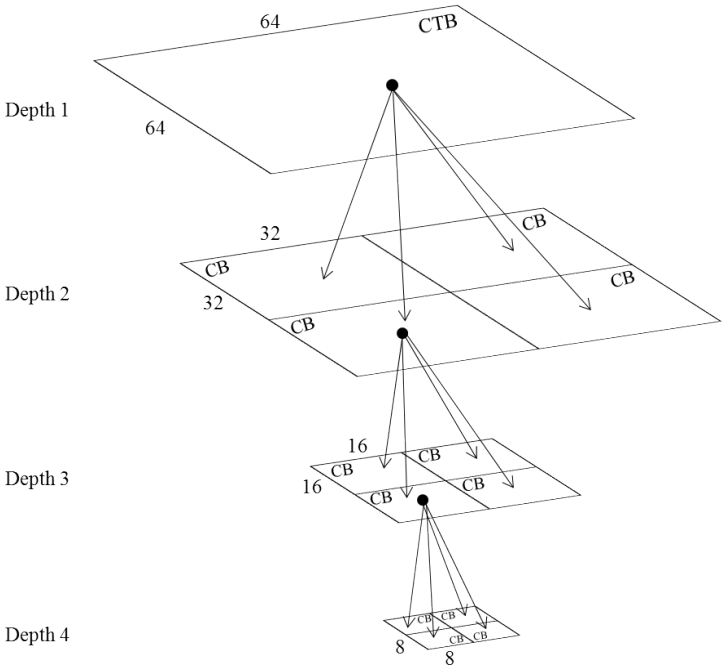
2.4.1 Basic Coding Structures

Each picture of a digital video is partitioned into Coding Tree Units (CTUs), which are considered the basic processing units of the HEVC standard. In turn, each CTU contains luma and chroma Coding Tree Blocks (CTBs). Considering the 4:2:0 subsampling format, the luma CTB covers a square picture area of *LxL* samples, while the chroma CTBs are each *L/2xL/2* samples sized. *L* can assume three different values: *16, 32,* and *64* samples. The definition of the CTB size, and hence the *L* value, may consider the picture behavior, memory, and computational requirements. On previous standards, the size of the basic processing unit was fixed at *16x16* samples. By

assuming variable CTB sizes, these structures can better adapt to the video content achieving better compression rates. In addition, supporting bigger CTB sizes is beneficial when HD and UHD video sequences are considered, since high resolution sequences are likely to present several homogeneous regions (SULLIVAN, OHM, *et al.*, 2012).

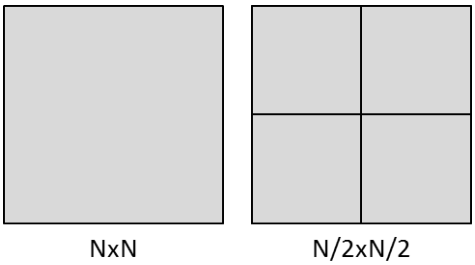
The luma and chroma CTBs can be considered Coding Blocks (CBs), although these CTBs can also be partitioned into multiple CBs via a quadratic tree (quadtree) partitioning scheme. The CBs may be recursively split, as shown in Figure 2.3, until they reach the minimum size allowed by the HEVC standard (defined as an *8x8* samples). The combination of one luma and two chroma CBs, assuming the 4:2:0 subsampling, derive a Coding Unit (CU).

Figure 2.3 – Quadtree luma CTB partitioning into luma CBs.



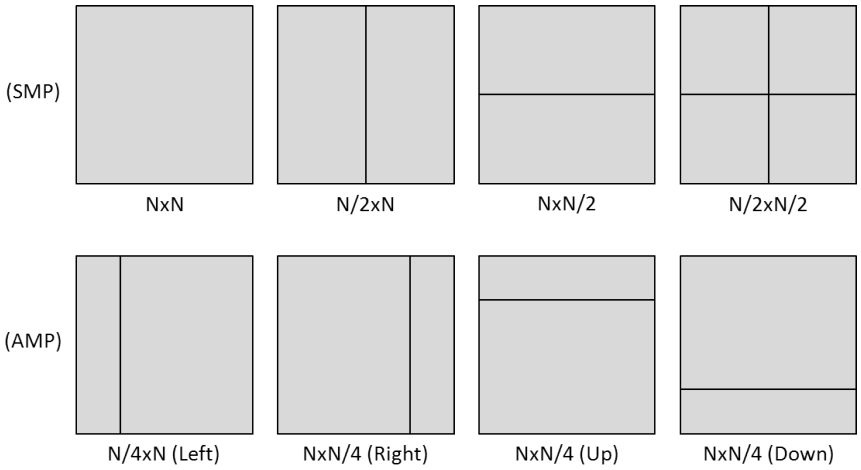
Each CU of a picture has to be signaled with a prediction mode (intra-frame of inter-frame). When the CU is signaled with the intra-frame prediction mode, it can assume two different Prediction Unit (PU) patterns, as shown in Figure 2.4. The *N/2xN/2* PU pattern is only adopted if the CU reaches the last depth of the quadtree partitioning, where the CU size is *8x8* luma pixels. In this case, the PUs occupy a *4x4* pixels area. For the remaining depths, the PU assumes the same size of the CU in which it is located (*NxN* pattern).

Figure 2.4 – Splitting PU patterns of an intra-predicted CU.



If the CU is signaled with the inter-frame prediction, it can assume the same PU patterns of the intra-frame prediction, under the same rules. However, there are six additional splitting patterns which divide the CU into two PUs. Figure 2.5 presents all the supported patterns. The patterns are divided into symmetric motion partitioning (SMP) and asymmetric motion partitioning (AMP). The AMP patterns are only allowed in luma CUs larger than *16x16* pixels to minimize memory bandwidth.

Figure 2.5 – Splitting PU patterns of an inter-predicted CU.

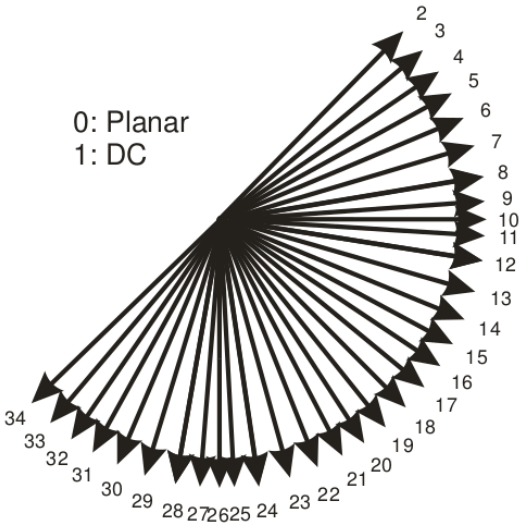


The residuals generated by the prediction steps are transformed and quantized. This residual block can be partitioned into square Transform Units (TUs). Approximations of the Discrete Cosine Transform (DCT) and the Discrete Sine Transform (DST) are applied on the generated TUs.

2.4.2 Intra-Frame Prediction

The HEVC standard performs a directional intra-frame prediction that analyzes the sample information of neighboring PUs' boundaries already encoded. There are 33 different angular orientations that can be used by this prediction step, and two other modes called DC and planar (refer to Figure 2.6). The angular intra-frame prediction modes can be applied to PUs sized from *4x4* up to *32x32* samples.

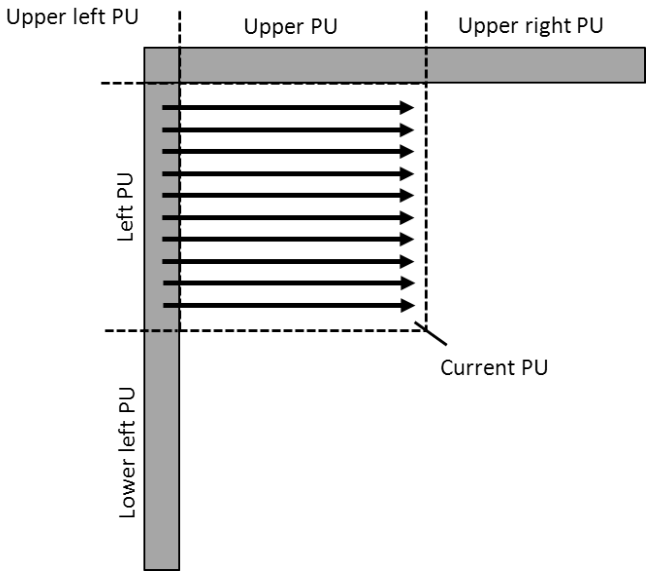
Figure 2.6 – The 35 intra-prediction modes.



Sullivan, Ohm, *et.al.* (2012, p. 1649-1668)

The horizontal and vertical angular modes represented, respectively, by numbers 10 and 26 in Figure 2.6, predict the current PU by performing a simple copy of the information from the neighboring PUs already encoded. Figure 2.7 presents a graphical example of the prediction process using the horizontal angular mode.

Figure 2.7 – Intra-frame prediction using horizontal angular mode.



The remaining intra-frame prediction angular modes calculate the weighted average (interpolation) of the information from neighboring PUs to define the components of the current prediction unit. The orientation of each angular mode defines which information from the nearby PUs will be considered in the calculation. The DC mode predicts the current PU by calculating a simple average based on its reference samples. This value is then copied to all the components of the current PU. In turn, the planar mode uses linear functions that consider horizontal and vertical samples to predict the PU information.

Considering that the intra-frame prediction process presents a wide range of possibilities to be analyzed, the only way to define the optimal prediction mode among the 35 available is testing them all. However, testing 35 modes per PU would require a big computational effort. To diminish this, several algorithms propose different techniques which are able to reduce the amount of modes to be analyzed. One of them, proposed by the HEVC developers, is the Most Probable Modes (MPM) strategy, which points the three intra MPM of the current PU based on the intra-prediction modes used by the neighboring PUs. This strategy is valid since the prediction modes tend to be similar between nearby prediction units (ZHAO, ZHANG, *et al.*, 2011).

2.4.3 Inter-Frame Prediction

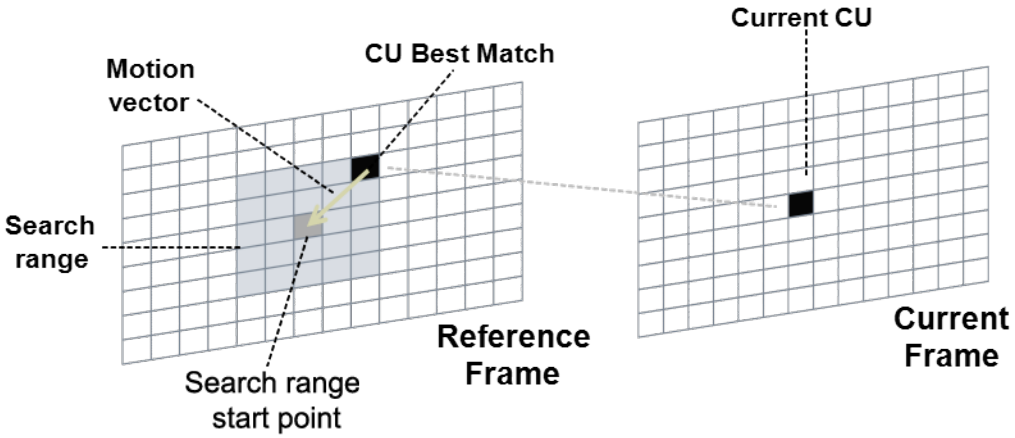
In the inter-frame prediction, the motion estimation step is responsible for predicting the frame being encoded, which is called current frame, based on comparisons with previous encoded frames (called reference frames). This can be considered one of the main modules of the video CODEC since it can lead to high bit-rate savings. However, this enhanced coding efficiency requires high computational effort (MONTEIRO, 2012). The elevated frame rate makes possible to note a high similarity between temporally neighboring frames, as shown in Figure 2.8.

Figure 2.8 – (A) and (B) show consecutive frames, while (C) shows the residual difference between (A) and (B) frames.

Richardson (2003).

In addition to the similarity between frames, the high frame rate implies that if there is any movement it tends to be small for the majority of the cases. Furthermore, the motion tends to be related to the neighboring blocks motion (ZHAO, GUO, *et al.*, 2012). Due to that, it is defined in the reference frame a search range region for the current CU being encoded, as shown in Figure 2.9. Delimiting this area aims to reduce the number of comparisons performed, and also to reduce the ME computational effort. Although, the size of the search range can be as big as the whole picture, even that it would not lead to an efficient method in terms of computational effort (MONTEIRO, 2012).

Figure 2.9 –Motion estimation process representation.



Adapted from Porto (2008).

After the search range is defined, it is necessary to find the best matching between the current PU being encoded and the candidate blocks from the reference frame which are inside the delimited area. There are several algorithms to perform the search; the exhaustive full search travels through all the CUs from the search range area. Besides bringing the best matching among all blocks from the search range, this algorithm is unfeasible due to its enormous computational cost that drastically increases with the search range size (MONTEIRO, 2012). The HEVC standard considers the fast Test Zonal Search (TZ Search) algorithm, which turns the search process faster as neither all the CUs from the search range are compared (URBAN, DÉFORGES and NEZAN, 2012). Following the comparison, after the best match is found, it is generated a motion vector (MV) that points the position of the matched block in the reference frame. As mentioned in Section 2.4.1, a single CU can be split into multiple PUs. In this case a complete search is performed and one associated MV is found for each PU.

Analyzing the behavior of the MVs generated by the ME process, the HEVC developers perceived a huge correlation between neighboring motion vectors. Willing to exploit this correlation, it is performed an Advanced Motion Vector Prediction (AMVP) step, which makes use of spatial and temporal correlation of neighboring partitions to predict the MV of the current block (ZHAO, GUO, *et al.*, 2012). The AMVP generates a set of candidates used to differentially encode the actual MV, i.e., the best candidate block is encoded in relation to the AMVP predicted vector.

The motion compensation process complements the ME process, since it uses the motion vectors produced by the motion estimation to generate the predicted frame. To perform this, the MC accesses the reference frames that are present in the memory. Then, it locates the best matches indicated by the MVs to build the predicted frame. Finally, the MC subtracts the predicted frame from the original one to generate a residual frame that will serve as input for the transform module.

2.4.4 Direct and Inverse Transform

Present only on the encoder side, the direct transform is responsible to transform the information from the spatial to the frequency domain. To perform this, the HEVC standard uses the Discrete Cosine Transform (DCT). The DCT disassociates the samples and reorganizes the coefficients without discarding any information (BAYER

and CINTRA, 2010). However, the coefficients generated by the DCT are used by the quantization step, where some are discarded to reduce the amount of data.

There are several versions of the DCT, however as digital videos signal propagates in two dimensions, the two-dimensional version of the DCT is considered in the HEVC standard. This version requires a high computational effort, although it could be achieved the same result by applying the one-dimensional DCT two times: one horizontally and one vertically. Additionally, the HEVC applies the Discrete Sine Transform (DST) in *4x4* intra-predicted TUs. This transform is considered only in this particular case since it would not lead to substantial bit-rate savings in bigger TUs (SULLIVAN, OHM, *et al.*, 2012).

In contrast to the direct module, the inverse transform is present both in the encoder and the decoder. On the decoder side, the inverse transform is required to reconstruct and present the frames properly. While on the encoder side this module guarantees that the reconstructed frames will be used as reference to the frames to be encoded.

2.4.5 Direct and Inverse Quantization

The quantization module uses the disassociated samples generated by the transform step to eliminate the redundant content. The amount of information to be discarded by the quantization is controlled by the quantization parameter (QP). There are a total of 52 QP levels. The larger the QP, the greater will be the impact on visual quality of the encoded video, and the greater will be the amount of information discarded by the quantization step.

However, applying a constant QP level during all the encoding process would not be efficient since the picture content varies along time. Taking into consideration a less detailed picture area, it could be applied a higher QP as more information can be discarded, whereas in regions that present more details, the quantization parameter should be smaller to preserve this information. For that reason, the HEVC standard adopts the Rate Distortion Optimized Quantization (RDOQ) (GWEON and LEE, 2012), where four candidates using different QP levels are generated to select the most suitable one.

The inverse quantization step is present both in the encoder and the decoder sides for the same reason of the inverse transform.

2.4.6 Entropy Coding

The HEVC standard adopts the Context Adaptive Binary Arithmetic Coding (CABAC) as its main entropy coding tool. This step is responsible to achieve high bit-rate savings in the video encoding process (SZE and BUDAGAVI, 2008). To achieve that, the CABAC has to select the probability models of each syntactic element based on its context. This step is refined along encoding time as new elements are collected.

Choosing the context of each element directly impacts on the coding efficiency. For this reason, the CABAC algorithm used in the HEVC standard considers the context information of neighboring elements, and also the quadtree depth information. This can be considered an upgrade of the previous CABAC algorithm, since it can now achieve better compression rates with lower computational effort when compared to its previous version (SULLIVAN, OHM, *et al.*, 2012).

2.4.7 Filters

Following the reconstruction of the frames, two filters are applied: the Deblocking Filter (DBF) and the Sample Adaptive Offset (SAO). Considering that the video coding process is performed on a block manner, some boundary artifacts can be noticed on the encoded frame. These artifacts compromise the visual quality of the image, and the DBF goal is to eliminate the blocking effects. To achieve that, the DBF is applied on the samples located on the TUs and PUs' boundaries.

Before storing the reconstructed frames, the SAO filter is applied. The SAO considers the local gradient of a given sample and uses a table with pre-defined values to apply an adaptive offset. This filter is very efficient to diminish undesirable quantization effects, contributing to enhance the visual quality of the encoded video.

2.4.8 Video Coding Configurations

One remarkable characteristic of the HEVC is its highly customizable profile. The JCT-VC group adopted structures that can be parameterized, becoming easier to adapt the standard to achieve good performance based on the target application. However, the development group has provided some standard configurations that facilitate the comparison between the video coding tools. The coding configurations defined are: All Intra (AI), Low Delay, which is divided into Low Delay P (LP) and Low Delay B (LB), and Random Access (RA) (BOSSEN, 2012).

2.4.8.1 All Intra

The All Intra configuration treats all the frames of a video sequence as intra-predicted frames, hence no temporal redundancy is exploited. Since the inter-frame prediction is not considered, and so the motion estimation/compensation, the compression efficiency of the AI configuration do not lead to high bit-rate savings. However, not applying MC and ME makes the encoding process faster, as these coding tools are computationally costly (SINANGIL, CHANDRAKASAN, *et al.*, 2012). Another advantage of the All Intra configuration is its error resilience, since the errors are not propagated along time as happens in inter-frame prediction (PIÑOL, TORRES, *et al.*, 2013).

This configuration is mainly suitable for applications that need to treat the frames independently, such as static picture encoding (NGUYEN and MARPE, 2012). Also, applications that are concerned about encoding time, but that are not worried about bandwidth, can benefit from the AI coding configuration.

2.4.8.2 Low Delay

The Low Delay achieves huge bit-rate savings when compared to the AI coding configuration. This happens since, in addition to the intra-frame prediction performed by the AI configuration, the Low Delay also considers the inter-frame prediction (LI, SULLIVAN and XU, 2012). The temporal prediction only considers previous frames, based on the display order, to be used as reference frames. Thus, in the Low Delay configuration every frame is encoded and transmitted according to the video display order. For this reason, after decoding, the frame can be immediately presented.

There are two types of Low Delay configurations: Low Delay P and Low Delay B. These approaches treat the first picture as an intra-predicted frame, however for the following pictures the LP uses only P (uniprediction) frames, therefore it considers only one previous reference frame (note: a single reference is used to build the prediction,

however, multiple frames may be used to search for the best matching) to perform the inter-prediction. While the LB configuration adopts B (biprediction) frames, so it can use up to two previous reference frames for the inter-prediction.

2.4.8.3 Random Access

The Random Access coding configuration does not follow the display order to encode the frames. It defines an Intra Period that lasts approximately one second, and starts with an intra-predicted frame. The remaining frames are B frames, which can use as reference previous and/or future frames according to the video display order. After one period finishes, one more Intra Period starts with another intra-predicted frame that is used as a Clean Decoded Refresh (CDR) frame to diminish the error propagation impact (SULLIVAN, OHM, *et al.*, 2012).

This configuration incurs in additional delay both on the encoder and decoder sides. The encoding delay exists as future frames might be used as reference to encode the current frame, while the decoding delay occurs as the frames are not received in the display order. For this reason, the RA configuration may be adopted by applications that can tolerate delay, such as pre-recorded videos for storage or broadcasting (PIÑOL, TORRES, *et al.*, 2013).

2.4.9 Parallel Encoding Tools

In contrast to the development of the previous H.264/AVC standard, where the parallelism support was an after-thought, some parallelism tools were incorporated in the HEVC standard. Parallelism support by video CODECs is essential nowadays, since they require high computational effort as the video resolution is constantly increasing (VANNE, VIITANEN, *et al.*, 2012). Additionally, the dissemination of parallel architectures on consumer multimedia device market encourages the development of solutions that exploit parallel processing.

Thus, to address the high computational effort required by the HEVC standard, and to make use of the recent parallel architectures, the JCT-VC group proposed novel parallelism-oriented tools. Using these tools, it is possible to exploit coarse and fine grain parallelism during the encoding and decoding process. The tools that enable parallel processing in the HEVC standard will be better described on Section 3.

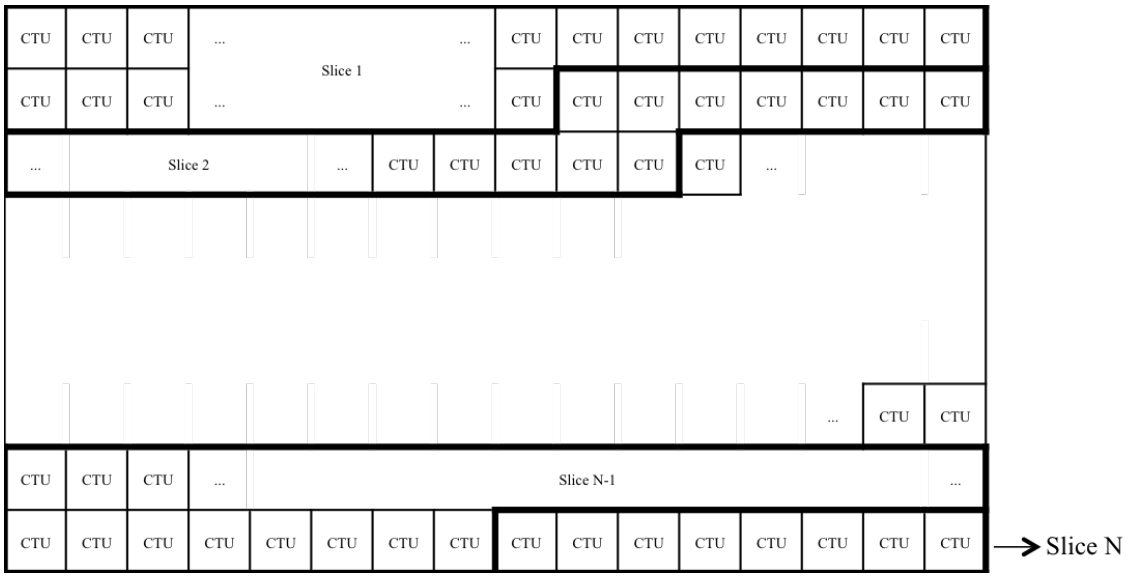
3 HEVC PARALLELISM-ORIENTED TOOLS

Current multimedia applications involve a huge amount of data, including video CODECs that nowadays are responsible to deal with beyond-HD video resolutions. For this reason, consumer electronic devices such as tablets, smartphones and computers, are adopting even more parallel-enabled architectures to deal with the high computational effort required by such applications (NVIDIA, 2010). To support parallel processing in the video encoding/decoding process, the latest standards proposed three tools: Slices, Tiles and Wavefront Parallel Processing (WPP).

3.1 Slices

Slices are data structures that were already adopted by the previous H.264/AVC standard, and are also present in the HEVC standard. Its main objectives are resynchronization in the case of data loss and packetized transmission (FULDSETH, HOROWITZ, *et al.*, 2012). A slice is a self-contained structure that is able to partition a picture in a sequence of CTUs, which are processed in raster scan order. Each picture of a video sequence can be partitioned into one or more slices, as represented in Figure 3.1.

Figure 3.1 – Example of a picture partitioning into slices.



There are three possible slice types: *(i)* I slice, *(ii)* P slice, and *(iii)* B slice. When the I slice (also called *intra slice*) is considered, all the CTUs inside this slice use only the intra-frame prediction. While in P slices, both intra- and inter-frame prediction can be used, however it considers only one reference to perform the inter-prediction. Finally, B

slices can use both prediction types like P slices, although they consider two references to perform the inter-prediction.

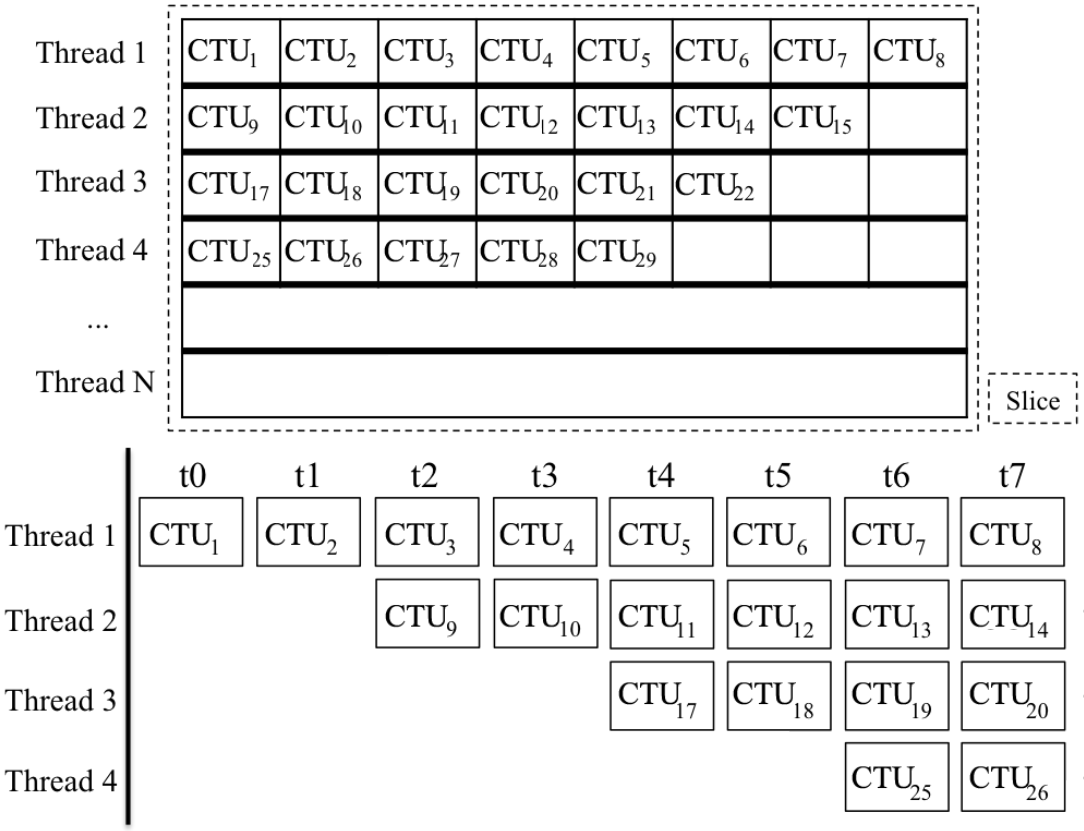
In the HEVC standard, the slices were upgraded in relation to the H.264/AVC and can be composed by slice segments. These segments can be considered *independent* or *dependent.* Independent segments do not need previous encoded segments to perform its encoding process. While in dependent slice segments, the value of each segment header depends on the coding result of previous slice segments. This flexibility is used to enhance the parallelism potential of the novel parallelism-oriented tools proposed in HEVC, as the coding latency can be reduced by using the independent segment structure (SULLIVAN, OHM, *et al.*, 2012).

As mentioned, slices can be considered self-contained, for this reason they can also be considered independent. In this case, each CTU prediction step is only able to access information that is located inside the slice in which it is placed. Hence, it is not possible to cross slice boundaries. This independence makes viable to exploit parallelism by the use of slices, where each slice can be processed in parallel. However, the definition of a slice incurs in the insertion of additional headers in the bit-stream. Therewith, while defining more slices would increase the parallelism potential, it would also impact even more the coding efficiency as more contexts will be broken by slice boundaries and more headers are going to be inserted in the final bit-stream (FULDSETH, HOROWITZ, *et al.*, 2012). To diminish the impact of using parallel approaches in the video encoding/decoding process, the HEVC developers proposed novel-parallelism tools.

3.2 Wavefront Parallel Processing

The Wavefront Parallel Processing (WPP), when enabled, split a slice into CTU rows that can be processed in parallel. Figure 3.2 shows a slice (entire picture) with WPP enabled, where a different thread processes each CTU row. The first row is processed normally, in raster scan order. The second, and the rows after the second, have to wait until two CTUs from the previous row are processed to begin processing one CTU. The WPP have to wait for two CTUs above to be processed, because intra-frame and motion vector prediction depend on the information of the directly above and the above right CTU. Additionally, the entropy coding also uses the context from these previous encoded CTUs to benefit from the maximum context possible on the next thread (POURAZAD, DOUTRE and NASIOPOULOS, 2012).

Figure 3.2 – CTU processing of a WPP enabled slice and the CTU timeline wavefront processing.

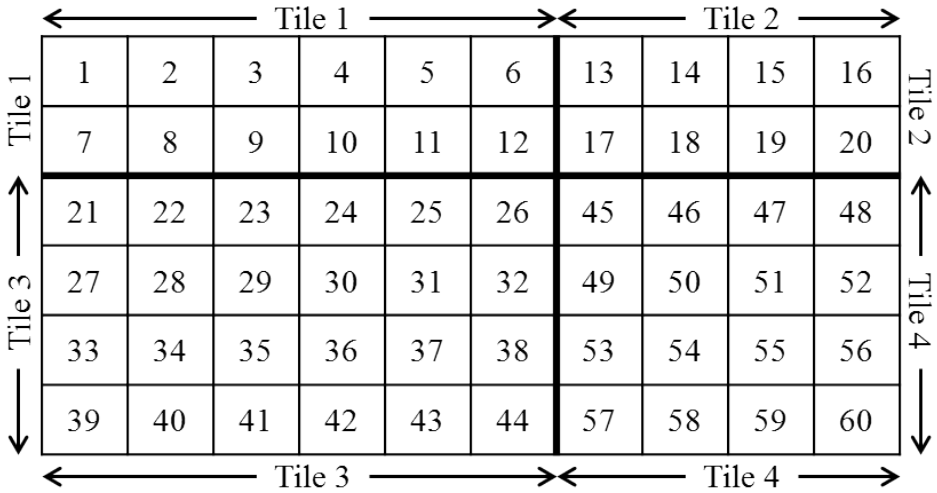


Since the WPP rows present certain dependencies, the parallelism is achieved with a two-CTU delay, as can be noticed in the timeline parallel processing of Figure 3.2. However, as the WPP rows are not considered independent since no coding dependencies are broken, they can lead to better visual quality than slices (BORDES, CLARE, *et al.*, 2012).

3.3 Tiles

Tiles are another novel parallelism-oriented tools proposed in the HEVC standard. They consist of rectangular partitions that are created by defining horizontal and vertical boundaries across a picture (CHI, MESA, *et al.*, 2012). These boundaries must coincide with the CTU boundaries, as can be seen in Figure 3.3. Tiles can be either uniform or non-uniform spaced. In uniform tiles, the encoder evenly distributes the number of CTUs inside the partitions, whereas in non-uniform tiles the user can freely set the number of CTUs per tile. Although, the definition of multiple tile partitions per frame has a size restriction, where the minimum size defined is *256x64* samples per tile (BROSS, HAN, *et al.*, 2013). For instance, if it is considered the maximum size that a CTU can assume (*64x64* samples), the minimum tile size would be four CTUs wide per one CTU high.

Figure 3.3 – CTU scanning order of a picture split into four tiles.



Another peculiarity of tile partitioning is that it changes the CTUs raster scanning order to a tile-constrained raster scanning order, which is represented in Figure 3.3, where each CTU number represents the order of scan. First, the CTUs within Tile 1 numerated from 1 to 9 are raster scanned. After that, the CTUs within Tile 2 are also raster scanned. This process is repeated tile-by-tile until the whole picture is scanned. This description matches a sequential scanning, but all tiles could be processed in parallel since there is no data dependency between each other.

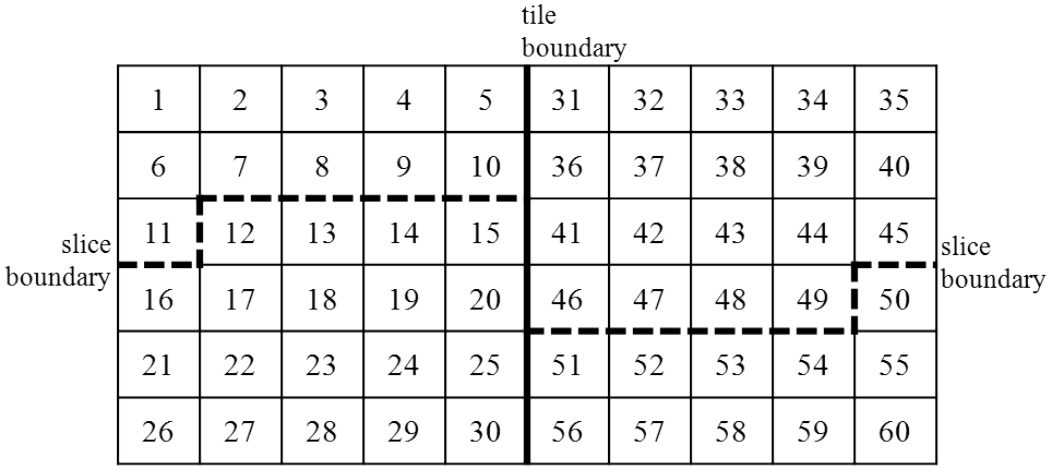
Just like slices, tiles are considered independent since all coding and prediction dependencies such as motion vectors, intra-frame neighboring samples, entropy coding context, are broken through its boundaries (MISRA, SEGALL, *et al.*, 2013), reseting all coding tools that use this context information to perform the data compression. The only tools allowed across tile boundaries are the filters, which are used to reduce the blocking effect and are described in Section 2.4.7. A bit-stream flag, named *LFCrossTileBoundaryFlag*, enables or disables the application of these filters. When this flag is reset, the data communication between neighboring tiles is null. Otherwise, if the flag is set, it reflects in inter-tile communication. Considering parallel processing, enabling inter-tile communication may compromise the parallelism potential, however it would also lead to better video quality. In contrast, if *LFCrossTileBoundaryFlag* is not enabled the video quality will be reduced once blocking effect at tile boundaries will remain unchanged, though the parallelism potential will increase as no inter-tile communication will exist (BROSS, HAN, *et al.*, 2013).

As all tiles in a frame can be processed in parallel there is no guarantee that the tiles are processed in order. For this reason the encoder and decoder must know where each tile is located into the frame/bit-stream to be able to reconstruct the complete picture. One way to do that is by inserting markers in the bit-stream that indicate the tiles entry points. The other way is sending the offsets in the slice header containing the tile (FULDSETH, HOROWITZ, *et al.*, 2012).

It was mentioned before that the WPP rows are defined inside a slice. So as WPP and slices, tiles and slices can also coexist in the same picture. To achieve that, each CTU of a slice must belong to one unique tile, or each CTU of a tile must belong to one unique slice (SULLIVAN, OHM, *et al.*, 2012). Figure 3.4 illustrates the first condition,

where there are two slices inside the first tile and another two slices inside the second tile.

Figure 3.4 – Picture partitioning with slices inside tiles.



Although slices and tiles can be together defined in a video frame, for design simplicity the HEVC standard does not support the coexistence of WPP and tiles (SULLIVAN, OHM, *et al.*, 2012). However, these tools could be combined to exploit both fine grain (WPP) and coarse grain (tiles) parallelism.

3.4 Comparison of Parallel Tools

The slices were the first structure considered to exploit parallel processing in the video coding process. Besides being used to this end, slices were originally proposed to deal with transmission errors. Usually the size of the slices is defined based on the maximum transmission unit (MTU) size of the network being considered (MISRA, SEGALL, *et al.*, 2013). Since the slices usually assume raster scanning order shapes, and their size is mainly based on the network characteristics, often the slice boundaries split the picture where there is increased context correlation. Furthermore, each defined slice carries an extra header that introduces a non-negligible overhead into the final bit-stream (MISRA, SEGALL, *et al.*, 2013). In conclusion, taking into consideration the slice header overhead and the fact that no coding context is examined to define the slice partitions, using slices to exploit parallel processing is not recommended as it highly compromises the achievable coding efficiency. To suppress the need of efficient parallelization tools, the HEVC developers adopted the WPP and Tiles (CHI, MESA, *et al.*, 2012) (MISRA, SEGALL, *et al.*, 2013) (FULDSETH, HOROWITZ, *et al.*, 2011).

Tile partitions do not present the same header issue that compromises the coding efficiency when using slices (MISRA, SEGALL, *et al.*, 2013). However, as well as slices, tile partitions break the context across its boundaries, being able to be independently processed. This leads to an increased parallelism potential when compared to the WPP approach (CHI, MESA, *et al.*, 2012) (ZHOU, SZE and BUDAGAVI, 2012), which presents a certain delay due to the coding dependencies from previous rows. In addition, the thread dependency can lead to memory coherence issues that can compromise the overall coding efficiency. Yet, by not breaking the coding dependencies across its boundaries the WPP achieves better coding efficiency results than tile partitions (SULLIVAN, OHM, *et al.*, 2012).

Nevertheless, tile partitions adopt rectangular regions that makes easier to employ regions of interest (ROI) video coding (MISRA, SEGALL, *et al.*, 2013). Hence, assuming that the size and location of non-uniform tile partitions can be freely set, it is possible to analyze the picture and the context information to define the tiling pattern on a suitable way. Trying to encapsulate highly correlated picture regions inside the same tile would reduce the impact on coding efficiency caused by the tile boundaries context break (CHI, MESA, *et al.*, 2012). Taking into consideration the highlighted points, it is presented in Table 3.1 a summarized comparison between the three tools. The positive characteristics of each tool are bolded.

Table 3.1 – Comparison between coding tools.

	Slices	WPP	Tiles
Custom Partitions	No	No	**Yes**
Header Overhead	Yes	**No**	**No**
Context Breaks	Yes	**No**	Yes
Parallelism Delay	**No**	Yes	**No**
Thread Dependency	**No**	Yes	**No**

The tile is the only parallelism tool that can have its size and location freely set, and this can be considered a positive characteristic to exploit the video frame content by defining custom partitions. As already mentioned, the slices are the only tools that add extra headers on the final bit-stream, compromising the achievable coding efficiency. Another key aspect to be considered on the comparison is the context breaks, which can also compromise the coding efficiency. The WPP is the only tool that does not break the context. However, by not breaking the context, the parallelism using WPP structures is achieved by the cost of a two-CTU delay, compromising the parallelism potential. In addition to that, the thread dependency required by the WPP can lead to some issues regarding accessing data in memory. Taking this into consideration, it can be noticed that the tile is the tool that presents most positive points between the compared tools.

Thus, in this work it is proposed to study and reduce the undesirable effects of using tiles as a tool to provide parallelism in the video coding process. To better understand the coding effects posed by the use of this novel parallelism-oriented tool, we present in Section 3.5, an extensive evaluation on the impacts related to the use of tiles.

3.5 Fixed Tile Partitioning Evaluation

As mentioned before, the HEVC enables the definition of uniform and non-uniform spaced tiles. In uniform tiles, the encoder equally divides the number of CTUs among the total number of tile partitions. While in non-uniform spaced tiles it is possible to define the number of tiles along rows and columns, and also their sizes based on the number of CTUs per tile. In the HEVC Test Model (HM) reference software (BOSSEN, FLYNN and SÜHRING, 2013), after defined, the same tile partitioning pattern will be employed for all next frames to be encoded. This fixed tiling pattern facilitates the marking of the tiles entry points in the bit stream, as the location of tile boundaries in the frames will always be the same.

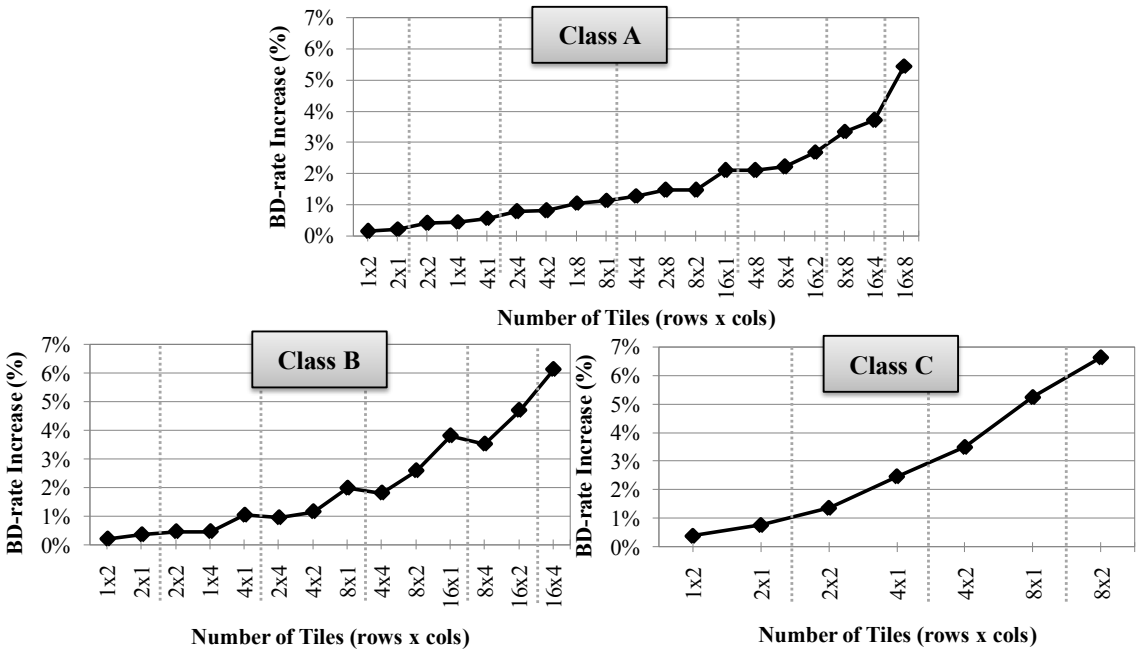
However, using fixed tiling patterns can bring load balance and coding efficiency issues. It occurs, because even if load balanced tiles are defined for the first frame, considering the mutable profile of digital videos, the computational effort of the defined tiles can also change. Hence, the tiling pattern can become unbalanced during encoding time. The same can happen in terms of coding efficiency, since efficient tiles that consider the correlations of a picture can turn to split undesirable regions if the tiling pattern does not adapt with the video frames.

To better understand the impact caused by the use of fixed tiling patterns during encoding time, an analysis of the coding efficiency losses caused by this approach is presented in Figure 3.5. The coding efficiency is measured in terms of the Bjontegaard Delta Bit-rate (BD-rate), which is a metric that represents the bit-rate percentage difference between two videos for the same video quality (BJONTEGAARD, 2001). In the evaluation video sequences encoded using fixed uniform tile partitioning in relation to the same sequence with only one tile per picture are compared, i.e., with no dependency breaks inside the video frame. The reference video sequences considered in the experiments are presented in Table 3.2. It is considered two benchmark video sequences from each of the first three video classes defined by the Common Test Conditions (BOSSEN, 2012). The experiments are performed using the HM reference software. The encoding follows the Random Access coding configuration. The number of tiles per picture varies in power of two, ranging from 2 to 128 tiles. However, the total number of tiles per picture is limited by the video resolution.

Table 3.2 – Considered benchmark video sequences.

Video Classes (Resolution)	**Video Sequence**
Class A (2560x1600)	*Traffic – 30 fps* *Nebuta – 60 fps*
Class B (1920x1080)	*ParkScene – 24 fps* *BQTerrace – 60 fps*
Class C (832x480)	*RaceHorses_C – 30 fps* *BQMall – 60 fps*

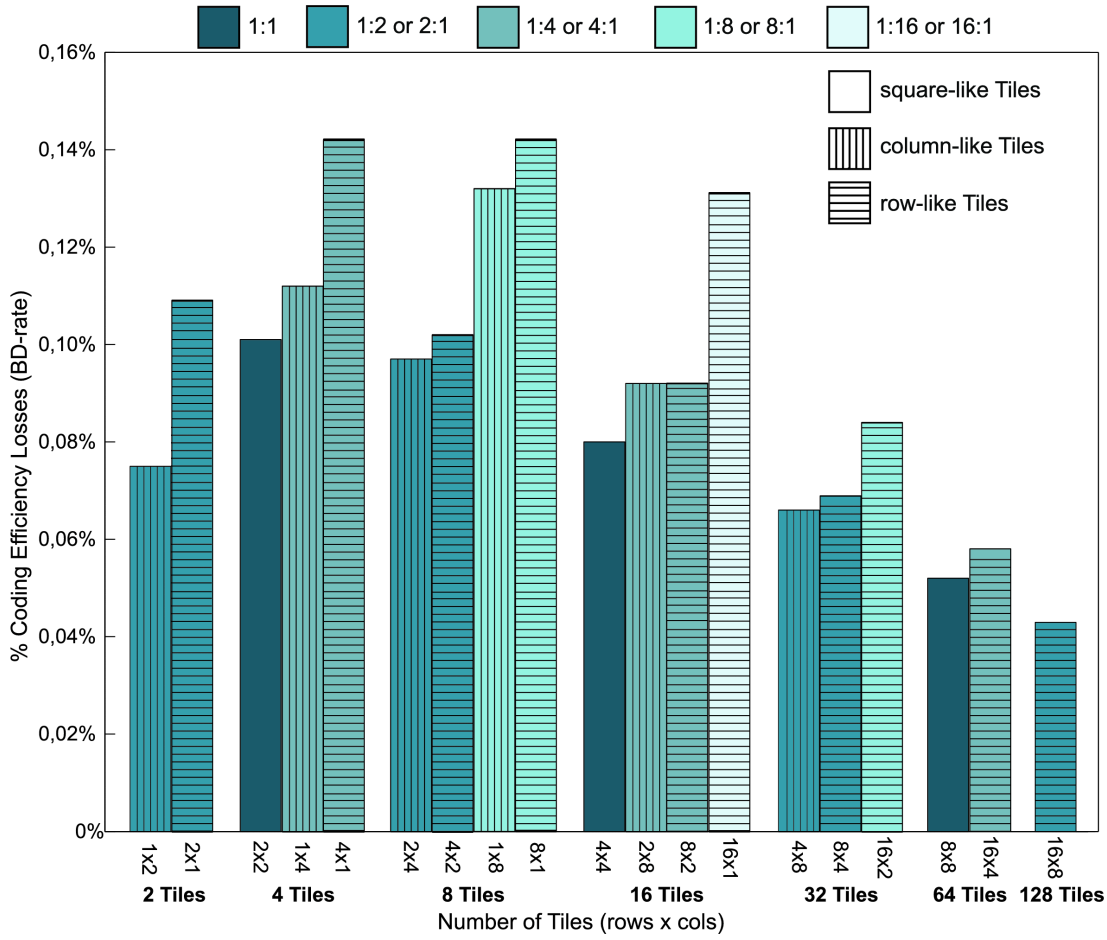
Figure 3.5 – BD-rate increase caused by various fixed uniform tiling patterns for Classes A, B, and C video sequences.



For all the video classes analyzed, the impact on coding efficiency increases as the number of tiles also increases. This happens since the more tiles are defined, the more contexts are broken. These context breaks, as mentioned before, reflect in coding losses (BD-rate increase). The BD-rate increase proved to be higher in lower resolution videos. While the top percentage BD-rate increase (worst case) in Class A videos reached 5.45% when 128 tiles (*16x8* tiles) where used, the worst case in Class B videos reached 6.12% with 64 tiles (*16x4* tile pattern). For Class C video sequences the impact was even higher, reaching 6.63% by dividing the frames with 16 tiles (*8x2* tiles). The high impact of splitting lower resolution videos with tiles is justified by the fact that these videos have smaller correlated areas, and then it is more difficult to cluster the highly correlated picture regions that sometimes are split by tile boundaries.

Aiming to analyze the related coding losses regarding tile shapes, it is presented in Figure 3.6 the percentage BD-rate increase per number of tiles. This analysis considers the tile patterns defined for Class A videos, since they group more tiling possibilities than the other video classes. Each of the seven bar groups in Figure 3.6 designates the total tile number per frame. The bars contained in the groups represent tiling patterns (relation between number of tiles in rows and columns) that can assume five different tile perspective groups: 1:1, 1:2 or 2:1, 1:4 or 4:1, 1:8 or 8:1, and 16:1 or 16:1. For example, a square tiling always assume the 1:1 perspective, since the number of tiles along rows and columns are the same. When the *8x4* tile pattern is considered, its related perspective is 2:1 since the number of tiles along the rows is twice the number of tiles along the columns. The orientation of the tile partitions (form factor) is also represented in Figure 3.6, where bars with horizontal, vertical and no texture represent respectively row-, column- and square-like tiles.

Figure 3.6 – Coding efficiency impact for different tiling configurations in Class A videos.



The square-like tile shapes, i.e., *2x2, 4x4,* and *8x8,* compromised less the coding efficiency when compared to row- and column-like shapes with the same number of tiles. This conclusion comes from the area/perimeter relation. The perimeter represents the tile boundaries where the dependencies are broken, and the area represents the number of CTUs within a tile. Therefore, the fewer dependencies are broken to encode a given number of CTUs, the better the rate-distortion performance, on average. In other words, the bigger the area/perimeter relation, the better the coding efficiency tends to be – it also depends of the video content. Square shapes represent the maximum area for a given perimeter (CHI, MESA, *et al.*, 2012).

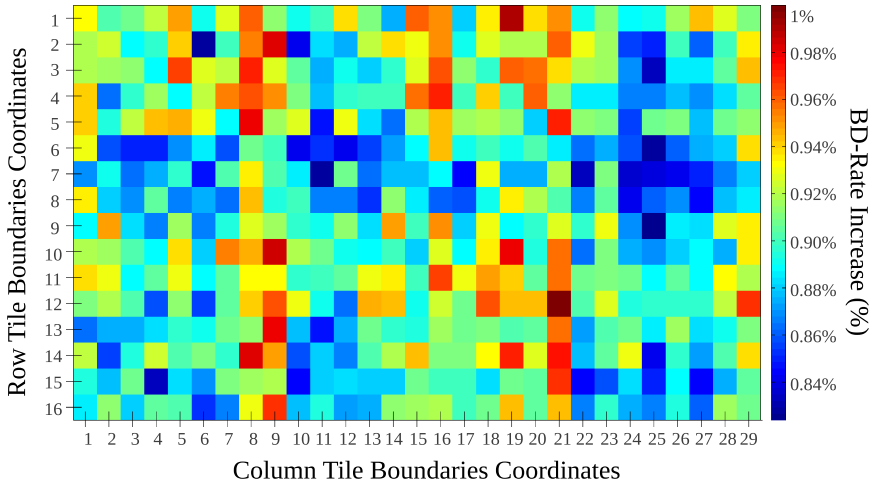
Even that tile partitions incur in coding efficiency losses, the impact caused by this novel-parallelism tools can be diminished if suitable tile shapes are chosen. Moreover, if the location of the tiles is also taken into consideration, it is possible to better exploit the picture context to attenuate the impact caused by tiles. These approaches, associated with an adaptive tile partitioning can enhance the coding efficiency achieved by tiles.

3.6 Challenges on Video Coding Parallelization

Even though novel parallelism-oriented tools were integrated to the HEVC standard, the parallelism support by the video coding process still arises several challenges. Besides being able to exploit parallel video coding by splitting the picture into independent partitions, as already mentioned, tiles inherently lead to coding efficiency

drawbacks. However, initial analysis shown that these losses could be reduced if suitable tiling patterns, i.e., square-like tiles are adopted. There are other techniques that can be exploited to further reduce the tile coding impact, such as placing tile boundaries where there is a natural context break in the picture. Hence, the dependency breaks caused by tile boundaries would not lead to elevated coding losses. To locate these suitable picture regions, an exhaustive analysis was performed. It is shown in Figure 3.7 a BD-rate map extracted after encoding the 1080p benchmark video sequence *BQTerrace* (BOSSEN, 2012) with every possible tile division using four tiles (*2x2* tile partitions) per picture. When considering four tiles, there is only one intersection point between the horizontal and vertical tile boundaries, the axis labels represent the coordinates of this intersection point in terms of CTUs. Each square presented in Figure 3.7 represents the BD-rate percentage increase when the vertical/horizontal intersection is located at that specific point – against the encoding process with one single tile. For instance, the square located at position [8,5] – a red square in this case – represents the BD-rate increase when the vertical tile boundary was placed at the 8th column of CTUs and the horizontal tile boundary was placed at the 5th row of CTUs.

Figure 3.7 – BD-rate increase of each possible *2x2* tile partitioning.



There are several points in the BD-rate map, represented by the dark blue squares that do not generate meaningful coding losses (BD-rate percentage increase). Besides, many of these points present less coding losses than the uniform-spaced tile partitioning (intersection [15,8] in Figure 3.7). These lower coding losses are achieved since the partitioning patterns are able to split the picture where there is a natural context break, hence clustering the highly correlated picture regions inside the same tile. It is important to state that this map considers one frame only. If the same map is extracted considering the next frame it would change the location of the regions where the coding efficiency impact is reduced. This happens since the natural video movement can displace the highly correlated regions between consecutive frames. Hence, arising the first challenge: *locate, for each frame, the intersection points that are able to group the correlated regions and split regions with different properties to enhance tiling coding efficiency.*

An additional detail that must be considered when using tiles to process in parallel is the load balancing. Since tiles are independent, it is desired to process every partition concurrently. The first and natural approach to achieve load balanced tiles would be to define uniform-spaced partitions. However, equally sized tiles do not guarantee that the

same encoding computational effort would be spent for all tiles. Since one tile can be composed by only homogeneous content, while another can be composed of a highly detailed content. This arises a second challenge: *increase the parallelism potential of using tiles on the video coding process.*

Taking into consideration the two challenges above mentioned the main objective of this work is to *develop solutions that are able to adapt the tiling pattern along encoding time, aiming to reduce the coding efficiency losses caused by the use of tiles*. It is also proposed *techniques able to enhance the parallelism potential of the adaptive tiles*, since this adaptive approach could lead to unbalanced workload.

4 ADAPTIVE TILE PARTITIONING SOLUTIONS

As already mentioned, encoding tiles in parallel is only possible because they are independent to each other. To be independent, tile boundaries break the coding dependencies across its limits, leading to a certain impact on coding efficiency. However, there are several regions in a video frame where the tile boundaries reflect in less coding efficiency impact. Applying a brute force technique to locate these suitable tiling regions, such as testing all the possibilities for a certain tiling pattern as performed in Figure 3.7, would make real-time encoding unviable due to the additional computational effort. Yet, the definition of efficient tiles can be supported by the use of image properties and encoding information. Based on this, it is possible to derive the highly correlated picture regions as the CTUs located on these regions are likely to share similar encoding information. Hence, defining the tile boundaries around the highly correlated CTU regions (matching the natural encoding context break) diminishes the impact of the context break caused by tile boundaries in the encoding and prediction structures.

Analyzing image properties gives a good hint on the location of the highly spatial-correlated regions. Another element to be considered is the encoding information, as it also points the picture regions where there is highly correlation of encoding structures. These are key factors to be analyzed since the CODECs consider it to support the overall encoding process. This work exploits several elements to locate the highly correlated picture regions that, if clustered inside the same tile partition, lead to better coding efficiency results. These elements are described on the following section.

4.1 Partitioning Maps

The image properties and the coding information that will support the location of highly correlated picture regions, and hence the definition of more coding efficient tiles, are called in this work *partitioning maps*. It is studied four partitioning maps that consider different image properties and encoding information with potential to be used in a tile partitioning algorithm: variance map, bit map, CTU's partitions map, and differential motion vectors (DMV) map. In this work it is considered the CTU granularity to generate all maps, since the tile boundaries must coincide with the CTU boundaries. The maximum CTU size (*64x64* pixels) is adopted for all the maps.

4.1.1 Variance Map

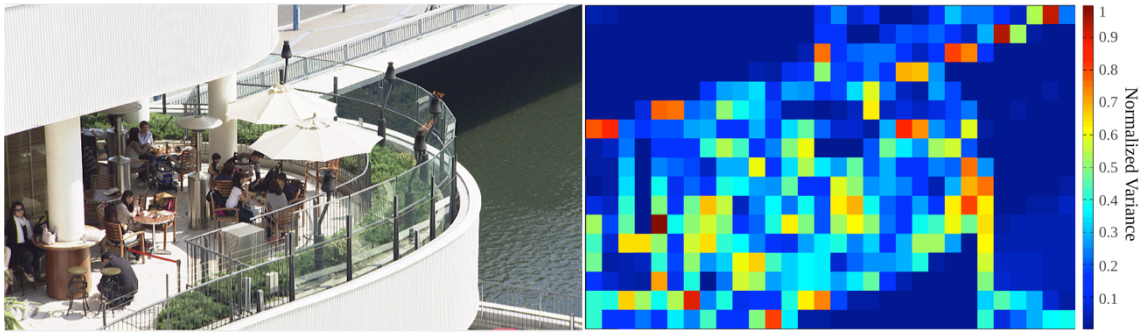
Variance is a measure of variability between the components of a data set, which calculates the distance of each element from the mean. The variance formula is defined in Eq. (4.1), where a subtraction between each element of the data set and the average set value is performed. The differences acquired are squared and, finally, the number of

elements in the data set divides the sum of the squares to achieve the variance of the set components (SHARMA, 2005).

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \qquad (4.1)$$

To acquire the variance partitioning map from a video frame, the variance of each CTU is extracted from the set of 4096 luminance elements (*64x64* pixels). By doing this, it is possible to identify both the homogeneous CTUs, presenting low variance, and the high textured CTUs, which present high variance (ZATT, 2012). In Figure 4.1 it is presented the variance map extracted from *BQTerrace* video sequence frame.

Figure 4.1 – *BQTerrace* frame and its variance map.

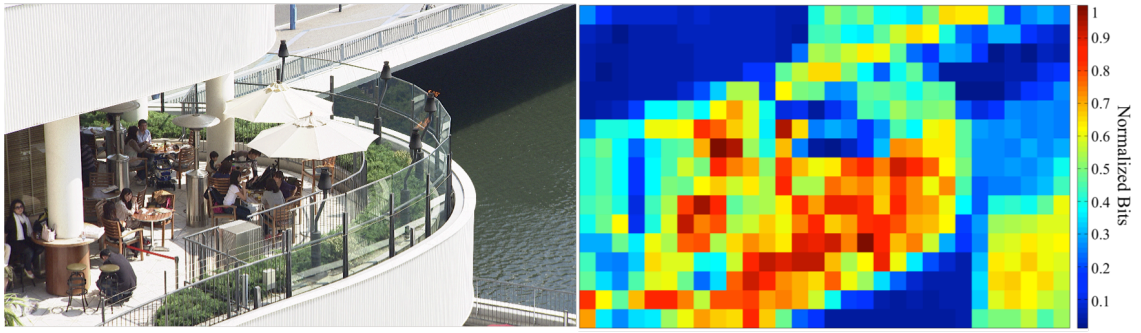


It can be noticed that the homogeneous CTUs are represented by squares with low variance (dark blue squares) in the variance map. These homogeneous regions share similar spatial encoding information, and hence are considered highly correlated with respect to the encoding process. Clustering these regions into the same tile avoid the non-negligible coding efficiency impact imposed by the spatial context breaks caused by tile boundaries, enhancing the coding efficiency of using these structures. It is important to point that the variance map is extracted from the raw picture, consisting of a pre-processing (before encoding) partitioning map.

4.1.2 Bit Map

In contrast to the variance map, the bit map is a post-processing partitioning metric. To achieve this map it is necessary to first encode an entire frame, and then acquire the amount of bits generated to represent each CTU of a frame. On the one hand, CTUs with fewer bits tend to represent homogeneous CTUs, since its pixels are highly correlated and good prediction tends to be easier to be obtained. On the other hand, CTUs that need more bits to represent its content usually have more texture, Figure 4.2 shows the bit map of a given *BQTerrace* frame where this intuitive assumption is demonstrated.

Figure 4.2 – *BQTerrace* frame and its bit map.

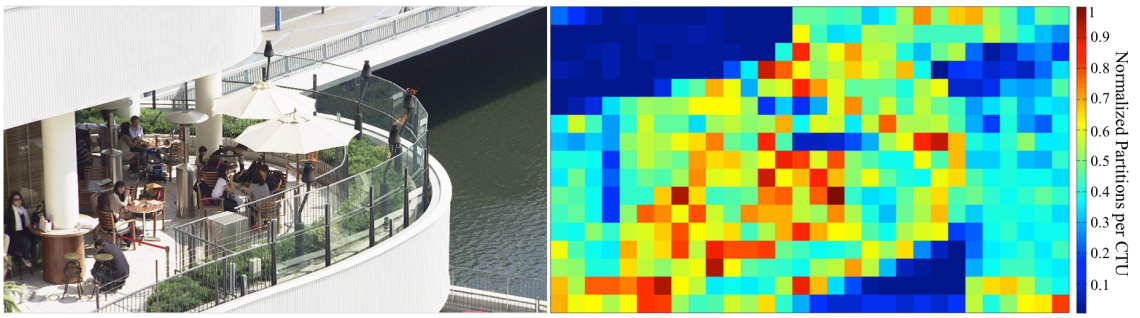


The amount of bits required to represent the information from the CTUs in homogeneous picture regions (top-left corner and mid-bottom regions) is smaller than in more detailed regions (center of the frame). Also, there is a certain correlation between the bit map and the variance map (refer to Figure 4.1). However, there is some information, such as the lightning effects present on the lower right corner of the picture, which are only observed in the bit map. This occurs since the pixels of that regions present little variation between them, and hence it achieves a small variance. Nevertheless, the amount of bits to represent this lightning effect and the water texture is not reduced since it is not trivial information to be represented, requiring more bits per CTU. Taking this into consideration, the bit map can highlight some information that is not possible to be acquired by analyzing the variance map. This kind of information, in some cases, is more relevant to support the definition of coding efficient tiles. However, the bit map has as disadvantage the fact that it can only be acquired after the frame encoding process.

4.1.3 CTU's Partitions Map

As mentioned in Section 2.4.1, the HEVC's CTU can be partitioned into smaller structures to perform suitable prediction. Although smaller partitions result in better video quality after encoding, each additional partition generates individual encoding information, which affects the final bit rate result. This way, the encoder defines that when homogeneous picture regions are considered, it is not necessary to generate diverse encoding information as the context between correlated pixels is availed. Hence, the less the number of partitions inside a CTU, the more homogeneous this CTU tends to be. In turn, if a textured CTU is considered, this CTU tends to reach deeper depths on the quadtree to generate suitable encoding information for different parts of the CTU. Figure 4.3 shows a map considering the number of partitions used per CTU after the encoding process.

Figure 4.3 – *BQTerrace* frame and its CTU's partitions map.



The number of partitions inside the CTUs located on the homogeneous picture regions is reduced. This happens, as mentioned before, since correlated pixels share similar image properties being able to be represented by a bigger CTU. Comparing to the variance and bit map, presented in Figure 4.1 and Figure 4.2 respectively, the CTU's partitions map is more similar to the bit partitioning map. It can be explained by the fact that more partitions into a CTU lead to more encoding information, since each partition have its individual information. The more encoding information inside a CTU, the greater is the amount of bits required to represent this CTU. Another similarity to the bit map is that the CTU's partitions map is obtained only after an entire frame is encoded, since the decision of how many partitions into each CTU is only made after encoding.

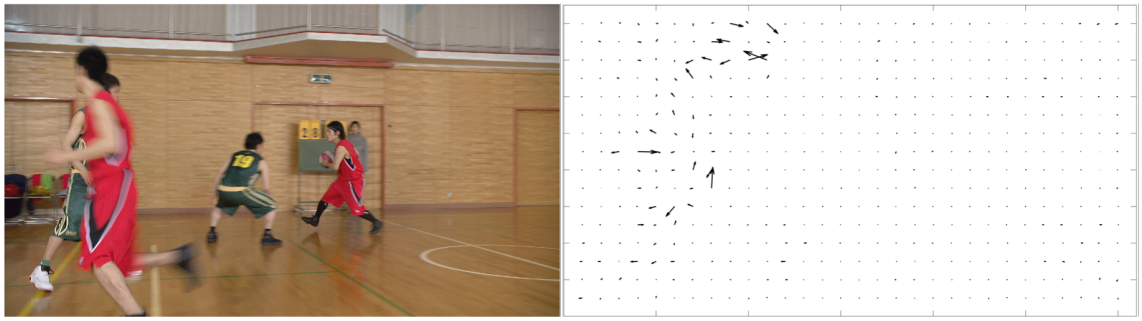
4.1.4 Differential Motion Vectors (DMV) Map

The motion vectors are responsible to point to the direction where the best match for the current block being encoded is located. This best match is located in a reference frame, which is a previously encoded frame. To achieve the DMV partitioning map at least two frames must be encoded. As can be noticed in Figure 4.4, the differential motion vectors are composed by two coordinates (*X, Y*). The DMV is extracted from the bigger CTU size (*64x64* pixels), hence each CTU contains two coordinates. Since the proposed algorithms need a single value per CTU, it is calculated the Euclidean Distance (ED) (GREENACRE and PRIMICERIO, 2013) for each CTU (CTU_{ED}) between both DMV coordinates (DMV_x and DMV_y), as showed in Eq. (4.2).

$$CTU_{ED} = \sqrt{DMV_x^2 + DMV_y^2} \tag{4.2}$$

To reduce the amount of information to be transmitted from the encoder to the decoder, the HEVC calculates the predictive motion vectors (PMVs) that summed to the differential motion vectors (DMVs) generate the current MVs (ZATT, 2012). Since the calculus of the DMVs is based on spatial information, picture regions that present similar MV context are likely to generate zeroed DMVs. This behavior is better perceived in Figure 4.4, where the differential motion field – set of differential motion vectors – of a frame from the *BasketballDrive* benchmark video sequence is presented.

Figure 4.4 – *BasketballDrive* frame and its differential motion field.



The frame presents four basketball players, which are the picture moving subjects. The bigger differential motion vectors are related to these moving subjects, especially to the player that is close to the camera. In contrast, it can be noticed that there are several zeroed DMVs, which are related to the background of the scene. Visually it is possible to infer that the background consists of a highly correlated region of the picture, which is confirmed by the differential motion field presented. These zeroed DMVs are more likely to have analogous motion vector context, and if tile boundaries are placed around these regions it is possible to match natural context breaks, reducing the coding efficiency impact caused by the use of tiles.

4.2 *Adapta* – Adaptive Tiling Algorithm

The Adaptive Tiling Algorithm (*AdapTa*) is able to dynamically choose suitable tile partitions for intra- and inter-predicted frames in order to cluster highly correlated picture regions. The purpose to do so is to reduce the impact on coding efficiency caused by the tile boundaries dependency breaks, which is the main objective of this thesis. To support the definition of suitable tiles, the algorithm makes use of the previous described partitioning maps to dynamically find the regions of interest, and define non-uniform tiles.

The *AdapTa* input is the partitioning map that will be used to define the tiling pattern. The algorithm is composed by two independent stages; the first one acquires the coordinates of the vertical tile boundaries, while the second one acquires the coordinates of the horizontal boundaries. Based on the partitioning map considered, the absolute differences between adjacent columns of the map are calculated and stored in a column difference matrix (*colDiffMtx*). This algorithm step if performed according to the Eq. (4.3). The same process is applied along the partitioning map rows, but the results are stored into *rowDiffMtx* (Eq. (4.4)). This step may be interpreted as a simple border detection technique.

$$colDiffMtx_{i,j-1} = \left|CTU_{i,j} - CTU_{i,j+1}\right| \tag{4.3}$$

$$rowDiffMtx_{i-1,j} = \left|CTU_{i,j} - CTU_{i+1,j}\right| \tag{4.4}$$

Each column of *colDiffMtx* and each row of *rowDiffMtx* will generate a *colSet* and a *rowSet* respectively. There will be *w-1* column sets (*colSets)*, where *w* is the width in CTUs of a picture. In the same way, there will be *h-1* row sets (*rowSets)*, where *h* is the height in CTUs of the picture. Following this, the *AdapTa* calculates the variance of the values contained in each column and row set according to Eq. (4.5) and Eq. (4.6) respectively.

$$colVarArray_{w-1} = \underset{i-1}{\overset{w-1}{\sigma^2}}\left(colSet_{colDiffMtx}\right)_i \quad (4.5)$$

$$rowVarArray_{h-1} = \underset{j-1}{\overset{h-1}{\sigma^2}}\left(rowSet_{rowDiffMtx}\right)_j \quad (4.6)$$

The variance is calculated to show how the differences are distributed along the sets. These arrays are then sorted in descending order to become a vertical tile boundaries ranking, for the *colVarArray*, and a horizontal tile boundaries ranking for the *rowVarArray*. The first array values represent the potentially best boundary indexes, since bigger values might represent natural context breaks of a picture. Although, the column indexes must be verified to guarantee that they fulfill the minimum tile size requirement (*256x64* pixels), as shown in Eq. (4.7). To assess this, the picture dimensions and the previous chosen tile boundary indexes must be considered. Since it is adopted the *64x64* pixels CTU, it is not necessary to test the minimum size of the horizontal tiles, because these tiles will be at least one CTU high, i.e., 64 pixels high.

After assessed the size of the vertical tiles, all the possible vertical tile boundary indexes (*idx*) are organized in a set called VTB_n (Vertical Tile Boundaries indexes), whereas the horizontal tile boundary indexes are organized in the HTB_m (Horizontal Tile Boundaries indexes) set.

$$VTB_n = \{idx \in colVarArray \mid size(idx) \geq 256\times64\} \quad (4.7)$$

The final *AdapTa* step consists of picking the first *Tv* elements from the VTB_n set and the first *Th* elements from the HTB_m set, as shown in Eq. (4.8) and Eq. (4.9). It is important to note that *Tv* is equal to the total number of vertical tiles minus 1, since there is always one less tile boundary than the total number of partitions. The same is adopted for *Th*, where it is equal to the total number of horizontal tiles minus 1.

$$VTB_{Tv} = first(Tv, VTB_n) \quad (4.8)$$

$$HTB_{Th} = first(Th, HTB_m) \quad (4.9)$$

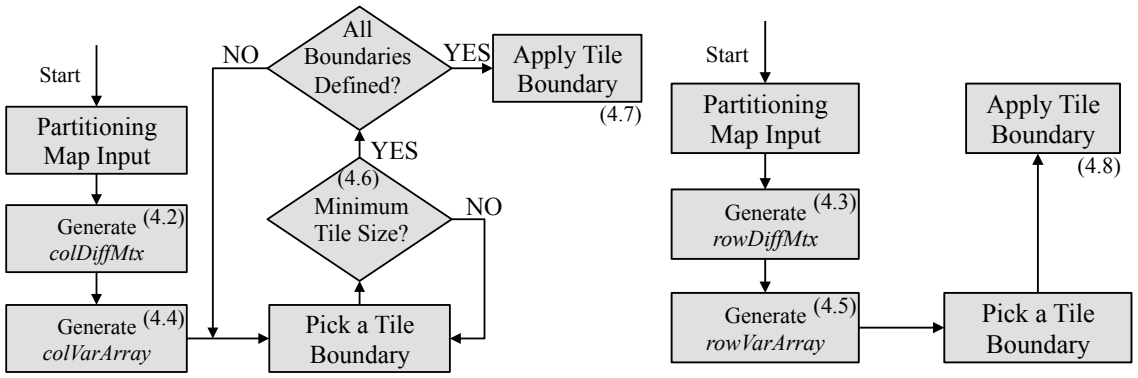
Each one of the performed steps is described on the complete Algorithm 4.1. To better visualize, the *AdapTa* algorithm is summarized on a flowchart; Figure 4.5 (a) presents the steps performed to choose the vertical tile boundaries, while Figure 4.5 (b) shows the choosing of the horizontal tile boundaries. The numbers next to the boxes refers to the equations already mentioned.

Algorithm 4.1 – *AdapTa* – Adaptive Tiling Algorithm

Input: Partitioning Map $\mathbf{M_{i,j}}$, Horizontal Tiles **Th**, Vertical Tiles **Tv**

1: **for** each CTU row/column ∈ $\mathbf{M_{i,j}}$ **do**
2: $\mathbf{colDiffMtx_{i,j-1}}$ ← get diff. between adjacent columns
3: $\mathbf{rowDiffMtx_{i-1,j}}$ ← get diff. between adjacent rows
4: **for** each row ∈ $\mathbf{rowDiffMtx_{i-1,j}}$ **do**
5: $\mathbf{rowVarArray_{w-1}}$ ← get variance along rows
6: **for** each column ∈ $\mathbf{colDiffMtx_{i,j-1}}$ **do**
7: $\mathbf{colVarArray_{h-1}}$ ← get variance along columns
8: sort $\mathbf{rowVarArray_{w-1}}$ & $\mathbf{colVarArray_{h-1}}$ on descendant order
9: $\mathbf{HTB_m}$ ← $\mathbf{rowVarArray_{w-1}}$
10: **for** each *idx* ∈ $\mathbf{colVarArray_{h-1}}$ **do**
10: **if** size(*idx*) ≥ 256x64 **then**
11: $\mathbf{VTB_n}$ ← $\mathbf{colVarArray_{idx}}$
12: $\mathbf{HTB_{Th}}$ ← pick the first **Th** row indexes ∈ $\mathbf{rowVarArray_{w-1}}$
13: define horizontal tiles's size
14: $\mathbf{VTB_{Tv}}$ ← pick the first **Tv** column indexes ∈ $\mathbf{colVarArray_{h-1}}$
15: define vertical tiles's size

Figure 4.5 – Flowchart of *AdapTa* algorithm to define (a) vertical and (b) horizontal tiles.



(a) (b)

This algorithm is applied to define the tile boundaries for each frame. The number of tiles per frame does not vary, but the size and location of the tiles are adapted according to the partitioning map input. The map can be related to the current frame being encoded, or it can be related to previous encoded frames. Adopting different partitioning maps can lead to different tiling patterns, since the partitioning maps provide varied information. An extensive analysis of the efficiency of each map is presented on Chapter 5.

4.3 Tile Scheduling Scheme

It is expected that bigger tiles reflect in bigger workload, while small tiles can be processed with less computational effort. Considering this, our *AdapTa* algorithm can define unbalanced tile partitions, since non-uniform tiles can assume different sizes in a

picture. This happens since the main focus of the proposed algorithm is to enhance the coding efficiency of the tiles by exploiting the highly correlated picture regions, with no concern about the size of the tiles. This way, defining tiles with similar workloads to also enhance the parallelism-potential of the algorithms was an after-thought. To reduce the impact that unbalanced tiles can cause on the parallelism-potential, it is also proposed a tile scheduling scheme.

Taking into consideration the intra-predicted frames, defining unbalanced tiles for a single frame is not an issue that will cause great impact in the parallelism-potential. After all, the unbalanced tiles from different frames, if properly scheduled along encoding time, can become balanced at the end of the encoding process. Exploiting the load balancing along encoding time is only possible, because the intra-predicted frames do not present inter-frame dependencies. Thus, it is viable to begin processing tiles from future frames in display order, while neither all the tiles from the current frame have been processed. However, when the inter-predicted frames are considered, the inter-prediction temporal dependencies limit the parallelism between frames. Due to that, it is proposed a tile scheduling scheme to enhance the parallelism potential of the proposed algorithm for inter-predicted frames.

The proposed tile scheduling scheme consists in an analysis of the tiles' area, aiming to schedule a similar area-based workload between the available processing units. The scheme is described in Algorithm 4.2, and is performed right after the *AdapTa* algorithm is applied.

Algorithm 4.2 – Tile Scheduling Scheme

```
1: given P processing units and T tiles
2: let Tw be the tile width and let Th be the tile height
3: for each t ∈ T do
4:       calculate t_i area ← Tw_i * Th_i
5: sort T according to area in descendant order
6: for each t ∈ T do
7:       schedule t_i to min(P) //processing unit with less area based workload
```

The first step of the proposed scheme is to calculate each tile area based on its width (*Tw*) and height (*Th*) in terms of CTUs (lines 3-4 of Algorithm 4.2). Then, the tiles are descendant sorted (line 5) based on the area occupied. This order is adopted to organize the tiles from the largest to the smallest one. In the proposed scheme the tile's area is used to define its workload.

Following this, the tiles are scheduled in descendant order to the processing unit that has less accumulated area-based workload (lines 6-7). Hence, if the tiling defines a tile that occupies an extensive area of the picture and the others consist of smaller areas, it is possible to schedule the multiple small tiles to one individual unit willing to balance the workload. Chapter 5 shows an analysis of the proposed tile scheduling scheme.

5 EXPERIMENTAL SETUP AND RESULTS

The experiments are performed using the HEVC Test Model (HM) 9.2, following the Common Test Conditions (BOSSEN, 2012) defined by the HEVC standard developers. We analyze a total of nine benchmark video sequences, being four from Class A (*2560x1600* pixels) and the remaining five from Class B (*1920x1080* pixels). These video sequences, presented in Table 5.1, are considered as the tiles were proposed to exploit parallelism, which is mainly advantageous to be employed in high definition content. To easily identify each sequence, a label is assigned to each video sequence.

Table 5.1 – Benchmark video sequences considered.

Video Class	Label	Video Sequence	FPS
Class A (*2560x1600*)	A1	*NebutaFestival_10bits*	60
	A2	*PeopleOnStreet*	30
	A3	*SteamLocomotiveTrain_10bits*	30
	A4	*Traffic*	60
Class B (*1920x1080*)	B1	*BasketballDrive*	50
	B2	*BQTerrace*	60
	B3	*Cactus*	50
	B4	*Kimono*	24
	B5	*ParkScene*	24

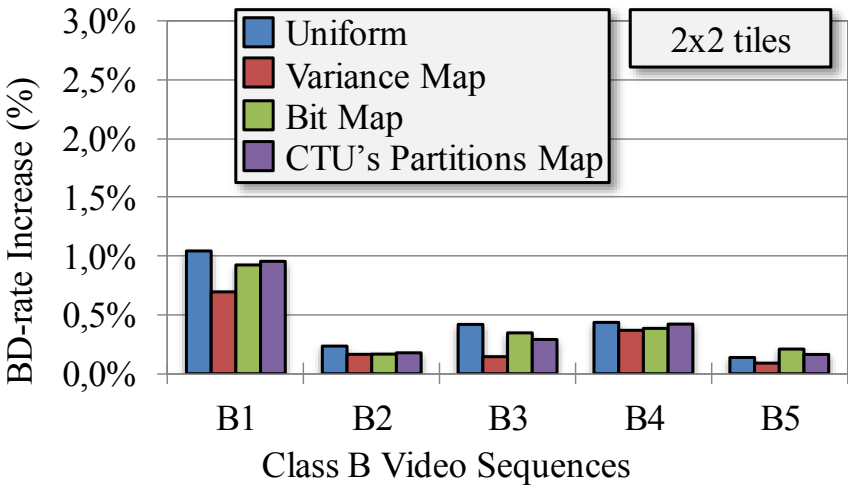
To use as reference, all the video sequences are first encoded using a single tile (*1x1* tile pattern) according to the All Intra (AI) and the Low Delay (LD) coding configurations. For the sake of comparison, another three tiling patterns are adopted: *2x2, 3x3,* and *4x4* tiles, totalizing four, nine, and sixteen partitions per picture, respectively. These tiling patterns are encoded both using the proposed adaptive tiling algorithm, and the uniform-spaced approach. The comparison between the approaches is a way to assess the efficiency of using a content-aware adaptive approach (represented by our adaptive tiling algorithm), versus a size-aware or non-content-aware approach (represented by uniform-spaced tiles). All the results regarding coding efficiency are presented in terms of BD-rate (BJONTEGAARD, 2001) percentage increase (less is better) in comparison to the *1x1* tiling pattern.

5.1 All Intra Experimental Results

The first experiments consider only intra-predicted frames. This way, the All Intra coding configuration is adopted in this test case scenario. We consider three different partitioning maps as the algorithm input: variance map, bit map, and CTU's partitions map. The BD-rate increase caused by our proposed algorithm and the increase caused

by using one single tile are presented in Figure 5.1. This test scenario considers Class B videos partitioned with *2x2*. Figure 5.2 and Figure 5.3 bring the results of the analogous experiments for *3x3* and *4x4* tiling scenarios, respectively.

Figure 5.1 – BD-rate increase using 2x2 tiles (AI in Class B sequences).



In four (B1, B2, B3, and B4) out of five Class B video sequences, the BD-rate increase caused by the tile partitions defined by the *AdapTa* algorithm was smaller than the BD-rate increase caused by the uniform-spaced tiles. This better coding efficiency is achieved since the tile partitions of our algorithm are defined based on the picture content, whereas for uniform-spaced tiles the partitions are defined based only on the picture dimensions. The relative smallest BD-rate increase comparing our algorithm to the uniform approach is perceived in the *BasketballDrive* (B1) video sequence. In this case, our algorithm saves around 0.35% BD-rate compared to the uniform approach when using the variance partitioning map as input information.

The variance map is the most efficient partitioning map to be used as input by the *AdapTa* algorithm for intra-predicted frames, where it saves over 0.4% the BD-rate for the *Kimono* (B4) video sequence using *3x3* tiles, as shown in Figure 5.2. The variance map is the only partitioning map that achieves better coding efficiency than the uniform-spaced tiles for every Class B test scenario. In some cases, such as the B2 video sequence in Figure 5.2 and Figure 5.3, defining the tile boundaries based on the bit map or the CTU's partitions map leads to more BD-rate savings than the uniform-spaced approach and also than the variance map. However, this occurs on specific cases.

This better coding efficiency achieved by the variance map is due to the fact that this is the only map that is extracted previously to the encoding process (pre-processing), so the partitioning map that will support the tile definition is related to the current picture being encoded. The remaining partitioning maps are extracted from the previous encoded frame (post-processing), due to that the map extracted from this frame shows information for this particular frame, then the definition of suitable tile boundaries is compromised if the changes between the information of the previous and the current frame are not negligible. This issue is mainly noticed on the AI profile, since no temporal context is exploited.

This post-processing issue occurs in *ParkScene* (B5) video sequence using *2x2* tiles (Figure 5.1), where the bit map and the CTU's partitions map achieve higher BD-rate

than the uniform-spaced tiles: 0.072% and 0.026% higher BD-rate for the bit map and CTU's partitions map, respectively. This same issue affects other two video sequences when using *3x3* tiles (Figure 5.2). For the *BasketballDrive* (B1) video sequence the BD-rate increase of using bit map is 0.068% higher than the uniform-spaced tiles, while using the CTU's partitions map this increase is 0.116% higher. These partitioning maps also compromise the coding efficiency of the *Cactus* (B3) video sequence, where the bit map adds 0.095% in BD-rate, and the CTU's partitions map adds 0.002% BD-rate. This same issue is also present when *4x4* tile partitions are used (Figure 5.3), where for the B1 video sequence it achieves 0.192% and 0.087% higher BD-rate than the uniform-spaced approach considering the bit and CTU's partitions map, respectively.

Figure 5.2 – BD-rate increase using 3x3 tiles (AI in Class B sequences).

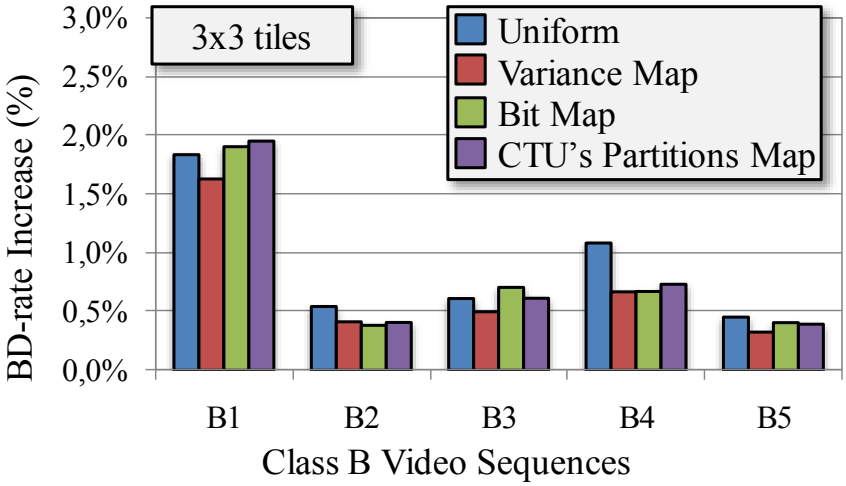
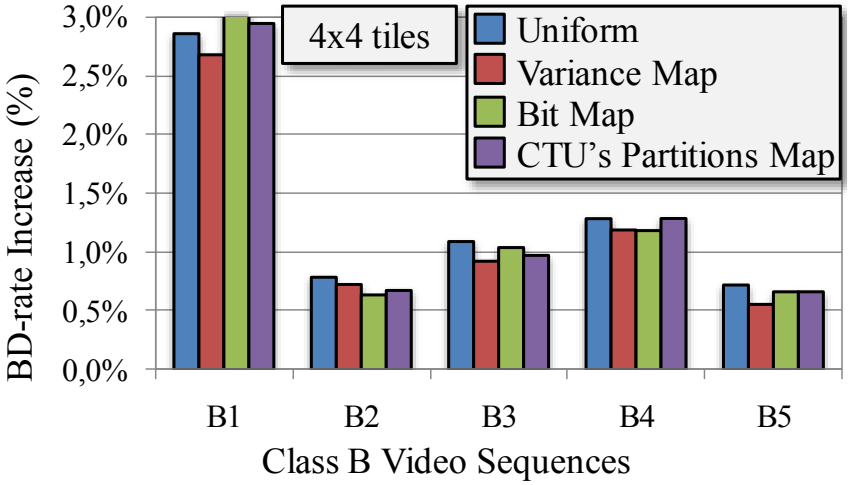
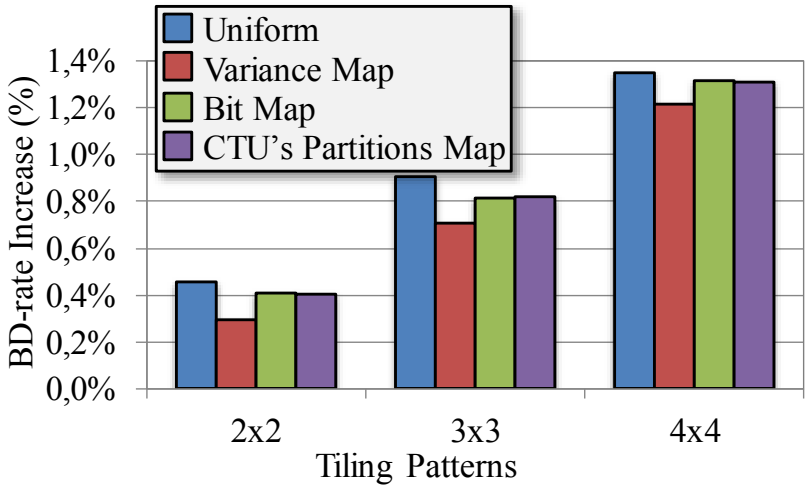


Figure 5.3 – BD-rate increase using 4x4 tiles (AI in Class B sequences).



Even though the post-processing partitioning maps compromise the definition of efficient tile partitions by the *AdapTa* algorithm for intra-predicted frames in some cases, on average the bit map and the CTU's partitions map reflect in smaller BD-rate increase than the uniform-spaced tiles. After calculating the average BD-rate increase of the five considered Class B video sequences, our adaptive tiling algorithm defines more coding efficient tiles than the ones defined by the uniform approach for all the partitioning maps adopted, as can be seen in Figure 5.4.

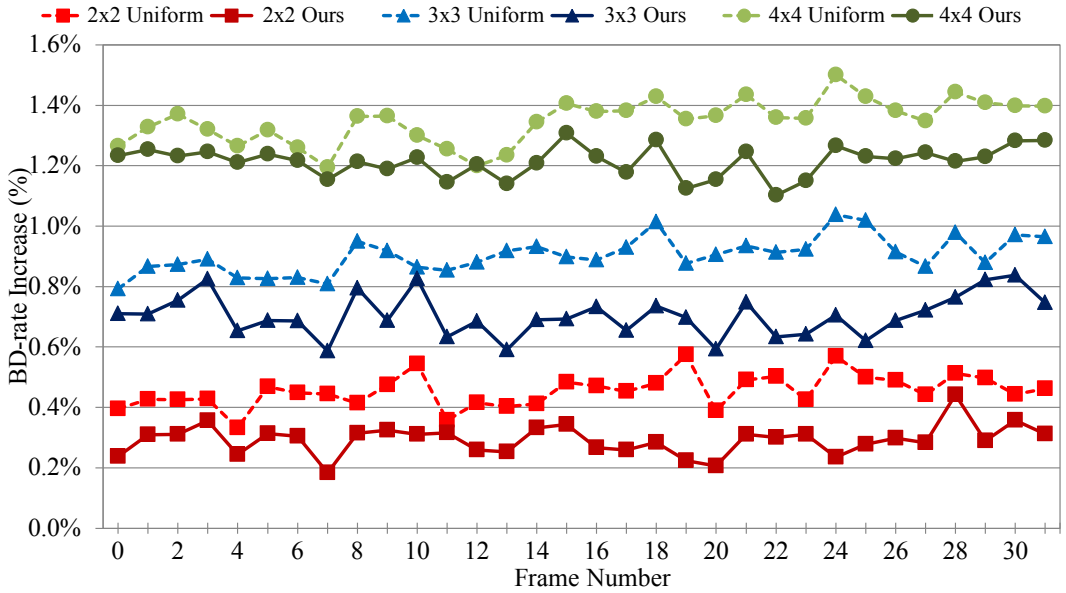
Figure 5.4 – Average BD-rate increase per tiling pattern using AI in Class B videos.



On average, the CTU's partitions map and the bit map save around 0.08% of the BD-rate when compared to the uniform approach. As already mentioned, the variance map is the best partitioning map to support the definition of coding efficient tiles by our adaptive algorithm. The average BD-rate increase in Figure 5.4 proves this fact, where it is saved 0.162%, 0.199%, and 0.134% of the BD-rate for the *2x2, 3x3,* and *4x4* tiling patterns respectively, comparing the uniform tiles to the tiles defined by the variance map.

Achieving the most coding efficient tiles among all the cases when using the variance map is justified when analyzing frame by frame. In Figure 5.5 we present a temporal analysis considering the first 32 frames of the Class B videos. The values are extracted after encoding each frame to generate the BD-rate per frame when compared to encoding with one tile. Then, the average value per frame considering the five Class B video sequences analyzed is extracted.

Figure 5.5 – Increase in BD-rate frame by frame using AI in Class B videos.

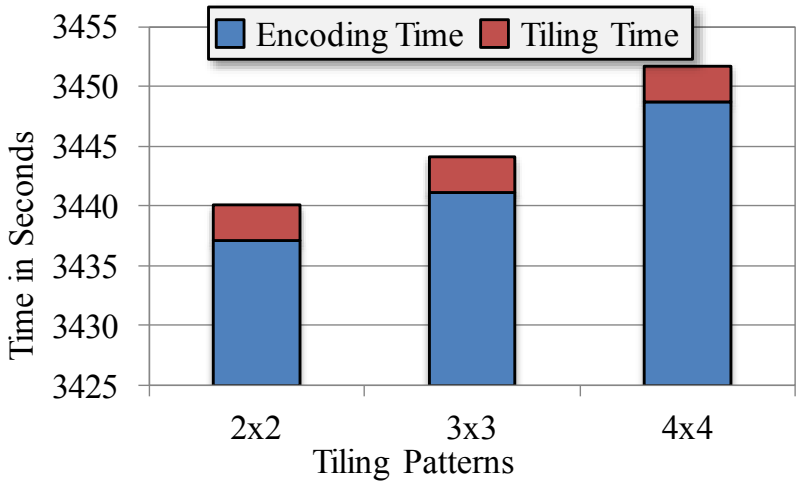


Through all the intra-predicted frames, the *AdapTa* algorithm defined more efficient tile partitions than the ones defined by the uniform-spaced approach. Only frame 12 of

the *4x4* tiling pattern have nearly the same coding efficiency, though higher, as the uniform approach. In this case, the similar result is achieved since the uniform-spaced tiles approach is already a good solution.

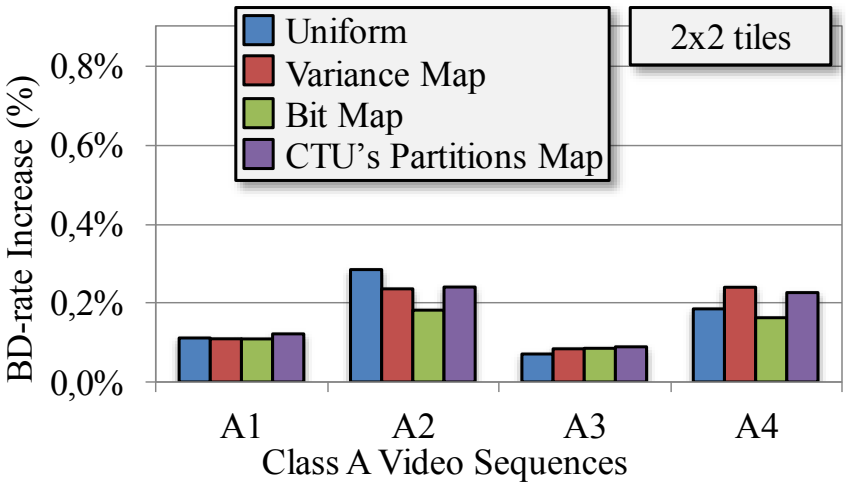
In addition to the coding efficiency analysis of tiles defined based on the variance map, we also present a coding delay analysis. Even supporting the definition of the most coding efficient tiles by the *AdapTa* algorithm for Class B videos using the All Intra coding configuration, the variance map must be extracted before encoding each frame. Therefore, this pre-processing extraction step implies in extra encoding time which is called *tiling time*. However, Figure 5.6 shows that the tiling time is responsible for a negligible delay, because for all the tiling patterns this delay represents on average less than 0.1% of the total encoding time (encoding time summed to tiling time). Moreover, it is worth to mention that typically for real-world encoders the variance extraction is already performed to accelerate processes such as the mode decision. Thus, the variance map extraction does not incur in additional processing cost.

Figure 5.6 – Variance map pre-processing delay in Class B video sequences.



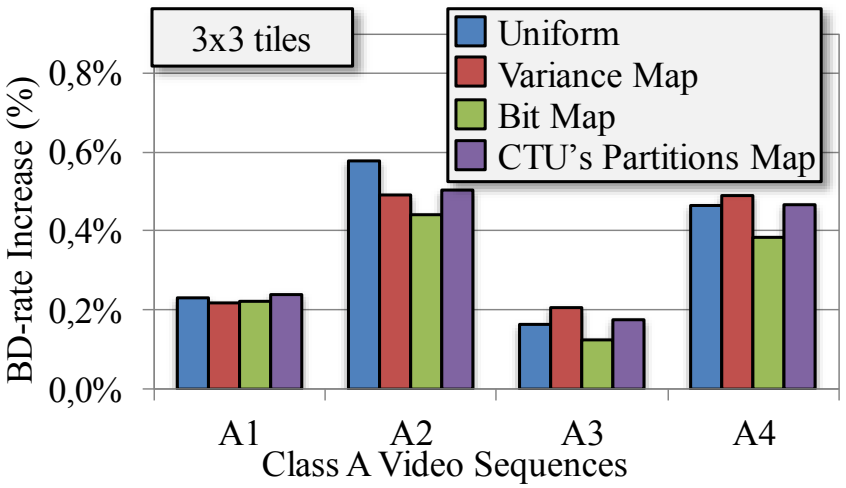
In addition to the Class B video sequences, we also analyze the four Class A benchmark video sequences defined by the Common Test Conditions. We present in Figure 5.7 the BD-rate increase caused by our adaptive tiling algorithm using *2x2* tiles in Class A video sequences. We support the definition of the tile partitions considering the same three partitioning maps used for the Class B experiments. The results are also compared to the uniform-spaced approach.

Figure 5.7 – BD-rate increase using 2x2 tiles (AI in Class A sequences).



Unlike what happened through all the Class B experiments, considering *2x2* tiles in Class A benchmark video sequences, the partitioning map that supports the definition of the most coding efficient tiles is the bit map. The bit map is responsible to achieve lower BD-rate increases in most of the cases for the *2x2* tiling pattern: 0.135% against 0.163% of average BD-rate increase from the uniform tile partitions. However, for the *SteamLocomotiveTrain* (A3) video sequence the most coding efficient tiles were defined using the uniform-spaced partitions, which caused 0.014% less impact on coding efficiency than our solution. The BD-rate savings using *2x2* tiles in Class A videos are smaller than the ones achieved during the Class B experiments. This happens since the Class A video sequences do not present pictures with striking highly correlated regions, turning the definition of suitable tile boundaries more difficult. The same issue is perceived when using *3x3* tiles, as shown in Figure 5.8.

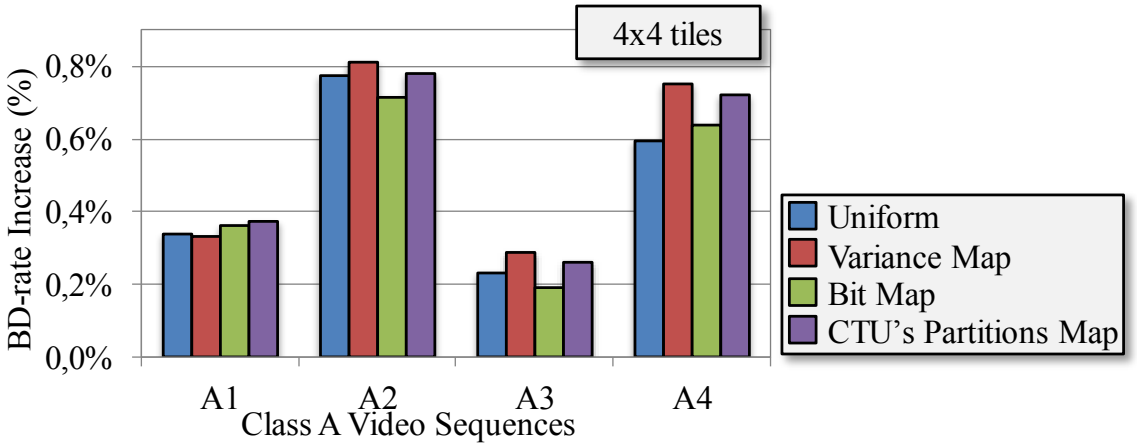
Figure 5.8 – BD-rate increase using 3x3 tiles (AI in Class A sequences).



Besides being able to define more coding efficient tiles than the uniform-spaced approach, in the best case our solution is limited to save 0.136% BD-rate when compared to the uniform tiles. Considering the same test scenario (bit map as partitioning map and the *3x3* tiling pattern) for Class B videos, the best case scenario achieved 0.413% of BD-rate savings comparing to the uniform approach. This reinforces the idea that Class A video sequences do not present highly correlated picture regions as the ones present in Class B videos. The lack of highly correlated picture

regions is better noticed when more tile boundaries are defined, since locating one suitable region is difficult, locating more regions is even more difficult. Figure 5.9 proves that, since the *4x4* tiling pattern achieves the worst experimental results considering the All Intra coding configuration in Class A video sequences.

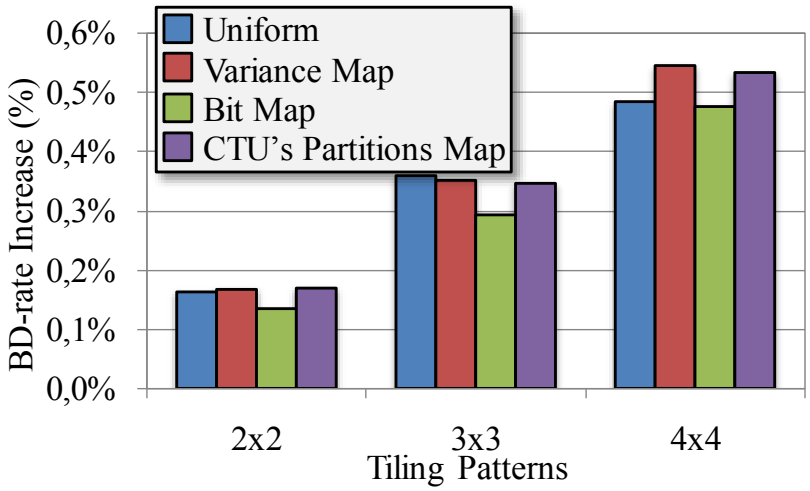
Figure 5.9 – BD-rate increase using 4x4 tiles (AI in Class A sequences).



Using the variance map to support the definition of tile boundaries, our algorithm is more efficient than the uniform-spaced tiles in only one video sequence (*NebutaFestival* – B1), achieving 0.332% BD-rate versus 0.338% of the uniform approach. Considering the CTU's partitions map, our algorithm does not define more coding efficient tiles, while using the bit map it is able to save more BD-rate compared to the uniform-spaced tiles in two video sequences: *PeopleOnStreet* (B2) and *SteamLocomotiveTrain* (B3). For the B2 video sequence our algorithm saves 0.06% BD-rate when compared to the uniform approach, whereas for the B3 sequence this savings are around 0.04%. However, among all the considered tiling patterns, on average our adaptive tiling algorithm causes less impact on coding efficiency when using the bit map.

As Figure 5.10 shows, the average BD-rate increase of our solution using the bit map is smaller than the other partitioning maps and also than the uniform tiles. The BD-rate increase achieved by our algorithm using the bit map to support the tiles definition is 0.028%, 0.066%, and 0.008% smaller than the BD-rate increase achieved by the uniform-spaced tiles for the *2x2*, *3x3*, and *4x4* tiling patterns respectively.

Figure 5.10 – Average BD-rate increase per tiling pattern using AI in Class A videos.



The experiments show that it is possible to define more coding efficient tile partitions if the picture content or the encoding information is considered. Analyzing this information, our algorithm is capable to find the natural picture context breaks. Thus, defining the tile boundaries on these regions lead to more coding efficient tiles as the impact caused by the tile boundary dependency break matches a natural context break. The most suitable partitioning map to search for these regions of interest depend on the video sequence characteristics, such as picture content, video resolution, and frame rate. As the experiments show, for Class A videos (*2560x1600* pixels) which do not present many highly correlated picture regions, the bit map is responsible to support the definition of the most coding efficient tiles. Considering the bit map as input, for the best case our algorithm is capable to save 0.066% of the BD-rate on average. Due to the lack of highly correlated picture regions, the savings are limited. However, when Class B videos are considered these BD-rate savings reach over 0.4% in some cases when using the variance map as input. This happens since there are more highly correlated picture regions in the Class B video sequences used in our experiments.

The experiments here presented considered only intra-predicted frames and three different partitioning maps (variance map, bit map, and CTU's partitions map) to support the definition of tile boundaries by the *AdapTa* algorithm. On the following section we present the experimental results considering inter-predicted frames, and an additional partitioning map that considers temporal context: the DMV map.

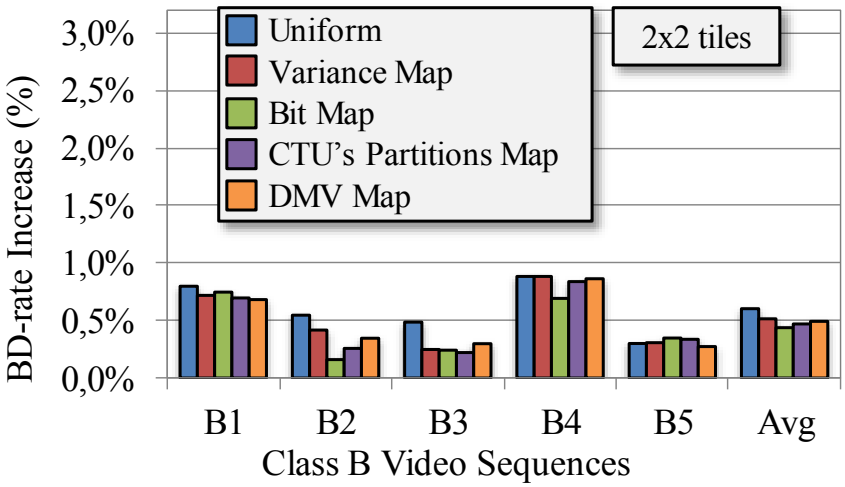
5.2 Low Delay Experimental Results

Besides being able to define more coding efficient tiles for intra-predicted frames, we also present experiments that consider inter-predicted frames. The experiments here performed adopt four different partitioning maps to define the tiling patterns: variance map, bit map, CTU's partitions map, and DMV map. When the DMV map is considered it is used the variance map to define the tiling pattern for the first two frames, since the motion estimation process is not performed until a reference frame is available. After two encoded frames it is possible to acquire the DMV map generated by the motion estimation. We use the Low Delay as coding configuration, because the coding and display order are the same. This is beneficial because the considered post-processing partitioning maps used to support the definition of tile boundaries are extracted from the

previous encoded frame. If this previous encoded frame is also a temporal neighboring frame in display order, we guarantee higher correlation between the extracted map and the current frame. Using the Random Access coding configuration, for instance, the temporal gap between the previous encoded frame and the current frame could be larger, leading to smaller frame correlation.

Figure 5.11 shows the experimental results using *2x2* tiles in Class B videos considering the Low Delay coding configuration. Along with the four considered partitioning maps used as input by our adaptive algorithm, Figure 5.11 also shows the BD-rate increase of the uniform-spaced tiles approach and the average BD-rate increase of the five Class B video sequences.

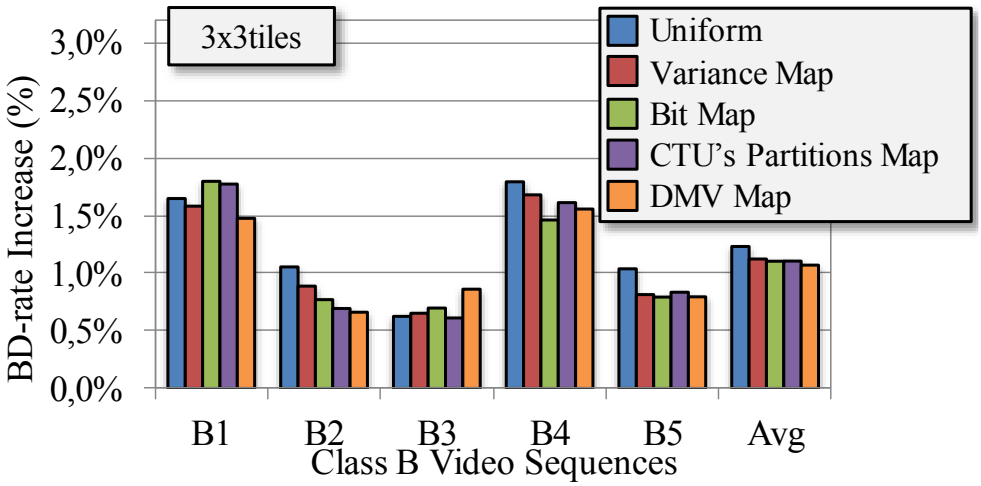
Figure 5.11 – BD-rate increase using 2x2 tiles (LD in Class B sequences).



Even though the variance map benefits from the fact that it is extracted from the current frame being encoded, when inter-predicted frames are considered the pre-processing partitioning map turns not to be very effective. As can be seen, the bit map is the most effective partitioning map being able to save 0.386% BD-rate when compared to the uniform-spaced tiles in the best case scenario (B2 video sequence). On average, the bit map is the algorithm input that leads to the definition of the most coding efficient tiles, increasing only 0.438% the BD-rate in comparison to encoding with the *1x1* tile pattern. The uniform-spaced tiles increase in 0.603% the BD-rate comparing to the same reference.

Taking into consideration that inter-predicted frames exploit the temporal context to perform the encoding process, using post-processing partitioning maps to support the tile definition is a good practice. Analyzing the average BD-rate increase using *2x2* tiles, the three post-processing maps: bit map, CTU's partitions map, and DMV map achieve better coding efficiency, increasing in 0.438%, 0.470%, and 0.493% the BD-rate respectively, than the pre-processing variance partitioning map that increased in 0.515% the BD-rate. The same happens when using *3x3* tiles, as presented in Figure 5.12.

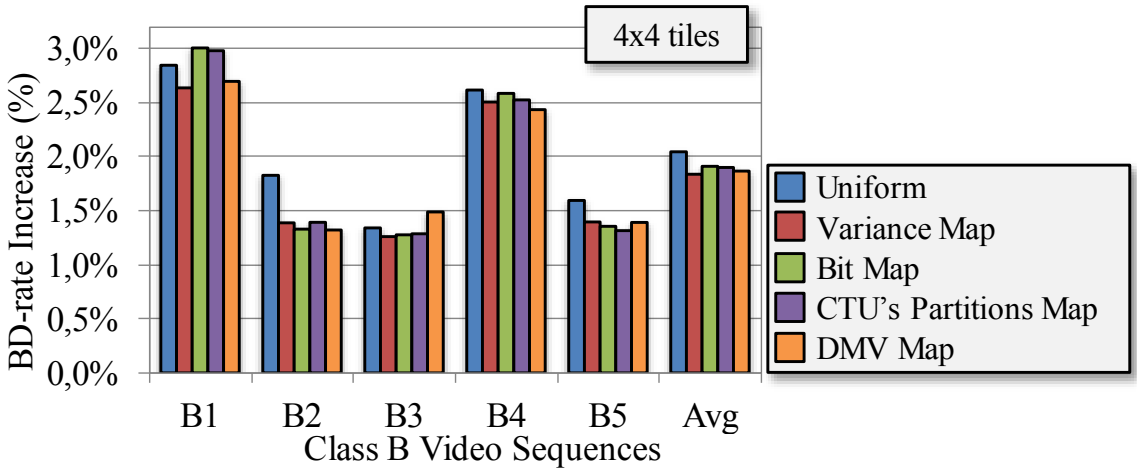
Figure 5.12 – BD-rate increase using 3x3 tiles (LD in Class B sequences).



Adopting the *3x3* tiling pattern the bit map does not present results as good as the ones using *2x2* tiles. Instead of it, the DMV map is the partitioning map that supports the definition of the most coding efficient tiles for Class B videos. Comparing to the uniform-spaced tiles, the DMV map causes more impact on coding efficiency only for the B3 video sequence, reaching 0.861% of BD-rate increase against 0.625% of BD-rate increase caused by the uniform tiles. For the remaining video sequences, it reflects in less coding efficiency impact as can be noticed on the average results, where the DMV map saves 0.162% BD-rate when compared to the uniform-spaced approach.

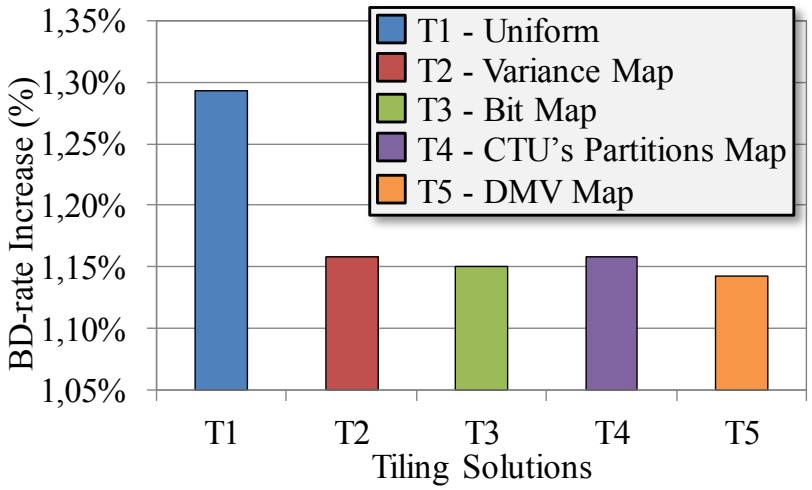
Besides causing less impact on coding efficiency, as more tile boundaries are defined in a picture, the effectiveness of the post-processing partitioning maps such as the bit and the CTU's partitions maps tends to decay. This happens since the amount of bits necessary to encode a given CTU (bit partitioning map), and also the number of partitions inside a CTU (CTU's partitions map) changes especially on the CTUs located next to the tile boundaries. These changes compromise the process of locating suitable regions to define the tile boundaries, because depending on the location of this "changed CTUs" it can indicate a non-correlated block inside a highly correlated picture region. The more tiles are defined, the greater will be the CTUs compromised by this issue. Due to this fact, the results regarding the use of *4x4* tiles (Figure 5.13) show that the impact on coding efficiency caused by the definition of tiles based on post-processing partitioning maps is higher than the impact caused when the variance map (pre-processing partitioning map) supports the definition of the tiles.

Figure 5.13 – BD-rate increase using 4x4 tiles (LD in Class B sequences).



The variance map is the only partitioning map that leads to the definition of tiles that cause less impact on coding efficiency than the uniform-spaced tiles for all the considered Class B video sequences. On average, the BD-rate increase using this map reaches 1.836%, while the uniform approach reaches 2.044%; representing 0.208% of BD-rate savings. Even though the variance map is the partitioning map that supports the definition of the most coding efficient tiles, the post-processing maps also define better tile partitions in terms of coding efficiency when compared to the uniform-spaced tiles. Considering the B2 (*BQTerrace*) video sequence, the DMV map achieves over 0.5% of BD-rate savings compared to the uniform approach. On average, the DMV map saves 0.178% BD-rate, the CTU partitions map saves 0.145%, and the bit map achieves 0.134% of BD-rate savings, all compared to the uniform approach. We also analyze the average BD-rate increase of the five tiling approaches after accumulating the results for the three tiling patterns considering all the five Class B video sequences, as shown in Figure 5.14.

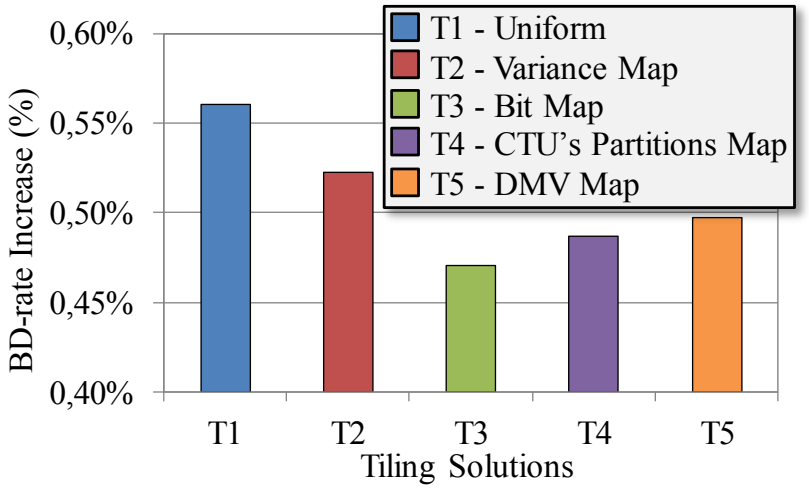
Figure 5.14 – Accumulated average results for Class B videos.



For the accumulated results, all the partitioning maps cause less impact on coding efficiency than the uniform approach. The BD-rate savings comparing our solution to the uniform approach range from 0.135% to 0.150%. The DMV map supports the definition of the best tiles in terms of coding efficiency, increasing in 1.143% the BD-rate while the uniform approach increases in 1.293% the BD-rate (0.150% of BD-rate

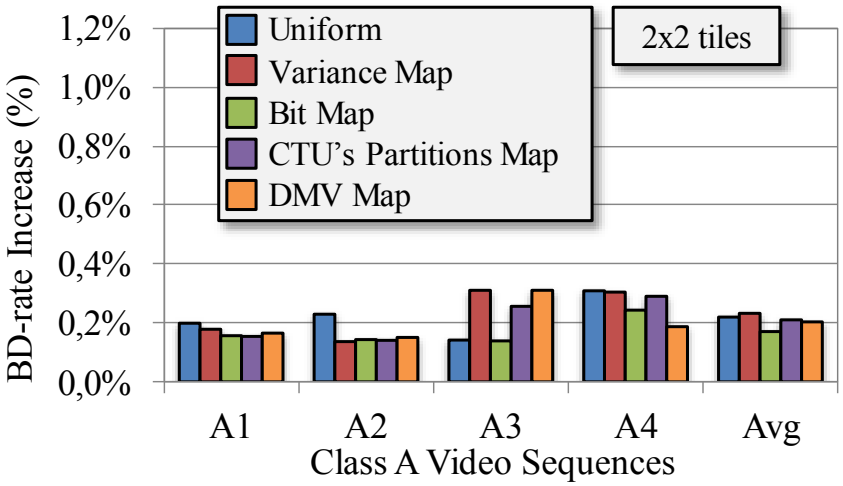
savings) when compared to encoding with one single tile (*1x1* tiling pattern). Seeing Class A video sequences, the developed algorithm also achieves better results than the uniform-spaced approach, as presented in Figure 5.15.

Figure 5.15 – Accumulated average results for Class A videos.



The accumulated results for the *2x2*, *3x3*, and *4x4* tiling patterns in Class A videos show that the BD-rate savings range from 0.038% to 0.089%. As can be noted, the savings are limited due to the behavior and content of the considered benchmark video sequences. The main factor that limits the savings is that the tile boundaries split the entire frame, and then some picture regions are split in undesirable locations. This is more common to occur in videos with bigger resolution since the tile boundaries range is also bigger. This issue is noted in the A3 (*SteamLocomotiveTrain*) video sequence adopting *2x2* tiles, where in Figure 5.16 it is possible to perceive that the variance, the DMV, and the CTU's partitions maps cause more impact on coding efficiency than the uniform approach.

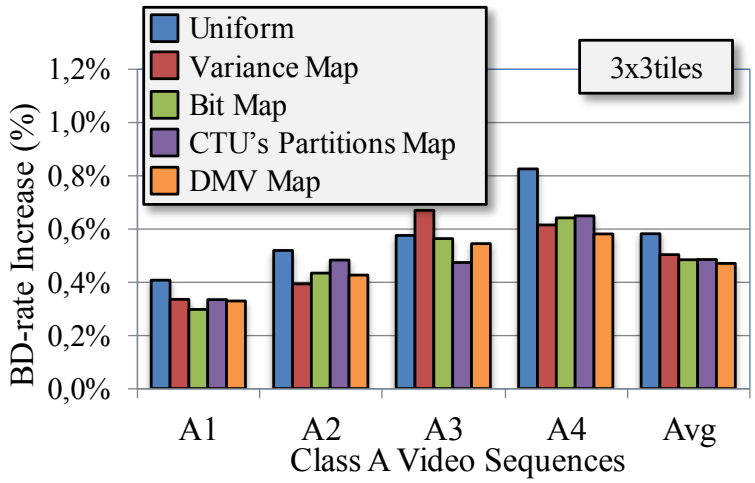
Figure 5.16 – BD-rate increase using 2x2 tiles (LD in Class A sequences).



Another point to note is the limited coding efficiency gain in Class A videos adopting *2x2* tiles. Seeing the best case scenario, achieved when considering the A4 (*Traffic*) video sequence and the DMV partitioning map, the BD-rate savings reach 0.122% comparing to the uniform approach. For the *3x3* tiling pattern, the best results

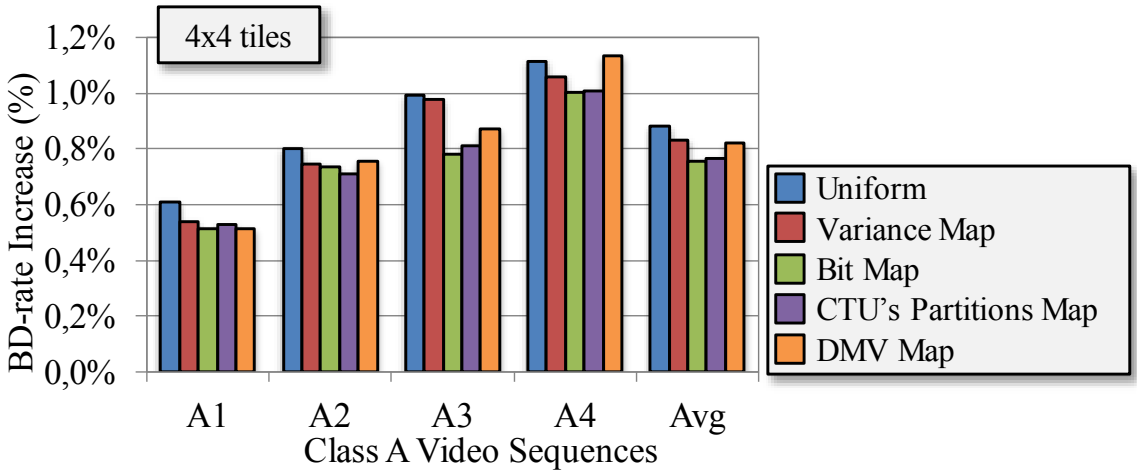
are achieved for the same video sequence and partitioning map, however the savings reach 0.244% as shown in Figure 5.17.

Figure 5.17 – BD-rate increase using 3x3 tiles (LD in Class A sequences).



The DMV map achieves the best results on average only when adopting *3x3* tiles. In this case, it increases in 0.47% the BD-rate compared to encoding with one single tile, while the uniform-spaced tiles' increase reach 0.582%. For the remaining two tiling patterns *2x2* (refer to Figure 5.16) and *4x4* (presented in Figure 5.18) the partitioning map that supports the definition of the most coding efficient tiles is the bit map. Adopting this map our solution saves 0.049% and 0.121% of the BD-rate when compared to the uniform-spaced approach for the *2x2* and *4x4* tiling patterns respectively.

Figure 5.18 – BD-rate increase using 4x4 tiles (LD in Class A sequences).



The results regarding inter-predicted frames show that for the Class A video sequences the partitioning map that supports the definition of the most coding efficient tiles is the bit map, considering all three tiling patterns together. Since the inter-prediction is considered, we also analyze the DMV map. This is an important context to be analyzed, since the motion vector prediction plays an important role in the resulting coding efficiency (SINANGIL, CHANDRAKASAN, *et al.*, 2012). Due to this fact, exploiting the DMV map resulted in the definition of the most suitable tiles in terms of coding efficiency in Class B video sequences. However, the other partitioning maps in

most of the cases also supported the definition of more coding efficient tiles by our algorithm than the ones defined by adopting the uniform-spaced approach.

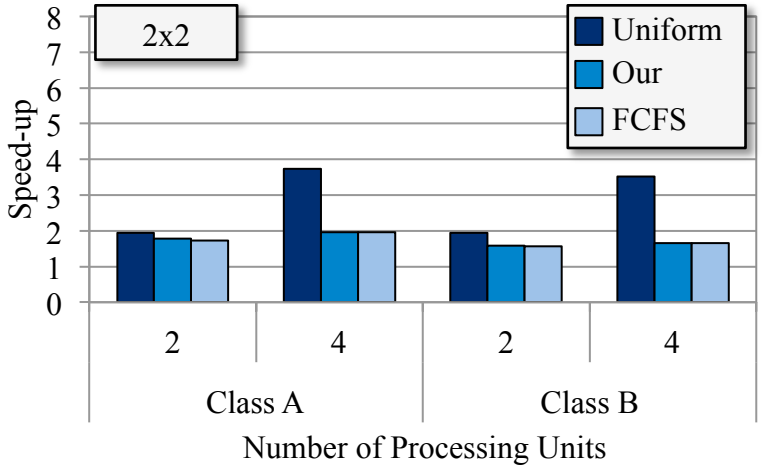
Yet, our algorithm uses non-uniform tiles to define the partitions to match natural picture context breaks. Then, different sized tiles are defined to be appropriately located on these key regions. Taking this into consideration, and the fact that the tiles are proposed to exploit parallel video encoding/decoding, different sized tiles can lead to unbalanced workload. Although non-uniform tiling may result in this parallelism issues for a single frame, for intra-prediction this can be solved by properly scheduling tiles of different frames. However, when the inter-prediction is adopted there are temporal dependencies that limit the scheduling of tiles from neighboring frames resulting in possible processing stalls. Due to this fact, we propose in the following section a tile scheduling scheme to enhance the parallelism potential of tiles defined by our adaptive algorithm.

5.3 Parallelism Results

As shown on the previous section, achieving high coding efficient tiles is possible by clustering the highly correlated picture regions inside the same tile partition. Since these regions might not have similar sizes, we adopt non-uniform tiles to match any specific size. At the same time this approach diminishes the impact on coding efficiency caused by tiles, it also leads to unbalanced workload as different sized tiles might require different computational effort to be encoded. This is mainly noted on inter-predicted frames, where the temporal dependencies pose issues to the possibility of workload balancing by scheduling tiles from different frames. Hence, we also propose a tile scheduling scheme to enhance the parallelism potential of the proposed adaptive tiling algorithm for inter-predicted frames.

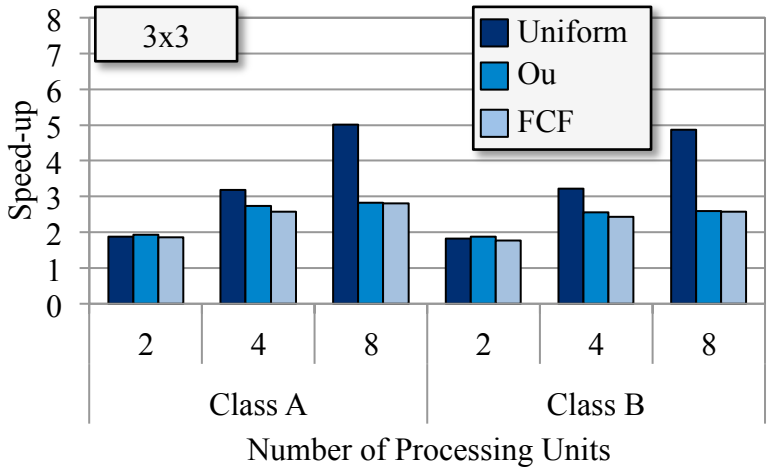
To assess the efficiency of the proposed tile scheduling scheme, it is considered a test scenario that involves scheduling the tiles of the *2x2*, *3x3,* and *4x4* tiling patterns to 2, 4, and 8 processing units. To acquire the results it was simulated a parallel environment that is able to acquire the time spent to encode each tile based on the encoding time of each CTU. The number of tiles and processing units is a required input for the simulations. For the sake of simplicity it is considered a tile based temporal dependency, where the tiles from the current frame must wait until their collocated tiles from the reference frame are encoded. If the dependencies were considered in a CTU level granularity, the achieved speed-ups would be higher. After applying our tiling algorithm, the defined tiles are scheduled following our proposed scheduling scheme, and also a First Come First Serve (FCFS) scheduling scheme (FEITELSON and RUDOLPH, 1995). Both solutions were compared to the uniform-spaced tiles, which are considered to achieve the best load balancing when tiles are defined. In Figure 5.19 it is presented the speed-up results for the *2x2* tiling pattern. In this case, it is not considered 8 processing units since there are not enough tiles in a frame (*2x2* tiles lead to a total of 4 tiles) for this number of processing units.

Figure 5.19 – Speed-up using 2x2 tiles.



When we consider fewer processing units (2) than the total number of tiles (4), our scheduling scheme achieves better speed-up results for both video classes: around 1.78 and 1.59 times faster than the sequential approach for Classes A and B, respectively. For this number of processing units, our scheduling scheme achieves better speed-up than the FCFS approach that is 1.71 and 1.55 times faster than a serial execution. The same behavior can also be observed when *3x3* tiles are considered (Figure 5.20).

Figure 5.20 – Speed-up using 3x3 tiles.

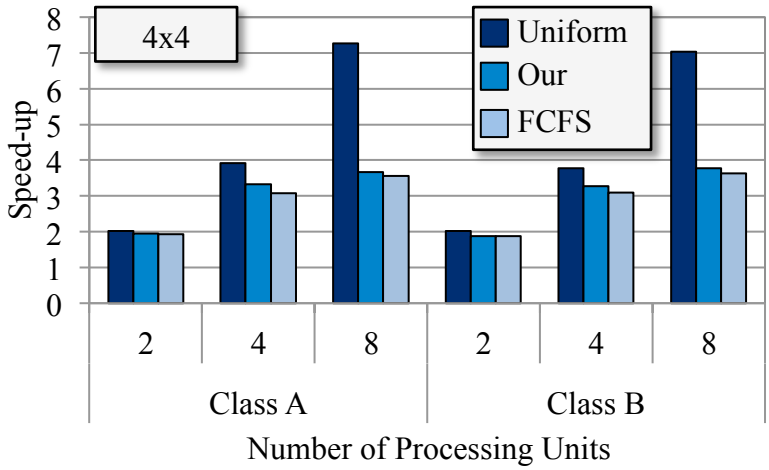


For a total of 9 tile partitions and 2 processing units, our scheduling scheme achieves better speed-up than the ones achieved by uniform-spaced tiles. For Class A video sequences, our scheme is around 1.92 times faster, whereas for the uniform partitioning the speed-up achieved was around 1.87. When Class B videos are considered, our scheme achieves 1.88 speed-up, against 1.82 achieved when the uniform tiles are adopted. This happens since equally sized tiles are not guaranteed to require the same computational effort to be encoded. In some cases, applying our adaptive tiling algorithm can lead to tiles with more accurate area/computational effort ratio, hence out tile scheduling scheme will be able to efficiently balance the workload among the available processing units. When 4 processing units are considered, our scheduling scheme do not achieve better speed-up results than the uniform-spaced tiles, however when compared to the FCFS approach our scheme speed-up is on average 0.3 higher (considering both video classes). For 8 processing units the speed-ups between

the approaches are similar, though the ones achieved by our scheduling scheme are slightly better.

In Figure 5.21, for a limited number of processing units it also achieves good performance. The average speed-up is 1.91 and 3.3 when considering 2 and 4 processing units, respectively. Besides this processing unit limitation, our scheduling scheme shows to be more efficient on average (around 1.1 times) than a conventional FCFS scheduling for all test scenarios.

Figure 5.21 – Speed-up using 4x4 tiles.



To prevent this limitation, it is possible to combine the WPP with the proposed adaptive tiling algorithm. This way, it would be able to exploit parallelism at tile partition level (coarse grain) and also inside each tile by parallelizing its rows (fine grain). However, this solution is proposed as future work.

6 CONCLUSIONS AND FUTURE WORKS

This work presented an adaptive tiling algorithm that is able to reduce the coding efficiency impact caused by the use of tile partitions during the encoding process. Tiles are independent rectangular partitions of a picture that can be processed totally in parallel. To be independent, tiles break the coding dependencies across its boundaries reflecting in coding efficiency impacts.

To reduce the inherent coding efficiency impact of using tiles, we proposed the *Adapta,* an adaptive algorithm that is able to dynamically choose suitable locations of a picture to define the tiling patterns along encoding time. The main objective is to achieve coding efficient tiles, i.e., tiles that cause reduced coding efficiency impact. To define suitable tile partitions, the algorithm looks for highly correlated picture regions to cluster these portions of the frame inside a same tile. This way it is possible to match the tile boundaries with the natural context break of the picture, diminishing the coding efficiency impact caused by the dependency breaks generated by tile boundaries. To locate the highly correlated picture regions the algorithm adopts partitioning maps as its input argument. We considered four different partitioning maps: variance map, bit map, CTU's partitions map, and DMV map.

Analyzing the partitioning maps and adopting non-uniform spaced tiles to cluster the highly correlated picture regions, it was able to define more coding efficient tiles when compared to the uniform approach. Experiments with intra-predicted frames proved that the proposed algorithm saved over 0.4% of the BD-rate when compared to the uniform approach. While for inter-predicted frames, these savings are over 0.5% comparing to uniform-spaced tiles. For all the test case scenarios, the proposed algorithm achieved better coding efficiency using at least one of the considered partitioning maps. Since the algorithm is mainly concerned about analyzing the picture content to define coding efficient tiles, in some cases the use of non-uniform tiles can lead to load balance issues. To enhance the parallelism potential of the non-uniform tiles defined by the proposed algorithm, we also proposed a tile scheduling scheme. This scheme aims to schedule a similar area-based workload between the available processing units. Experiments showed that for a limited number of processing units our tile scheduling scheme achieved near optimal speed-ups. In some cases our scheme was even faster than the uniform approach, which is considered to achieve the best load balance when tiles are defined.

To avoid this processing unit limit, it is proposed as future work to adopt the WPP approach inside the tile partitions. This way, it is possible to exploit both levels of parallelism: coarse grain, represented by the tile partition itself; and fine grain, represented by the WPP rows contained inside each tile partition. Another option to improve the workload balancing among processing units is to implement workload prediction techniques based on the previously encoded frames.

It is also proposed to investigate the relation between the partitioning maps and the video sequence behavior. As could be noticed in the experiments, the partitioning map that defines the most coding efficient tiles depend on the video sequence behavior and characteristics, and also on the tiling pattern adopted. Hence, it is desirable to trace a relation between the partitioning maps and the video sequences to define which map (or combination of maps) is best suited to encode each sequence or frame.

Finally, our investigations led to the observation that, although effective, adaptive tiling techniques are limited by the tiling format rigid rules (the frame must be completely divided by horizontal/vertical boundaries). Note that very frequently a tile boundary that is accurately defined onto natural picture context breaks, leading to reduced coding drawback at that region, also divides highly correlated regions, resulting in meaningful coding efficiency losses. Therefore, as a future work, it is planned the evaluation of flexible tiling formats in order to achieve a more precise frame tiling and further reduce of eliminate the negative impacts in coding efficiency posed by the tiles.

REFERENCES

AFONSO, V. **Desenvolvimento de Arquiteturas para Estimação de Movimento Fracionária Segundo o Padrão HEVC**. Individual Work. Federal University of Pelotas. Pelotas: UFPEL, 2012.

AGOSTINI, L. **Desenvolvimeto de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC**. PhD Thesis. Federal University of Rio Grande do Sul. Porto Alegre: UFRGS, 2007. p. 173.

AHN, Y. J. et al. Complexity model based load-balancing algorithm for parallel tools of HEVC. In: VISUAL COMMUNICATIONS AND IMAGE PROCESSING, 2013... **Proceedings**. Kuching: [s.n.], 2013. p. 1-5.

BAYER, F.; CINTRA, R. Image Compression Via a Fast DCT Approximation. **IEEE Latin American Transactions**. [S.l.:s.n.], v. 8 n. 6, p. 708-713, 2010.

BJONTEGAARD, G. **Calculation of average PSNR differences between RD-curves**. VCEG. [S.l.]. 2001.

BORDES, P. et al. An overview of the emerging HEVC standard. In: INTERNATIONAL SYMPOSIUM ON SIGNAL, IMAGE, VIDEO AND COMMUNICATIONS, 2012... **Proceedings**. Valenciennes: [s.n.], 2012.

BOSSEN, F. **Common HM test conditions and software reference configurations (JCTVC-K1100)**. JCT-VC. Shanghai. 2012.

BOSSEN, F.; FLYNN, D.; SÜHRING, K. **HM Software Manual**. JCT-VC. [S.l.]. 2013.

BROSS, B. et al. **High Efficiency Video Coding (HEVC) text specification draft 10**. JCT-VC. Geneva. 2013.

CHI, C. et al. Parallel Scalability and Efficiency of HEVC Parallelization Approaches. **IEEE Transactions on Circuits and Systems for Video Technology**. [S.l.:s.n.], v. 22 n. 12, p. 1827-1838, 2012.

DEPRÁ, D. A.; ROSA, V.; BAMPI, S. A novel hardware architecture design for binary arithmetic decoder engines based on bitstream flow analysis. In: ANNUAL SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, 2008... **Proceedings**. [S.l.]: ACM, 2008. p. 239-244.

FEITELSON, D.; RUDOLPH, L. Job Scheduling Strategies for Parallel Processing. INTERNATIONAL PARALLEL PROCESSING SYMPOSIUM PROCEEDINGS, 1995... **Proceedings**. Santa Barbara: Springer-Verlag, 1995.

FULDSETH, A. et al. **Tiles**. JCT-VC. Geneva. 2011.

FULDSETH, A. et al. Tiles for Managing Computational Complexity of Video Encoding and Decoding. In: PICTURE CODING SYMPOSIUM, 2012... **Proceedings**. Krakow: [s.n.], 2012. p. 389-392.

GHANBARI, M. **Standard Codecs: Image Compression to Advanced Video Coding**. The Institution of Electrical Engineers. United Kingdom. 2003.

GREENACRE, M.; PRIMICERIO, R. Measures of Distance between Samples: Euclidean. In: MULTIVARIATE ANALYSIS OF ECOLOGICAL DATA, 2013. [S.l.]: Fundación BBVA, 2013. Cap. 4, p. 47-59.

GWEON, R.; LEE, Y. N-level quantization in HEVC. In: IEEE INTERNATIONAL SYMPOSIUM ON BROADBAND MULTIMEDIA SYSTEMS AND BROADCASTING, 2012... **Proceedings**. Seoul: [s.n.], 2012. p. 1-5.

JUURLINK, B. et al. **Scalable Parallel Programming Applied to H.264/AVC Decoding**. [S.l.]: Springer, v. 13, 2012.

LI, B.; SULLIVAN, G.; XU, J. Compression performance of high efficiency video coding (HEVC) working draft 4. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2012... **Proceedings**. Seoul: [s.n.], 2012. p. 886-889.

MISRA, K. et al. An Overview of Tile in HEVC. **IEEE Journal of Selected Topics in Signal Processing**. [S.l.:s.n.], v. 7 n. 6, p. 969-977, 2013.

MONTEIRO, E. **Implementação e Análise de Algoritmos para Estimação de Movimento em Processadores Paralelos tipo GPU (Graphics Processing Units)**. Master Thesis. Federal University of Rio Grande do Sul. Porto Alegre: UFRGS, 2012. p. 114.

NETFLIX. Blog: Netflix. **Netflix**, 2013. Available in: <http://brasilblog.netflix.com/2013/09/a-mais-alta-qualidade-de-hd-ja-esta.html>. Access in: 8 February 2014.

NGUYEN, T.; MARPE, D. Performance analysis of HEVC-based intra coding for still image compression. In: PICTURE CODING SYMPOSIUM, 2012... **Proceedings**. Krakow: [s.n.], 2012. p. 233-236.

NVIDIA. **The Benefits of Multiple CPU Cores in Mobile Devices**. [S.l.]. 2010.

OHM, J. et al. Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC). **IEEE Transactions on Circuits and Systems for Video Technology.** [S.l.:s.n.], v. 22 n. 12, p. 1669-1684, 2012.

OU, Y. F.; ZENG, H.; WANG, Y. Perceptual quality of video with quantization variation: A subjective study and analytical modeling. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2012... **Proceedings**. Orlando: [s.n.], 2012. p. 1505-1508.

PIÑOL, P. et al. An Evaluation of HEVC using Common Conditions. In: XXIV JORNADAS DE PARALELISMO, 2013... **Proceedings**. Madrid: [s.n.], 2013. p. 133-138.

PORTO, M. **Arquiteturas de Alto Desempenho e Baixo Custo em Hardware para a Estimação de Movimento em Vídeo Digitais**. PhD Thesis. Federal University of Rio Grande do Sul. Porto Alegre: UFRGS, 2008. p. 101.

POURAZAD, M. T.; DOUTRE, C.; NASIOPOULOS, P. HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC? **IEEE Consumer Electronics Magazine.** [S.l.:s.n.], v. 1 n. 3, p. 36-46, 2012.

PURI, A.; CHEN, X.; LUTHRA, A. Video coding using the H.264/MPEG-4 AVC compression standard. **Signal Processing: Image Communication**. [S.l.]: Elsevier, v. 19 n. 9, p. 793-849, 2004.

RICHARDSON, I. **H.264/AVC and MPEG-4 Video Compression - Video Coding for Next-Generation Multimedia**. Chichester: John Wiley and Sons, 2003.

RICHARDSON, I. **The H.264 Advanced Video Compression Standard**. 2nd ed. [S.l.]: John Wiley and Sons, 2010.

SAMPAIO, F. **Energy-Efficient Memory Hierarchy for Motion and Disparity Estimation in Multiview Video Coding**. Master Thesis. Federal University of Rio Grande do Sul. Porto Alegre: UFRGS, 2013. p. 120.

SHARMA, A. K. **Text Book of Elementary Statistics**. 1st ed. New Delhi: Discovery Publishing House, 2005.

SINANGIL, M. et al. Memory cost vs. coding efficiency trade-offs for HEVC motion estimation engine. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2012... **Proceedings**. Orlando: [s.n.], 2012. p. 1533-1536.

SULLIVAN, G. J. et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology.** [S.l.:s.n.], v. 22 n. 12, p. 1649-1668, 2012.

SZE, V.; BUDAGAVI, M. Parallel CABAC for Low Power Video Coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2008... **Proceedings**. San Diego: [s.n.], 2008. p. 2096-2099.

URBAN, F.; DÉFORGES, O.; NEZAN, J. Optimization of the motion estimation for parallel embedded systems in the context of new video standards. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XXXV, 2012... **Proceedings**. San Diego: [s.n.], 2012.

VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **IEEE Transactions on Circuits and Systems for Video Technology.** [S.l.:s.n.], v. 22 n. 12, p. 1885-1898, 2012.

WIEGAND, T. et al. Overview of the H.264/AVC Video Coding Standard. **IEEE Transaction on Circuits and Systems for Video Technology.** [S.l.:s.n.], v. 13 n. 7, p. 560-576, 2003.

WIEGAND, T.; SULLIVAN, G.; LUTHRA, A. **Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)**. Joint Video Team (JVT). Geneva, p. 269. 2003.

ZATT, B. **Energy-Efficient Algorithms and Architectures for Multiview Video Coding**. PhD Thesis. Federal University of Rio Grande do Sul. Porto Alegre: UFRGS, 2012. p. 236.

ZHAO, L. et al. Fast mode decision algorithm for intra prediction in HEVC. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING, 2011... **Proceedings**. Tainan: [s.n.], 2011. p. 1-4.

ZHAO, L. et al. Simplified AMVP for High Efficiency Video Coding. In: IEEE VISUAL COMMUNICATIONS AND IMAGE PROCESSING, 2012... **Proceedings**. San Diego: [s.n.], 2012. p. 1-4.

ZHOU, M.; SZE, V.; BUDAGAVI, M. Parallel Tools in HEVC for High-Throughput Processing. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XXXV, 2012... **Proceedings**. San Diego: SPIE, 2012.

ANNEX A – RESUMO EM PORTUGUÊS

Algoritmo de *Tiling* Adaptativo Baseado em Regiões Altamente Correlacionadas de um Quadro para o Padrão de Codificação de Vídeos de Alta Eficiência

A.1. Resumo

Esta dissertação de mestrado propõe um algoritmo adaptativo que é capaz de dinamicamente definir partições *tile* para quadros intra- e inter-preditos com o objetivo de reduzir o impacto na eficiência de codificação. *Tiles* são novas ferramentas orientadas ao paralelismo que integram o padrão de codificação de vídeos de alta eficiência (HEVC – *High Efficiency Video Coding standard*), as quais dividem o quadro em regiões retangulares independentes que podem ser processadas paralelamente. Para viabilizar o paralelismo, os *tiles* quebram as dependências de codificação através de suas bordas, gerando impactos na eficiência de codificação. Este impacto pode ser ainda maior caso os limites dos *tiles* dividam regiões altamente correlacionadas do quadro, porque a maior parte das ferramentas de codificação usam informações de contexto durante o processo de codificação. Assim, o algoritmo proposto agrupa as regiões do quadro que são altamente correlacionadas dentro de um mesmo *tile* para reduzir o impacto na eficiência de codificação que é inerente ao uso de *tiles.* Para localizar as regiões altamente correlacionadas do quadro de uma maneira inteligente, as características da imagem e também as informações de codificação são analisadas, gerando mapas de particionamento que servem como parâmetro de entrada para o algoritmo. Baseado nesses mapas, o algoritmo localiza as quebras naturais de contexto presentes nos quadros do vídeo e define os limites dos *tiles* nessas regiões. Dessa maneira, as quebras de dependência causadas pelas bordas dos *tiles* coincidem com as quebras de contexto naturais do quadro, minimizando as perdas na eficiência de codificação causadas pelo uso dos *tiles.* O algoritmo adaptativo proposto é capaz de poupar mais de 0.4% e mais de 0.5% o BD-rate para quadros intra-preditos e inter-preditos, respectivamente, quando comparado com *tiles* uniformes, solução que não considera o contexto do quadro para definir as partições *tile*.

A.2. Introdução

Os recentes avanços nos dispositivos multimídia possibilitaram a gravação e a reprodução de vídeos de alta definição. Atualmente, diversos tipos de tecnologias, incluindo dispositivos móveis e serviços de fluxo de vídeo através de rede de dados, são capazes de lidar com altas resoluções de vídeo como HD720p e HD1080p (NETFLIX, 2013). Esse tipo de conteúdo envolve uma alta quantidade de dados, que só podem ser transmitidos ou armazenados empregando a codificação de vídeos. O codificador de vídeos avançado H.264 (H.264/AVC – H.264 *Advanced Video Coding* (WIEGAND, SULLIVAN and LUTHRA, 2003)), o qual é amplamente utilizado, foi um ponto chave

para possibilitar a manipulação de vídeos de alta definição. Porém, o desejo dos usuários por melhor qualidade e maior imersão traz a necessidade de manipular resoluções de vídeo ainda maiores, como as resoluções 4K e 8K. Visto que o padrão H.264/AVC não é otimizado para lidar com ultra-altas resoluções de vídeo (SULLIVAN, OHM, *et al.*, 2012), o padrão de codificação de vídeos de alta eficiência (HEVC – *High Efficiency Video Coding standard* (BROSS, HAN, *et al.*, 2013)) foi desenvolvido pelo *Joint Collaborative Team on Video Coding* (JCT-VC) e atingiu sua primeira versão padronizada no primeiro semestre do ano de 2013.

O principal objetivo do JCT-VC era desenvolver um novo padrão de codificação de vídeos capaz de reduzir pela metade a taxa de *bits*, e manter a mesma qualidade visual quando comparado com o padrão anterior. Estudos recentes mostram que as reduções na taxa de *bits* variam entre 36% e 70% (OHM, SULLIVAN, *et al.*, 2012). Apesar disso, essa taxa reduzida de *bits* é atingida a um custo de duas a dez vezes mais esforço computacional. Para lidar com este alto esforço computacional, o padrão HEVC além de suportar a manipulação de vídeos de ultra-alta definição, também foca em outro aspecto chave: o uso de paralelismo para auxiliar no processo de codificação e decodificação de vídeos (POURAZAD, DOUTRE and NASIOPOULOS, 2012).

O suporte ao paralelismo é um recurso muito importante nas aplicações multimídia atuais, como codificadores e decodificadores de vídeos, pois este tipo de aplicação envolve uma alta quantidade de dados. Além disso, a adoção de arquiteturas paralelas pelos novos dispositivos como *tablets*, *smartphones* e computadores, está aumentando cada vez mais (NVIDIA, 2010). No padrão H.264/AVC o paralelismo era explorado utilizando *slices*, os quais delimitam regiões independentes que são capazes de ser codificadas e decodificadas paralelamente. Apesar disso, estas estruturas foram originalmente propostas para reduzir as perdas de codificação relacionadas a erros de transmissão. Assim, utilizar *slices* para explorar paralelismo mostrou-se muito ineficiente, pois eles requisitam a inserção de cabeçalhos adicionais no fluxo de *bits* final (JUURLINK, ALVAREZ-MESA, *et al.*, 2012). Levando isso em consideração, novas ferramentas orientadas ao paralelismo, como as partições *tile*, foram propostas durante o desenvolvimento do padrão HEVC.

As partições *tile* dividem o quadro em regiões retangulares, as quais são definidas por bordas horizontais e verticais. O número de partições pode ser livremente definido, podendo adotar formatos uniformes e não uniformes. Quando *tiles* uniformes são considerados o codificador divide o quadro em partições de mesmo tamanho, porém, quando *tiles* não uniformes são utilizados o tamanho e o formato (o fator forma pode ser definido mas deve ser retangular) de cada partição pode ser livremente definido. Essas partições são consideradas independentes, e dessa forma podem ser processadas em paralelo somente porque as dependências de codificação são quebradas através das bordas dos *tiles*. Ainda assim, essas quebras refletem em perdas de codificação, pois isso compromete o processo de algumas ferramentas de codificação como a codificação de entropia, a intra- e a inter-predição, as quais são altamente baseadas na informação de contexto para atingir taxas de compressão mais altas. Porém, as perdas causadas pelas quebras de dependências são reduzidas quando comparadas às perdas causadas por *slices,* pois cabeçalhos adicionais não são inseridos no fluxo de *bits* final (CHI, MESA, *et al.*, 2012).

Alguns trabalhos, como (MISRA, SEGALL, *et al.*, 2013) e (FULDSETH, HOROWITZ, *et al.*, 2012) comparam a eficiência de codificação de *slices* e *tiles* quando utilizados para explorar o paralelismo, provando que a ferramenta proposta no

padrão HEVC causa menos impacto no processo de codificação. Em (CHI, MESA, *et al.*, 2012) os autores realizam a mesma comparação dos trabalhos anteriormente citados, porém uma ferramenta de paralelismo adicional é considerada. Além de comparar a eficiência de codificação de *slices* e *tiles*, também é analisado o impacto causado pelo uso do *Wavefront Parallel Processing* (WPP) para exploração do paralelismo. Esta é outra ferramenta orientada ao paralelismo proposta no padrão HEVC, a qual divide o quadro em partições linha que podem ser processadas paralelamente. Neste caso o WPP atingiu os melhores resultados com relação a eficiência de codificação, porém as partições definidas por essa técnica não são totalmente independentes, porque algumas informações da linha previamente codificada são utilizadas como referência para codificar a linha atual. Ao mesmo tempo que essas dependências atingem uma melhor eficiência de codificação, elas também comprometem o potencial de paralelismo porque necessitam que haja uma intercomunicação entre as unidades de processamento que codificam as linhas do WPP.

Os trabalhos anteriormente mencionados realizam somente uma comparação entre as ferramentas de processamento paralelo para codificação de vídeos. Se pode notar que existe uma relação entre a eficiência de codificação e o potencial de paralelismo atingido pelas ferramentas analisadas. Apesar disso, nenhum trabalho considera a possibilidade de reduzir o impacto na eficiência de codificação inerentemente causado pelo uso dessas ferramentas orientadas ao paralelismo, sem abdicar da exploração do processamento paralelo no processo de codificação e decodificação. Levando isso em consideração, surge um desafio: *adotar as novas ferramentas orientadas ao paralelismo do padrão HEVC durante o processo de codificação de vídeos, reduzindo as perdas de eficiência de codificação inerentemente causadas pelo uso dessas ferramentas.*

A.2.1. Objetivos do Trabalho

Este trabalho propõe um algoritmo adaptativo que é capaz de dinamicamente escolher partições *tile* adequadas, visando reduzir o impacto na eficiência de codificação causado pelo uso dessa ferramenta orientada ao paralelismo. O algoritmo analisa diversas estratégias, como características da imagem e informações de codificação, para definir as partições *tile*. A definição dos *tiles* é baseada em partições não uniformes, pois o formato e o tamanho podem ser livremente definidos. Este trabalho também propõe um método de escalonamento de *tiles* alternativo para diminuir as questões relativas ao não balanceamento de carga quando *tiles* não uniformes são utilizados. Este método distribui a quantidade de trabalho adequada para as diversas unidades de processamento, visando melhorar o potencial de paralelismo.

A.3. Ferramentas Orientadas ao Paralelismo

As aplicações multimídia atuais são compostas por uma grande quantidade de dados, incluindo os codificador e decodificadores de vídeo, que já são capazes de tratar vídeos com ultra-alta resoluções. Tratar essa grande quantidade de dados requer alto esforço computacional, e por isso os dispositivos atuais já contam com arquiteturas paralelas (NVIDIA, 2010). Visando explorar essas funcionalidades, o padrão de codificação de vídeos propôs duas novas ferramentas orientadas ao paralelismo: o *Wavefron Parallel Processing* (WPP) e os *Tiles*.

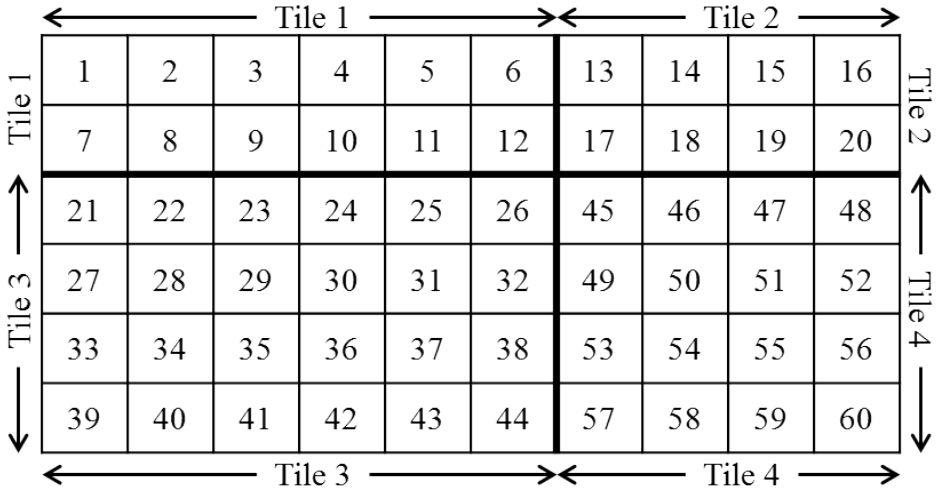
O WPP consiste em dividir um *slice* em linhas de *Coding Tree Units* (CTUs) que podem ser processadas paralelamente. Apesar disso, existem dependências entre as CTUs de diferentes linhas, o que faz com que o paralelismo utilizando WPP seja

atingido com dois blocos de atraso. Além do WPP, os desenvolvedores do padrão HEVC também propuseram os *Tiles,* os quais são partições retangulares consideradas totalmente independentes e que podem ser processadas totalmente em paralelo, sem nenhum atraso. Visto que o foco do trabalho é na utilização de *tiles,* estas partições serão descritas em maiores detalhes na seção seguinte.

A.3.1 Tiles

Tiles são novas ferramentas orientadas ao paralelismo propostas no padrão HEVC. Elas consistem em partições retangulares que são criadas através da definição de bordas horizontais e verticais em um quadro de um vídeo (CHI, MESA, *et al.*, 2012). Estas bordas devem coincidir com as bordas das CTUs, como pode ser observado na Figura A.3.1. Os *tiles* podem assumir formatos uniformes e não uniformes. Nos formatos uniformes o próprio codificador distribui igualmente o número de CTUs dentro de cada partição *tile,* já em *tiles* não uniformes o usuário pode livremente definir a quantidade de CTUs dentro de cada *tile.* Apesar disso, existe uma restrição quanto ao tamanho mínimo de um *tile,* o qual deve ser maior do que *256x64 pixels* (BROSS, HAN, *et al.*, 2013). Por exemplo, se o maior tamanho de um CTU for considerado (*64x64 pixels*), o tamanho mínimo de um *tile* será quatro CTUs de largura por uma CTU de altura. Outra característica dos *tiles* é que eles alteram a ordem de varredura das CTUs, como pode ser visto na Figura A.3.1.

Figura A.3.1 – Ordem de varredura de CTUs em um quadro com quatro tiles.



Os *tiles* são considerados independentes, pois todas as dependências de codificação e predição, como vetores de movimento, amostras vizinhas da predição intra-quadros e o contexto da codificação de entropia, são quebradas através das bordas dos *tiles.* A única ferramenta que é capaz de atuar através das bordas são os filtros, os quais são utilizados para reduzir o efeito de blocagem. Para controlar isso, um parâmetro é utilizado para ligar ou desligar a aplicação dos filtros através dos *tiles* (BROSS, HAN, *et al.*, 2013).

Considerando que todos os *tiles* de um quadro podem ser processados paralelamente, não há garantia que eles sejam processados na sua ordem natural. Por isso, o codificador e o decodificador devem ter conhecimento de onde cada *tile* está localizado no quadro, para que ele possa ser montado e reconstruído. Isso pode ser feito adicionando marcadores no fluxo de *bits* final, ou através de deslocamentos

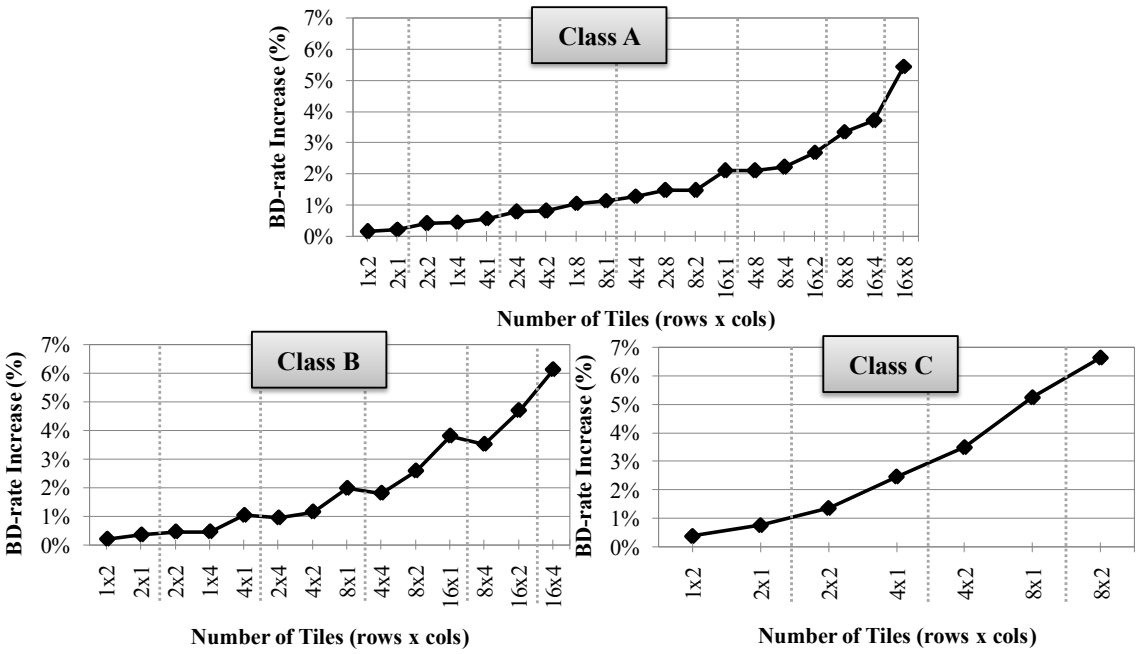
(FULDSETH, HOROWITZ, *et al.*, 2012). Além disso, para viabilizar o processamento paralelo dos *tiles* eles não apresentam dependências entre si. Com isso, todas as dependências de codificação e predição são quebradas através de suas bordas. Embora isto permita explorar o paralelismo, as quebras também refletem em um certo impacto na eficiência de codificação, por isso é apresentado na próxima seção um experimento para analisar o comportamento dos *tiles* para codificação de diferentes resoluções de vídeo.

A.4. Avaliação de Particionamento Fixo de *Tiles*

Ao ser definido um particionamento utilizando *tiles* no *software* de referência HM (HEVC *Test Model*) do padrão HEVC, este padrão é fixado e utilizado na codificação de todos os quadros que compõe a sequência de vídeo. Este particionamento fixo facilita a marcação que indica a localização de cada uma das bordas *tile* presentes nos quadros, porém este caráter fixo pode apresentar problemas de balanceamento de carga e eficiência de codificação na medida em que o conteúdo do quadro é alterado e deslocado. Visando analisar e entender a influência de um particionamento utilizando padrões de *tiling* fixos, foram realizados experimentos para verificar o impacto causado por essas partições na eficiência de codificação.

A eficiência de codificação é medida em termos do Bjontegaard Delta Bit-rate (BD-rate), a qual é uma métrica que representa o percentual da diferença da taxa de *bits* entre dois vídeos de mesma qualidade (BJONTEGAARD, 2001). Na avaliação, mostrada na Figura A.4.1, foram considerados particionamentos *tile* uniformes e fixos, os quais foram comparados com a mesma sequência utilizando um único *tile* (sem quebra de dependências). Os experimentos seguiram as Condições Comuns de Teste (BOSSEN, 2012) do padrão HEVC, e consideraram de 2 a 128 *tiles* por quadro (limitado pela resolução de cada sequência de vídeo).

Figura A.4.1 – Aumento de BD-rate causado por vários tipos de padrão de *tiling* uniforme para sequências de vídeo Classes A, B e C.



Considerando todas as sequências de vídeo analisadas, se pode perceber que o impacto na eficiência de codificação cresce na medida em que mais *tiles* são utilizados.

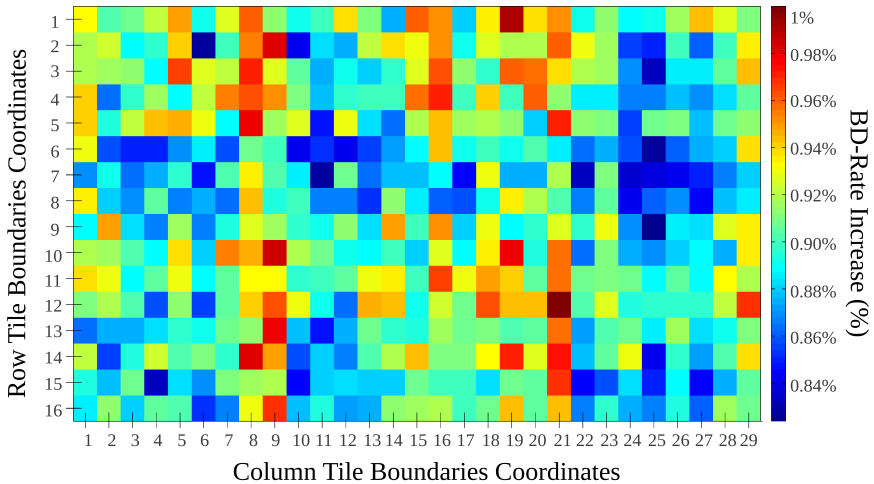
Isso acontece, pois quanto mais partições *tile* são definidas, mais quebras de contexto acontecem, e com isso a eficiência de codificação decresce. Além disso, o impacto na eficiência de codificação causado pelos *tiles* é maior em vídeos de resolução menor. Enquanto em vídeos Classe A (*2560x1600 pixels*) o aumento do BD-rate no pior caso foi de 5.45% utilizando 128 *tiles,* em vídeos Classe C (*832x480 pixels*) esse impacto chegou a 6.63% utilizando apenas 16 *tiles*. Isso pode ser explicado porque em resoluções menores as bordas das partições *tile* tendem a cortar regiões que comprometem mais o BD-rate do que em resoluções de vídeo maiores.

Outro comportamento verificado foi que *tiles* que, quanto mais próximo do formato quadrado é uma partição, menor é o impacto causado por ela na eficiência de codificação. Isso se explica através da relação área/perímetro, onde o perímetro representa a borda dos *tiles,* ou seja, onde as dependências são quebradas. Já a área é representada pela número de blocos dentro de um *tile.* Portanto, quanto maior for essa relação, melhor tende a ser a eficiência de um *tile,* e é possível notar que partições quadradas apresentam a maior relação área/perímetro possível (CHI, MESA, *et al.*, 2012). Embora isso tenha sido verificado nos experimentos, o formato de um *tile* não garante sua eficiência, pois isso também vai depender do conteúdo de cada um deles.

A.5. Desafios de Paralelização na Codificação de Vídeos

Mesmo que a utilização de partições *tile* com formato próximo do quadrado levem a definição de *tiles* mais eficientes em termos de codificação, existem outras técnicas que podem levar a definição de *tiles* adequados. Uma maneira de fazer isso é definir as bordas dos *tiles* em regiões do quadro que apresentam uma quebra natural de contexto. Dessa forma, as quebras inseridas pelas bordas das partições não causariam tanto impacto na eficiência de codificação final. Para investigar a procedência dessa hipótese, foi realizado um experimento exaustivo que testou todos os tipos de particionamento utilizando *2x2 tiles* para um quadro da sequência de vídeo BQTerrace (*1920x1080 pixels*) (BOSSEN, 2012). Quando são considerados *2x2 tiles* existe apenas um ponto de intersecção entre as bordas horizontais e verticais. Cada um dos quadrados da Figura A.5.1 representa o aumento de BD-rate quando o ponto de intersecção está localizado naquele ponto específico.

Figura A.5.1 – Aumento de BD-rate causado por cada particionamento utilizando *2x2 tiles.*



Como pode ser observado, existem diversos pontos na Figura A.5.1 que apresentam reduzidas perdas na eficiência de codificação (quadrados azul escuros). Muitos desses pontos apresentam impactos ainda menores do que os causados pelo particionamento uniforme (intersecção [15,8] na Figura A.5.1). Esses pontos que geram *tiles* mais eficientes foram obtidos através da definição de bordas que coincidem com quebras naturais de contexto nos quadros do vídeo. Com isso, agrupar regiões altamente correlacionadas dentro de um mesmo *tile* pode levar a definição de partições mais eficientes. Dessa maneira, surge um primeiro desafio: *localizar, para cada quadro, os pontos de intersecção que são capazes de agrupar regiões correlacionadas e dividir regiões com diferentes propriedades de codificação, visando melhorar a eficiência dos tiles.*

Considerando que os *tiles* foram definidos para explorar processamento paralelo na codificação de vídeos, o balanceamento de carga entre as unidades de processamento deve ser levado em conta. A solução imediata seria utilizar *tiles* de mesmo tamanho, porém isto não garante que eles irão requerer o mesmo esforço computacional para serem codificados, pois isto também depende de seu conteúdo. Com isso, surge um segundo desafio: *aumentar o potencial de paralelismo da utilização de tiles durante o processo de codificação de vídeos.*

Levando em consideração os dois desafios apontados, este trabalho tem como objetivo principal *desenvolver soluções que sejam capazes de adaptar o padrão de tiling ao longo da codificação de vídeos, visando reduzir as perdas na eficiência de codificação causadas pelo uso de tiles.* Além disso, se propões *melhorar o potencial de paralelismo dos tiles adaptativos,* uma vez que eles podem definir partições não balanceadas.

A.6. Soluções de Particionamento *Tile* Adaptativas

Como pode ser visto na Figura A.5.1, existem diversos pontos de um quadro que levam a definição de *tiles* mais eficientes. Porém, realizar uma varredura exaustiva previamente a codificação de cada um dos quadros de um vídeo se torna inviável. Em vista disso, é proposta a utilização de mapas de particionamento, os quais são gerados após a obtenção de informações de codificação e características do vídeo. Através disso, é possível derivar as regiões altamente correlacionadas dos quadros para que elas possam ser agrupadas dentro de um mesmo *tile,* causando menor impacto na eficiência de codificação. A seguir, são descritos os quatro mapas de particionamento utilizados por este trabalho para definição de *tiles* mais eficientes.

A.6.1. Mapa de Variância

A variância mede a variabilidade entre os elementos de um grupo de dados através do cálculo da distância de cada elemento com relação à média do grupo. A fórmula da variância é definida segundo a Eq. (A.6.1) (SHARMA, 2005).

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \qquad \text{(A.6.1)}$$

Para obter o mapa de variância de cada quadro, a variância de cada CTU de *64x64 pixels* é extraída. Com isso é possível identificar as CTUs com conteúdo homogêneo (que apresentam baixa variância), e as CTUs com mais textura (que apresentam variância mais alta) (ZATT, 2012). As regiões homogêneas do mapa de variância tendem a ter informação espacial similar, e por isso são consideradas altamente

correlacionadas com relação ao processo de codificação. Agrupando essas regiões em um mesmo *tile* tornará o impacto causado pelas quebras de contexto menor. Este mapa é extraído previamente a codificação de cada quadro do vídeo, por isso o consideramos um mapa pré-processado.

A.6.2. Mapa de *Bits*

Comparando com o mapa de variância, o mapa de *bits* é considerado pós-processado, pois para ser obtido no mínimo um quadro deve ser codificado. A quantidade de *bits* necessária para representar uma determinada CTU é fornecida pelo próprio codificador. Com isso, quanto menor for a quantidade de *bits*, mais homogênea tende a ser o conteúdo dessa CTU, e quanto mais *bits* forem necessários para representar uma CTU, maior quantidade de detalhes esta CTU tende a ter.

O mapa de *bits* é capaz de apontar as regiões homogêneas do quadro, ou seja, as regiões que compartilham contexto semelhante entre si. Apesar disso, as informações destacadas por este mapa são diferentes das destacadas pelo mapa de variância, por isso é interessante utilizar diferentes mapas para definir o particionamento utilizando *tiles* de maneira mais eficiente.

A.6.3. Mapa de Partições por CTU

No padrão HEVC, as CTUs podem ser particionadas para que a predição possa ser realizada de forma mais adequada. Dessa forma, o codificador define que quando CTUs homogêneas são consideradas, não é necessário gerar diversas informações de codificação porque o contexto entre os *pixels* correlacionados podem ser aproveitadas. Assim, quanto menor for o número de partições dentro de uma CTU, mais homogênea essa CTU tende a ser. Já se a CTU apresentar maior quantidade de detalhes, este tende a ser mais particionada.

Considerando isto, o número de partições dentro de cada CTU do quadro pode indicar as regiões homogêneas e altamente correlacionadas de um quadro. Assim como o mapa de *bits,* as informações sobre o número de partições dentro de cada CTU são geradas pelo codificador, e por isso é necessário que um quadro seja previamente codificado para que esta informação seja obtida.

A.6.4. Mapa de Vetores de Movimento Diferenciais

Os vetores de movimento são responsáveis por apontar para a direção onde está localizado o melhor casamento entre o bloco atualmente sendo codificado e um bloco previamente codificado. Para isso que estes vetores sejam obtidos, no mínimo dois quadros devem ser codificados. Visto que os vetores apontam uma direção, eles são compostos por duas coordenadas (*X,Y*). Porém, para que um mapa de particionamento possa ser gerado a partir destes vetores, é necessário apenas um valor por CTU. Por isso, é calculada a Distância Euclidiana (GREENACRE e PRIMICERIO, 2013) entre cada uma das coordenadas dos vetores de movimento diferenciais, como é mostrada na Eq. (A.6.2).

$$CTU_{ED} = \sqrt{DMV_x^2 + DMV_y^2} \qquad \text{(A.6.2)}$$

Para reduzir a quantidade de informação a ser transmitida, o padrão HEVC busca predizer os vetores de movimento com base na informação dos vetores de movimento vizinhos (ZATT, 2012). Como este cálculo é baseado em informações espaciais, regiões

que apresentam contexto similares tendem a geradas vetores de movimento diferenciais próximos do zero. Estes vetores tendem a ter contexto análogo, e caso as bordas dos *tiles* sejam definidas ao seu redor, coincidindo com quebras naturais de contexto do quadro, o impacto na eficiência de codificação causado pelos *tiles* pode ser reduzido.

A.7. *AdapTa – Adaptive Tiling Algorithm*

O *Adaptive Tiling Algorithm* (*AdapTa*) é capaz de dinamicamente escolher partições *tile* que agrupam regiões altamente correlacionadas dos quadros. O propósito disso é reduzir o impacto na eficiência de codificação causado pelas quebras de dependência geradas pelas bordas dos *tiles.* Para auxiliar na definição de *tiles* eficientes, o algoritmo faz uso dos mapas de particionamento para encontrar as regiões de interesse dos quadros.

O algoritmo proposto é composto por dois estágios independentes, o primeiro adquire as coordenadas das bordas verticais dos *tiles,* já o segundo estágio consiste em adquirir as coordenadas das bordas horizontais dos *tiles.* O parâmetro de entrada do algoritmo é o mapa de particionamento a ser considerado. De posse desse mapa, são extraídas diferenças absolutas entre colunas e linhas adjacentes, organizando esses resultados em duas matrizes: *colDiffMtx* (Eq. (A.7.1)) e *rowDiffMtx* (Eq. (A.7.2)).

$$colDiffMtx_{i,j-1} = \left|CTU_{i,j} - CTU_{i,j+1}\right| \tag{A.7.1}$$

$$rowDiffMtx_{i-1,j} = \left|CTU_{i,j} - CTU_{i+1,j}\right| \tag{A.7.2}$$

Cada coluna e linha das matrizes *colDiffMtx* e *rowDiffMtx*, respectivamente irão gerar conjuntos de dados intitulados *colSets* e *rowSets.* O próximo passo do algoritmo consiste em extrair a variância de cada um destes conjuntos de acordo com as Eq. (A.7.3) e Eq. (A.7.4).

$$colVarArray_{w-1} = \underset{i-1}{\overset{w-1}{\sigma^2}}\left(colSet_{colDiffMtx}\right)_i \tag{A.7.3}$$

$$rowVarArray_{h-1} = \underset{j-1}{\overset{h-1}{\sigma^2}}\left(rowSet_{rowDiffMtx}\right)_j \tag{A.7.4}$$

A variância é calculada para verificar como as diferenças estão distribuídas ao longo do conjunto, evitando que máximos locais interfiram na definição das bordas dos *tiles*. Uma vez extraída as variâncias dos conjuntos linha (*rowSets)* e dos conjuntos coluna (*colSets*), estas são organizadas em vetores chamados *rowVarArray* e *colVarArray,* respectivamente. Estes vetores são organizados em ordem decrescente para serem utilizados como um *ranking*, onde os primeiros valores tendem a indicar as potenciais melhores coordenadas para as bordas dos *tiles.*

O próximo passo do algoritmo é garantir que os *tiles* verticais respeitem a regra do tamanho mínimo por *tile* (*256x64 pixels*). Com isso, todos os possíveis índices a serem definidos são organizados em conjuntos chamados *VTB* (*Vertical Tile Boundaries indexes*) e *HTB* (*Horizontal Tile Boundaries indexes*). Por fim, basta selecionar a quantidade de índices necessária para definição dos *tiles* verticais e horizontais*,* como mostram, respectivamente, as Eq. (A.7.5) e Eq. (A.7.6). Sempre devem ser selecionadas uma borda a menos do que a quantidade total de *tiles.*

$$VTB_{Tv} = first(Tv, VTB_n) \quad \text{(A.7.5)}$$

$$HTB_{Th} = first(Th, HTB_m) \quad \text{(A.7.6)}$$

Cada um dos estágios do algoritmo é apresentado no pseudocódigo mostrado em Algoritmo A.7.1.

Algoritmo A.7.1 – *AdapTa* – Adaptive Tiling Algorithm

Input: Partitioning Map $\mathbf{M_{i,j}}$, Horizontal Tiles **Th**, Vertical Tiles **Tv**

1: **for** each CTU row/column ∈ $\mathbf{M_{i,j}}$ **do**
2: $\mathbf{colDiffMtx_{i,j-1}}$ ← get diff. between adjacent columns
3: $\mathbf{rowDiffMtx_{i-1,j}}$ ← get diff. between adjacent rows
4: **for** each row ∈ $\mathbf{rowDiffMtx_{i-1,j}}$ **do**
5: $\mathbf{rowVarArray_{w-1}}$ ← get variance along rows
6: **for** each column ∈ $\mathbf{colDiffMtx_{i,j-1}}$ **do**
7: $\mathbf{colVarArray_{h-1}}$ ← get variance along columns
8: sort $\mathbf{rowVarArray_{w-1}}$ & $\mathbf{colVarArray_{h-1}}$ on descendant order
9: $\mathbf{HTB_m}$ ← $\mathbf{rowVarArray_{w-1}}$
10: **for** each *idx* ∈ $\mathbf{colVarArray_{h-1}}$ **do**
10: **if** size(*idx*) ≥ 256x64 **then**
11: $\mathbf{VTB_n}$ ← $\mathbf{colVarArray_{idx}}$
12: $\mathbf{HTB_{Th}}$ ← pick the first **Th** row indexes ∈ $\mathbf{rowVarArray_{w-1}}$
13: define horizontal tiles's size
14: $\mathbf{VTB_{Tv}}$ ← pick the first **Tv** column indexes ∈ $\mathbf{colVarArray_{h-1}}$
15: define vertical tiles's size

Este algoritmo é aplicado para definir os índices das bordas dos *tiles* para cada um dos quadros de um vídeo. O número total de tiles por quadro não varia, mas o tamanho e a localização de cada *tile* é adaptada de acordo com o mapa de particionamento que serve como parâmetro de entrada para o algoritmo.

A.8. Esquema de Escalonamento de *Tiles*

É esperado que *tiles* maiores requeiram maior esforço computacional para serem codificados, enquanto *tiles* menores tendem a necessitar menor esforço. Levando isso em consideração, o algoritmo *AdapTa* pode definir partições *tile* não balanceadas, pois os *tiles* não uniformes podem assumir diferentes tamanhos. Visando reverter o problema causado pelo desbalanceamento de carga causado pela definição de *tiles* não uniforme, também é proposto um esquema de escalonamento de *tiles.*

Quando a predição intra-quadros é aplicada, a definição de *tiles* desbalanceados em um quadro não compromete o potencial de paralelismo. Afinal, *tiles* de diferentes quadros podem ser escalonados ao longo do tempo para que se obtenha um balanceamento de carga adequado ao final da codificação. Isto só é possível, porque a predição intra-quadro não apresenta dependências entre quadros vizinhos. Porém, quando a predição inter-quadros é aplicada essas dependências existem, impossibilitando que *tiles* de diferentes quadros sejam escalonados ao longo do tempo. Por isso, foi proposto um esquema de escalonamento de *tiles* para aumentar o potencial de paralelismo de quadros que utilizam predição inter-quadros.

Este esquema consiste em analisar a área de cada *tile,* visando escalonar cargas similares (baseadas na área de cada partição) entre as unidades de processamento disponíveis. O esquema proposto está descrito no Algoritmo A.8.1, e é aplicado após a definição de *tiles* realizada pelo algoritmo *AdapTa.*

Algoritmo A.8.1 – Tile Scheduling Scheme

1: given **P** processing units and **T** tiles
2: let **Tw** be the tile width and let **Th** be the tile height
3: **for** each $t \in$ **T do**
4: calculate t_i area ← $\mathbf{Tw_i}$ * $\mathbf{Th_i}$
5: sort **T** according to area in descendant order
6: **for** each $t \in$ **T do**
7: schedule t_i to *min(***P***)* //processing unit with less area based workload

O primeiro estágio do esquema consiste em calcular a área em termos de CTUs de cada um dos *tiles* do quadro (linhas 3-4 do Algoritmo A.8.1). Em seguida, os *tiles* são organizados em ordem decrescente (linha 5) de acordo com a área ocupada. Por fim, o último estágio consiste em escalonar os *tiles* em ordem decrescente para as unidades de processamento que possuem menor carga acumulada (linhas 6-7).

A.9. Resultados e Configuração dos Experimentos

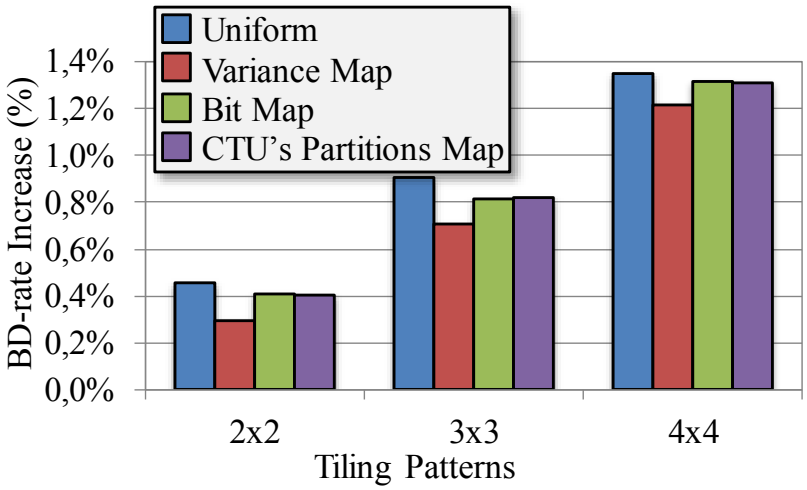
Os experimentos foram realizados utilizando o *software* de referência HEVC *Test Model* (HM) 9.2, seguindo as Condições Comuns de Teste (BOSSEN, 2012). Foram analisadas um total de nove sequências, quatro de vídeos Classe A (*2560x1600 pixels)* e cinco de vídeos Classe B (*1920x1080 pixels).*

Para serem utilizados como referência, todas as sequências foram primeiramente codificadas utilizando um único *tile* por quadro (padrão *1x1*) de acordo com as configurações de codificação *All Intra* (AI) e *Low Delay* (LD). Foram adotados outros três padrões de *tiles: 2x2, 3x3* e *4x4,* totalizando quatro, nove e dezesseis partições por quadro, respectivamente. Estes padrões foram codificados utilizando tanto o algoritmo *AdapTa,* quanto o particionamento uniforme. Comparando estas duas soluções é possível relacionar particionamentos que consideram o conteúdo dentro de cada *tile* contra particionamentos que consideram apenas o tamanho dos *tiles.* Todos os resultados que consideram eficiência de codificação foram medidos em termos de aumento percentual do BD-rate (BJONTEGAARD, 2001) com relação ao particionamento utilizando *1x1 tiles.*

A.9.1. Resultados para Configuração *All Intra*

Os resultados referentes à média das cinco sequências de vídeo por particionamento *tile* estão mostrados na Figura A.9.1. Pode ser percebido que independentemente do mapa de particionamento utilizado como parâmetro de entrada por nosso algoritmo, os *tiles* definidos pelo *AdapTa* se mostraram mais eficientes dos que os *tiles* uniformes.

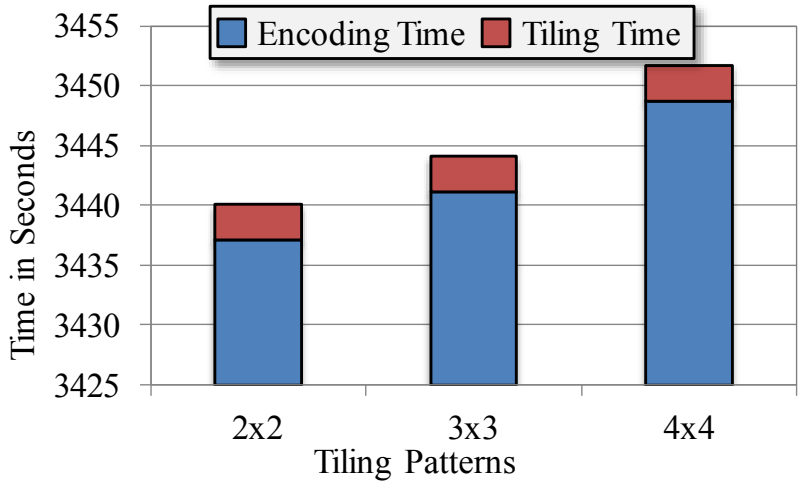
Figura A.9.1 – Aumento médio no BD-rate causada por cada padrão *tile* usando configuração AI em vídeos Classe B.



O mapa de variância foi responsável por definir os *tiles* mais eficientes dentre todos os casos de teste. Quando comparado com o particionamento uniforme, o algoritmo *AdapTa* utilizando o mapa de variância foi capaz de salvar 0.162%, 0.199% e 0.134% de BD-rate para os particionamentos *2x2, 3x3* e *4x4,* respectivamente. Embora não tenham obtido resultados tão bons, os *tiles* definidos pelo mapa de *bits* e o mapa de partições por CTU foram capazes de reduzir 0.08% do BD-rate quando comparados com o particionamento de *tiles* uniforme.

Embora o mapa de variância tenha definido os *tiles* mais eficientes, este mapa deve ser adquirido previamente à codificação de cada quadro. Por isso, este passo para aquisição do mapa acarreta em um tempo extra de codificação. Apesar disso, como é mostrado na Figura A.9.2, o tempo necessário para aquisição desse mapa pode ser considerado desprezível, pois ele representa na média menos de 0.1% do tempo total de codificação.

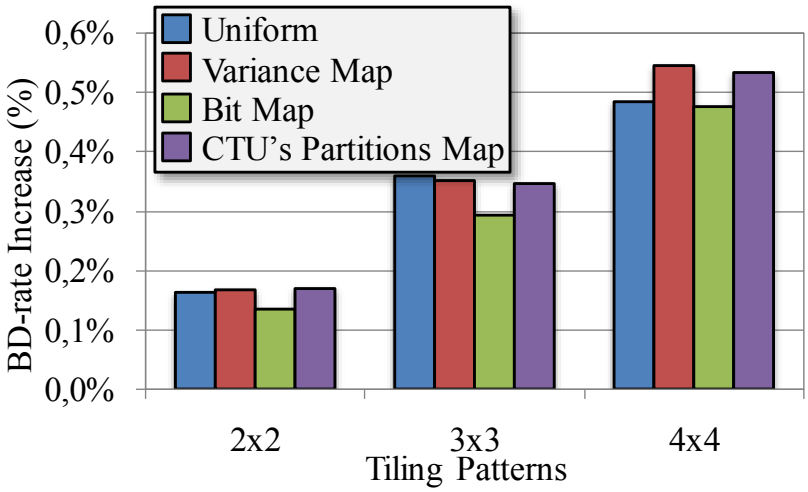
Figura A.9.2 – Atraso para aquisição do mapa de variância em vídeos Classe B.



Também é apresentado, na Figura A.9.3, a média com relação aos experimentos considerando a configuração de codificação *All Intra* e vídeos Classe A. Ao contrário do que aconteceu nos experimentos com vídeo Classe B, para vídeos Classe A os *tiles* mais eficientes foram definidos quanto o mapa de *bits* foi utilizado, o qual foi o único

que obteve melhores resultados quando comparado com o particionamento uniforme independentemente do padrão *tile* utilizado.

Figura A.9.3 – Aumento médio no BD-rate causada por cada padrão *tile* usando configuração AI em vídeos Classe A.



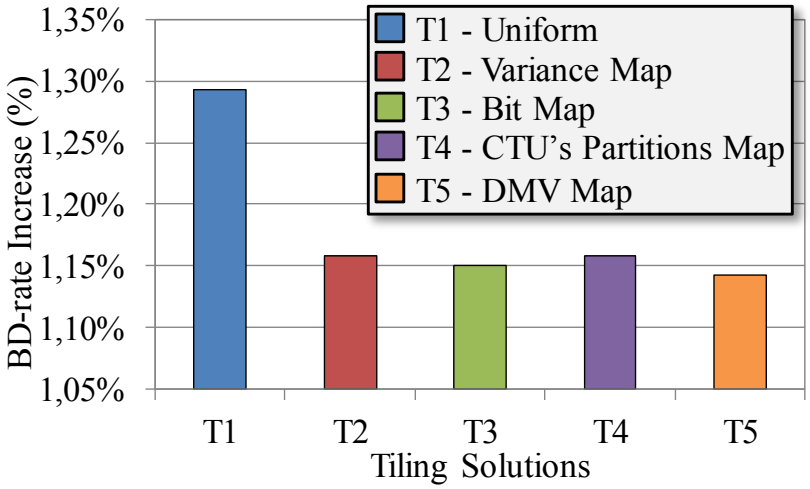
O aumento no BD-rate causado pelos *tiles* definidos pelo algoritmo proposto utilizando o mapa de *bits* foi 0.028%, 0.066% e 0.008% menor do que o aumento causado pelos *tiles* uniformes utilizando os padrões *2x2, 3x3* e *4x4,* respectivamente. Embora o impacto causado tenha sido inferior, estes ganhos são considerados limitados. Isso acontece, pois os vídeo Classe A considerados possuem poucas regiões altamente correlacionadas, o que dificulta a definição de *tiles* adequados.

Os experimentos mostram que é possível definir *tiles* mais eficientes se o conteúdo do quadro e as informações de codificação são consideradas. Analisando estas informações, o algoritmo proposto é capaz de encontrar as quebras naturais de contexto dos quadros. Assim, definindo as bordas dos *tiles* nessa regiões leva à definição de *tiles* mais eficientes, pois as quebras causadas por suas bordas coincidem com quebras de contexto naturais do quadro. Considerando vídeos Classe B, a redução no impacto do BD-rate foi superior a 0.4% no melhor caso.

A.9.1. Resultados para Configuração *Low Delay*

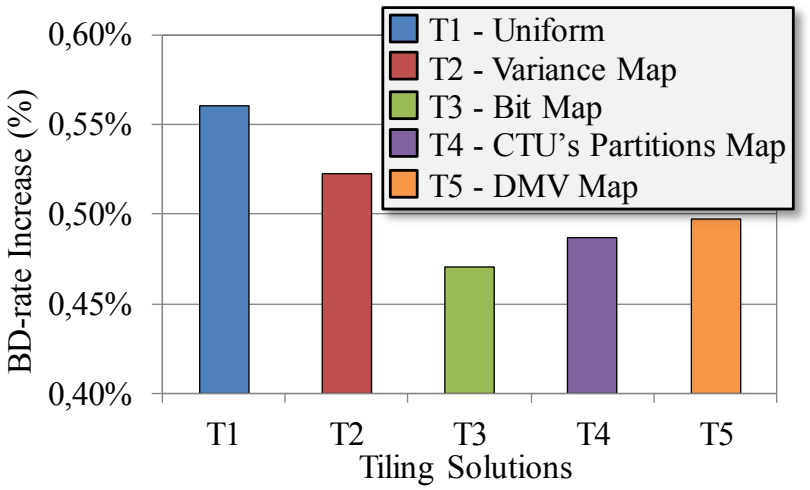
Além de considerar os três mapas de particionamento considerados nos experimentos para configuração *All Intra,* na configuração *Low Delay* também foi utilizado o mapa de particionamento de vetores de movimento diferenciais. Como este mapa só pode ser obtido depois que dois quadros já tenham sido codificados, os particionamentos *tile* para os dois primeiros quadros foi definido utilizando como parâmetro de entrada o mapa de variância. Os experimentos referentes à média acumulada de todos os tipos de particionamento (*2x2, 3x3* e *4x4*) em vídeos Classe B são apresentados na Figura A.9.4.

Figura A.9.4 – Resultados médios acumulados para vídeos Classe B.



Pode ser percebido que todos os mapas de particionamento considerados como parâmetro de entrada para o algoritmo proposto definiram *tiles* mais eficientes do que os uniformes. Comparando o *AdapTa* com o particionamento uniforme, as reduções no BD-rate variaram entre 0.135% e 0.150%. O mapa de vetores de movimento diferenciais foi o mapa que definiu as partições mais eficientes, causando um aumento de 1.143% no BD-rate, enquanto o particionamento uniforme aumentou o BD-rate em 1.293% (0.150% de redução no BD-rate) quando comparado com o particionamento utilizando *1x1 tiles*. Analisando os experimentos considerando vídeos Classe A, apresentados na Figura A.9.5, percebe-se que o algoritmo proposto também é capaz de atingir melhores resultados quando comparado ao particionamento uniforme.

Figura A.9.5 – Resultados médios acumulados para vídeos Classe A.

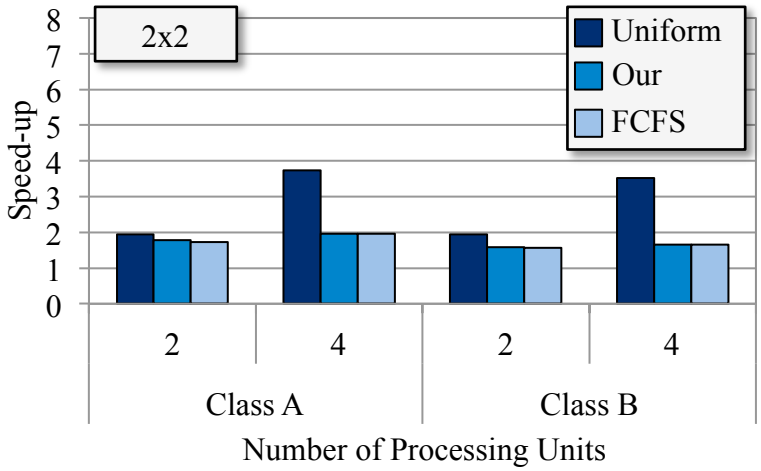


Quando vídeos Classe A são considerados, as reduções no BD-rate variaram entre 0.038% e 0.089%. Essas reduções podem ser consideradas limitadas quando comparadas com as reduções obtidas em vídeos Classe B. Isso acontece, pois os vídeos Classe A não apresentam muitas regiões altamente correlacionadas, o que dificulta a definição de *tiles* mais eficientes. Este mesmo efeito também pôde ser percebido na configuração *All Intra,* onde as reduções também foram limitadas.

A.9.3. Resultados de Paralelismo

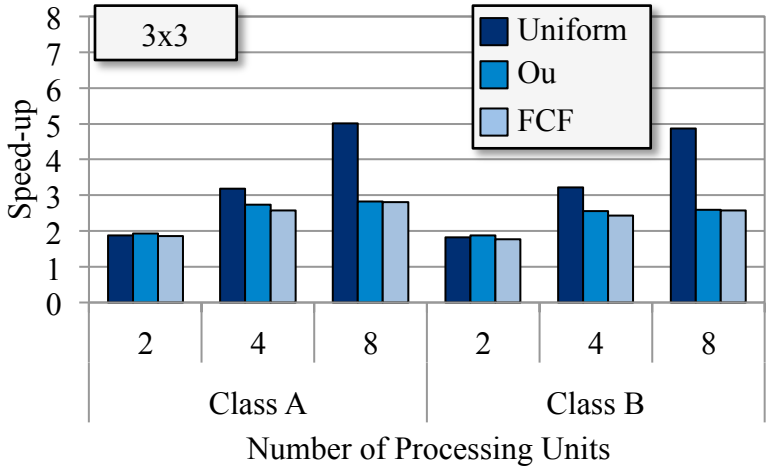
Como pode ser visto, a definição de *tiles* não uniformes que consideram o conteúdo do vídeo para serem definidos levam a partições mais eficientes em termos de eficiência de codificação. Apesar disso, os *tiles* definidos pelo algoritmo proposto não garantem o balanceamento de carga, pois isso um esquema de escalonamento também foi proposto. Os experimentos referentes a este esquema consideraram os mesmo três padrões de *tiling* já considerados (*2x2, 3x3* e *4x4)*, os quais foram escalonados para duas, quatro e oito unidades de processamento. Para fins de comparação também foi utilizado o escalonamento *First Come First Serve* (FCFS) (FEITELSON e RUDOLPH, 1995), o qual simula o esquema de escalonamento convencional do sistema operacional. Com isto, foram utilizados *tiles* uniformes com o escalonamento FCFS, bem como *tiles* definidos pelo algoritmo *AdapTa* utilizando tanto o escalonamento FCFS, quanto o esquema de escalonamento proposto. Na Figura A.9.6 são apresentados os resultados de *speed-up* referentes ao particionamento utilizando *2x2 tiles.*

Figura A.9.6 – *Speed-up* usando *2x2 tiles.*



Quando são consideradas menos unidades de processamento (2) do que o número total de *tiles* (4), o esquema de escalonamento proposto atinge melhores resultados de *speed-up* para ambas classes de vídeo: em torno de 1.78 e 1.59 vezes mais rápido do que o esquema sequencial para vídeos Classe A e B, respectivamente. Para este número de unidades de processamento, o esquema de escalonamento proposto atinge melhores resultados, inclusive, do que o escalonamento FCFS, o qual é 1.71 e 1.55 vezes mais rápido do que a solução sequencial. Os resultados referentes ao uso de *3x3 tiles* são apresentados na Figura A.9.7.

Figura A.9.7 – *Speed-up* usando *3x3 tiles*.

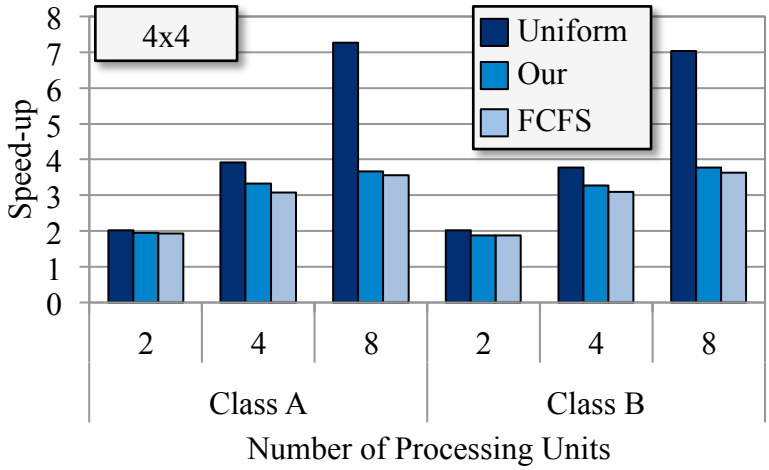


Para um total de nove partições *tile* e considerando duas unidades de processamento, o esquema de escalonamento proposto atinge melhores resultados de *speed-up* do que os atingidos pelo particionamento utilizando *tiles* uniformes. Para vídeo Classe A, o esquema proposto é próximo de 1.92 vezes mais rápido do que o tempo sequencial, enquanto o particionamento uniforme atingiu *speed-up* de 1.87. Para vídeo Classe B, o esquema proposto atinge *speed-up* de 1.88, contra 1.82 atingidos quando *tiles* uniformes são utilizados. Isso acontece, pois *tiles* de mesmo tamanho não garantem que o mesmo esforço computacional será necessário para sua codificação. Em alguns casos, aplicando o algoritmo *AdapTa* podem ser gerados *tiles* com melhor relação área/esforço computacional, e com isso o esquema de escalonamento proposto irá eficientemente balancear a carga entre as unidades de processamento disponíveis.

Apesar disso, quando quatro unidades de processamento são consideradas, o esquema proposto não atinge melhores resultados do que os atingidos por *tiles* uniformes. Porém, quando comparado com o esquema FCFS, o *speed-up* do escalonamento proposto é em média 0.3 maior (considerando as duas classes de vídeo).

Na Figura A.9.8, o escalonamento proposto também atinge boa performance para um número limitado de unidades de processamento. O *speed-up* médio é 1.91 e 3.3 quando são consideradas, respectivamente, duas e quatro unidades de processamento. Embora exista a limitação com relação ao número de unidades de processamento, o esquema proposto é mais eficiente na média (próximo de 1.1 vezes melhor) do que um escalonamento FCFS convencional (considerando todos os cenários de teste).

Figura A.9.8 – *Speed-up* usando *4x4 tiles*.



Para prevenir essa limitação do número de unidades de processamento, é possível combinar duas soluções de ferramentas orientadas ao paralelismo: *tiles* e o *Wavefront Parallel Processing* (WPP). Dessa forma, seria possível explorar o paralelismo no nível da partição *tile* (grão grosso), bem como dentro de cada *tile* através da paralelização de suas linhas de CTUs (grão fino). Apesar disso, essa é uma solução proposta como trabalho futuro.

A.10. Conclusão e Trabalhos Futuros

Este trabalho apresentou um algoritmo adaptativo para definição de partições *tile* que é capaz de reduzir o impacto causado por estas ferramentas na eficiência de codificação. *Tiles* são partições retangulares independentes que podem ser processadas totalmente em paralelo. Estas partições são consideradas independentes, pois quebram as dependências de codificação através de suas bordas, comprometendo a eficiência de codificação.

Para reduzir este impacto inerente ao uso dos *tiles* foi proposto o *AdapTa,* o qual é um algoritmo adaptativo que é capaz de escolher dinamicamente localizações adequadas dos quadros para definir os padrões de *tiling.* O principal objetivo é definir *tiles* mais eficientes, ou seja, que causem menor impacto na eficiência de codificação. Para definir partições *tile* adequadas, o algoritmo procura por regiões altamente correlacionadas do quadro para agrupá-las dentro de um mesmo *tile.* Desta maneira, é possível definir as bordas dos *tiles* em regiões do quadro onde existem quebras naturais de contexto, diminuindo o impacto causado pelas quebras de dependência geradas pelas bordas dos *tiles.* Para localizar as regiões altamente correlacionadas dos quadros, o algoritmo adota mapas de particionamento como argumento de entrada. Foram considerados quatro mapas de particionamento: mapa de variância, mapa de *bits,* mapa de partições de CTUs e mapa de vetores de movimento diferenciais.

Analisando os mapas de particionamento e adotando *tiles* não uniformes para agrupar as regiões altamente correlacionadas, foi possível definir partições *tile* mais eficientes quando comparadas com as partições *tile* uniformes. Experimentos utilizando predição intra-quadros provaram que o algoritmo proposto foi capaz de salvar 0.4% de BD-rate quando comparado com *tiles* uniformes. Quando considerada a predição inter-quadros, a redução em BD-rate atingido pelo algoritmo proposto foi superior a 0.5% quando comparado ao padrão uniforme de partições *tile*. Considerando todos os

cenários de teste, o algoritmo proposto atingiu melhor eficiência de codificação ao utilizar como entrada um dos mapas de particionamento. Visto que o algoritmo se preocupa principalmente em analisar o conteúdo dos quadros para definir *tiles* mais eficientes, em alguns casos o uso de partições não uniformes pode levar a problemas de balanceamento de carga. Para aumentar o potencial de paralelismo dos *tiles* não uniformes definidos pelo algoritmo proposto, também foi proposto um esquema de escalonamento de *tiles.* Este esquema visa escalonar cargas de trabalho similares, baseadas na área de cada *tile,* entre as unidades de processamento disponíveis. Experimentos mostraram que para um número limitado de unidades de processamento nosso esquema de escalonamento de *tiles* atingiu *speed-ups* próximos do ótimo. Em alguns casos o esquema proposto foi ainda mais rápido do que os *tiles* uniformes, os quais são considerados a melhor solução em termos de balanceamento de carga.

Para lidar com o limite de unidades de processamento, é proposto como trabalho futuro adotar o *Wavefront Parallel Processing* (WPP) dentro das partições *tile*. Desta forma, é possível explorar os dois níveis de paralelismo: grão grosso, representado pelas partições *tile,* e grão fino, representado pelas linhas do WPP contidas dentro de cada *tile.* Outra opção para melhorar o balanceamento de carga entre as unidades de processamento seria desenvolver técnicas de predição de esforço computacional baseado em quadros previamente codificados.

Também é proposto investigar a relação entre os mapas de particionamento e o comportamento das sequências de vídeos. Como foi observado nos experimentos, o mapa capaz de definir *tiles* mais eficientes depende do comportamento do vídeo e também de suas características, bem como do padrão de *tiling* adotado. Portanto, é desejável traçar uma relação entre os mapas de particionamento e as sequências de vídeo para definir qual mapa (ou combinação de mapas) é mais adequada para cada vídeo ou quadro.

Por fim, as investigações realizadas mostraram que, mesmo efetivas, as técnicas para definição de *tiles* de forma adaptativa são limitadas pelo formato das partições (o quadro deve ser completamente dividido por bordas verticais/horizontais). Nota-se que, devido a este formato dos *tiles*, frequentemente uma borda que é adequadamente definida em regiões do quadro onde há quebra de contexto natural, causando reduzidos impactos na eficiência de codificação, também acaba por dividir regiões altamente correlacionadas do quadro, o que leva a comprometer a eficiência dos *tiles* definidos. Considerando isso, planeja-se como trabalho futuro avaliar formatos flexíveis de *tiles* para que as regiões altamente correlacionadas dos quadros possam ser melhor exploradas, e que isso não comprometa outras regiões dos quadros. Dessa forma, será possível definir *tiles* mais precisos e que impactem menos a eficiência de codificação.

ANNEX B – DETAILED RESULTS

Table B.1 – BD-rate increase per tiling pattern vs. 1x1 tiles (AI in Class B videos).

Tiling	Video Sequence	Uniform	Variance	Bit Map	CTU Partitions
2x2	**B1**	1.045%	0.697%	0.926%	0.957%
	B2	0.235%	0.165%	0.167%	0.178%
	B3	0.420%	0.145%	0.348%	0.291%
	B4	0.437%	0.370%	0.386%	0.423%
	B5	0.138%	0.090%	0.210%	0.164%
3x3	**B1**	1.835%	1.628%	1.903%	1.950%
	B2	0.542%	0.412%	0.382%	0.407%
	B3	0.610%	0.498%	0.705%	0.613%
	B4	1.083%	0.667%	0.670%	0.732%
	B5	0.452%	0.325%	0.405%	0.392%
4x4	**B1**	2.859%	2.680%	3.051%	2.946%
	B2	0.786%	0.725%	0.636%	0.674%
	B3	1.090%	0.923%	1.039%	0.973%
	B4	1.285%	1.189%	1.183%	1.286%
	B5	0.720%	0.555%	0.663%	0.663%

Table B.2 – Average BD-rate increase per tiling pattern vs. 1x1 tiles (AI in Class B videos).

Tiling	Uniform	Variance	Bit Map	CTU Partitions
2x2	0.455%	0.293%	0.407%	0.403%
3x3	0.905%	0.706%	0.813%	0.819%
4x4	1.348%	1.214%	1.314%	1.308%

Table B.3 – BD-rate increase per tiling pattern vs. 1x1 tiles (AI in Class A videos).

Tiling	Video Sequence	Uniform	Variance	Bit Map	CTU Partitions
2x2	**A1**	0.111%	0.109%	0.109%	0.122%
	A2	0.285%	0.236%	0.182%	0.240%
	A3	0.071%	0.084%	0.085%	0.089%
	A4	0.185%	0.240%	0.163%	0.226%
3x3	**A1**	0.231%	0.218%	0.222%	0.239%
	A2	0.578%	0.492%	0.442%	0.504%
	A3	0.164%	0.206%	0.125%	0.176%
	A4	0.465%	0.490%	0.384%	0.467%
4x4	**A1**	0.338%	0.332%	0.362%	0.373%
	A2	0.774%	0.811%	0.714%	0.780%
	A3	0.231%	0.288%	0.191%	0.261%
	A4	0.595%	0.751%	0.638%	0.721%

Table B.4 – Average BD-rate increase per tiling pattern vs. 1x1 tiles (AI in Class A videos).

Tiling	Uniform	Variance	Bit Map	CTU Partitions
2x2	0.163%	0.167%	0.135%	0.169%
3x3	0.359%	0.352%	0.293%	0.346%
4x4	0.484%	0.545%	0.476%	0.534%

Table B.5 – BD-rate increase per tiling pattern (and average) vs. 1x1 tiles (LD in Class B videos).

Tiling	Video Sequence	Uniform	Variance	DMVs	Bit Map	CTU Partitions
2x2	**B1**	0.798%	0.718%	0.682%	0.747%	0.696%
	B2	0.546%	0.417%	0.346%	0.160%	0.257%
	B3	0.486%	0.248%	0.298%	0.242%	0.221%
	B4	0.883%	0.883%	0.863%	0.692%	0.838%
	B5	0.300%	0.307%	0.273%	0.348%	0.337%
	Average	0.603%	0.515%	0.493%	0.438%	0.470%
3x3	**B1**	1.647%	1.581%	1.475%	1.796%	1.773%
	B2	1.054%	0.887%	0.661%	0.770%	0.693%
	B3	0.625%	0.652%	0.861%	0.697%	0.611%
	B4	1.792%	1.679%	1.556%	1.461%	1.612%
	B5	1.038%	0.814%	0.795%	0.792%	0.835%
	Average	1.231%	1.123%	1.070%	1.103%	1.105%
4x4	**B1**	2.844%	2.635%	2.695%	3.004%	2.979%
	B2	1.826%	1.387%	1.321%	1.329%	1.392%
	B3	1.340%	1.260%	1.488%	1.276%	1.286%
	B4	2.615%	2.504%	2.434%	2.584%	2.524%
	B5	1.594%	1.395%	1.391%	1.355%	1.315%
	Average	2.044%	1.836%	1.866%	1.910%	1.899%

Table B.6 – BD-rate increase per tiling pattern (and average) vs. 1x1 tiles (LD in Class A videos).

Tiling	Video Sequence	Uniform	Variance	DMVs	Bit Map	CTU Partitions
2x2	**A1**	0.199%	0.179%	0.166%	0.157%	0.154%
	A2	0.230%	0.137%	0.151%	0.144%	0.141%
	A3	0.142%	0.311%	0.311%	0.139%	0.256%
	A4	0.309%	0.305%	0.187%	0.244%	0.290%
	Average	0.220%	0.233%	0.204%	0.171%	0.211%
3x3	**A1**	0.407%	0.335%	0.329%	0.297%	0.334%
	A2	0.519%	0.393%	0.427%	0.434%	0.483%
	A3	0.576%	0.670%	0.545%	0.564%	0.474%
	A4	0.826%	0.615%	0.581%	0.642%	0.649%
	Average	0.582%	0.503%	0.470%	0.484%	0.485%
4x4	**A1**	0.610%	0.540%	0.515%	0.513%	0.531%
	A2	0.801%	0.747%	0.757%	0.736%	0.711%
	A3	0.991%	0.976%	0.871%	0.779%	0.813%
	A4	1.114%	1.060%	1.132%	1.003%	1.009%
	Average	0.879%	0.831%	0.818%	0.758%	0.766%