

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

MATEUS GRELLERT DA SILVA

**Computational Effort Analysis and Control
in High Efficiency Video Coding**

Dissertação apresentada como requisito parcial
para a obtenção do grau de Mestre em Ciência
da Computação

Prof. Dr. Sergio Bampi
Orientador

Prof. Dr. Bruno Zatt
Co-orientador

Porto Alegre, fevereiro de 2014.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Grellert, Mateus

Computational Effort Analysis and Control in High Efficiency Video Coding [manuscrito] / Mateus Grellert da Silva – 2014

89 f.:il.

Orientador: Sergio Bampi; Co-orientador: Bruno Zatt

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

1. Computational Effort Analysis. 2. Complexity Control 3. HEVC 4. Video Coding I. Bampi, Sergio. II. Zatt, Bruno. III. Computational Effort Analysis and Control in High Efficiency Video Coding.

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

TABLE OF CONTENTS

TABLE OF CONTENTS.....	3
LIST OF ABBREVIATIONS AND ACRONYMS	6
LIST OF FIGURES.....	8
LIST OF TABLES	9
RESUMO	10
ABSTRACT	11
1. INTRODUCTION.....	12
1.1. Problem formulation and motivation	15
1.2. Work proposal and organization	15
1.3. Methodology	16
2. HIGH EFFICIENCY VIDEO CODING	17
2.1. Prediction in the HEVC Standard	18
2.1.1. Intra-Prediction	20
2.1.2. Inter-Prediction	20
2.2. Transforms and Quantization	21
2.3. Entropy Coding	22
2.4. Rate-Distortion Optimization.....	22
2.5. Final Remarks	23
3. STATE OF THE ART.....	24
3.1. Common Test Conditions and Metrics.....	24
3.2. Computational Effort Analysis	27
3.3. Complexity Reduction.....	28
3.3.1. Intra-Prediction	28
3.3.2. Inter-Prediction	28

3.3.3.	Other HEVC components	29
3.4.	Complexity Control.....	29
3.4.1.	Multilevel Complexity Control.....	30
3.4.1.1.	Frame level control	31
3.4.1.2.	Per frame complexity control.....	32
3.4.1.3.	Results.....	32
3.5.	Final Remarks	33
4.	HEVC COMPUTATIONAL EFFORT ANALYSIS	35
4.1.	HEVC Model Encoding Parameters.....	35
4.2.	Mechanisms for Measuring the Computational Effort.....	36
4.2.1.	Execution Time.....	36
4.2.2.	Application Profiling	36
4.2.3.	Computational Effort Modeling	37
4.3.	The Arithmetic Complexity Metric.....	37
4.4.	Test Conditions	38
4.4.1.	Computational Effort Measurements	38
4.5.	Computational Effort Analysis and Discussions.....	39
4.5.1.	Maximum Throughput at Default Conditions.....	39
4.5.2.	Inter-sequence Complexity Scaling	40
4.5.3.	Inter-Frame and Intra-Frame Complexity Distribution.....	41
4.5.4.	Encoder Sensitivity Analysis	42
4.5.4.1.	Complexity and Bitrate Results.....	42
4.5.4.2.	The Rate-Complexity Efficiency metric	45
4.6.	Final Remarks	46
5.	HEVC COMPUTATIONAL CONTROL	47
5.1.	Control Feedback Loop	47
5.2.	Computational Management of HEVC Encoders	47
5.2.1.	Control Unit	49
5.2.2.	Parameter Sets	50
5.2.2.1.	Parameter Set Analysis	51
5.2.3.	Complexity Estimation	52
5.2.4.	Budgeting Algorithms.....	52
5.2.4.1.	Top-Down Budgeting (TDB).....	53
5.2.4.2.	Bottom-Up Budgeting (BUB).....	53
5.2.4.3.	Priority-based Budgeting (PBB)	53
5.3.	Final Remarks	56
6.	RESULTS AND DISCUSSION	57
6.1.	Computation Management Schemes Comparison	58
6.1.1.	PID-based Controller Analysis	58
6.1.2.	Budgeting Strategies Analysis	60

6.2.	In-Depth Analysis of the Computation Management Scheme.....	62
6.2.1.	Controllability and Adaptability	62
6.2.2.	Related Work Comparison.....	63
6.2.3.	Comparison with HEVC Model reference	64
7.	CONCLUSION AND FUTURE WORK	67
	APPENDIX A SEQUENCES USED IN THE TESTS	68
A.1	Class A sequences	68
A.2	Class B sequences	69
A.3	Class C sequences	69
A.4	Class D sequences	70
A.5	Class F sequences.....	71
	APPENDIX B RESUMO EM PORTUGUÊS	72
B.1	Resumo	72
B.2	Introdução.....	72
B.2.1	Formulação do Problema e Motivação	75
B.2.2	Proposta e Metodologia	75
B.3	Análise de Esforço Computacional na codificação HEVC.....	76
B.3.1	Métricas de Complexidade	76
B.3.2	Configurações de Teste.....	76
B.3.3	Parâmetros Avaliados	77
B.4	Controle Computacional em Codificadores HEVC.....	78
B.4.1	Controlador PID.....	80
B.4.2	Conjuntos de Parâmetros	80
B.4.3	Algoritmo de Budgeting	80
B.4.4	Resultados	82
B.5	Conclusão e Trabalhos Futuros	85
	REFERENCES.....	86

LIST OF ABBREVIATIONS AND ACRONYMS

AC	Arithmetic Complexity
ALF	Adaptive Loop Filter
AMP	Asymmetric Motion Partition
BD	Bjøntegaard Difference
BR	Bit Rate
BUB	Bottom-Up Budgeting
CABAC	Context-Adaptive Binary Arithmetic Coding
CFG-LUT	Configuration Lookup Table
CMS	Computation Management Scheme
CTC	Common Test Conditions
CTU	Coding Tree Unit
CTU-HT	CTU History Table
CU	Coding Unit
DCT	Discrete Cosine Transform
DF	Deblocking Filter
FME	Fractional Motion Estimation
GOP	Group of Pictures
HadME	Hadamard Motion Estimation
HD	High Definition
HEVC	High Efficiency Video Coding
HM	HEVC Model
IME	Integer Motion Estimation
LB	Low Delay with B frames
MaxCTUd	Maximum CTU quadtree Depth
MaxTUd	Maximum TU quadtree Depth
ME	Motion Estimation
MPEG	Moving Picture Experts Group

MSE	Mean Squared Error
PBB	Priority-Based Budgeting
PID	Proportional Integral Derivative
PS	Parameter Set
PSNR	Peak Signal-To-Noise Ratio
PU	Prediction Unit
QP	Quantization Parameter
RA	Random Access
RD	Rate-Distortion
RDO	Rate-Distortion Optimization
RDOQ	RDO Quantization
RMD	Rough Mode Decision
RQT	Residual Quadtree
SAD	Sum of Absolute Differences
SAO	Sample Adaptive Offset
SATD	Sum of Absolute Transformed Differences
SCC	Sistema de Controle de Complexidade
SMP	Symmetric Motion Partition
SP	Set Point
SR	Search Range
SSE	Sum of Squared Differences
TDB	Top-Down Budgeting
TU	Transforms Unit
TZSearch	Test Zone Search
UHD	Ultra-High Definition

LIST OF FIGURES

Figure 1.1: Encoding time among frames and among CTUs in a frame (sequence: BasketballDrill).....	15
Figure 2.1: HEVC encoding loop.....	17
Figure 2.2: (a) A CTU and its correspondent decision quadtree. Each PU partition (b) is evaluated in each node.....	19
Figure 2.3: An arbitrary block B and its transformed/quantized coefficients	22
Figure 3.1: Random Access temporal structure	24
Figure 3.2: Low Delay temporal structure.....	25
Figure 3.3: RD curves for two targets and the differential areas (gray filling) used in the BD-BR (a) and BD-PSNR (b) calculation	26
Figure 3.4: Proposed complexity control scheme (source: (KANNANGARA, RICHARDSON and MILLER, 2008)).....	31
Figure 4.1: Coding complexity comparison of sequences with the same resolution (class A: 2560x1600; class B: 1920x1080)	40
Figure 4.2: Encoding time among frames in the BasketballDrill sequence	41
Figure 4.3: Encoding time spent on each CTU in the frame	41
Figure 4.4: AC savings of each configuration with respect to c0	43
Figure 4.5: Time savings for each coding configuration	43
Figure 4.6: bitrate increase of each configuration with respect to c0	45
Figure 4.7: Rate-Complexity Efficiency results for all configurations.....	46
Figure 5.1: Block diagram of a feedback control loop as per (OGATA, 2013)	47
Figure 5.2: Conceptual model of a Computation Management Scheme	48
Figure 5.3: Average Arithmetic Complexity Savings of each Parameter Set with respect to PS0	51
Figure 5.5: Bottom-Up-Budgeting algorithm.....	54
Figure 5.6: Priority-Based Budgeting Algorithm.....	55
Figure 6.1: Computation results of both controllers. (Sequence: BasketballDrill)	59
Figure 6.2: Uniform Budgeting algorithm used for comparison.....	60
Figure 6.3: Frame-wise PS distribution for each algorithm (sequence: BasketballDrill)	61
Figure 6.4: HM-CMS performance under varying SPs.....	62
Figure 6.5: Encoding cycles when the HM-CMS is switched off and on again	63
Figure 6.6: PSNR/bitrate charts for two sequences:(a) SteamLocomotiveTrain (class A) and (b) ChinaSpeed (class F).....	66
Figura B.1: Tempo de codificação entre quadros (à esquerda) e entre CTUs de um mesmo quadro (à direita) (sequência: BasketballDrill).....	75
Figura B.2: Resultados de RCE para cada configuração (c0 utilizada como base de comparação)	78
Figura B.3: Diagrama do Sistema de Controle de Complexidade utilizado neste trabalho	79
Figura B.4: Algoritmo PBB.....	81
Figura B.5: Análise do controlador PID.....	82
Figura B.6: Análise de adaptabilidade do SCC	82

LIST OF TABLES

<i>Table 1.1: Coding tools defined in H.264/AVC and HEVC</i>	14
<i>Table 2.1: Enabled/Disabled partitions for each prediction mode and each CU</i>	19
<i>Table 3.1: Frame rate, PSNR and bitrate results (source: (KANNANGARA, RICHARDSON and MILLER, 2008))</i>	33
<i>Table 3.2: Summary of the complexity solutions discussed</i>	34
<i>Table 4.1: Test configurations for the computational effort analysis</i>	39
<i>Table 4.2: Estimated maximum FPS at 2 GHz using the hypothetical co-processor (using HM default configurations and AC as complexity metric)</i>	39
<i>Table 4.3: Configurations used in the sensitivity analysis</i>	42
<i>Table 4.4: Difference between time and AC complexity savings</i>	44
<i>Table 5.1: Parameter Sets used in the CTU budgeting</i>	50
<i>Table 5.2: AC, time, bitrate and PSNR variations of each PS with respect to PS0 (default configuration)</i>	51
<i>Table 6.1: Test setup for the CMS analysis</i>	57
<i>Table 6.2: RD and MSE results of both controllers</i>	59
<i>Table 6.3: Comparison results of each budgeting strategy against the HM reference</i>	60
<i>Table 6.4: Overall parameter Set usage for each algorithm (sequence: BasketballDrill)</i>	61
<i>Table 6.5: Comparison with Related Work (Target savings: 40% and 50%)</i>	63
<i>Table 6.6: BR-BR results for all sequences, using both temporal structures (Target savings: 40%)</i>	64
<i>Table 6.7: BR-BR results for all sequences, using both temporal structures Target savings: 50%)</i>	65
<i>Tabela B.1: Ferramentas de codificação suportadas pelos padrões H.264/AVC e HEVC</i>	74
<i>Tabela B.2: Configurações utilizadas na análise de esforço computacional</i>	77
<i>Tabela B.3: Configurações utilizadas nas análises (c0: configuração padrão)</i>	78
<i>Tabela B.4: Conjuntos de Parâmetros (PSs) utilizados no SCC</i>	80
<i>Tabela B.5: Resultados de tempo e de compressão/qualidade de cada PS</i>	80
<i>Tabela B.6: Configurações de teste para as análises do SCC</i>	83
<i>Tabela B.7: Resultados de BR-BR para todas as sequências usando as configurações RA e LB (redução alvo: 40%)</i>	83
<i>Tabela B.8: Resultados de BR-BR para todas as sequências usando as configurações RA e LB (redução alvo: 50%)</i>	84
<i>Tabela B.9: Comparação com trabalhos relacionados (Alvo de redução: 40% 4 50%)</i>	84

RESUMO

Codificadores HEVC impõem diversos desafios em aplicações embarcadas com restrições computacionais, especialmente quando há restrições de processamento em tempo real. Para tornar a codificação de vídeos HEVC factível nessas situações, é proposto neste trabalho um Sistema de Controle de Complexidade (SCC) que se adapta dinamicamente a capacidades computacionais variáveis. Considera-se que o codificador faz parte de um sistema maior, o qual informa suas restrições como disponibilidade da CPU e processamento alvo para o SCC. Para desenvolver um sistema eficiente, uma extensiva análise de complexidade dos principais parâmetros de codificação é realizada. Nessa análise, foi definida uma métrica livre de particularidades da plataforma de simulação, como hierarquia de memória e acesso concorrente à unidade de processamento. Essa métrica foi chamada de Complexidade Aritmética e pode ser facilmente adaptada para diversas plataformas. Os resultados mostram que o SCC proposto atinge ganhos médios de 40% em complexidade com penalidade mínima em eficiência de compressão e qualidade. As análises de adaptabilidade e controlabilidade mostraram que o SCC rapidamente se adapta a diferentes restrições, por exemplo, quando a disponibilidade de recursos computacionais varia dinamicamente enquanto um vídeo é codificado. Comparado com o estado da arte, o SCC atinge uma redução de 44% no tempo de codificação com penalidade de 2.9% na taxa de compressão e acréscimo de 6% em BD-bitrate.

Palavras-Chave: codificação de vídeo, HEVC, controle de complexidade, análise de complexidade, avaliação de complexidade.

Computational Effort Analysis and Control for High Efficiency Video Coding

ABSTRACT

HEVC encoders impose several challenges in resource-/computationally-constrained embedded applications, especially under real-time throughput constraints. To make HEVC encoding feasible in such scenarios, an adaptive Computation Management Scheme (CMS) that dynamically adapts to varying compute capabilities is proposed in this work. It is assumed that the encoder is part of a larger system, which informs to the CMS its restrictions and requirements, like CPU availability and target frame rate. To effectively develop and apply such a scheme, an extensive computational effort analysis of key encoding parameters of the HEVC is carried out. For this analysis, a platform-orthogonal metric called “Arithmetic Complexity” was developed, which can be widely adopted for various computing platforms. The achieved results illustrate that the proposed CMS provides 40% cycle savings on average at the cost of small RD penalties. The adaptability and controllability analyses show that the CMS quickly adapts to different constrained scenarios, e.g., when the executing HEVC encoder requires more or less computation from the underlying platform. Compared to state of the art, the CMS achieves 44% encoding time savings while incurring a minor 2.9% increase in the bitrate and 6% increase in BD-bitrate.

Keywords: video coding, HEVC, complexity control, computational effort analysis, complexity assessment.

1. INTRODUCTION

Digital devices have undergone a non-ceasing evolution chain, in which products such as palm tops are considered obsolete, and smart phones can be regarded as top-technology for no longer than a year. Naturally, this fast-paced technological race has affected the consumer market, establishing an increasing demand for cheaper and more powerful mobile devices, better quality media services, etc.

Digital video applications are typical examples of this current scenario. The old habit of sharing information through text in a webpage is steadily being replaced by video content located in databases such as YouTube. In addition, videoconferences are also becoming more frequent as the communication technology improvements provide larger bandwidths, and stereo or multi-view videos are also increasingly common, mostly due to the recent hype in 3D applications. To quantify this trend, a forecast paper published by Cisco shows that the internet-video bandwidth will go from 57% in 2012 to 69% in 2017 (CISCO, 2012).

The main concern that arises from this fact is that, as the use of digital video increases, so does the need for larger resolutions and higher frame rates. These two parameters, along with the number of views in multi-view sequences, are directly proportional to the bandwidth required to transmit a video, as well as to the memory capacity needed to store them. This can actually be quantified for uncompressed sequences. The equation below shows the bandwidth required to transmit an uncompressed video sequence considering a 4:2:0 color sampling:

$$BW = N_{VIEWS} \times W \times H \times FPS \times B_{depth} \times 1.5 \text{ bits/s} \quad (1)$$

In (1), N_{VIEWS} represents the number of views (one for single-view sequences and 2 or more for multi-view), B_{depth} , the size of a luminance or chrominance sample in bits (usually 8), whereas W , H , and FPS are respectively the width, height and frame rate of the sequence. Note that the 4:2:0 color subsampling is already a mechanism for compressing data, since it defines that for each 4 luminance samples, there exists only 2 chrominance samples (one for each layer). That explains the 1.5 multiplication factor in the formula (if 4:2:2 subsampling was considered, defining one chrominance sample for each luminance one, this factor would be 3).

For an uncompressed Full HD (1920×1080 pixels, also called 1080p/i) sequence captured at 30 frames per second, a bandwidth of 746.5 Mbps is required to transmit this video in real time. This also means that if we want to store a 60-minute Full HD sequence, it would take 336 GB to do so. These requirements are clearly prohibitive with current technology, especially if portable devices with tighter constraints like smart phones and tablets are considered. Therefore, compressing this information using video-coding techniques is imperative to enable the broad utilization of this media.

Video coding can be roughly described as a process that explores frames and regions inside frames (blocks) in search for redundant information that can somehow be compressed by exploiting such redundancies. This naturally generates a stream of bits that must be decoded whenever the video is displayed. There are several video-coding techniques available, each performing the same task through a different manner, so video-coding standards were created in order to allow a common language between encoders and decoders in different platforms.

Current state-of-the-art video encoders generate a bit stream in conformance to the H.264/AVC standard, which had its draft formally approved in 2003 (ITU-T, 2003). Although the market has adopted H.264/AVC as a commercial solution in the last 10 years, many video-coding experts claim that the limited compression rate achieved by this standard will soon become unsatisfactory, especially when the future expectations for media services are considered. In fact, the Motion Picture Experts Group (MPEG) released a report in 2011, stating the factors that justified the need for a new video-coding standard (JCT-VC, 2011) based on the following claims:

- High-definition (HD) devices (displays and cameras) are affordable for consumer usage today, whereas the currently available Internet and broadcast network capacity is not sufficient to transfer a large amount of this content. This is further aggravated if we consider, the next generation of ultra-HD (UHD) content and devices, such as 4K×2K displays for home applications and digital cameras with increased resolution.
- For mobile terminals, quality of video services using resolutions today such as QCIF at low frame rates and low bit rates is largely unacceptable. Anticipating that lightweight HD resolutions such as 720p (1280×720 pixels) or even beyond will be introduced in the mobile sector to provide similar perceptual quality as for the home applications, lack of sufficient data rates as well as the prices to be paid for transmission will remain a problem for the long term.
- The limitations mentioned above are becoming even more severe in real-time services like videoconferencing.
- Demands for video content (when current compression technology is used) are increasing faster than the network infrastructure is able to economically carry, both for wireless and wired networks.

Therefore, a new generation of video compression technology that has sufficiently higher compression capability than the existing AVC standard is needed.

With that in mind, a group of experts from ITU-T and ISO/IEC (named Joint Collaborative Team on Video Coding – JCT-VC) started a joint effort towards the next-generation video-coding standard. The result from this collaboration is the High Efficiency Video Coding (HEVC) standard, which had its first draft released in April 2013 (ITU-T, 2013). The results achieved by HEVC show that it outperforms H.264/AVC in terms of coding efficiency by 23% on average (using the Bjøntegaard Difference metric) (LI, SULLIVAN and XU, 2012), (BJONTEGAARD, 2008).

The HEVC standard employs a block-based hybrid coding architecture, combining motion-compensated prediction and transform coding with high-efficiency entropy coding. However, in contrast to previous video coding standards, it provides a flexible quadtree coding block partitioning structure that enables the use of large and multiple sizes of coding, prediction, and transforms blocks. It also employs improved intra

prediction and coding, adaptive motion parameter prediction and coding, a new loop filter and an enhanced version of context-adaptive binary arithmetic coding (CABAC) entropy coding (KIM, MCCANN, *et al.*, 2013). New high-level structures for parallel processing are also employed. A more detailed discussion on this standard is presented in Chapter 2. As seen in Table 1.1, despite the similarities with its predecessor, the data structures and the techniques defined in HEVC are more sophisticated than those of H/264/AVC.

Table 1.1: Coding tools defined in H.264/AVC and HEVC

	H.264/AVC	HEVC
Data structures	Macroblock, block	CTU, CU, PU, TU
Intra-prediction	9 modes	35 modes
Inter-prediction	4 modes	8 modes
Transforms	2 modes	8 modes (RQT)
Filters	DF	SAO, DF

These innovations are key components that enable HEVC superior coding performance with respect to H.264/AVC, but they also convey an issue that has been subject of research for many years: the computational requirements of real HEVC encoding applications.

In this dissertation, the terms complexity and computational effort will be treated as interchangeable terms, since this has already been established in the video-coding community.

Practical applications introduce many factors that cannot be ignored by complex systems such as video encoders. For instance, if a camera needs to record and instantly transmit a video (such as live transmissions), then real-time encoding is needed, which means that more than twenty frames per second must be encoded. This is further aggravated if the resources used by the encoder are shared among other components of the device, meaning that a variable amount of computation is available at each time interval. Furthermore, if the device is battery-powered, the remaining battery life may be just enough to perform a very low-complexity encoding task.

All these scenarios demand solutions that somehow reduce encoding complexity. One possible way to solve this is to design a low-complexity encoder that reduces a fixed amount of computation compared to the traditional approach. However, this implies that the same process will be carried out even when there are more resources available to achieve a better compression. Alternatively, an encoding system that is capable of adapting itself according to the underlying hardware characteristics is a more promising solution.

A plethora of solutions that tackle video-coding complexity issues can be found in the literature, and the most relevant ones are discussed in Chapter 3. These solutions can be separated into two categories: the ones based in complexity reduction, and the ones targeting complexity control. Complexity reduction in video coding fundamentally consists in replacing a component of the system by one that is faster and does a similar job. This is done by designing new lightweight algorithms and faster heuristics. In contrast, complexity control relies on adjusting the encoding parameters in order to achieve different levels of complexity. This takes advantage of the fact that many encoder components have input parameters that can be dynamically switched, and many

of these parameters directly affect the complexity of the process. Both solutions are valid, but complexity reduction has been exhaustively studied for years, and its lack of adaptability limits its applications, whereas complexity control employs adaptability in its very definition. Additionally, a limited number of works focusing on complexity management are found in the current literature, indicating that significant potential innovations can be sought after. Even the ones already published are oblivious to the characteristics of the underlying platform.

1.1. Problem formulation and motivation

To achieve complexity scalability in a system, complexity control techniques can be used. However, this kind of solution introduces new challenges for video-coding systems. Figure 1.1a shows the encoding time distribution for each frame when encoding the *BasketballDrill* sequence with the HEVC reference implementation (KIM, MCCANN, *et al.*, 2013). Three observations can be made: (1) the amount of computation spent is not evenly distributed among its frames; (2) some frames apparently require much more time than others (represented as peaks in the chart); and (3) based on the gaps present among the curves, it seems QP has also an important influence on encoding time. Furthermore, by observing Figure 1.1b, it becomes clear that time distribution is also uneven inside a frame, showing that some Coding Tree Units (CTUs) require more computations to be encoded than others.

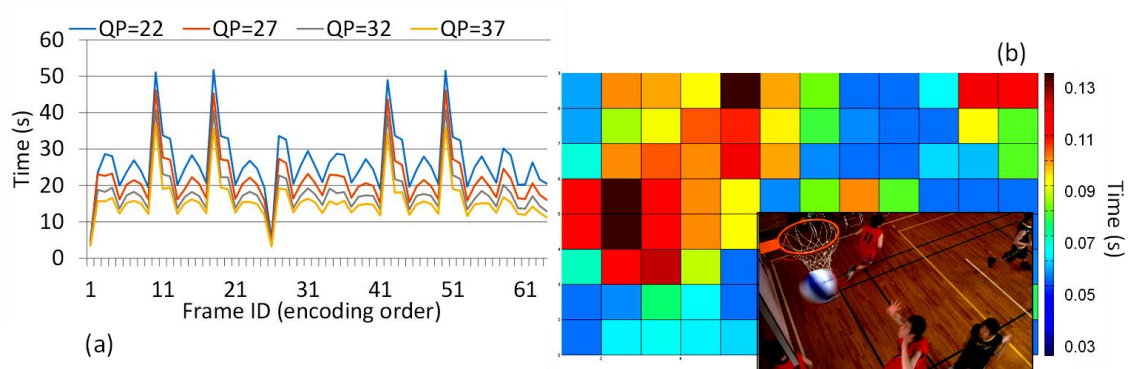


Figure 1.1: Encoding time among frames and among CTUs in a frame (sequence: *BasketballDrill*)

This profiling first indicates that it is important to initially understand and model the HEVC complexity instead of blindly finding solutions that reduce it. Second, designing complexity control schemes involves non-trivial tasks, such as: (1) analyzing and characterizing the sensitivity of the video encoding parameters with respect to their complexity associated with each of them; (2) designing a controller that behaves adequately in an actual encoding system and studying its stability, controllability and observability; (3) discovering the best policy to assign budgets among the frames in a sequence; (4) investigating the best way to distribute the available complexity budget among the CTUs of a frame; and (5) monitoring the actual complexity achieved on-the-fly and reacting promptly when it is straying from the target complexity.

1.2. Work proposal and organization

This work targets the investigation of complexity control techniques for HEVC encoders. In this dissertation two novel contributions can be pointed out:

- Chapter 4 presents a computational effort analysis of key HEVC parameters based on a novel platform-orthogonal metric defined in this work, and;
- A novel adaptive Computation Management Scheme (CMS) for HEVC encoding that dynamically adapts itself to computationally-constrained situations. The CMS and all investigations related to it are presented in Chapter 5.

The proposed CMS was designed considering the existence of an interface to the underlying hardware that informs the CPU status, such as operating frequency and processor availability, as well as the target frame rate required. With this input, a budget for each frame is computed and then distributed among the Coding Tree Units (CTUs).

To explain the context in which this work is inserted, Chapter 2 introduces the HEVC standard and its reference software implementation, and Chapter 3 describes complexity-related solutions found in the literature. The CMS results are discussed in Chapter 6. To conclude, Chapter 7 presents the final remarks and foreshadows the following steps of this research.

1.3.Methodology

The results from the computational effort analysis were used as grounding knowledge to design the components of the Computational Management Scheme later described in Chapter 5. In order to extract results for both the analysis and the CMS, the contributions were all implemented in the HM reference in C++ (KIM, MCCANN, *et al.*, 2013). The test conditions used were the ones stated by the JCT-VC group, as described in Chapter 3.1.

Different versions of the CMS were investigated, using different budgeting strategies and controllers. To elect the best solution, bitrate savings and control efficiency were compared. The elected CMS was then tested in terms of its controllability and adaptability, as well as against related work. Encoding time and compression efficiency comparisons with the HEVC Model were also performed.

2. HIGH EFFICIENCY VIDEO CODING

Despite its many innovations, which will be duly explained in this chapter, HEVC encoding shares many similarities with H.264/AVC. Prior to encoding, a sequence is separated into Groups of Pictures (GOPs), which are formed by one or more frames. Following, each frame undergoes the encoding loop, which is depicted in Figure 2.1.

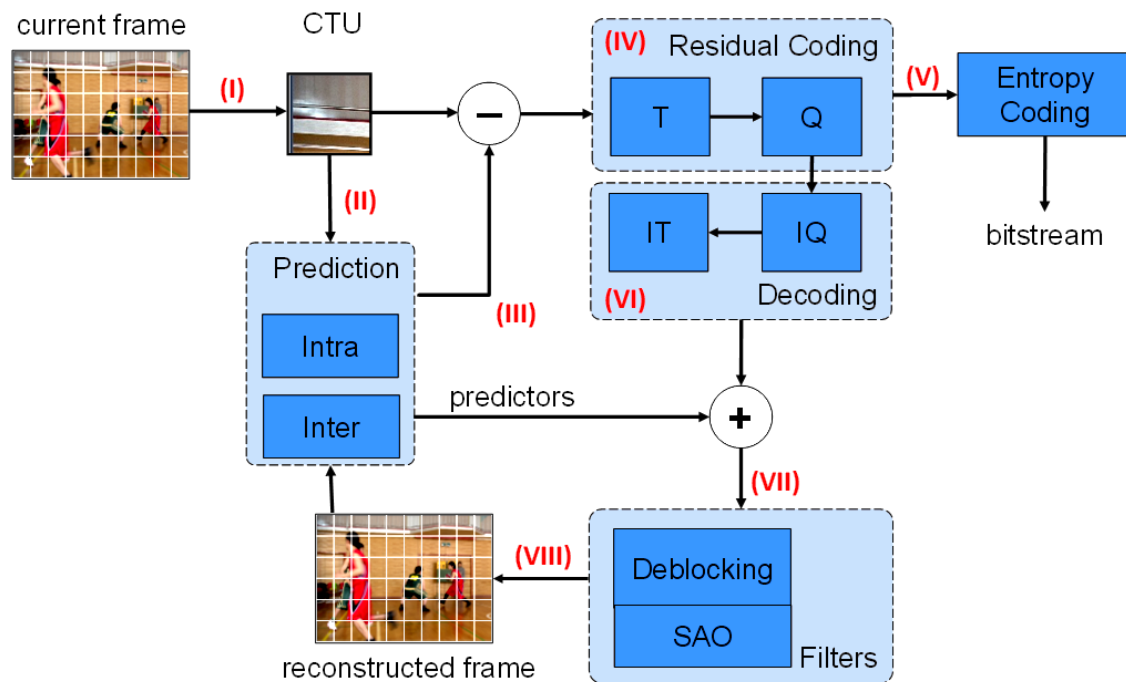


Figure 2.1: HEVC encoding loop

The process can be roughly described as follows: (I) retrieve the next CTU from the input frame, which is sent to the (II) prediction stage in which its redundancies are exploited. (III) The predicted block is then subtracted from the current CTU, producing a residue (this can be interpreted as an error). (IV) The residual information serves as input for the transforms and quantization stage, in which the least relevant information for our visual system is discarded. (V) The quantized output is sent to the entropy encoder, generating the final bit stream that can be either stored or transmitted. Following, (VI) the quantization data are also sent to a decoding stage, after which the residue is reconstructed and added to the predicted samples yet again. (VII) The resulting block is then filtered to remove visual artifacts introduced by the encoding process (due to its block-based operation). Finally, (VIII) this information is stored as a reconstructed frame, which will be used as a reference in the next encoding iterations.

HEVC introduced more complex data structures. This standard defines that each CTU can be encoded in a variety of distinct ways with the possibility to iteratively subdivide themselves into four smaller Coding Units (CUs) to allow more flexibility, forming a quadtree structure. Each CU can be further subdivided into Prediction Units (PUs) during prediction and into Transforms Units (TUs) at the transforms stage. This flexibility was designed to achieve the best possible relation between the compression efficiency and quality, represented by a metric called Rate-Distortion cost. Each new subdivision introduces significant amount of computations to determine the best encoding mode.

Aside from the CTU structure and its subdivisions, other novel techniques defined in HEVC also contribute to its coding efficiency. These tools are all implemented in the HEVC Model (HM) (KIM, MCCANN, *et al.*, 2013), a reference software developed by JCT-VC experts. The HM implements every tool and technique supported by HEVC (although not all of them are enabled under default configurations), so it is widely adopted as a testbed in the video coding community. The next sections describe the most important tools and coding modes supported in this standard. Following, the algorithm used to decide the best mode among the many possibilities available is explained.

2.1. Prediction in the HEVC Standard

The current HEVC version defines that a frame must be subdivided into CTUs, which are broadly analogous to Macroblocks in the H.264/AVC standard (BROSS, HAN, *et al.*, 2011). The largest CTU size currently supported is 64×64 luminance samples, although this may increase when Ultra HD applications hit the market. As with Macroblocks, the chrominance samples are also part of the CTU. This size is fixed throughout and must be set prior to encoding a sequence.

To provide a more flexible coding structure, each CTU can be divided into Coding Units. The reason behind this is to separate distinct regions of a frame with similar redundancies that can be exploited by the prediction process. The process occurs recursively, so a 64×64 CTU is divided into four 32×32 CTUs, and each of these can be subdivided into four 16×16 CTUs. This is repeated until the smallest CTU size is reached (currently defined as 8×8), or when a terminating condition is satisfied (since the HM reference implements heuristics in its mode decision), implicitly forming a quadtree. Figure 2.2 (a) shows an example of a quadtree partitioning.

The following sections describe the intra- and inter-prediction processes, granting more focus to the latter due to its dominant share in the overall computational effort.

2.1.1. Intra-Prediction

This prediction mode is responsible for compressing information by exploiting redundant information in a frame, such as homogeneous regions or repeating patterns. This type of prediction is used in notorious image compressing standards, such as JPEG (PENNEBAKER and MITCHELL, 1993). The abstract notion behind the inter-prediction is simple: use only the borders of a block to predict the information of its remaining pixels.

HEVC defines 35 ways to predict the information. 33 of them, called directional modes, are useful on repeating patterns, such as diagonal or straight lines. The other two are called DC mode, which predicts that the average sample is repeated throughout the block; and planar mode, that repeats information from more than one border, useful in gradient patterns. For 64×64 PUs, only 5 modes are evaluated, and on 4×4 PUs, 17 modes are considered.

In order to minimize computations, a heuristic called Rough Mode Decision (RMD) is implemented in the HM software. In RMD, only a subset of the 33 directional modes available is evaluated, instead of testing all of them. After the best direction is selected, a refinement step is applied, testing the neighboring directions.

2.1.2. Inter-Prediction

The inter-prediction is based on exploiting redundant information among two or more frames in a sequence. High frame rates and absence of movement are two typical situations that raise the odds of achieving good compression through inter-prediction.

During inter-prediction, HEVC defines that a PU can be coded under five modes: inter, merge, SKIP, Pulse Code Modulation (PCM), and intra. The first four will be described in the following paragraphs.

- **Inter:** when a PU is inter-coded, its motion information is derived from the Motion Estimation (ME) process. The ME implements a block-matching algorithm that follows a given search pattern. The main goal is to find in the reference frame (a previously encoded frame) the most similar block to the one being encoded, and reuse this information to provide compression. Different ME algorithms were proposed throughout the years, each with a different search pattern or different starting point. The starting point for the search can be, for instance, the collocated position in the reference frame, or the same as motion vector of the surrounding PUs. Two ME algorithms are implemented in the HM software: Full Search and Test Zone Search (TZSearch), the latter being a fast algorithm that reduces ME complexity with minimum RD penalties (SCHWARZ, HINZ and SUEHRING, 2010).
- **SKIP:** This mode significantly contributes to compression gains, because no movement or residual information is transmitted (since the encoder infers there was no movement at all). In this mode, the PU partition is invariably square (8×8 up to 64×64 , represented always as a $2N \times 2N$ PU).
- **Merge:** consists in deriving the motion parameters for a PU based on the information obtained from spatial and temporal neighbors. This provides significant compression gains, because only the merge flag and the PU index are to

the bitstream; the motion information is inferred from the pointed PU. The merge mode is used for SKIP-coded PUs, but can also be used for inter-coded ones.

- **PCM:** this mode is a rudimentary coding technique that consists in simply sending the raw pixels to the bitstream. This mode was already available for intra-coded Macroblocks in H.264/AVC, but it can also be used in inter-coded PUs in HEVC. Nevertheless, PCM is disabled during inter-prediction in the default HM implementation.

The ME process, performed in inter-coded PUs, is responsible for most of the encoding computational effort. This process is divided into Integer ME (IME) and Fractional ME (FME). During IME, the block-matching algorithm uses one or more reference frames to look for the most similar match. The same process is applied in the FME, but blocks with fractional pixels are also computed. Since the fractional pixels are not originally represented in the sequence, they must be interpolated using 4-, 7- or 8-tap filters, further increasing the computational effort of this process.

The FME introduces motion vectors that represent movements smaller than a pixel, which is possible especially with high frame rates. In HEVC, half-pixel, quarter-pixel and 1/8-pixel (chrominance pixels only) precisions are supported. Luminance half-pixels are interpolated using an 8-tap filter, and quarter-pixels with a 7-tap filter. Chrominance pixels are always interpolated with 4-tap filters. As opposed to H.264/AVC, the fractional part of ME in the HM software cannot be disabled by any encoding parameter, so fractional vectors are always expected in the bitstream.

As pointed in (VANNE, VIITANEN, *et al.*, 2012), IME and FME share 17% and 54% of the overall coding computational effort, adding up to 71%. This shows that the ME computations pose a serious concern, so it must be prioritized in complexity solutions that aim to achieve significant reduction.

2.2. Transforms and Quantization

The residue produced between the current CTU and the one predicted is sent to a process that can be referred to as residual coding. During this process, the transforms, quantization, and their inversed counterparts are applied, thus it is also called T/Q/IQ/IT loop.

The transforms are responsible for translating pixel values to the frequency domain. This process also compacts the energy in the upper-left side of the block, which is useful for quantization and entropy that follow it. Like in prediction, the HEVC transforms also have different modes. The data structure used in this step is called TU, and it can assume sizes of 4×4 up to 32×32 samples, introducing a mode decision tree, called Residual Quadtree (RQT), that is similar to the CTU one. Note that one RQT is nested in each CU of the CTU quadtree, aggravating the computational costs of the latter. In the HM implementation, the transform is performed by partial butterfly structure for low computational complexity (KIM, MCCANN, *et al.*, 2013).

Next in the residual coding process is the quantization. With the transformed coefficients of a block given as input, it is possible to use Human Visual System (HVS) theories to discard the frequencies that are not relevant to our vision. The threshold used to decide the frequency range discarded is calculated based on a very important parameter called Quantization Parameter (QP). Higher QP values increase the frequencies discarded, causing significant compression gains (the explanation for this is explained

in the next section), but this also increases the quality losses, since frequencies that are actually relevant to our vision get discarded as well. The opposite occurs for lower QPs. In the HEVC Model, a technique called Rate-Distortion Optimized Quantization (RDOQ) is employed. The basic concept behind the RDOQ is to use different QP values for each coefficient, given both its impact on the bitrate and quality. To estimate the impact of a coefficient in the number of bits, the tabulated values stored in CABAC (employed in the entropy coding) are used (KIM, MCCANN, *et al.*, 2013).

2.3. Entropy Coding

This is the component in which the compressed bitstream is generated. The entropy coding reduces statistical redundancies by applying data compression techniques similar to the ones used in famous algorithms such as ZIP. The entropy of a data set (in this context, a block of pixels) is measured based on how predictable each data point (or pixel) is. This explains why transforms and quantization are so important for compressing visual information. The transforms energy-compaction property combined with the quantization discarding produce low entropy regions in the lower-right side of a Macroblock. This is better illustrated in Figure 2.3, which shows a block B operated in a Discrete Cosine Transform (DCT) and then quantized.

$$\begin{array}{ccc}
 \mathbf{B} = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} & \mathbf{Q(T(B))} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \\
 \Downarrow \text{Transforms} & \Downarrow \text{Quantization} & \\
 \mathbf{T(B)} = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} & &
 \end{array}$$

Figure 2.3: An arbitrary block B and its transformed/quantized coefficients

In HEVC, a lossless compression technique based on arithmetic coding named CABAC is used. This algorithm implements efficient compression based on statistical analysis, which is updated online, grating adaptability to the video characteristics.

2.4. Rate-Distortion Optimization

Based on what was explained so far, it is possible to conclude that there are many combinations possible to encode a single CTU: different prediction modes, partition sizes, transforms sizes, quantization parameters etc. In order to decide which one is the best, a decision called Rate-Distortion Optimization (RDO) is used.

As the name suggests, this decision is based on optimizing the Rate-Distortion Cost, which is calculated as follows:

$$RD_{cost} = Distortion + \lambda_{mode} * B_{mode} \quad (2)$$

In (2), λ and B are respectively a Lagrange multiplier and the cost in bits of a particular mode (e.g., the bit cost to encode a 64×64 CU using SKIP mode). The Lagrange multipliers for each mode are hardcoded and defined in the standard (KIM, MCCANN, *et al.*, 2013). The distortion corresponds to the SAD (Sum of Absolute Differences), SATD (Sum of Absolute Transformed Differences), or SSE (Sum of Squared Error) result using the residual block (current block minus predicted block).

Note that, to obtain the value of B_{mode} , it is necessary to go through the transforms and quantization steps. This introduces a great computation overhead, especially during prediction, because the RD cost of each CU must be calculated during the CTU quadtree decision, so transforms and quantization are called many times just to decide the best prediction mode. The same occurs during RQT decision, in which entropy must also be performed to decide the best TU size. Therefore, reducing the number of nodes in both, CTU and RQT quadrees is an important way of reducing the overall complexity.

2.5. Final Remarks

The HEVC standard defined many novel techniques, but the excessive number of configurations supported by this standard cause many computations to be performed. If complexity-constrained applications are considered, it is important to find alternative solutions that reduce the computational effort of this process. The number of modes tested are clearly one of the aspects that could be reduced, since only one is effectively used to encode the video.

All the tools and techniques implemented in this standard must work for different applications, from broadcasting to video-surveillance, each with particular requirements. With that in mind, most of these tools were designed to work under several operation modes, which are assigned with encoding parameters. It is possible exploit this characteristic to provide complexity scalability to encoders. For instance, it is possible to change the ME search range, reducing the number of blocks evaluated during this process. With fewer computations, the time to encode the sequence will naturally be shorter, as shown in Chapter 4.

Hence, instead of designing new algorithms for this standard, which could cause decoding incompatibility, the idea of using the parameters to scale computational effort up or down depending on the application requirements sounds a more promising approach. In addition, as reported in the state-of-the-art survey presented in the following chapter, this kind of solution is at present poorly explored in the literature, so there are many venues open to new research in this field. This work sets out to make contributions in that direction.

3. STATE OF THE ART

This chapter aims to initially explain the common methodology and metrics employed in most video coding complexity works. Following, the most relevant publications targeting complexity are presented and discussed.

3.1.Common Test Conditions and Metrics

Video encoders have several applications, from broadcasting to videoconferencing on smartphones, thus video-coding standards must provide a wide range of configuration sets in order to support them all. Nonetheless, this can become a problem when it is necessary to compare two different implementations, since it is important that the same configuration was used in both cases to maintain fair comparisons.

With that in mind, the JCT-VC made an effort on creating a document that describes the Common Test Conditions (CTC) for the HEVC Model reference (BOSSEN, 2011). This document establishes several conditions, for instance how the reference frames should be distributed in a GOP. Two distributions are defined: Random Access (RA) and Low Delay (LB, the B stands for B frames). The RA and LB distributions, often referred to as temporal structures, are depicted in Figure 3.1 and Figure 3.2.

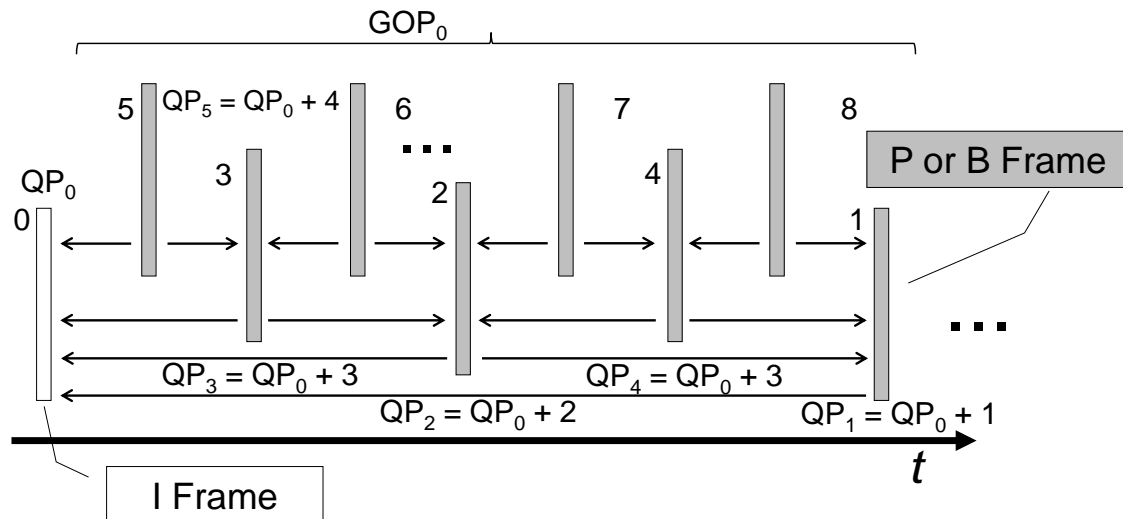


Figure 3.1: Random Access temporal structure

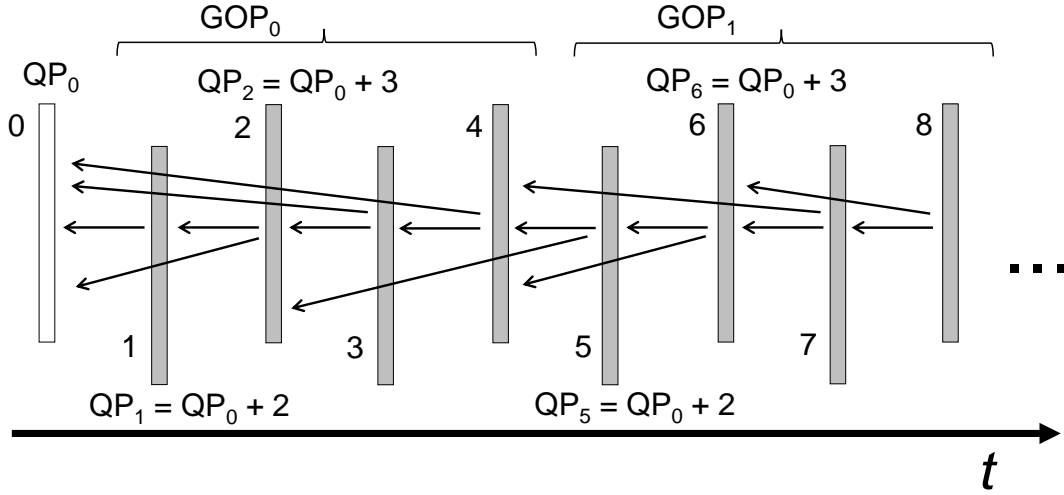


Figure 3.2: Low Delay temporal structure

The CTC document also defines six classes (from A to F) of sequences that should be used when running tests on HEVC. For a list of all the sequences used in this work, consult . They are classified according to their resolution and target application. For instance, classes A to D refer to broadcasting applications, whereas class F targets user-generated content, such as videoconferences and gaming.

The metrics employed to measure encoding efficiency are also discussed in the CTC. This document encourages the use of metrics based on the Bjøntegaard Difference (BD) (BJONTEGAARD, 2008). These metrics are called BD-bit rate (BD-BR), measured in percentage, and BD-Peak Signal-to-Noise Ratio (BD-PSNR), measured in dB. To calculate them, the following steps must be executed (using the HM software):

1. The sequences are encoded under two target configurations: one considered as reference, the other being the object of comparison. For each target, four QP values (22, 27, 32, and 37, as established in the CTC) are used, producing eight bit rate/PSNR pairs (four from each target).
2. A third-degree interpolating function, that uses the four pairs of each target as input, is then used to generate two Rate-Distortion (RD) curves, which will be referred to as REF and TEST.
3. The differential area between both curves is integrated, using the X-axis as reference for BD-BR and the Y-axis for BD-PSNR. The final value is obtained from dividing the integral by its interval. For understanding purposes, the BR-BR formula is displayed in the equation below:

$$BD_{BR} = \frac{\int_a^b (REF_{PSNR}(x) - TEST_{PSNR}(x)) dx}{b - a} \quad (3)$$

In (3), $REF_{PSNR}(x)$ and $TEST_{PSNR}(x)$ respectively represent the PSNR values on the REF and TEST curves, whereas a and b are the second minimum and second maximum PSNR values from both curves. The BD-PSNR formula is analogous, but the bit rate values are used instead. This is better illustrated in Figure 3.3 (consider each mark as the results for QPs 22, 27, 32 and 37, from right to left).

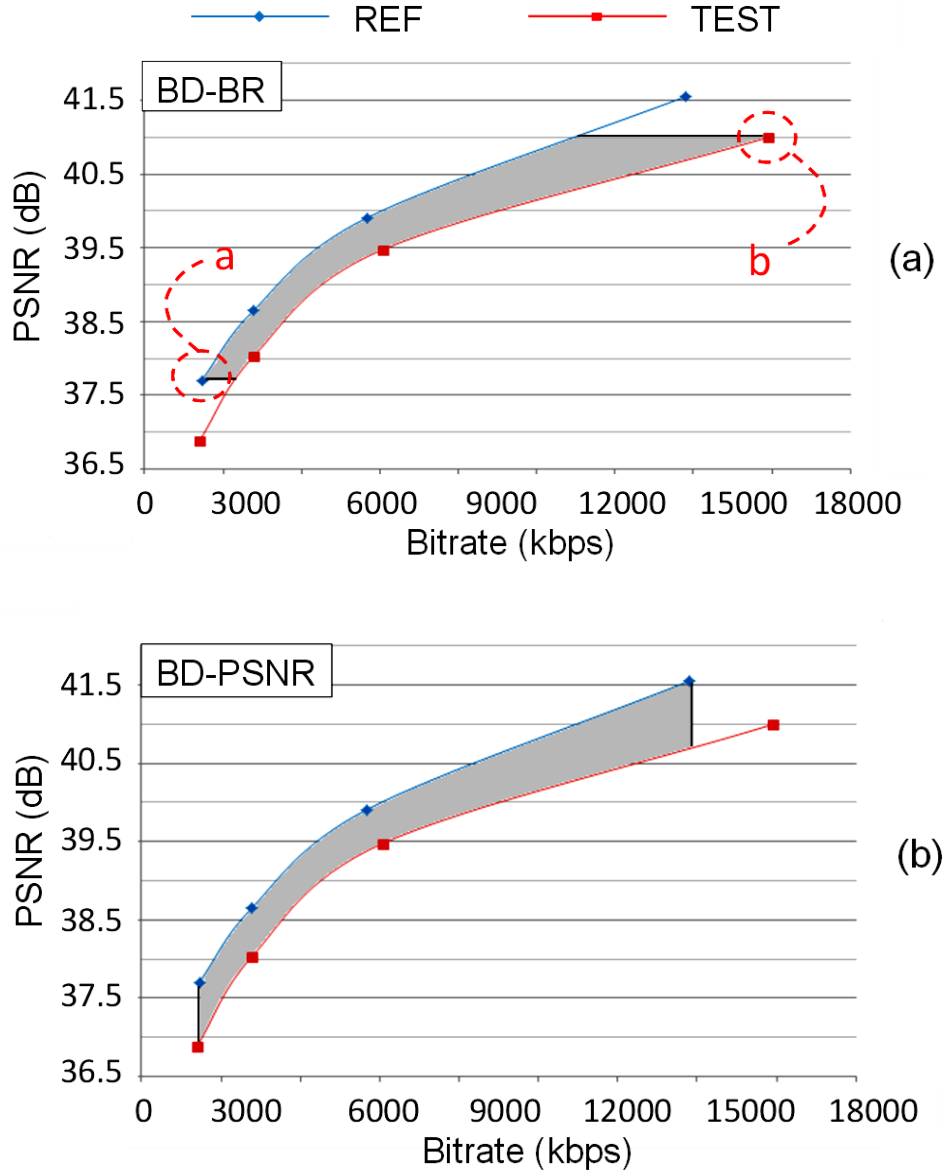


Figure 3.3: RD curves for two targets and the differential areas (gray filling) used in the BD-BR (a) and BD-PSNR (b) calculation

The BD-BR metric can be interpreted as how much better is the bitrate of the test subject compared to its reference considering the same quality. Thus, most of the works found divulge only BD-BR results, given that this value is sufficient to determine how one target compares against the other (BJONTEGAARD, 2008).

Complexity issues have been studied in previous standards, especially H.264/AVC. In addition, many complexity solutions for HEVC have already been published despite the fact that it was just recently standardized. The following subsections present publications targeting both standards, since H.264/AVC solutions can sometimes be adapted to work in HEVC as well. The references were classified into three categories: (1) computational effort analysis; (2) complexity reduction; and (3) complexity control.

3.2. Computational Effort Analysis

Extensive and elucidative work can be found with detailed characterization of HEVC computational effort/complexity. These references are of utmost importance in the pursuit of solutions that tackle complexity issues, since they save a lot of work otherwise spent on running a series of tests.

(VANNE, VIITANEN, *et al.*, 2012) presents a comparative Rate-Distortion-Complexity analysis of HEVC and AVC codecs. The paper is mostly dedicated on cross-checking both encoders, which is not interesting in this work. However, the authors present an analysis showing the computational effort distribution of the HEVC components, pointing that the most complex ones are the Fractional ME, the Integer ME and the T/Q/IQ/IT loop, consuming 54%, 17% and 14% of the encoding time respectively. Secondly, the authors classified the most complex encoding functions as the Interpolation of FME pixels, with 35% of the computational effort share, the SATD calculation, with a 16% share, and lastly the SAD calculation, consuming 11% of the encoding time. The profiling tool used in these results was the VTune profiler by Intel (INTEL, 2013). These results reaffirm and quantify a fact that is widely known in this community: the tools and techniques related to the inter-prediction are the ones responsible for most of the computations spent to encode a video.

A more fine-grained analysis (also using the VTune as profiler) of several HEVC coding parameters is presented in (CORREA, ASSUNÇÃO, *et al.*, 2012b). Initially, the authors describe an analysis to identify the most relevant coding parameters among a wide range of possibilities. This analysis consisted in evaluating the PSNR, bitrate and complexity variations after varying a single parameter in each test. The most relevant parameters were elected as the ones that presented a PSNR variation of at least 0.1 dB, bitrate increase of at least 1.5% and a complexity increase of at least 5%.

With the most relevant parameters selected, the authors created sixteen test configurations, each containing the cumulative result from changing a single parameter with respect to the previous one. The authors justify the cumulative approach by stating that many works perform computational effort profiling by changing a single parameter and checking variations with a baseline configuration, but doing so neglects the interdependencies among parameters. The results from this analysis are particularly interesting, as they show that the parameters that affect the inter-prediction (e.g. ME search range) are the ones responsible for the largest computational effort increases. Although some parameters and tools were removed in the current HEVC version, this information might still prove useful.

Another observation made by the authors is that the filters (SAO, DF and ALF) and the AMP partitions should take priority when considering which tools should be enabled in a complexity-constrained environment, since they increase PSNR and bitrate performance at the expense of small computational effort overhead compared to other tools such as prediction and transforms. ALF was withdrawn in the current HEVC recommendation and will not be discussed in this work.

The last analysis presented in this paper shows that the overall computational effort decreases slightly when the QP increases. The authors propose that this happens due to the smaller amount of processed data with higher QP values, since more zero coefficients are generated after the quantization process.

3.3. Complexity Reduction

This section presents a collection of works found in the literature that propose solutions for HEVC complexity issues based on heuristics faster than the ones already defined in the standard. The works are classified according to the coding stage affected.

3.3.1. Intra-Prediction

(JIANG, HANJIE and YAOWU, 2012) presents a fast mode decision algorithm for the intra-prediction process based on applying a Sobel filter on the picture to identify regions with more details. The authors claim that, based on optical flow theories, the gradient direction of a pixel represents its maximum variation, whereas the perpendicular direction indicates the minimum. Thus, before calling the intra-prediction method, the gradients of each pixel are calculated, identifying the dominant gradient direction of the PU being encoded. Lastly, the intra-mode directions evaluated are the ones perpendicular to the dominant gradient. The results show a 20% time savings with a 0.74% increase in BD-BR

In the work (KIM, YANG, *et al.*, 2011), the authors use a heuristic that limits how deep the quadtree goes on intra CTUs. This kind of technique is usually referred to as Early Termination. Two methods are proposed: the first one evaluates the best mode of the CU immediately above (in terms of depth) in the quadtree. If the best mode from the above CU is the same as the one for the current CU and if the current CU size is the same as the current TU size, the quadtree partitioning is interrupted. The second method considers a mode decision heuristic that was actually adopted by HEVC, called Rough Mode Decision. RMD consists in evaluating a smaller set of the 35 available directional modes. Based on the best result from the RMD, a refinement is applied, evaluating only the modes similar to the best one from RMD. By combining both methods, the results point a 23% time reduction, with a 0.9% BD-BR increase. The RMD was actually implemented in the HEVC reference, showing the efficacy of this solution.

Other works related to intra-prediction were found. To cite a few: (ZHAO, ZHANG, *et al.*, 2011), (YAN, HONG, *et al.*, 2012), (SHEN, ZHANG and AN, 2013), and (SHI, AU, *et al.*, 2013). However, since this dissertation focuses mostly on inter-prediction, they will not be discussed here.

3.3.2. Inter-Prediction

The authors in (LENG, SUN, *et al.*, 2011) employ a frame-level fast mode decision, which is based on the counting the occurrence of CTU depths. If the occurrence of a given depth d is much smaller than $d+1$, then d is not evaluated for all CTUs of the next frame. Secondly, a CTU-level decision evaluates the neighboring CTUs to decide the depth interval that will be evaluated in the current CTU. The average results correspond to a 45% reduction in the ME execution time, whereas bitrate is increased by 0.01% and PSNR is decreased by 0.066dB. However, the authors did not present results in accordance with the CTC, so it is difficult to define the quality of the results.

(CHOI, PARK and JANG, 2011) proposes a CTU Early Termination algorithm based on the occurrence of the SKIP mode. The algorithm is rather simple: if SKIP mode is selected as best mode in the current depth, then the following depths are not evaluated. The authors present statistics showing that if SKIP is selected as best mode for depth d , the next depth has a 5% chance of having a better mode. Encoding time is reduced by 42%, with a 0.6% BD-BR increase.

The solution presented in (CASSA, NACCARI and PEREIRA, 2012) is composed of two fast mode decisions. The first one, named *Top Skip*, consists in skipping the larger CU sizes. To support this decision, the authors claim that there's a low probability of finding the best mode in these CUs, especially in high-resolution pictures. Therefore, the *Top Skip* decision starts from the largest CU found on the previously encoded CTU. The second decision tries to remove the lower CUs based on an Early Termination algorithm, which defines a threshold value based on the standard deviation of the residual samples. The results presented show a 40.3% encoding time reduction, with a 1.91% bitrate increase and 0.007dB BD-PSNR decrease.

The following works also target inter-prediction fast mode decisions: (KIM, YANG, *et al.*, 2012), (SHEN, YU and CHEN, 2012) and (XIONG, LI, *et al.*, 2014). The inter-prediction is the most computation exhaustive process in the encoding loop, therefore many researchers focus on trying to reduce its complexity. The references cited here are only some examples from many.

3.3.3. Other HEVC components

In (TENG, HANG and CHEN, 2011), the authors present a fast decision for the TU mode decision. The proposed solution reverses the evaluation order of the possible TU mode, prioritizing smaller TU sizes. The authors also define some Early Termination conditions: when the PU transformed coefficients after the transforms are all zero, then quadtree partitioning is interrupted. The divulged results show a 49% average time reduction in the TU mode decision time (defined as RQT), with a 0.31% BD-BR increase and a 0.013 BD-PSNR decrease.

Many other works that target a specific HEVC component can be found. (MIYAZAWA, MURAKAMI, *et al.*, 2012) presents an algorithm that disables the Adaptive Loop Filter (ALF) when certain conditions are met. (JOO, CHOI and LEE, 2013) proposes a fast algorithm for SAO encoding based on the best intra mode.

3.4. Complexity Control

This section describes the most relevant works on complexity control found in the literature, as well as one that targets thermal control. These solutions make use of the complexity scalability that is provided by many HEVC tools due to their high configurability. The main challenge consists in how to efficiently scale computational effort up and down in order to maintain quality and compressions requirements.

(MIRTAR, DEY and RAGHUNATHAN, 2012) presents a solution based on dynamic thermal control for an H.264/AVC encoder. The goal of this work was to reduce coding computational effort before the CPU reaches temperature levels considered too high. The authors initially defined different thermal complexity profiles, with varying frequencies and voltages. The second step consisted on running the H.264/AVC encoder for each profile in order to create offline computational effort estimations for all sequences. These results were stored in what they called a *Thermal Policy Characterization Table* (TPC). The authors do not detail how the TPC used in the presented solution was built, but apparently a single sequence was used as training set. To circumvent the offline training issues (e.g., videos that stray from the training set), the authors present an online tuning equation in order to scale the estimations up or down. The results point a 2.4 dB average PSNR drop and a bitrate 4% higher. These results were extracted from running three sequences only, compromising their accuracy.

The work presented in (HUIJIBERS, OZCELEBI and BRIL, 2011) proposes a complexity-scalable Motion Estimation algorithm. This method limits the number of candidates evaluated in the ME process, based on the amount of computation available for each Macroblock. Computation budgeting is decided upon the pre-quantized residue generated for the collocated Macroblock in the previous frame: if the residue is considered too high, more computation is granted to the current Macroblock; otherwise, less candidates will be evaluated for it. This routine is called before each Macroblock is processed to compensate cases when the resources are over/under-used, leaving concerns regarding the computation overhead introduced by this solution. The results from this work are only presented as charts, with no quantified values to allow comparisons.

In (TAN, LEE, *et al.*, 2009), a solution for H.264/AVC encoders that initially selects SKIP mode for all Macroblocks of a wavefront (Macroblocks without interdependence) is proposed. After this initial step, the algorithm refines each Macroblock until the available resources are used up. This can be interpreted as a bottom-up budgeting strategy. The budgeting is calculated by using a single parameter that represents the percentage of available resources. The results in this work are also presented as charts.

In (CORREA, ASSUNÇÃO, *et al.*, 2012a), a complexity control solution is presented. The authors initially show that the maximum depth of collocated CTUs remains the same through a considerable number of consecutive frames. Based on that conclusion, they designed an algorithm that classifies frames as unconstrained (the regular encoding is performed), and constrained, in which a fast mode decision is applied to limit the maximum quadtree depth (Early Termination). The main contribution in this work is the fact that the authors developed a model that dynamically classifies frames as constrained or unconstrained by using online-based estimations. For a 60% target complexity (40% time savings), the results divulged show a 6.29% BD-BR increase and an actual encoding time savings of 38%.

3.4.1. Multilevel Complexity Control

This section describes the work of (KANNANGARA, 2006). This Ph.D. thesis deserves a separate section because it contains the most well documented work on complexity control, although it targets H.264/AVC encoders. The solution proposed in this work, later published in (KANNANGARA, RICHARDSON and MILLER, 2008), is a multilevel complexity control scheme for real-time H.264/AVC encoding. The goal of this work is to enable stable frame rates when encoding videos in a computationally constrained application, such as mobile phones. The authors defend that frame rate drops were especially unpleasant for users in videoconferences and other mobile applications. They designed a solution that combines a frame-level controller with a per-frame complexity controller, as shown in Figure 3.4.

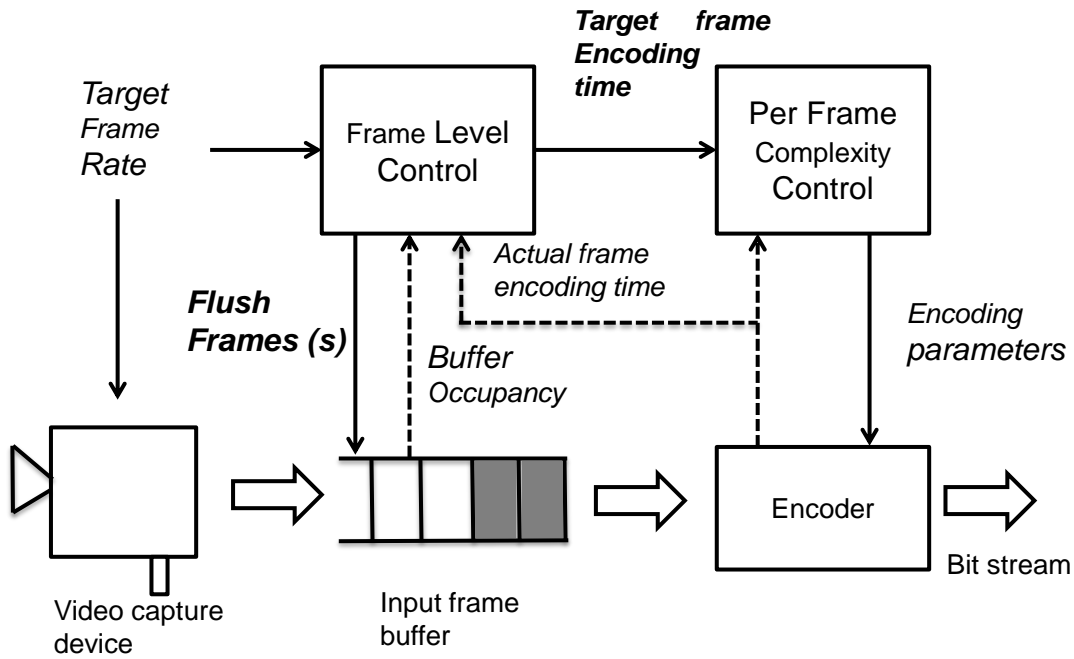


Figure 3.4: Proposed complexity control scheme (source: (KANNANGARA, RICHARDSON and MILLER, 2008))

In Figure 3.4, the Input Frame Buffer has limited capacity, thus it is necessary to drop the frames received (Flush frame(s) signal) when it fills up. The goal established by the authors was to design a solution that is capable of minimizing this event. The main contributions in this work, Frame level Control and Per Frame complexity control, are explained in the following subsections.

3.4.1.1. Frame level control

The Frame level Control component is responsible for maintaining a stable frame rate during encoding. The algorithm calculates the available time to encode the next frame based on the time this frame was kept in the buffer. This value serves as input for the Per Frame complexity control, which applies a fast SKIP mode decision to make sure the target encoding time is met.

The authors present a theoretical discussion to model the encoding time available for the next frame based on a series of equations and equivalence relations, yielding the final equivalence shown below:

$$T_{target}^n = T_f - \frac{1}{B} * [T_{st}^n - T^{last} + T_f * (Level_{Buffer} - 1)] \quad (4)$$

In (4), T_{target}^n is the available time for frame n , B stands for maximum buffer capacity, T_f is the constant time to receive a new frame, T_{st}^n is the instant frame n starts encoding, T^{last} is instant in which the last frame was stored in the buffer, and $Level_{Buffer}$ means the amount of frames stored in the buffer. To summarize, the time to encode the next frame is reduced as more frames are stored in the buffer.

One of the problems found in this section was the premise the authors used to support their math, which stated that there is low movement variation among neighboring frames, thus there is a small deviation in the computation required to encode them. This might be true in H.264/AVC encoders, but as shown in the first

chapter of this work (see Figure 1.1) there encoding time distribution among frames may present high variations.

3.4.1.2. *Per frame complexity control*

This component implements a fast SKIP mode decision based on altering the RD-cost equation originally used in H.264/AVC encoders. The complexity variable is added to this equation, as shown in the equation below:

$$J = D + \lambda_r * R + \lambda_c * C \quad (5)$$

In (3), D stands for the distortion of a residual block, R is the quantity of bits, and C is the complexity associated to a particular coding mode (inter, intra or SKIP). The terms λ_r and λ_c are Lagrangean multipliers.

Complexity control is achieved by altering the number of SKIP-encoded Macroblocks. If computation restrictions are tighter, more Macroblocks are encoded as SKIP and vice-versa. Therefore, (5) is separated into two equations:

$$J_{code} = D_{code} + \lambda_r * R_{code} + \lambda_c * C_{code} \quad (6)$$

$$J_{SKIP} = D_{SKIP} \quad (7)$$

In order to effectively reduce complexity, these values are estimated before invoking the ME. The traditional method uses the actual values achieved after ME is performed instead of estimations. The authors defend that it is not possible to precisely estimate the complexity required to encode a Macroblock, so C_{code} was fixed as 1.3. They also state that it is no way of accurately predicting D_{code} and R_{code} , thus this value is estimated from the collocated Macroblocks in the previous frame. Finally, D_{SKIP} is calculated as the plain difference between the current Macroblock and the one pointed by the predicted motion vector.

The final decision, which tells whether to encode a Macroblock as SKIP or not, is obtained by solving the following inequality:

$$D_{code} + \lambda_r * R + \lambda_c \geq D_{SKIP} \quad (8)$$

Complexity scalability is granted through the variable λ_c , which is recalculated for each frame. The calculation for this variable also involved a series of mathematical discussions that will not be reproduced in this paper. In short, it is based on a feedback control loop that uses the time taken encode previous frames and the time available to encode the current frame as input.

3.4.1.3. *Results*

Table 3.1 presents the results achieved in this work. Only small resolution sequences were used, since the authors focused on mobile applications only. Note that both PSNR and bitrate were compromised, but the frame rate was almost doubled in many cases.

Table 3.1: Frame rate, PSNR and bitrate results (source: (KANNANGARA, RICHARDSON and MILLER, 2008))

N	Sequence	Processing Resources	Normal Encoder			Managed Complexity Encoder		
			Frame rate	PSNR dB (avg)	BR kbps	Frame rate	PSNR (avg.)	BR kbps
1	Claire	50%	7.65	40.74	23.7	15	40.17	30.8
2	Claire	33%	5.10	40.71	19.51	11.96	39.72	26.16
3	Claire	20%	3.10	40.65	14.8	6.38	39.74	19.9
4	Carphone	50%	6.99	37.56	79.9	9.26	37.11	91.8
5	Carphone	33%	4.63	37.54	59.6	5.89	37.17	65.95
6	Mother& Daughter	50%	7.00	38.55	30.37	13.3	38.08	37.27
7	Mother& Daughter	33%	4.70	38.56	24.65	8.50	38.13	30.53
8	Mother& Daughter	20%	2.90	38.53	18.54	4.60	38.17	22.83
9	Foreman	50%	7.80	36.47	91.36	9.75	36.08	100.7 1
10	Foreman	33%	5.02	36.50	68.22	6.00	36.19	75.43

3.5.Final Remarks

The computational effort assessment works discussed present metrics based on the encoding time using a profiling tool. Although it is possible to enable a less platform-dependant estimate by disabling SSE/MMX instructions, there still might exist some instructions that cause different execution times among different architectures. In addition, these estimates include overhead cycles spent on function calls and context switching, which would not be present in VLSI implementations of HEVC encoders.

The complexity reduction solutions presented are not capable of dynamically adapting to the hardware properties, so quality degradation is always introduced in the sequence, even when there are resources available to reduce this unwanted side effect. The same applies for the opposite situation, if computations must be reduced by more than 40% for whatever the reason, none of these solutions will make it possible. The latter scenario is even more troubling in real-time applications, as it causes a frame rate drop. As pointed in (OU, MA, *et al.*, 2011), frame rate drops reduce the QoS (Quality of Service), so they should be avoided at all costs if quality real-time encoding is the application goal.

Table 3.2 shows the results published by the authors of the works discussed in this chapter. The bitrate results are often quoted in the papers, while the BD-BR is not divulged in the older references, which explains the missing data in Table 3.2.

Table 3.2: Summary of the complexity solutions discussed

Target	Reference	BD-BR	Δ bitrate	Time Savings
Reduction /HEVC	(JIANG, HANJIE and YAOWU, 2012)	0.74%	N/A	20%
	(MIYAZAWA, MURAKAMI, <i>et al.</i> , 2012)	0.9%	N/A	23%
	(TENG, HANG and CHEN, 2011)	0.31%	N/A	49% (RQT time only)
	(LENG, SUN, <i>et al.</i> , 2011)	N/A	0.01%	45% (ME time only)
	(CHOI, PARK and JANG, 2011)	0.6%	N/A	42%
	(KIM, YANG, <i>et al.</i> , 2012)	0.45%	N/A	34.5%
	(CASSA, NACCARI and PEREIRA, 2012)	1.91%	N/A	40.3%
	(SHEN, YU and CHEN, 2012)	1.88%	0.17%	41.4%
Control/ H.264	(MIRTAR, DEY and RAGHUNATHAN, 2012)	N/A	4%	N/A
	(KANNANGARA, RICHARDSON and MILLER, 2008)	N/A	19%	50%
Control/ HEVC	(CORREA, ASSUNÇÃO, <i>et al.</i> , 2012a)	6.29%	3.44%	40%

Only one solution complexity control for the HEVC standard is currently found in the literature (CORREA, ASSUNÇÃO, *et al.*, 2012a), and it controls a single coding parameter (CTU quadtree flexibility). As discussed earlier, many components in an HEVC encoder present scalable complexity depending on the input parameters. In addition, it does not really consider an interface with the hardware properties, as it relies simply on a target percentage of the encoding time, thus no hardware-driven adaptability is given. Lastly, none of the works mentioned present a detailed discussion about the characteristics of the controller, which is important if their schemes are to be used in a real application.

As few works on complexity management were advanced so far, further and more in-depth investigations are still warranted, like the ones presented in the next sections. Even in the few complexity management references found, none of them proposes a solution that dynamically adapts itself to the underlying hardware constraints. Thus, there is a gap to be filled in complexity control solutions for HEVC.

4. HEVC COMPUTATIONAL EFFORT ANALYSIS

This chapter initially explains important aspects in order to guarantee that all the results presented are comprehensible, such as describing the HEVC encoding parameters and the Arithmetic Complexity (AC) metric defined in this work. This is followed by an extensive analysis of HEVC encoding computational effort, considering how it behaves under several different conditions. The conclusions from this analysis were used to ground the experimentations later discussed in Chapter 5, in which the second main contribution of this work, the Computation Management Scheme, is explained.

4.1. HEVC Model Encoding Parameters

HEVC has many encoding parameters, each affecting the computational effort of one or more coding components. The ones that are most relevant to the purposes of this work are detailed in the following paragraphs:

- **Max Partition Depth (MaxCUd):** defines the maximum CTU quadtree depth. Each new depth means one extra CU subdivision. The CTC configuration assigns a maximum depth of 4, so starting from 64×64 , CUs can be as small as 8×8 (three subdivisions). This parameter has a direct impact not only in the prediction computational effort, as the residual coding is also affected, since each CU node also contains a nested RQT quadtree. The entropy module is also less used with smaller partition depths, because less RD costs are calculated.
- **Search Range (SR):** determines the ME search range for the ME. This usually causes the search to terminate sooner (unless some other terminating condition is raised), so this parameter can be used to reduce the inter-prediction computations.
- **Asymmetric Motion Partition (AMP):** enables/disables the evaluation of AMPs during inter-predicted CUs. This parameter had much impact in early HEVC versions, but the current ones present fast heuristics that derive motion information for these partitions without calling the ME process. Nonetheless, the RD cost is still calculated for each when they are enabled, so they cannot be neglected.
- **Hadamard ME (HadME):** enables SATD during the FME. This parameter also affects the inter-prediction computational effort, as the SATD involves more arithmetic operations than SAD, making it more complex.
- **Max TU Depth (MaxTud):** as the name suggests, defines how deep the RQT quadtree goes. This affects the transforms, quantization and entropy

modules, since it involves the TU size decision based on the RD-cost. The default value is 3, so a TU can assume a size as small as 4×4 .

- **RDOQ:** enables the RDOQ technique, through which different QP values are evaluated for each coefficient, electing the ones that bring the best RD tradeoff. The quantization computational effort is reduced when this parameter is switched off.

Aside from these parameters, the GOP structure can also be configured. Two GOP parameters can be pointed as important for complexity: (1) number of reference frames and (2) reference frames indices. The first, because ME is repeated for each reference frame, so cutting this parameter in half will also reduce ME computational effort; the second, because it determines how close is the reference frame (in displaying order), which alters the odds of finding good matches during prediction. Thus, close references raise the chances of finding the best match with few ME iterations.

There are other parameters important for components that are not contemplated in this work, such as SAO filtering enabler, PCM enablers etc. Some fast mode decision heuristics can also be disabled, but since they are enabled by default, and since the aim of this work is to further reduce HEVC complexity, they will not be tampered with.

4.2. Mechanisms for Measuring the Computational Effort

There are basically three ways to measure the computational effort of an application: (1) running the application on a given platform and measuring the time it took until it finishes; (2) using a profiling tool to get cycle-accurate results and more detailed information; and (3) building a theoretical model that is accurate enough to a particular goal. Each approach will be further explained in the following sections.

4.2.1. Execution Time

Extracting the execution time of an application is the most elementary manner of quantifying computational effort. Most programming languages have timing libraries that make it possible to acquire these results automatically, assuming the application source code is available. The strong platform-dependency allowed with the lack of support to more detailed measurements (such as modular timing) are two important disadvantages from this approach. However, modular timing can be achieved by explicitly inserting checkpoints in the code.

4.2.2. Application Profiling

This method consists in running a profiling tool that provides a much more detailed analysis of how the computational effort is distributed in an application. Not only cycle-accurate results can be obtained, as some tools provide cache access statistics and fine-grained profiling (modular or functional profiling).

There are many profiling tools available:

- **GNU Profiler (Gprof):** a famous open-source profiling tool that requires the application to be compiled with a special flag first. Afterwards, each run generates an output file that must then be sent for analysis. The final result is a fine-grained time profile of each function organized as a call graph. The percentage over the overall execution time and the number of calls are also accounted.

- **VTune Analyzer:** this is the profiling tool for x86 architectures provided by Intel. The VTune Analyzer provides detailed information of the hardware components as well, such as hardware event profiling. Stack sampling, thread profiling, and dynamically generated code profiling are just a few of the features provided by this tool. However, the license for this product is paid, which restricts its wide applicability.
- **Gem5:** the Gem5 simulator is a modular platform for computer system architecture research, encompassing system-level architecture as well as processor microarchitecture. This powerful tool provides cross-platform profiling, which is especially useful when investigating different architectures. In addition, the memory hierarchy can be explicitly configured, defining capacity, access latency, input width and many others. The results include cycle counters, cache hits/misses, bytes read/written from memories etc. MIPS, ARM, x86 and other common architectures are supported. To profile in a different platform, a cross-compiler not provided with the tool is required. The tool and its source code are openly available.

Although the cross-platform feature provided by gem5 reduces the platform dependency, the fact that the source code must be cross-compiled to the target architecture can be quite cumbersome, since some compilers are hard to find or require paid licenses.

4.2.3. Computational Effort Modeling

Theoretical models are also an important alternative for platform-orthogonal profiling. To model a given application, one can borrow from algorithm analysis theories (e.g., asymptotical notation) or specific properties of each application. Deep understanding of the application is required in order to produce accurate results. Complexity profiling can be used as a starting point, followed by designing a general model for any platform. This is an interesting alternative when the application runs on many different platforms, each with its particularities. Video encoders fit in this category, as they have applications in recording cameras, smartphones and high-end supercomputers as well.

In this work, the computational effort was modeled based on the most common functions used in HEVC encoders. The resulting model was abridged in a metric defined in this work as Arithmetic Complexity, which will be explained in the next section.

4.3. The Arithmetic Complexity Metric

The references for computational effort analysis mentioned in Chapter 3.2 are useful in most cases, but they measure the computational effort based in processing time, which is a platform-dependent metric. Many instruction sets implement Single Instruction Multiple Data (SIMD) instructions for video/signal processing subroutines, such as SAD calculation on Intel SSE2 (INTEL, 2013), so it is difficult to accurately estimate how processing time scales from one hardware architecture to another. The level of parallelism allowed in current multi-processor/multi-GPU systems-on-chip is so high that the processing time is highly dependent on specific code optimizations, platform set up, compiler technologies, and the like. Thus our study focus on the raw amount of operations required, leaving aside the important issue of various degrees of hardware parallelism exploitation that may be needed to deal with this complexity.

An alternative way to measure complexity is to consider the amount of arithmetic operations in a process. This method is simple and free from instruction set particularities. In this context, we propose the use of common video coding operations (SAD, SATD, SSE, and transforms) as basic units. Therefore, in order to quantify the complexity, we defined the Arithmetic Complexity, as presented in the equation below:

$$AC = N_{SAD} \cdot c_{SAD} + N_{SSE} \cdot c_{SSE} + N_{SATD} \cdot c_{SATD} + N_T \cdot c_T \quad (9)$$

In (9), N_{op} and c_{op} represent the amount of occurrences of an operation and the number of cycles spent by this operation, respectively. Since HEVC works with variable block sizes, these values are normalized to a 64x64 CTU (i.e., 32x32 SAD calculations count as 1/4 SAD operation). Accordingly, the values of c_{op} are accounted for a 64x64 CTU as well.

The reason for using the arithmetic functions involved in the prediction and transforms calculations comes from the fact that 85% of the coding computational effort lies in these components, as pointed in (VANNE, VIITANEN, *et al.*, 2012). Other arithmetic operations such as the ones involved in the filtering and in the quantization processes were not considered, since they represent a small computational effort share (15%, from which pre-/post-processing and other helping functions are also included).

It is important to emphasize that the AC metric is oblivious to the memory hierarchy in the system. This was intentionally aimed, because modeling the memory hierarchy involves a large set of variables, such as number of cache levels, capacity of the main memory as well as of each cache in each level. Furthermore, the latencies of read/write operations for each component should also be parameterized. Therefore, for the sake of simplicity, the time spent on memory elements is not considered.

4.4. Test Conditions

The results discussed in this chapter were obtained from a large set of experiments that followed a common configuration, which is described in the following sections. The sequences used in the tests were the ones recommended in the CTC document.

4.4.1. Computational Effort Measurements

In this work, both time and AC values are presented as complexity results. The time values are based on the reports outputted by the HM software, which use the C++ timing functions to calculate the encoding time.

To obtain the AC results, a hypothetical co-processor that contains hardware accelerators was assumed. The CPU calls the co-processor to execute these functions when the distortions (SAD, SSE or SATD) and the transforms must be calculated. This was done to present results in a high-throughput platform instead of being hindered by known General Purpose Processor limitations. To obtain coherent cycle counts for each function, we investigated solutions in the literature that propose VLSI architectures for the SAD, SATD, SSE, and transforms.

Only two works were found: (NALLURI, ALVES and NAVARRO, 2013) and (AHMED, SHAHID and REHMAN, 2012). (NALLURI, ALVES and NAVARRO, 2013) proposes a highly parallel 64x64 SAD architecture that is capable of processing a 64x64 CU in 64 cycles, so we used that for c_{SAD} . In (AHMED, SHAHID and REHMAN, 2012), an N-Point DCT architecture for HEVC encoders is presented. Although the authors do not show real hardware synthesis results, they defend that the

proposed solution is capable of processing a 32x32 TU in 136 cycles. Since a 64x64 encompasses four 32x32 TUs, we assigned c_T to 544 cycles. Since no architectures for the SATD and SSE operations were found, we approximated c_{SATD} to 256, since it takes four times the number of SAD arithmetic operations in the HM reference implementation. In addition, c_{SSE} was also set to 256, because multipliers in general require more cycles than adders.

Since memory hierarchy is not modeled in the AC metric, we also assumed total access to the reference samples. This can be achieved by caching all the current CTU and reference frame data in the co-processor to make sure throughput is not compromised.

Table 4.1 summarizes the test setup used in the computational effort analysis discussed in the following section. In the encoder parameter sensibility analysis, only a subset from these sequences were considered, given the prohibitive amount of simulations that would be required to encode all of them. For every other result, all sequences from each class were encoded. Both AC and time (available in the HM reference output stream) were recorded, in order to provide a discussion on platform-dependent results.

Table 4.1: Test configurations for the computational effort analysis

Running Platform*		Intel Core i7, 3.4 GHz, 4 cores, 16 GB Memory,			
Co-processor**		$c_{SAD}=64$	$c_{SATD}=256$	$c_{SSE}=256$	$c_T=544$
Class	Resolution	Sequences (see Appendix A)	Frame Rate	Frame Count	
A	2560x1600	<i>PeopleOnStreet</i>	30	64	
B	1920x1080	<i>Kimono</i>	24	64	
C	832x480	<i>BasketballDrill</i>	50	64	
D	416x240	<i>BlowingBubbles</i>	50	64	
F	1024x768	<i>ChinaSpeed</i>	30	64	

*for time results **for AC results

4.5. Computational Effort Analysis and Discussions

4.5.1. Maximum Throughput at Default Conditions

Using the AC results as metric, the maximum throughput of the default HM configuration at a given operation frequency can be estimated. For this approximation we assume a processor running at 2 GHz and 100% CPU availability. In addition, the co-processor discussed in section 4.3 was considered to assign cycle estimations. The results are displayed in Table 4.2 for different QP values. Every sequence from each class was used (see (BOSEN, 2011) for the list).

Table 4.2: Estimated maximum FPS at 2 GHz using the hypothetical co-processor (using HM default configurations and AC as complexity metric)

Class	QP=22	QP=27	QP=32	QP=37
A	2.43	2.78	3.15	3.49
B	5.14	5.92	6.50	6.98
C	23.05	25.26	27.79	30.35
D	114.18	124.70	135.74	145.62
F	14.93	15.54	16.27	16.95

As Table 4.2 shows, even the modeled system containing a very high-throughput co-processor with state-of-the-art VLSI implementations is not able to achieve real-time encoding sequences from classes A, B and F using the HM default configuration. Some applications support frame rates as low as 23 frames per second, so Class C sequences can be considered real-time encodable for these cases.

4.5.2. Inter-sequence Complexity Scaling

This analysis accounted the AC results for sequences with the same resolution from classes A and B in order to determine if the computational effort is also similar. The results illustrated in Figure 4.1 are organized as follows: class A sequences, namely *NebutaFestival*, *PeopleOnStreet*, *SteamLocomotive*, and *Traffic* are respectively represented as A1-A4; the same goes for the ones from class B, *BasketballDrive* (B1), *BQTerrace* (B2), *Cactus* (B3), *Kimono* (B4) and *ParkScene* (B5). The average (AVG in Figure 4.1) and the standard deviation (STDD) considering the variation in each QP are also represented.

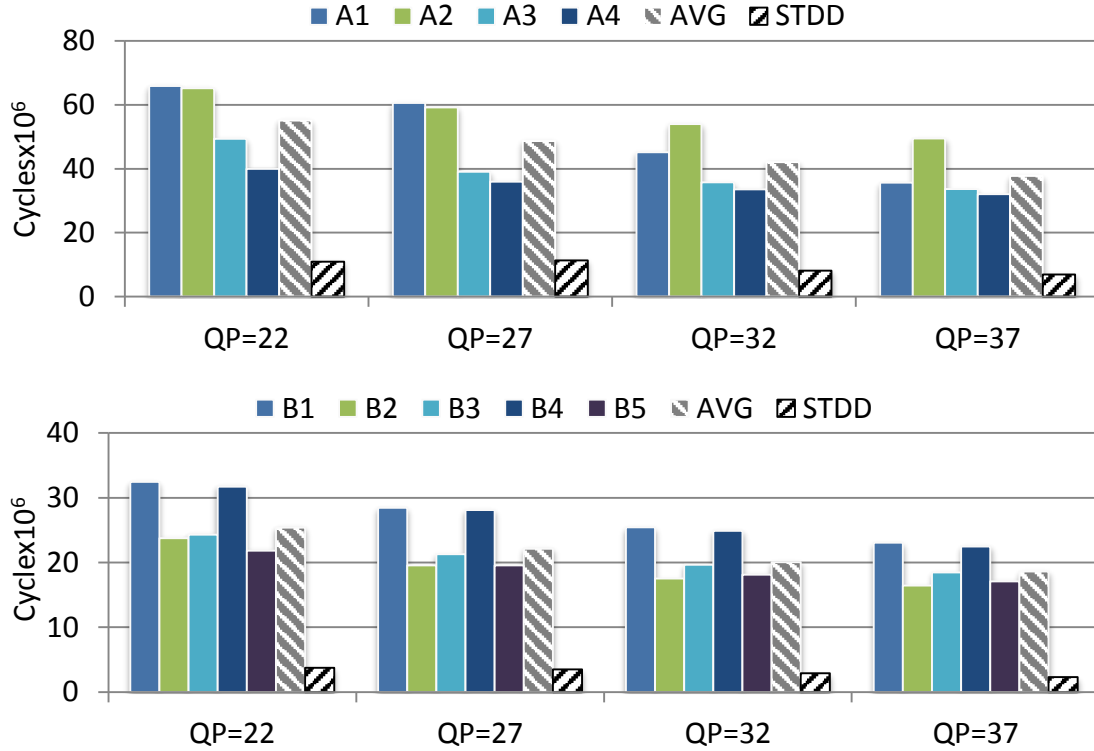


Figure 4.1: Coding complexity comparison of sequences with the same resolution (class A: 2560x1600; class B: 1920x1080)

As the charts indicate, considering only the resolution of a sequence is a good starting point to determine its coding complexity, but it definitely cannot be considered the only factor, since complexity can deviate significantly among sequences with the same resolution. To quantify, results for class A sequences attained a deviation of 18% (for a QP of 37) up to 23% (QP=27), whereas class B presented deviations between 12.5% (QP=37) and 16% (QP=27). Therefore, to make efficient complexity solutions that work for all sequences, it is important to implement mechanisms that dynamically adapt to the particular characteristics of a video instead of singularly working based on average estimations.

4.5.3. Inter-Frame and Intra-Frame Complexity Distribution

Since the same process is applied for every frame in a sequence, it is natural to infer that the time spent to encode a frame is really close to the average. However, this is not the case in HEVC encoders, as revealed in the chart containing the encoding time for each frame of the *BasketballDrill* sequence, displayed in Figure 4.2.

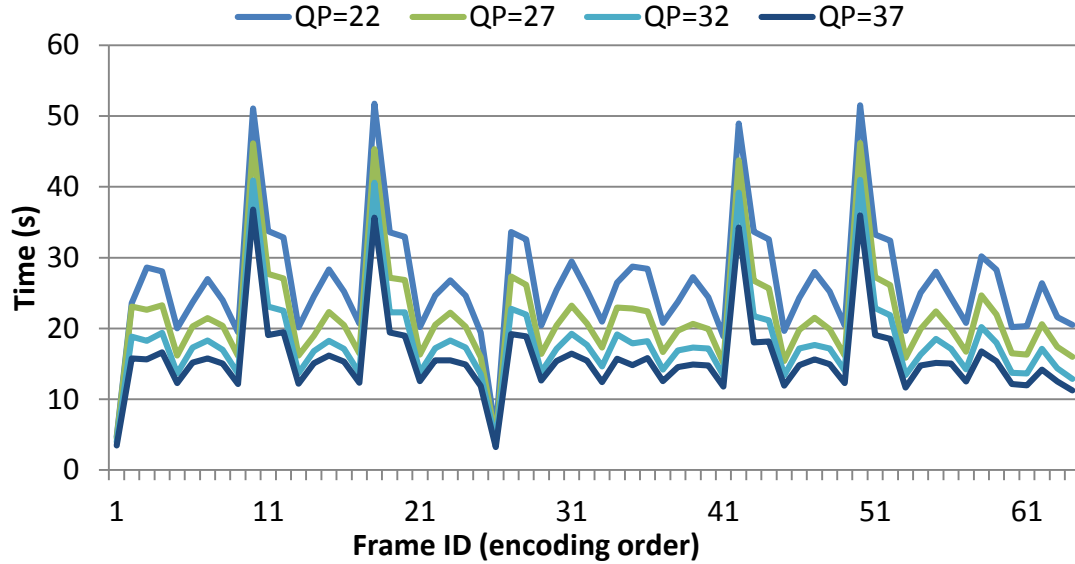


Figure 4.2: Encoding time among frames in the *BasketballDrill* sequence

As depicted in the chart, some frames require much more time (up to three times more) to be encoded than others. The fact that this happens periodically indicates that the GOP structure is the main reason behind these discrepancies. The peaks in the chart are called anchor references, which are the first inter-coded frames in a GOP. The reference frames used to encode anchors are temporally distant (in the RA structure, 8 frames away), so the objects have a higher chance of being displaced between both frames. Therefore, the ME algorithms tend to require more iterations to find the best solution.

The same mistake happens when one infers that the time to encode the CTUs in a frame is evenly distributed. As shown in Figure 4.3, some CTUs can require up to four times more computations than others. By observing the frame and its correspondent time map, it can be easily concluded that the regions presenting movement (i.e. the players) are requiring more time, while the regions that do not move (basically the floor) are encoded faster.

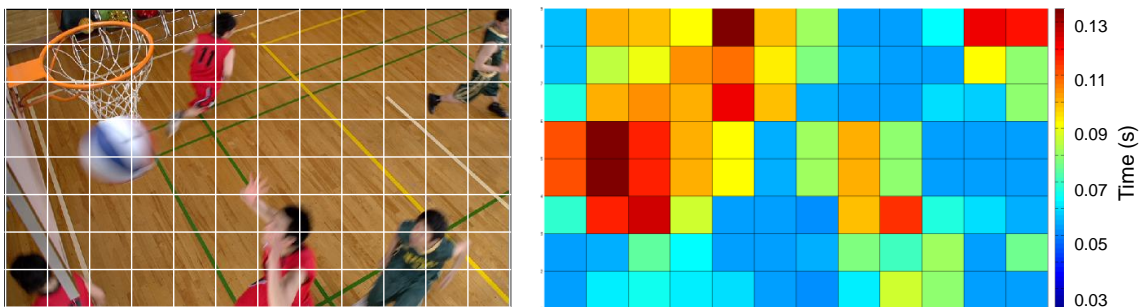


Figure 4.3: Encoding time spent on each CTU in the frame

Both results indicate it is important to study complexity solutions with mechanisms that are able to adapt to the video content on two levels: inter-frame and intra-frame. In doing so, the computations spent to encode a sequence will be much better employed.

4.5.4. Encoder Sensitivity Analysis

In order to measure the importance of each parameter in the encoding complexity and quality, a sensitivity analysis was performed. This kind of analysis consists in keeping track of how the target of evaluation (complexity or bitrate) changes as parameters change with respect to a reference model. In this work, the reference model was the default configuration in the HM implementation, including the fast decisions already enabled.

The coding configurations used in the complexity sensitivity analysis are listed in Table 4.3. The default configuration (c0 in Table 4.3) was used as starting point, and the others were defined by varying a single parameter from c0. The parameters were switched aiming to reduce the computational effort when compared to the default configuration, because this is the final goal of this research. The varied parameter in each case is displayed in bold in Table 4.3. Single-parameter switching was done in order to perform an accurate sensitivity analysis by drawing the relation between each coding parameter and the coding efficiency. The *FME* and *Max. RF* in the table respectively stand for Fractional ME (switched on and off through hardcoding) and maximum number of reference frames, which is not exactly a parameter as well, but can be configured by altering the GOP structure in the configuration files.

Table 4.3: Configurations used in the sensitivity analysis

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
AMP	1	0	1	1	1	1	1	1	1	1	1	1	1
FME	1	1	0	1	1	1	1	1	1	1	1	1	1
Had ME	1	1	1	0	1	1	1	1	1	1	1	1	1
Max. CTUd	4	4	4	4	1	2	3	4	4	4	4	4	3
Max. RF	4	4	4	4	4	4	4	1	4	4	4	4	3
SR	64	64	64	64	64	64	64	64	16	32	8	64	64
Max TUd	3	3	3	3	3	3	3	3	3	3	3	1	2

Since there are 13 configurations, each of these used to encode five sequences for four QP values, a total number of 260 simulations must be performed. The goal of this analysis is to identify the parameters that reduce complexity (measured with the AC metric previously defined)) and use this knowledge to design the management schemes. However, in order to minimize RD losses, it is necessary to consider how compression efficiency (bitrate) is affected as well.

4.5.4.1. Complexity and Bitrate Results

The chart depicted in Figure 4.4 shows the complexity (measured in AC) savings of each configuration with respect to c0.

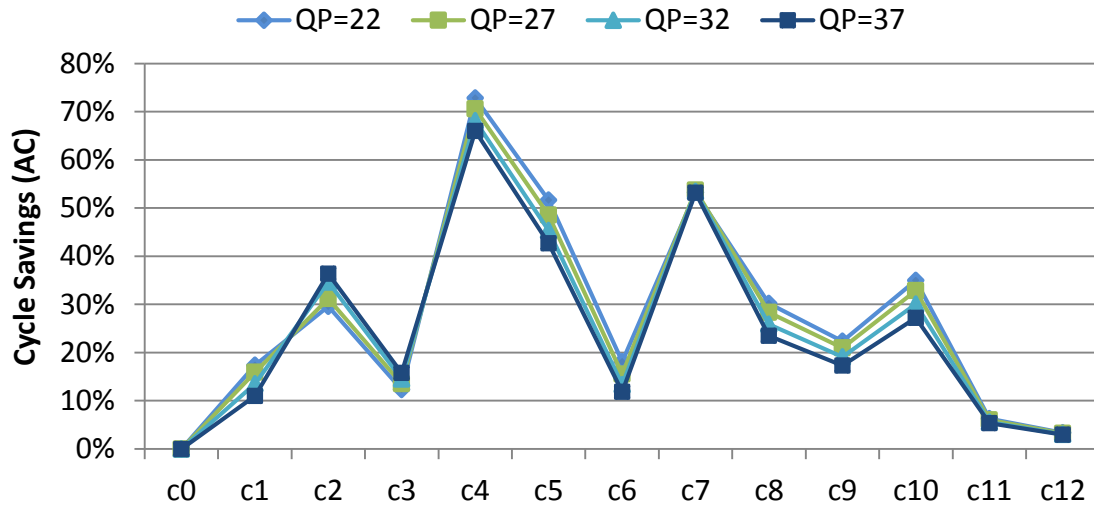


Figure 4.4: AC savings of each configuration with respect to c0

It is possible to observe that peak computational effort savings were achieved under configurations c2, c4, c7 and c10, achieving savings of up to 36.4%, 72.8%, 53.7%, and 34.9% respectively. This means that the max CTU depth, max RF, FME and max TU depth parameters are in that order the ones that have the greatest impact in complexity. The potential computational effort reduction achieved by reducing the maximum CTU depth explains the fact that many complexity solutions are based on this approach. It can also be concluded that QP variations have little impact on the savings achieved for each parameter, since the maximum deviation is 7.76% at c10.

The second analysis consisted in evaluating how encoding time varied with each configuration. The results are plotted in Figure 4.5.

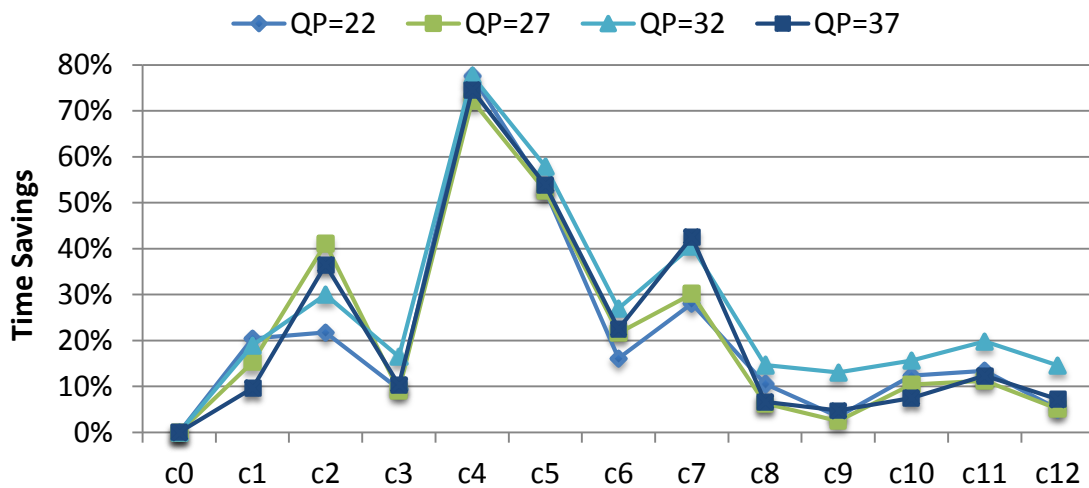


Figure 4.5: Time savings for each coding configuration

As the chart shows, the curves follow similar patterns, but some significant changes can be seen on configuration c7, c8, c9 and c10. For c7, the savings deviate by up to 25% for a QP of 22; for c8, 22% for QP equals 27; c9 had its largest deviation (19%) at QP 22; and c10, 20% when the was QP set to 37. The parameters related to these configurations, SR and max RF number, are directly related to the ME process. Since

the time savings were smaller than the AC ones, it is possible to infer that the MMX/SSE SAD-accelerating instruction (PSAD, available in the Intel architecture used in the simulations (INTEL, 2013)), help diminishing the complexity impact of this function. Thus, reducing the number of SAD calls (which is the main consequence of reducing the SR) does not have the same magnitude on both platforms.

Regardless of these particularities, both metrics present fairly similar savings for the configurations, as shown in Table 4.4. The results were averaged for all QP values. If AC results closer to the running platform were targeted, it would only be necessary to set the SAD cycles parameter to a much smaller value than the other ones.

Table 4.4: Difference between time and AC complexity savings

Configuration	AC Savings	Time Savings	Savings Difference
c0	0%	0%	0%
c1	14%	16%	2%
c2	33%	32%	1%
c3	14%	11%	3%
c4	69%	76%	6%
c5	47%	54%	7%
c6	15%	22%	7%
c7	53%	35%	18%
c8	27%	10%	17%
c9	20%	6%	14%
c10	31%	11%	20%
c11	6%	14%	8%
c12	3%	8%	5%
Average	26%	23%	8%

Aside from computation savings, it is also important to observe the bitrate results of each test as well to make sure the reduction achieved by each parameter is not achieved at the expense of significant bitrate increases. The bitrate variation of each configuration is displayed in Figure 4.6.

The most noticeable increases were in c2 (13.63% for QP equals 37), c4 (up to 36.2% for a QP of 27) and c5 (15.7%, $QP=27$ curve), respectively pertaining the FME and the max CTU depth parameters. These results show that the configurations related to the max CTU depth introduce significant bitrate increases. This proves that the impressive reduction achieved by altering this parameter may not always be worth it, since the compression penalties could compromise other requirements. Another remark from this chart is that reducing the max TU depth parameter sometimes cause coding gains, as shown in the $QP=27$ (c11), $QP=32$ (c11, c12), and in the $QP=37$ (c12) curves.

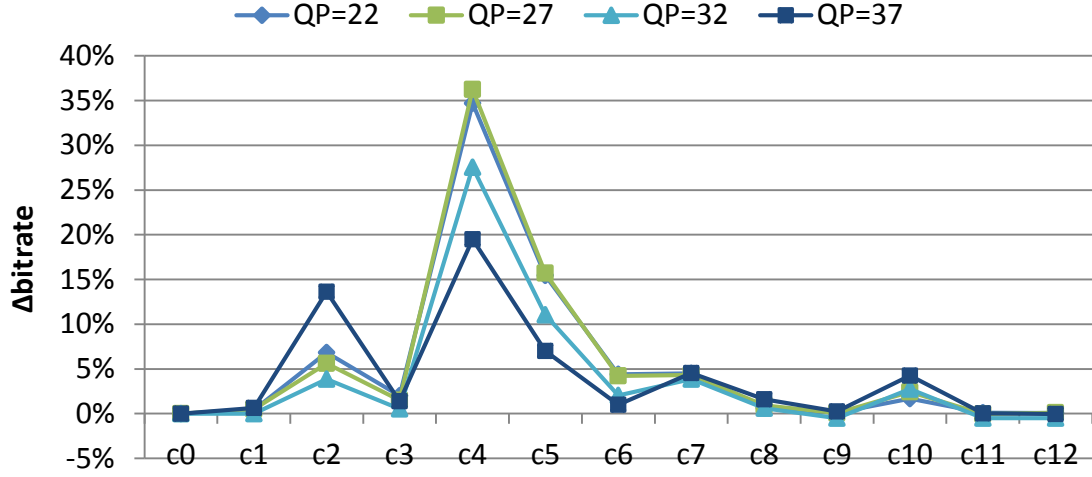


Figure 4.6: bitrate increase of each configuration with respect to c0

4.5.4.2. The Rate-Complexity Efficiency metric

If we compare the charts in both figures, we note that there is a direct correspondence between complexity savings and bitrate increase. However, there are cases in which the computational effort savings are similar, whereas bitrate increases are not. For instance, in the $QP=27$ curve, the c2 and c10 configurations reduced the computational effort by 31.2% and 32.9%, but c2 presented a 5.6% bitrate increase against 2.4% in c4. The difference may seem small, but many experts in the video-coding community already consider bitrate increases of more than 10% prohibitive.

Therefore, a metric that considers both complexity in terms of AC and the bitrate must be used. However, these terms have different magnitudes, since a 10% variation in bitrate in an encoder is more significant and more valuable than a computational effort savings of 10%. Therefore, both values were first normalized to a 1 to 10 scale, according to the following equation:

$$norm(x) = 1 + \frac{x - Min}{Max - Min} \times 9 \quad (10)$$

In (2), Max and Min represent the minimum and maximum values achieved in the simulations respectively. Each sequence class had its normalization calculated, since the AC average scales up with increasing video resolution. With the normalized values of complexity saving (ΔAC) and bitrate variation (ΔB), the Rate-Complexity Efficiency (RCE), a novel metric introduced in this dissertation, is calculated as shown in (11).

$$RCE = \frac{norm(\Delta AC)}{norm(\Delta B)} \quad (11)$$

The explanation for this metric is simple: larger values of RCE represent situations in which computational effort savings were achieved with small bitrate penalties and vice-versa. The RCE results are displayed in Figure 4.7.

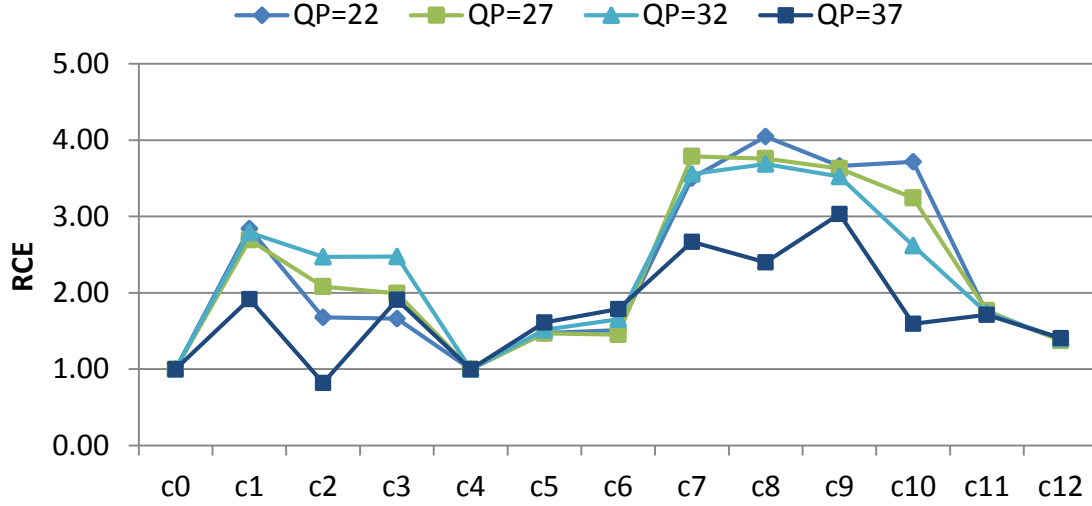


Figure 4.7: Rate-Complexity Efficiency results for all configurations

Note that c10 is now better ranked when compared to c2 due to its smaller bitrate increase. Based on the RCE results, the search range is a high-sensitivity parameter in terms of compounded bitrate and complexity. As shown for configuration c8 in Figure 4.7, the RCE efficiency is at its peak for a SR of 16, denoting an efficient parameter in terms of bitrate and complexity. From this analysis, setting maximum number of reference frames to 1 (c7) is also a prominent configuration, since it took the second position. Configurations c9 and c10 are out because they involve the SR parameter, which was already ranked, so the AMP parameter (disabled on configuration c1) takes the third place, followed by HadME (c3) and FME (c2), depending on the QP value. Note that MaxCUd (c3, c4) is not well ranked, due to its strong impact on bitrate. Chapter 5 will explain how this ranking was employed to model the Control Management Scheme designed in this work.

4.6.Final Remarks

The AC metric is very useful in this research, since it enables a computational effort profiling of the actual encoder components, leaving the time spent on memory access and multitasking aside. This enables running many simulations in parallel without having to worry about whether the encoding time will be affected, since this metric is no longer required to estimate computational effort savings. When the AC results were crosschecked with the timing ones, the comparisons pointed that this metric is rather accurate, further upholding its applicability.

In the inter-sequence and inter-frame analysis, it was proved that computation distribution is extremely uneven among sequences, among frames in a sequence and even among CTUs in a frame. Adaptive solutions that tackle all these levels can be very useful to make the best of the computation available in constrained applications.

The complexity sensitivity analysis presented in this chapter showed that the common parameter used in complexity solutions, the max CTU quadtree depth, is indeed capable of achieving high computational effort gains, but it also compromises bitrate results significantly if compared to other parameters. In light of these results, the use of the CTU depth parameter should actually be avoided to meet compression efficiency, and parameters like SR and AMP enabling should be prioritized.

5. HEVC COMPUTATIONAL CONTROL

This chapter begins by presenting an overview of the complexity management model assumed during the design of all the components and their alternative versions. Afterwards, each composing part of the Computation Management Scheme (CMS), i.e. the Control Unit and the Budgeting Unit, is described. Each design was evaluated and reported in Chapter 6.

5.1. Control Feedback Loop

According to (OGATA, 2013), a feedback control system can be assembled as the block diagram depicted in Figure 5.1.

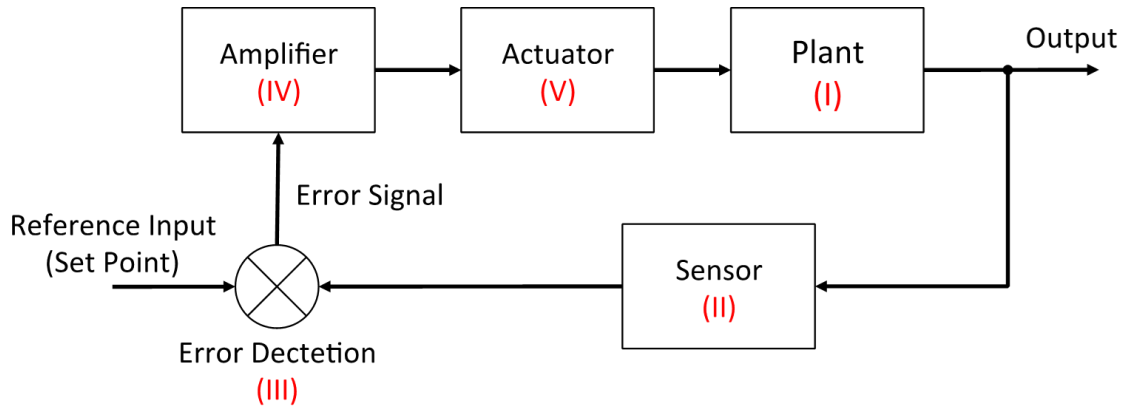


Figure 5.1: Block diagram of a feedback control loop as per (OGATA, 2013)

In Figure 5.1, the (I) Plant is the actual process that must be monitored and controlled (in this case, the HEVC encoder); the process output is also sent to a (II) Sensor, a component that takes records of how the system is behaving (i.e., how much computational effort is being spent); the Sensor reports this to the (III) Error Detector, which compares this with a reference input (namely the Set Point – SP), producing an error; this signal is then used as input for the (IV) Amplifier, which serves as an adjustment mechanism for the (V) Actuator, that changes the operating condition of the Plant in order to minimize the error of the next sample.

5.2. Computational Management of HEVC Encoders

In this work, a Computation Management Scheme was designed. Different implementations of some components were studied, in order to guarantee a final solution that is efficient. The general CMS model used in all cases was inspired in the feedback control loop aforementioned, as depicted in Figure 5.2. Each component is detailed in the following paragraphs:

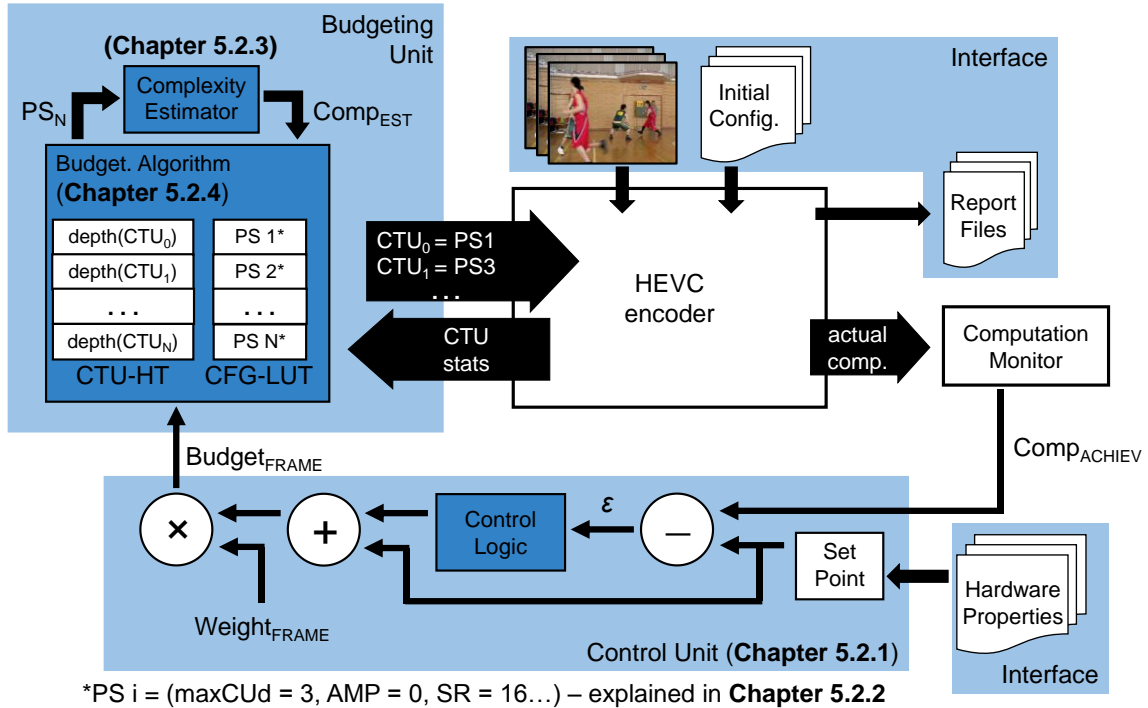


Figure 5.2: Conceptual model of a Computation Management Scheme

- **Interface:** a common interface with the underlying hardware properties is assumed, through which it is possible to receive relevant data, like CPU operation frequency, CPU availability, remaining battery life, and target FPS (Hardware Properties). This information is used to calculate a Set Point (SP). In addition, the Raw Input Frames and the default coding configuration (Initial Config.) are also received as input. Important results for analysis are outputted as Report Files.
- **Computation Monitor:** this component keeps track of how much computation is actually achieved by the HEVC encoder. The values are accumulated while the frame is encoded. When a new frame is encoded, the result is sent to the controller.
- **Control Logic:** computes the Frame Budget ($Budget_{FRAME}$) based on the achieved computation and the SP. In this work, a PID controller was employed (discussed in section 5.2.1, as well as other operations involved).
- **Configuration Lookup Table (CFG-LUT):** stores the Parameter Sets (PS) for each budget category. The sets defined in this work are presented and analyzed in Chapter 5.2.2.
- **CTU History Table (CTU-HT):** stores the depth of each CTU in the previous frame. The CTU depth is used in some budgeting policies to classify CTUs that require more or less computations
- **Complexity Estimator:** this component estimates the computational effort that will be required to encode the next frame using the Parameter Sets assigned to it as reference. Chapter 5.2.3 details how this is done.
- **Budgeting Algorithm:** this unit is responsible for distributing $Budget_{FRAME}$ among the CTUs in the frame. It is composed by the CTU-HT and the configuration CFG-LUT. This is a key component, as it ensures that more cycles are spent in CTUs that are more complex to encode, due to intense

movement or complex textures. Correspondingly, a smaller cycle budget is given to CTUs located in low-complexity regions of the frame, such as those with absence of movement. Chapter 5.2.4 details the different budgeting policies investigated in this work.

Connecting this model with the scheme presented in Figure 5.1, the HEVC encoder is the plant, and the Complexity Monitor replaces the sensor. The Error Detection is the SP/Comp_{ACHIEV} difference, followed by the Amplifier (Control Logic and following operations). Lastly the Budgeting Unit, in which the encoding parameters are actually tuned, represents the Actuator.

5.2.1. Control Unit

The Controller calculates the Budget_{FRAME} available to encode the next frame based on how far the achieved computation (reported by the Computation Monitor) is from the Set Point. In this work, the hardware specifications are used to calculate the SP by the following equation:

$$SP = \frac{Freq_{CPU} \times Avail_{CPU}}{TargetFPS} \quad (12)$$

In (12), $Freq_{CPU}$ and $Avail_{CPU}$ respectively stand for CPU frequency and availability. Since CPU frequency is given in cycles per second, the SP for each frame is obtained by simply dividing the cycles available by the target frame rate ($TargetFPS$). Note that changes in the CPU availability will change the SP as well, introducing adaptability to situations when the CPU cycles must be shared among other tasks.

The SP is subtracted from the computation achieved in the last encoded frame, generating an error signal (ϵ) that is sent to a controller. In this work, a Proportional, Integral, Derivative (PID) controller (ASTROM and HAGGLUND T., 1995) was implemented. A PID controller is a control loop feedback mechanism that is widely known in the industry due to its application in many systems. This controller involves three separate constant parameters, the proportional (P), integral (I), and derivative (D) values. The proportional value P depends on the present error, whereas I depends on the past values, and D can be considered as a prediction of the future ones. In Chapter 6.1.1, a more comprehensive analysis is presented in order to prove its efficiency.

The PID controller receives the difference between the SP and the actual output of the system, which is considered an error (e). With this, for each time instant t , the output is computed as follows:

$$C_{PID}(t) = K_p \epsilon + K_i \sum_{i=0}^t \epsilon(i)dt + K_d \frac{(\epsilon(t) - \epsilon(t - \Delta t))}{\Delta t} \quad (13)$$

This controller tries to match up its output with the Set Point at a certain speed, which depends upon the three constant parameters, namely K_p , K_i , and K_d . These coefficients must be finely tuned, since inefficient configurations may cause large oscillations in the controller output (the controller output rapidly crossing the set-point) or even cause the controller to diverge. There are different parameter-tuning **techniques** for the PID controller. In this work, the widely used Ziegler-Nichols method is employed (ZIEGLER and NICHOLS, 1993).

The PID output is then added to the initial SP, producing the cycle budget available for the next frame. However, as shown in Chapter 4.5.3, some frames require more

computations than others due to temporally distant references. Therefore, a weighting factor was defined in this work, and it is calculated as follows:

$$w_i = \frac{1 + \alpha \cdot (GOPSize)}{QP_{OFFSET}(i)} \quad (14)$$

Where $QP_{OFFSET}(i)$ denotes the QP increase of the i -th frame in the GOP, and α is obtained by solving the following equation:

$$\sum_{i=1}^{GOPSize} \frac{1 + \alpha \cdot (GOPSize)}{QP_{OFFSET}(i)} = GOPSize \quad (15)$$

By applying (7) for $GOP=8$, $\alpha=0.2$ is obtained. Finally, the frame budget is calculated by multiplying the weighting factor by the PID output plus the SP, as summarized in the following equation:

$$Budget(Frame_i) = w_i * (SP + C_{PID}(i)) \quad (16)$$

Following the loop, the Budgeting Unit is responsible for distributing this budget among the CTUs of the frame. As discussed in Chapter 4.5.3, the computation is not evenly distributed among CTUs in a frame, so it is necessary to define a mechanism that ensures this distribution is done efficiently. The strategy used in this work to accomplish this is assigning different encoding parameters for each CTU in the frame. These parameters were organized as Parameter Sets (PSs), and they are described in the following section.

5.2.2. Parameter Sets

Given the fact that some encoding parameters directly affect the computations spent to encode a sequence, it is possible to grant more or less computations for each CTU in a frame by assigning these parameters appropriately. Hence, five Parameter Sets were defined in this work.

The analysis presented in Chapter 4.5.4 was of utmost importance to define which parameters were going to be used to form the PSs. The parameters were ranked according to their RCE and then assigned to each PS until the target savings were reached. To exemplify: if a PS had a target saving of 40%, the first parameter assigned to it was $SR=32$ (20% AC savings), followed by $AMP=0$ (14%). Since only 6% was left, $MaxTUD=1$ (6% AC savings was assigned).

The starting point was the default values found in the HM reference, and the following sets were designed aiming to achieving computation savings 20% higher each time. Table 5.1 lists the PSs defined in this work (PSN stands for $N\%$ savings with respect to the default configuration).

Table 5.1: Parameter Sets used in the CTU budgeting

Param. Set	AMP	Had ME	Max CUD	SR	Max TUD	Max RF
PS0	1	1	4	64	3	4
PS20	1	1	4	32	3	4
PS40	0	1	4	32	1	4
PS60	0	0	3	16	1	4
PS80	0	0	3	8	1	1

The FME parameter discussed in Chapter 4.5.4 was not considered in this work due to the fact that it is not controlled by a parameter. In fact, the current HEVC version defines that the Fractional ME is a mandatory step. Note also that the CTU depth was only used for high target savings, due to the lack of better options, and even in those cases it was only reduced by one. Despite its high RCE, the number of Reference Frames was also avoided because it would otherwise cause savings higher than targeted.

5.2.2.1. Parameter Set Analysis

When the PSs were built, their complexity savings were only estimations based on the AC results for each separate parameter. However, the interdependencies among parameters caused by the complex structures involved in HEVC encoding may produce different results. Therefore, the sequences used in the analyses presented in Chapter 4 were also used to validate each PS savings. The results are presented in Figure 5.3.

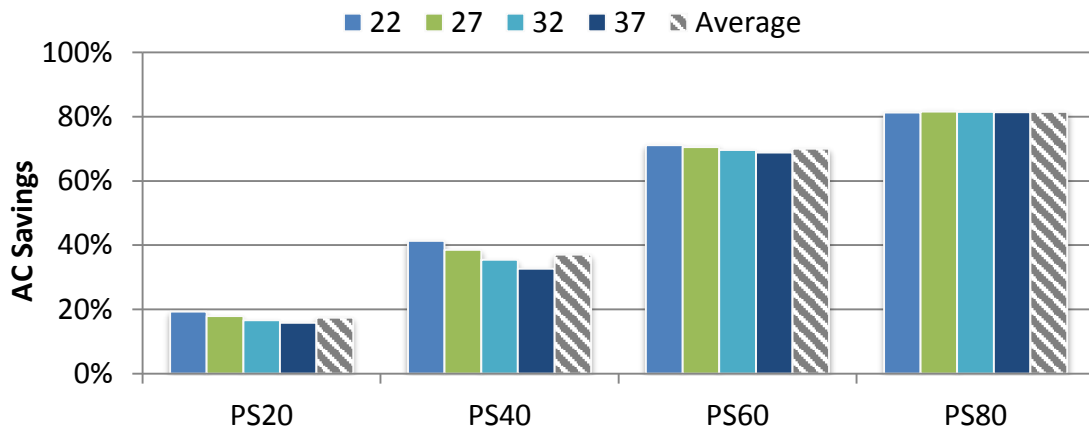


Figure 5.3: Average Arithmetic Complexity Savings of each Parameter Set with respect to PS0

As the chart portrays, each PS achieved savings very close to their targets. However, it is important to mention that this was not achieved on the first attempt. In fact, the sets had to be redefined twice until accurate results were reached.

In order to ensure that compression efficiency and image quality are not severely penalized, the bitrate and PSNR results are also presented in Table 5.2. In addition, the fact that these sets were designed using AC results from a specific (although hypothetical) platform raises concern regarding how they will perform in other applications. Therefore, the time savings for the running platform mentioned in Chapter 4.4 are also presented to gauge how savings differ.

Table 5.2: AC, time, bitrate and PSNR variations of each PS with respect to PS0 (default configuration)

	Target Savings	AC Savings	Time Savings	Δ BR	Δ PSNR (dB)
PS20	20%	17%	22%	0.1%	0.02
PS40	40%	38%	44%	1.1%	-0.04
PS60	60%	68%	64%	7.8%	-0.27
PS80	80%	80%	72%	10.8%	-0.33

Two observations can be raised: (1) the AC and time savings are very similar, but not exactly the same. This is not precisely a problem, since the Control Unit can circumvent frame budgeting misallocations, i.e., if more cycles than estimated are spent in a frame, the Control Unit will simply assign less cycles for the next frame. (2) The PS80 results show that a bitrate is increased by more than 10%, which is extremely unwanted in many applications, so this PSet should be used with caution. However, the fact that this increase was obtained with an 80% computational effort reduction, the results are still considered positive.

The parameters of each set and the average AC savings shown in Table 5.2 were recorded in a Configuration Lookup Table (CFG-LUT), which is used for budgeting and also for the complexity estimation, as detailed in the following section.

5.2.3. Complexity Estimation

During CTU budgeting, it is important to keep track of how much computational effort is available after each the PS is allocated. Therefore, a Complexity Estimator was implemented based on the savings achieved with each PS.

Initially, the estimator borrows the computations achieved in the last frame recorded in the Complexity Monitor. This value is then gradually reduced based on the savings achieved in each PS. The following equation better describes this component:

$$Comp_{EST}(i) = \frac{\sum_{CTU_0}^{CTU_N} Comp_{ACHIEV}(i-1) * (1 - \Delta AC(PS_N))}{\#CTUs} \quad (17)$$

In (17), the estimated complexity $Comp_{EST}$ for frame i is the averaged summation of the computation achieved in the previous frame ($Comp_{ACHIEV}$) multiplied by the savings $(1 - \Delta AC(PS_N))$ of the PS assigned for each CTU in the frame.

It is important to make it clear that this estimation uses offline (AC savings) and online (achieved complexity) measures in its formula, making it adaptive to video content. The estimated complexity is important to make sure the budgeting algorithm does not assign weighty budgets when there is not much more computation available.

5.2.4. Budgeting Algorithms

The last step performed in the Budgeting Unit is to actually distribute the frame budget among the CTUs. The steps involved in this process are following listed:

For each CTU in the frame:

1. Assign a given PS to current CTU (depends on the budgeting algorithm)
2. Update $Comp_{EST}$ with Complexity Estimation
3. Repeat 1 until condition $Comp_{EST} \cong Budget_{FRAME}$ is met

Step 2 can be easily solved by distributing the cycles equally in all CTUs and then selecting the PS that accordingly meets this budget. Nevertheless, this assumes that the computation required by all CTUs is the same. Since that is not the case, a more sophisticated budgeting strategy is required.

All of the strategies designed in this work were based on the premise that CTUs that were encoded with complex structures (smaller CUs) in the previous frame should be assigned more computation, because it is desired to spend more cycles on them to exploit the temporal correlation and maintain an efficient compression. Likewise,

accrediting fewer cycles to CTUs that were encoded with larger CUs is likely to introduce small compression penalties. Therefore, in this work the computational effort required for a CTU is based on the maximum CTU quadtree depth used to encode the collocated CTU in the previous frame. This information is recorded in a CTU History Table (CTU-HT).

From the several strategies investigated, the main ones will be presented in the following sections.

5.2.4.1. *Top-Down Budgeting (TDB)*

This algorithm initially assigns PS0 (highest computational effort) to every CTU in the frame. The second phase consists in demoting CTUs with smaller maximum quadtree depth until the budget is met. The demoting method reduces one complexity tier, meaning if a CTU is assigned to PS40, it will be reassigned to PS60 if demotion is called. Demotion starts with the CTUs that required fewer computations in the previous frame, since this will be likely less significant for them. This process is then repeated until the budget is met, or until all CTUs are already assigned with the smallest computational effort set (PS80). The pseudo-code implementation for this strategy is displayed in Figure 5.4.

5.2.4.2. *Bottom-Up Budgeting (BUB)*

The top-down budgeting applies the inverse strategy to distribute cycles: PS80 (lowest computational effort) is initially assigned to every CTU in the frame and updates cycle estimation. Afterwards, it starts promoting (analogous to demoting, except it increases tiers) the CTUs with larger maximum quadtree depth until the budget is met or no more promotions can be made. The promotion method is analogous to the demotion one (CTUs with PS40 are reassigned to PS60 and so on). Figure 5.5 shows the pseudo-code for this algorithm.

5.2.4.3. *Priority-based Budgeting (PBB)*

Both strategies explained above lack in some aspect. The bottom-up budgeting may never assign PS0 to complex CTUs, since it assigns intermediate Parameter Sets first. On the other hand, the top-down version also saves the PS that introduces most of the complexity savings (PS80) for last, reducing the chances of it being assigned to CTUs that will most probably fit best in this case. Hence, an algorithm to address these two limitations was designed, called The Priority-Based Budgeting (PBB) (in Figure 5.6). This algorithm starts by assigning respectively PS0 and PS80 to CTUs classified with low and high complexity. PS40 is allocated to the intermediate CTUs, but they can be demoted/promoted until the budget is met.

Algorithm 1: Top-Down Budgeting algorithm

```

input : Frame: current frame
input :  $B_F$ : frame budget

1 for each CTU in Frame do
2   set( $PS_0$ , CTU);
3    $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_0)$ ;
4    $Demote_{Depth} \leftarrow 1$ ;
5   while  $B_F \neq Est_{Cycles}$  do
6     for each CTU with DemoteDepth in Frame do
7       demote(CTU);
8        $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_x)$ ;
9     end
10     $Demote_{Depth} ++$ ;
11  end
12 end

```

Figure 5.4: Top-Down Budgeting algorithm

Algorithm 2: Bottom-Up Budgeting algorithm

```

input : Frame: current frame
input :  $B_F$ : frame budget

1 for each CTU in Frame do
2   set( $PS_{80}$ , CTU);
3    $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_{80})$ ;
4    $Promote_{Depth} \leftarrow 4$ ;
5   while  $B_F \neq Est_{Cycles}$  do
6     for each CTU with PromoteDepth in Frame do
7       promote(CTU);
8        $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_x)$ ;
9     end
10     $Promote_{Depth} --$ ;
11  end
12 end

```

Figure 5.5: Bottom-Up-Budgeting algorithm

Algorithm 3: Priority-Based Budgeting algorithm

```

input :  $Frame$ : current frame
input :  $B_F$ : frame budget

// Starts by assigning fix PSets based on collocated CTU
depth

1 for each CTU with Depth =1 in Frame do
    // assign few computations to every low complexity CTU
2   set( $PS_{80}$ , CTU);
3    $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_{80})$ ;
4 end

5 for each CTU with Depth =4 in Frame do
    // assign more computations to high CTU depths
6   set( $PS_0$ , CTU);
7    $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_0)$ ;
8 end

9 for each CTU with Depth =2,3 in Frame do
    // assign an intermediate amount of computations to other
    CTUs
10  set( $PS_{40}$ , CTU);
11   $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_{40})$ ;
12 end

    // Promotion/Demotion refinement in case there is more/less
    budget available

13  $Demote_{Depth} \leftarrow 1$ ;
14 while  $B_F \neq Est_{Cycles}$  do
15   for each CTU with  $Demote_{Depth}$  in Frame do
16     demote(CTU);
17      $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_x)$ ;
18   end
19    $Demote_{Depth} ++$ ;
20 end
21  $Promote_{Depth} \leftarrow 3$ ;
22 while  $B_F \neq Est_{Cycles}$  do
23   for each CTU with  $Demote_{Depth}$  in Frame do
24     promote(CTU);
25      $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_x)$ ;
26   end
27    $Promote_{Depth} --$ ;
28 end

```

Figure 5.6: Priority-Based Budgeting Algorithm

5.3.Final Remarks

All the topics discussed in this chapter can be referred as design space exploration of Computation Management Schemes. When designing each component, it became clear that this research involves many complex tasks, such as Parameter Set elaboration and budgeting strategies. There are still many questions to be unraveled. In particular, a deeper investigation on controllers would further enrich this discussion and could also bring improvements to the final CMS. This is one of the future investigations defined in this research.

The Parameter Set analysis presented showed that it is possible to achieve significant computational effort savings without appealing to rough techniques such as CTU quadtree pruning. Putting it in numbers, around 40% computational effort savings can be achieved while keeping this parameter unchanged. However, quadtree pruning was still required to achieve savings above 40%, but even in those cases the maximum depth permitted was not lower than three, meaning only 8×8 CTUs were left out of the large set of options.

The next chapter concludes the contributions in this work by presenting a comparative study of each budgeting algorithm, as well as a controller analysis and other interesting results.

6. RESULTS AND DISCUSSION

This chapter presents results from an extensive analysis of the different Computation Management Schemes designed in this work. Initially, the PID controller efficacy is put to the test, and soon after each budgeting algorithm was evaluated in order to point out the final CMS. This scheme is then compared against the HM reference implementation and also with solutions published in related work. The common test configurations used in this analysis are detailed in Table 6.1.

Table 6.1: Test setup for the CMS analysis

Sequences (consult Appendix A)	<i>all from Class A</i>
	<i>all from Class B</i>
	<i>all from Class C</i>
	<i>all from Class D</i>
	<i>all from Class F</i>
Frame Count	64
QP	22, 27, 32, 37
Processor Frequency	2 GHz
Processor Availability	50%, 60%
Target FPS (varies for each sequence and QP)	Class A: 2.4 – 3.5
	Class B: 5 – 7
	Class C: 23 – 30.3
	Class D: 113 – 145.6
	Class F: 15 – 17
PID Constants	$Kp, Ki, Kd =$ {0.036, 0.18, 0.018}

The target FPS was obtained from the results reported in Table 4.2, so each value represents the frame rate achieved under default conditions for a 2 GHz CPU. Thus, if CPU is assigned as 100% available, the execution would run normally without the need of a complexity controller. That said, CPU availability was assigned to 60% as a means to obtain a 40% time savings and therefore force the controller to work. The PID constants were the same used in a previous study (GRELLERT, SHAFIQUE, *et al.*, 2013).

6.1.Computation Management Schemes Comparison

The purpose of the first analysis presented in this section is to evaluate how the PID controller compares against a much simpler implementation. If the PID results prove to be more efficient, then it should be added to the CMS. Otherwise, the simpler controller is used. The second analysis aims to put the budgeting algorithms to the test, also using a more rudimentary implementation as target for comparisons. The best options from each analysis were then elected as the composing parts of the final CMS.

6.1.1. PID-based Controller Analysis

In order to prove the efficiency of the PID controller, a simpler control mechanism that computes the cycle budget and maintains the average computation equal to the SP was implemented in this work. Equation (6) shows how the output is calculated for this component, hereby referred to as average-convergence controller (AVGC).

$$C_{AVG}(t) = SP - Average(Actual_{COMP}) \quad (14)$$

In (5), $C_{AVG}(t)$ represents the difference between the SP and the averaged actual computation at a particular time t . As with the PID output, $C_{AVG}(t)$ is a correctness term that must then be added to the SP, yielding $Budget_{FRAME}$.

The quality of a controller can be defined as how close the actual computation is from the SP and how much overshoot (in this case represented as computational effort peaks) it produces. In our application, how close the computational effort is from its target for a given target FPS. While keeping that definition, the charts with the frame-by-frame results of both controllers implemented are illustrated in Figure 6.1 along with the target SP. The sequence used was *BasketballDrill*, RA was the temporal structure, and PBB was used as budgeting algorithm. Moreover, the SP was calculated based on a 60% CPU availability, and the target frame rate was gathered from the simulation mentioned in Chapter 4.5.1, which resulted a 28.9 FPS at QP=32 for this particular sequence.

Note that the PID controller leads to a more stable output, presenting a peak performance of 165.9k cycles at frame 40. In addition, the average achieved computation of 42.8k cycles is really close to the Set Point defined as 41.5k. In contrast, the average-based controller had a lower peak performance (143.5k cycles), but its achieved computation was farther from the SP (50.3k against 42.8k). Therefore, the PID controller was considered a better option, as it is more capable of achieving the SP despite its higher peak performance.

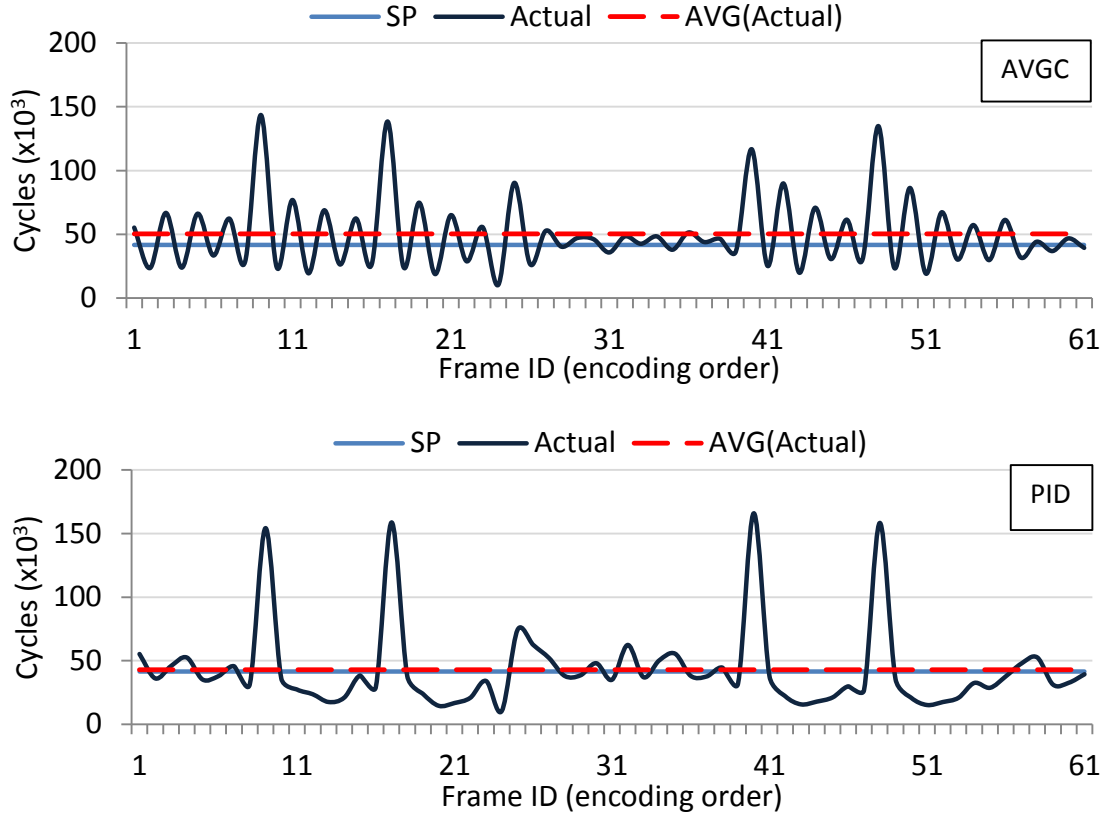


Figure 6.1: Computation results of both controllers. (Sequence: *BasketballDrill*)

To further assess the PID controller efficiency, the Mean Squared Error (MSE) between the SP and the actual cycle count was calculated as per the following equation:

$$MSE = \sqrt{\frac{\sum_{i=0}^{NF} (SP - Actual_{COMP})^2}{NF}} \quad (15)$$

In (15), NF is the number of frames encoded. The square root was applied in the result in order to keep the magnitude of the error, since it is squared in the summation. The second metric was the average PSNR of both outputs, defined as:

$$PSNR_{AVG} = \frac{4 * PSNR_Y + PSNR_U + PSNR_V}{6} \quad (16)$$

Table 6.2 shows the results using these metrics, as well as the bitrate difference.

Table 6.2: RD and MSE results of both controllers

	bitrate	PSNR _{AVG}	MSE
AVGC	845	35.8	30.46
PID	848	35.8	33.57

The results were very similar, and this was actually surprising, because it was expected better RD results with the AVGC, since it spent more cycles to encode. The MSE value of the PID controller is also larger, probably due to its higher peak performances, showing that the advantage on using the PID controller comes from its capability of achieving complexity results closer to the SP while keeping similar compression and quality.

6.1.2. Budgeting Strategies Analysis

To crosscheck the performance of each budgeting algorithm designed, the same approach from the previous section was used. In this case, a rudimentary budgeting algorithm that allocated the same PS for all CTUs was designed. This algorithm was named Uniform Budgeting (UB), and its pseudo-code is illustrated in Figure 6.2.

Algorithm 4: Uniform Budgeting algorithm	
<hr/>	
input :	<i>Frame</i> : current frame
input :	B_F : frame budget
1	$N_{CTU} = Frame_{width} * Frame_{height} / (64 * 64);$
2	for $PSx \leftarrow PS0$ to $PS80$ do
3	$Est_{Cycles} \leftarrow \text{updateEstimation}(PSx) * N_{CTU};$
	// if budget was not exceeded, update best PS
4	if $Est_{Cycles} \leq B_F$ then
5	$PS_{best} \leftarrow PSx;$
6	end
7	end
8	for <i>each CTU in Frame</i> do
9	set (PS_{best}, CTU);
10	end

Figure 6.2: Uniform Budgeting algorithm used for comparison

The reason behind designing and evaluating different budgeting strategies was to obtain not only better controllability (measured in MSE) by assigning more suited Parameter Sets, but also better RD results. Therefore, both aspects were subject to comparison. The results are shown in Table 6.3. In all cases, the *BasketballDrill* sequence and the PID controller were used.

Table 6.3: Comparison results of each budgeting strategy against the HM reference

	$\Delta\text{bitrate}$	$\Delta\text{PSNR}_{\text{AVG}}$	ΔAC	MSE
UB	3.62%	-0.09 dB	-38%	57.58
TDB	3.39%	-0.076 dB	-37%	95.23
BUB	3.16%	-0.076 dB	-37%	96.29
PBB	3.02%	-0.077 dB	-39%	53.31

In Table 6.3, it can be observed that all budgeting strategies presented very similar results in terms of achieved complexity, bitrate and PSNR. Although the Uniform Budgeting had slightly worse bitrate and PSNR results, it is still surprisingly more efficient than expected. Therefore, the best algorithm was elected based on a final comparison considering the MSE (previously defined in (15)). Hence, the PBB showed itself better than the UB in the RD results, and better than all others in terms of MSE. However, these results are still not convincing enough, because PBB is more complex than UB, so further analysis was carried out.

The second analysis was based on the distribution of the Parameter Sets, which is displayed in Figure 6.3. Note that both BUB and TDB allocated a single Parameter Set

(PS20) in four consecutive frames (10 – 14). This is extremely unwanted, because it eliminates the need of a budgeting algorithm to begin with. In addition, when four consecutive frames are poorly encoded, the chance of subjective quality reduction increases, as it raises the chances of visual perception. The UB also assigned only PS80 for frames 13 and 14. On the other hand, the PBB algorithm made sure at least two PSs were assigned in each frame (except for frame 12), granting better flexibility and, therefore, raising the odds of achieving better RD results.

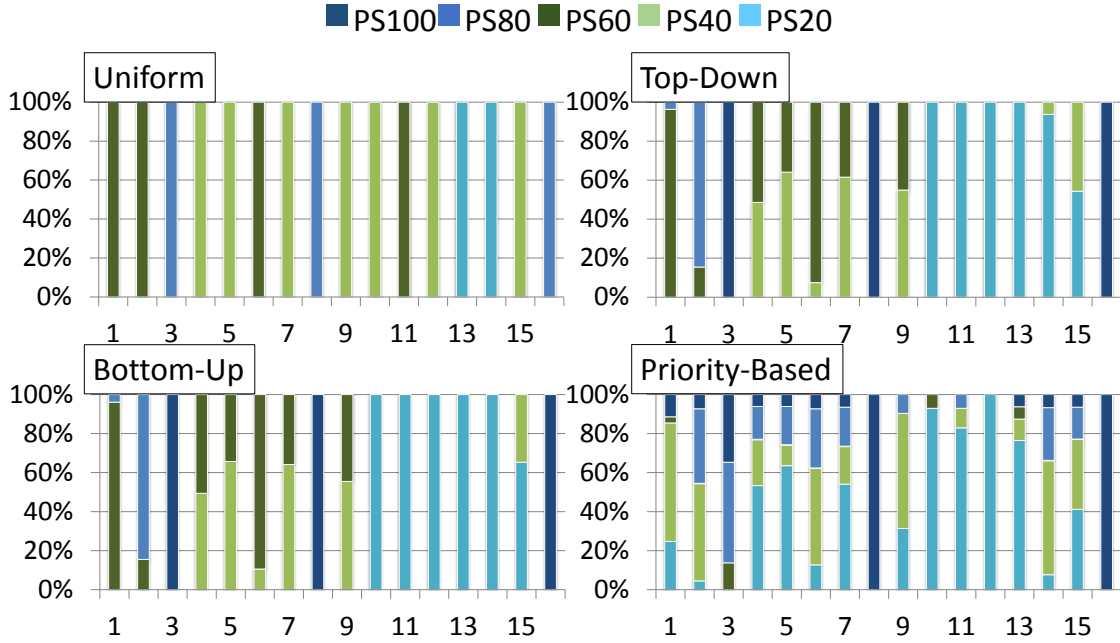


Figure 6.3: Frame-wise PS distribution for each algorithm (sequence: *BasketballDrill*)

To further validate the efficiency of the PBB algorithm, Table 6.4 shows that higher PSs occur more often, counterbalanced by an also higher occurrence of the lower PSs. This is exactly the strategy behind PBB: assigning very low computation to CTUs with lower complexity and vice-versa.

Table 6.4: Overall parameter Set usage for each algorithm (sequence: *BasketballDrill*)

	PS80	PS60	PS40	PS20	PS0
UB	7%	50%	23%	20%	0%
TDP	26%	27%	26%	8%	13%
BUB	27%	26%	26%	8%	13%
PBB	38%	28%	2%	17%	15%

To summarize, a CMS composed of a PID Controller and a PBB algorithm was elected the best option in terms of controllability and compression efficiency, and this configuration will be referred to as HM-CMS in the remainder of this dissertation. In the next section a detailed analysis of this scheme is presented. However, the other budgeting algorithms also showed good results and, depending on the application, might be easier to implement, so they can also be considered a reasonable solution.

6.2. In-Depth Analysis of the Computation Management Scheme

The following sections present detailed analysis of the designed HM-CMS, composed of a PID-based controller and PBB as budgeting algorithm. Initially, the controllability and adaptability of the HM-CMS are tested, followed by a comparison with state-of-the-art complexity control solutions. The last section compares the HM-CMS with the HM reference to measure the RD penalties introduced.

6.2.1. Controllability and Adaptability

This analysis aims to test how fast the HM encoder coupled with the CMS adapts when the available computation changes. This evaluation is important, because this type of scenario occurs often in embedded applications, since other processes could reduce the video encoder CPU share or battery limitations may dynamically affect the computation budget.

Figure 6.4 shows the result of running the class C *RaceHorses* sequence with RA structure, QP=32, and an initial SP of 100 (CPU availability to 100%) that decreases by 20% every 32 frames. This was stretched until a 20% share in order to determine the HM-CMS limits. For visualization purposes, the actual computation displayed in the chart was averaged for each GOP (8 frames).

Note in Figure 6.4 that the initial SP is actually superior to what the encoder demands, so the actual computation stood under the SP line. Actual complexity control starts after frame 32, when the SP changes from 100 to 80. Note that, in this case, the average computation was really close to the SP. The next reduction also was efficiently controlled, since the average computation (64.8k cycles) was still close to the SP of 60k cycles. After this, the actual computation gets farther from its target, showing the HM-CMS has limited control for lower complexity targets. The maximum variation is at frames (131 – 151), when the SP is 20k cycles while the average computations are 29k. However, the actual computations still followed the SP trend.

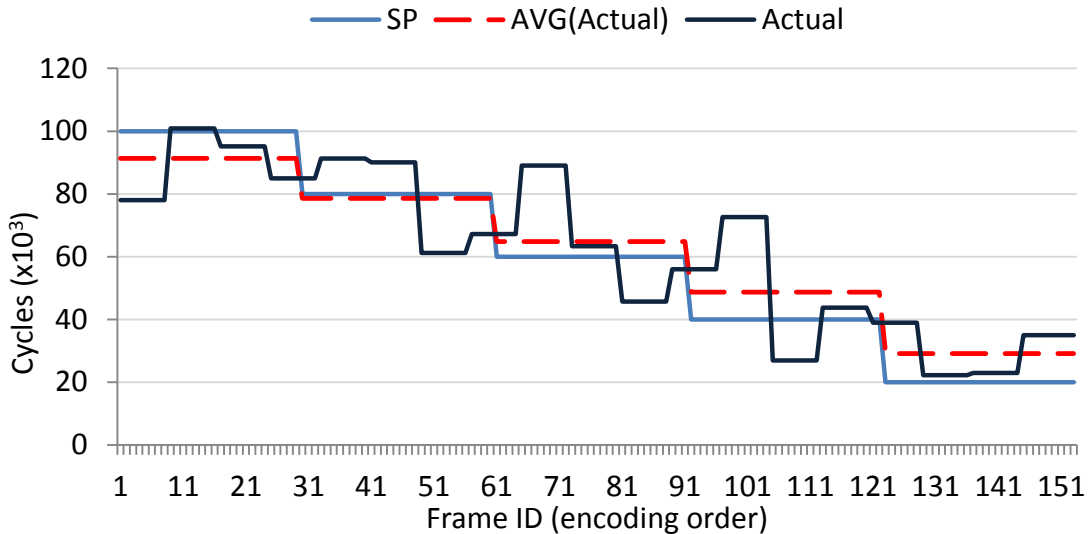


Figure 6.4: HM-CMS performance under varying SPs

Another possible situation is the CPU being periodically shared with other task, demanding a controlled encoder. When the task ends, the encoder has all the CPU for itself, so it can turn the CMS off, until another task is created, initiating the CMS once again. This scenario is depicted in Figure 6.5. When the CPU is shared, 60% of it is

dedicated to the controller (yielding a SP of 45k cycles). The chart shows that the average computation jumps from 47.5k to 70.1k cycles when the HM-CMS is turned off. When the CMS is turned on again, it takes around 20 frames until the computation starts to stabilize around the SP.

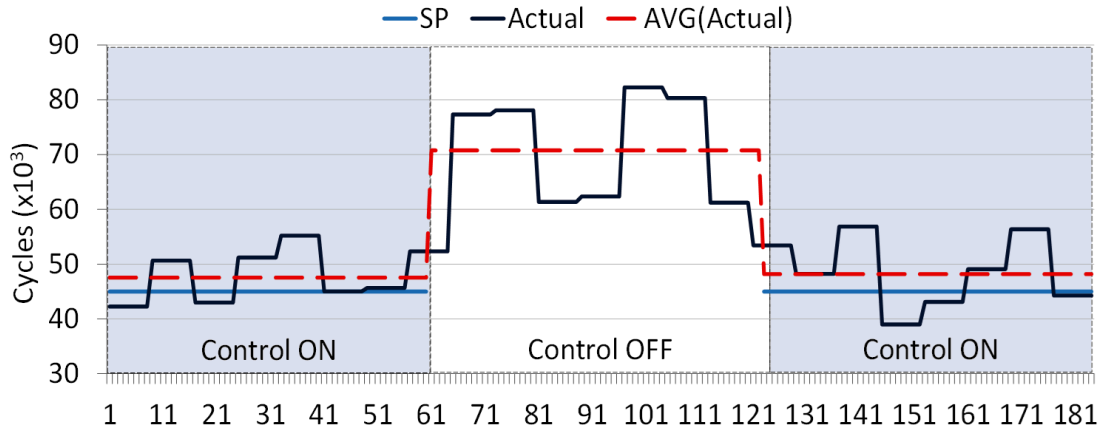


Figure 6.5: Encoding cycles when the HM-CMS is switched off and on again

From both analyses it became clear that the HM-CMS is successful in controlling the HEVC encoding complexity. However, there are still improvements to be made in order to guarantee effective control even for lower SPs.

6.2.2. Related Work Comparison

The following table shows a comparison between the HM-CMS and two relevant complexity control works recently published. Since in both cases the authors used encoding time as metric, this was also extracted from the HM-CMS results in order to provide a fair comparison. The authors divulged results for more than one complexity target. The ones selected were those that achieved savings similar to the ones achieved by the HM-CMS.

Table 6.5: Comparison with Related Work (Target savings: 40% and 50%)

Work	Target Savings	Time Savings	AC Savings	ΔY -PSNR	ΔY -bitrate	BD-BR
HM-CMS	40%	35%	40%	-0.08	1.84%	4.28%
	50%	44%	49%	-0.12	2.9%	6.02%
(CORREA, ASSUNÇÃO, <i>et al.</i> , 2012a)*	40%	38%	N/A	-0.07	3.44%	6.29%
(CORREA, ASSUNÇÃO, <i>et al.</i> , 2011)*	40%	37.6%	N/A	-0.11	1.26%	N/A

* class B sequences only

The comparison shows that the HM-CMS is a competitive solution among these state-of-the-art references, achieving better BD-BR results compared to both works.

For a target savings of 40% (first row in Table 6.5), the HM-CMS achieved very similar time savings compared to both works (the HM-CMS saved merely 3% and 2.6% less time). The bitrate results from HM-CMS are better than (CORREA, ASSUNÇÃO,

et al., 2012a), whereas the PSNR values were better than (CORREA, ASSUNÇÃO, *et al.*, 2011).

Still in Table 6.5, the results with target computational effort savings of 50% (second row) show that the HM-CMS slightly surpasses both works in computational effort savings (6% and 6.4% better savings). Bitrate values are also better for the HM-CMS compared to the results of (CORREA, ASSUNÇÃO, *et al.*, 2012a), but this work surpasses ours in objective quality (PSNR). Lastly (CORREA, ASSUNÇÃO, *et al.*, 2011) had PSNR and bitrate results a tad better.

It is important to add that the HM-CMS achieved a very high accuracy with its target, as opposed to the authors. Another important consideration is that only results for class B sequences were divulged in the references cited, so the results for all sequences may be very different.

6.2.3. Comparison with HEVC Model reference

This analysis evaluated the BD-bitrate results of the HM-CMS encoder running at 60% (Table 6.6) and 50% (Table 6.7) CPU availabilities. Every sequence from both RA and LB structures were encoded, adding up to 336 simulations. Class A sequences must be encoded for RA results only, which explains the missing values in the right half of the table. In addition, class E sequences (mandatory for LB, optional for RA) could not be downloaded from the FTP server that contains every sequence listed in the CTC due to changes in their sharing policy.

Table 6.6: BR-BR results for all sequences, using both temporal structures (Target savings: 40%)

Target Savings 40%	Random Access			Low Delay		
	Y	U	V	Y	U	V
Class A	2.2%	2.2%	2.0%	--	--	--
Class B	3.3%	3.8%	3.5%	2.7%	3.1%	3.4%
Class C	3.3%	5.9%	5.0%	4.4%	6.8%	5.3%
Class D	3.1%	2.1%	2.0%	7.1%	6.2%	5.8%
Class F	6.8%	5.9%	6.4%	5.6%	3.7%	4.2%
Overall	3.7%	4.0%	3.8%	4.8%	4.8%	4.6%
AC savings (%)	40%			40%		
Time Savings (%)	34%			36%		

Table 6.7: BR-BR results for all sequences, using both temporal structures Target savings: 50%)

Target Savings 50%	Random Access			Low Delay		
	Y	U	V	Y	U	V
Class A	3.7%	3.8%	3.5%	--	--	--
Class B	4.7%	5.1%	4.9%	3.6%	3.9%	4.3%
Class C	4.5%	7.2%	6.6%	5.9%	7.9%	7.1%
Class D	3.9%	2.8%	2.7%	8.9%	7.7%	7.7%
Class F	11.6%	10.6%	11.1%	7.9%	6.0%	6.5%
Overall	5.6%	5.9%	5.7%	6.4%	6.2%	6.3%
AC savings (%)	48%			50%		
Time Savings (%)	43%			45%		

The results show that significant encoding time savings were achieved with a slight average BD-BR increment. In Table 6.6, the lowest increments were obtained for classes A (in Random Access) and B (in the Low Delay results), indicating that the computational effort to encode HD sequences is better managed. This is probably due to the fact that more CTUs are available in an HD frame, allowing a better budgeting distribution. The worst results were with sequences from classes F (RA) and D (LB). Class F sequences target user-generated content (as observable in A.5), such as gaming, videoconferencing and slide presentations, so the worse results might be related to this particularity. Similar results were achieved in Table 6.7.

Lastly, detailed PSNR/bitrate charts for each corner of the RA results are depicted in Figure 6.6.

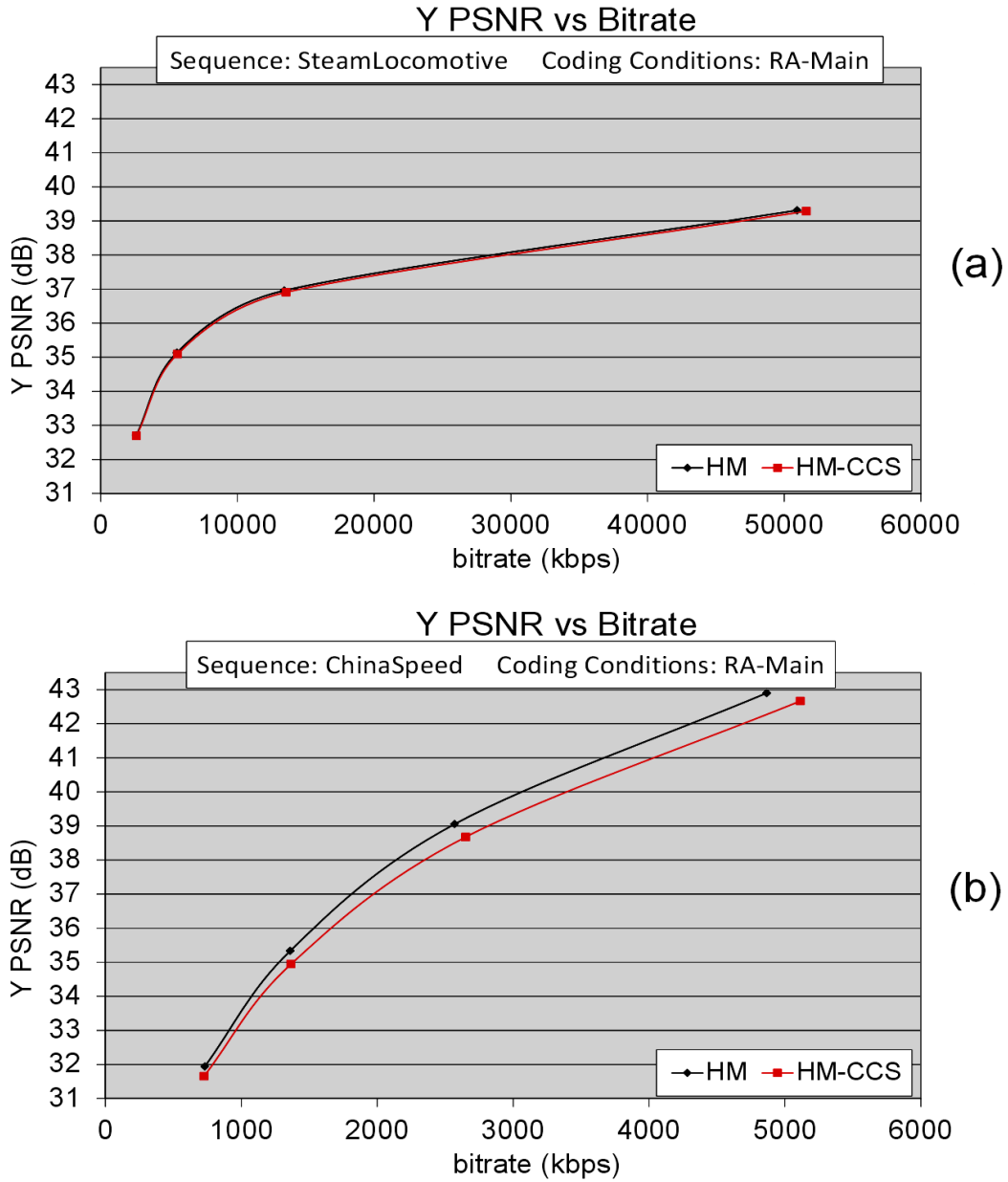


Figure 6.6: PSNR/bitrate charts for two sequences:(a) *SteamLocomotiveTrain* (class A) and (b) *ChinaSpeed* (class F)

These results further uphold what was aforementioned. The curves practically coincide for all QP values in Figure 6.6 (a). The largest bitrate increment in this case was 1.32% for a QP of 22 with a PSNR drop of 0.03 dB, and the smallest differences were at QP=37, in which bitrate increased by 0.54% and PSNR was lowered by 0.01 dB. In Figure 6.6 (b) they start at very similar points for QP=37 (PSNR drop of 0.3 dB while bitrate actually decreased by 0.8%), but the gap increases for smaller QP values, reaching its worst at QP 22, where bitrate increased by 5.08% and PSNR decreased by 0.24 dB.

The following chapter concludes this dissertation with the final thoughts and future investigations intended in this research.

7. CONCLUSION AND FUTURE WORK

This dissertation presented a comprehensive computational effort analysis of the HEVC standard, followed by the design of a Computation Management Scheme for HEVC encoders. The first analysis was extremely useful to understand how complexity behaves and the key encoding parameters related to it. This analysis took advantage of new metrics set forth herein as a new contribution: the Arithmetic Complexity (AC), as well as the metric of compound complexity-bitrate efficiency gain, both defined and explained in this work. Particularly the AC metric proved itself very convenient, as it is accurate, free from platform-specific limitations, and furthermore can be easily calculated during encoding.





Different Computation Management Schemes were designed and analyzed, and the one composed of a PID Controller and PBB budgeting algorithm was elected as the most efficient one. Afterwards, a detailed analysis over this best CMS was performed in order to test its adaptability, its controllability, and the RD efficiency achieved.

The RD results against the HM reference show that this solution can achieve 45% time savings with minor BD-bitrate penalties (6.3% BD-BR increase on average). When compared against related work, the HM-CMS showed itself a competitive solution, achieving better computational effort savings in one case and very similar PSNR and bitrate results.




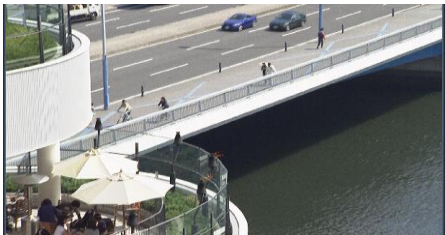
In order for the CMS to reach lower SPs, it is important to investigate further improvements, especially regarding the Parameter Sets (if a PS can greatly reduce the coding computational effort, it should be possible to achieve much higher speed-ups). This will, however, be penalized with increased bitrates. Further refining the AC metric to contemplate other encoding functions such as FME interpolations and RDOQ calls is intended. In addition, studying better implementations of the PID controller, perhaps by altering the K constants using a different tuning method, can prove useful. Other controllers are also considered, such as the Model Prediction Controller. The budgeting algorithms can also be widely explored, since there are many techniques that can be borrowed from Operating System workload management. Finally, designing complexity scalable algorithms for each component and combining this with the current CMS presents lots of potential discoveries as well.

APPENDIX A SEQUENCES USED IN THE TESTS


A.1 Class A sequences




Picture	Specifications	Scene Description
	Traffic, 2560x1600. 30 fps, 8 bits/sample	Shows the traffic on three lanes. The first two lanes are slightly jammed, so the cars move slowly. The third one is free, so cars drive by faster.
	PeopleOnStreet, 2560x1600. 30 fps, 8 bits/sample	Displays a large crowd walking in many directions. The camera remains still.
	NebutaFestival, 2560x1600 60 fps, 10 bits/sample	Shows a float of the Aomori Nebuta Festival. The camera pans very slightly in many directions, keeping the focus on the face of the warrior.
	SteamLocomotive, 2560x1600 60 fps, 10 bits/sample	Depicts a locomotive moving towards the camera. The locomotive produces lots of smoke. At a later instance, the locomotive is so close to the camera that the surroundings can no longer be seen.

A.2 Class B sequences


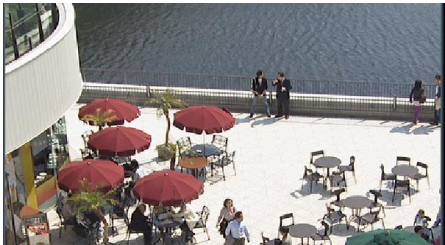

Picture	Specifications	Scene Description
	Kimono, 1920x1080, 24 fps, 8 bits/sample	A lady in a kimono walks around the woods. In frame 140, a change of scene happens, showing a Japanese house.
	ParkScene, 1920x1080, 24 fps, 8 bits/sample	Depicts a park in which cyclers ride by. The camera pans to the left slowly.
	Cactus, 1920x1080, 50 fps, 8 bits/sample	Shows many objects, including a pot with a cactus that rotates vertically. In addition, something similar to a weather vane with cards and an object with two tigers attached to it both rotate horizontally. The camera remains still.
	BasketballDrive, 1920x1080, 50 fps, 8 bits/sample	A group of players exercise a basketball match. Every player moves constantly. The camera pans to the right and to the left, always following the players.
	BQTerrace, 1920x1080, 60 fps, 8 bits/sample	The terrace above the square depicted in <i>BQSquare</i> is shown, but the camera moves up, revealing a road on which cars move in both directions. The camera continues to move up until the terrace is barely seen.

A.3 Class C sequences



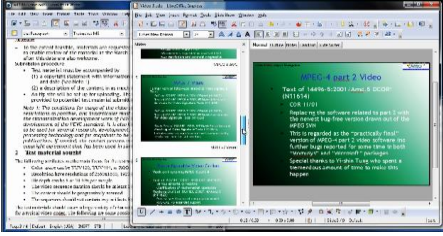

Picture	Specifications	Scene Description
	BasketballDrill, 832x480, 50 fps, 8 bits/sample	A group of players run in circles while taking turns to throw the ball against the board. While the players are constantly moving, the camera remains still.

Picture	Specifications	Scene Description
	BQMall, 832x480, 60 fps, 8 bits/sample	Depicts a typical day at the mall, with people coming and going from both directions. The camera steadily pans to the left. In the last seconds, fewer people are in the scene, so most of the activity comes from camera panning.
	PartyScene, 832x480, 50 fps, 8 bits/sample	Shows a party with kids enjoying themselves. Two kids circle around the tree on the left, while the one in the middle blows bubbles. The camera slowly zooms in on the girl in the middle. Bubbles fly around.
	RaceHorsesC, 832x480, 30 fps, 8 bits/sample	Depicts a group of horse riders moving around a grassy background. The camera follows their movement.

A.4 Class D sequences

Picture	Specifications	Scene Description
	BasketballPass, 416x240, 50 fps, 8 bits/sample	Another scene from the Basketball game, in which both the players and the camera move constantly.
	BQSquare, 416x240, 60 fps, 8 bits/sample	Shows a square with a restaurant, in which people spend leisure time. Initially only the square can be seen, but a body of water becomes visible on the background as the camera zooms out.
	BlowingBubbles, 416x240, 50 fps, 8 bits/sample	Two girls play with a bubble-blow. Many bubbles fly around, and the camera steadily zooms out through the entire sequence.
Same from RaceHorsesC	RaceHorses 416x240, 30 fps, 8 bits/sample	Same scene from RaceHorsesC with a smaller resolution.

A.5 Class F sequences

Picture	Specifications	Scene Description
	<p>ChinaSpeed, 1024x768, 30 fps, 8 bits/sample</p>	<p>A car racing game in which the background changes rapidly to provide the idea of fast movement. The foreground, containing the user interface, stays practically unchanged.</p>
	<p>BasketballDrillText, 832x480, 50 fps, 8 bits/sample</p>	<p>The same scene from <i>BasketballDrill</i> plus a caption box on the bottom, in which a sliding text is displayed</p>
	<p>SlideEditing, 1280x720, 30 fps, 8 bits/sample</p>	<p>Shows a computer screen while a user edits slides and text. Initially, the user rolls up/down the slides panel, then starts editing a slide, moving to the text editor afterwards. Aside from the parts being edited by the user, the remaining background remains still.</p>
	<p>SlideShow, 1280x720, 20 fps, 8 bits/sample</p>	<p>Shows a Microsoft PowerPoint presentation, with many inter-/intra-slide animations.</p>

APPENDIX B RESUMO EM PORTUGUÊS

Análise e Controle de Esforço Computacional para o padrão High Efficiency Video Coding

B.1 Resumo

Codificadores HEVC impõem diversos desafios em aplicações embarcadas com restrições computacionais, especialmente quando há restrições de processamento em tempo real. Para tornar a codificação de vídeos HEVC factível nessas situações, é proposto neste trabalho um Sistema de Controle de Complexidade (SCC) que se adapta dinamicamente a capacidades computacionais variáveis. Considera-se que o codificador faz parte de um sistema maior, o qual informa suas restrições como disponibilidade da CPU e processamento alvo para o SCC. Para desenvolver um sistema eficiente, uma extensiva análise de complexidade dos principais parâmetros de codificação é realizada. Nessa análise, foi definida uma métrica livre de particularidades da plataforma de simulação, como hierarquia de memória e acesso concorrente à unidade de processamento. Essa métrica foi chamada de Complexidade Aritmética e pode ser facilmente adaptada para diversas plataformas. Os resultados mostram que o SCC proposto atinge ganhos médios de 40% em complexidade com penalidade mínima em eficiência de compressão e qualidade. As análises de adaptabilidade e controlabilidade mostraram que o SCC rapidamente se adapta a diferentes restrições, por exemplo, quando a disponibilidade de recursos computacionais varia dinamicamente enquanto um vídeo é codificado. Comparado com o estado da arte, o SCC atinge uma redução de 44% no tempo de codificação com penalidade de 2.9% na taxa de compressão e acréscimo de 6% em BD-bitrate.

B.2 Introdução

Os dispositivos digitais estão em constante evolução, na qual produtos como *palm tops* são considerados obsoletos, e modelos de *smart phones* não podem ser considerados como última tecnologia por mais de um ano. Naturalmente, essa rápida corrida afetou o mercado consumidor, estabelecendo uma demanda crescente por dispositivos móveis mais baratos e poderosos, serviços de mídia de maior qualidade etc.

Aplicações de vídeo digital são exemplos típicos desse cenário. O antigo hábito de compartilhar informação com textos em uma página web está sendo gradativamente sendo substituído por conteúdo visual o qual é armazenado em bancos de dados de fácil acesso, como o YouTube. Além disso, videoconferências também se tornam mais frequentes à medida que as melhorias na tecnologias de comunicação proporcionam larguras de banda maiores, e vídeos multivistas também são cada vez mais comuns, devido à recente onda de aplicações de vídeo 3D. Para quantificar essa tendência, uma previsão publicada pela Cisco mostra que a banda gasta em vídeos na internet vai saltar de 57% em 2012 para 69% em 2017 (CISCO, 2012).

A principal preocupação que surge deste fato é que, à medida que o uso de vídeos digitais cresce, cresce também a necessidade por resoluções maiores e maior taxa de amostragem. Esses dois parâmetros, aliados ao número de vistas para sequências multivistas, são diretamente proporcionais à banda necessária para transmitir o vídeo, assim como a capacidade de memória necessária para armazená-lo. Esse valor pode ser quantificado para vídeos não comprimidos. A equação abaixo mostra a banda necessária para transmitir uma sequência de vídeo não comprimida considerando sub-amostragem de cores 4:2:0:

$$BW = N_{VIEWS} \times W \times H \times FPS \times B_{depth} \times 1.5 \text{ bits/s} \quad (1)$$

Em (1), N_{VIEWS} representa o número de vistas (acima de 1 para multivistas), B_{depth} , o número de bits para cada amostra de luminância ou crominância (usualmente 8), enquanto que W , H , e FPS são respectivamente a largura, a altura, e a taxa de amostragem da sequência. Note que a sub-amostragem de cores 4:2:0 já é um mecanismo de compressão de dados, posto que define que cada 4 amostras de luminância possuem somente duas amostras de crominância (uma para cada camada). Isso explica o fator de multiplicação 1,5 na fórmula (se a sub-amostragem 4:2:2 fosse levada em conta, na qual cada amostra de luminância possui uma amostra de crominância em cada camada, esse fator seria 3).

Para uma sequência *Full HD* (1920x1080 pixels) capturada a uma taxa de 30 quadros por segundo, é necessária uma banda de 746,5 Mbps para transmitir esse vídeo em tempo real. Isso também significa que, se é necessário armazenar um vídeo *Full HD* de 60 minutos, seria necessário 336 GB para isso. Esses valores são claramente proibitivos com as tecnologias atuais, especialmente se dispositivos portáteis com restrições ainda maiores, como *smart phones* e *tablets*, são considerados. Portanto, é imperativo comprimir essa informação com técnicas de codificação de vídeo para possibilitar a ampla utilização dessa mídia.

A codificação de vídeo pode ser abstratamente descrita como um processo que analisa quadros e regiões dentro dos quadros (chamadas blocos), buscando informações redundantes que podem de alguma forma ser comprimidas através da exploração dessas redundâncias. Isso naturalmente gera uma cadeia de bits que deve ser decodificada quando o vídeo é exibido. Existem diversas técnicas de codificação disponíveis, cada uma realizando a mesma tarefa de forma diferente, portanto padrões de codificação de vídeo foram criados para possibilitar uma linguagem comum entre codificadores e decodificadores implementados em plataformas diferentes.

Os codificadores de vídeo mais recentes no mercado geram um bitstream em conformidade com o padrão H.264/AVC, o qual teve sua norma publicada em 2003 (ITU-T, 2003). Apesar de o mercado ter amplamente adotado o H.264/AVC como solução comercial, muitos especialistas em codificação de vídeo defendem que a taxa de compressão atingida por esse padrão será em breve considerada insatisfatória, especialmente quando se consideram as futuras expectativas para serviços multimídia. De fato, o *Motion Picture Experts Group* (MPEG) lançou um relatório em 2011, enumerando os fatores que justificam a necessidade de um novo padrão de codificação de vídeo. São eles (JCT-VC, 2011):

- Dispositivos de alta definição (*High Definition* – HD) já se encontram no mercado atual a custos razoáveis, enquanto que a capacidade corrente das redes de comunicação e de Internet não é suficiente para transmitir uma grande quantidade desse conteúdo. Isso é ainda mais grave se considerarmos a próxima geração de conteúdos e dispositivos Ultra HD (UHD), como televisores 4Kx2K.
- Para dispositivos móveis, serviços de vídeo com resolução QCIF e baixa taxa de amostragem são amplamente inaceitáveis. Antecipando que resoluções HD, como a 720p (1280x720) ou até maiores, serão introduzidas no setor móvel para prover qualidade semelhante a de televisores, a falta de uma banda de transmissão suficiente, assim como o preço necessário para os serviços de transmissão necessários, continuarão sendo problemas de longa data.
- As limitações mencionadas acima são ainda mais severas em serviços de tempo real, como videoconferências.
- A demanda por conteúdo visual (utilizando as tecnologias de compressão atuais) estão crescendo mais rapidamente do que a infraestrutura de rede é capaz de suprir economicamente, tanto para redes com fio como para redes moveis.

Portanto, uma nova tecnologia de codificação de vídeos com capacidade de compressão suficientemente superior à do AVC se faz necessária para remediar os problemas listados.

Com isso em mente, um grupo de profissionais da ITU-T e da ISO/IEC formaram o *Joint Collaborative Team on Video Coding* (JCT-VC), com o objetivos de desenvolver o padrão de codificação de vídeo da próxima geração. O resultado desse esforço foi o padrão *High Efficiency Video Coding* (HEVC), o qual teve sua primeira norma publicada em abril de 2013 (ITU-T, 2013). Os resultados atingidos pelo HEVC

mostram que esse padrão ultrapassa em 23% o H.264/AVC em termos eficiência de codificação (usando a métrica *Bjontegaard Difference*) (LI, SULLIVAN and XU, 2012), (BJONTEGAARD, 2008).

O padrão HEVC define uma estrutura de codificação híbrida baseada em blocos, combinando predição com compensação de movimento e codificação de transformada com codificação de entropia altamente eficiente. No entanto, em contraste aos padrões anteriores, o HEVC oferece uma estrutura de bloco flexível que permite o uso de grandes e variados tamanhos de blocos na predição e nas transformadas. Ele também provê predição intra-quadros, predição de movimento adaptativa, um novo filtro, e uma versão melhorada do algoritmo CABAC, implementado na codificação de entropia. Como pode ser visto na Tabela B.1.

Tabela B.1: Ferramentas de codificação suportadas pelos padrões H.264/AVC e HEVC

	H.264/AVC	HEVC
Estruturas de Dados	Macrobloco, bloco	CTU, CU, PU, TU
Intra-predição	9 modos	35 modos
Inter-predição	4 modos	8 modos
Transformadas	2 modos	8 modos (RQT)
Filtros	DF	SAO, DF

Essas inovações são componentes-chave para que o HEVC atinja um desempenho de codificação superior ao do H.264/AVC, mas elas também introduzem um problema que é assunto de pesquisa há muitos anos: os requisitos computacionais necessários para aplicações HEVC de tempo real.

Nesta dissertação, os termos complexidade e esforço computacional serão tratados como sinônimos, visto que essa convenção foi estabelecida na comunidade de codificação de vídeo.

Aplicações práticas introduzem muitos fatores que não podem ser ignorados no desenvolvimento de sistemas complexos como codificadores de vídeo. Por exemplo, se uma câmera precisa gravar e instantaneamente transmitir o vídeo (como em transmissões ao vivo), então é necessário codificar essa informação em tempo real, o que geralmente significa que mais de vinte quadros por segundo devem ser codificados. Isso se torna ainda mais grave quando os recursos computacionais utilizados pelo codificador são compartilhados entre outros componentes do dispositivo, indicando que uma quantidade variável de computação se encontra disponível em cada intervalo de tempo. Além disso, se o dispositivo é alimentado por uma bateria, a vida útil desta pode ser suficiente somente para realizar uma codificação com baixo custo computacional.

Todos esses cenários requerem soluções que sejam capazes de alguma forma reduzir a complexidade da codificação. Uma maneira de resolver isso consiste em desenvolver um codificador de baixa complexidade que reduz uma quantia fixa de computação comparada a uma implementação tradicional. Contudo, isso indica que o mesmo processo vai ser executado mesmo quando há mais recursos computacionais disponíveis, possibilitando uma codificação de maior qualidade. Alternativamente, um subsistema de codificação que é capaz de adaptar os requisitos computacionais dinamicamente de acordo com as recursos do sistema global aponta uma solução mais interessante.

Uma grande seleção de trabalhos que estudam os problemas de complexidade na codificação de vídeo podem ser encontrados na literatura. Elas podem ser separadas em duas categorias: as que se baseiam em redução de complexidade; e as que se apresentam sistemas de controle de complexidade. Redução de complexidade em codificação de vídeo consiste fundamentalmente em substituir um componente do codificador por uma alternativa mais rápida que realiza uma tarefa similar. Isso é feito através do desenvolvimento de novos algoritmos menos complexos e heurísticas mais rápidas. Em contraste, soluções de controle de complexidade se baseiam em ajustar dinamicamente parâmetros de codificação para atingir diferentes níveis de complexidade. Essas soluções se aproveitam do fato de que muitos componentes presentes nos codificadores possuem parâmetros de entrada que podem ser dinamicamente alterados, e muitos desses parâmetros afetam diretamente a complexidade desse processo. As duas linhas são válidas, mas redução de complexidade tem sido exaustivamente estudada há bastante tempo, e a falta de adaptabilidade dessa linha limita sua aplicação, ao passo que controle de complexidade insere

adaptabilidade já na sua definição. Adicionalmente, um número limitado de trabalhos voltados a controle de complexidade se encontram na literatura atual, indicando que inovações significativas podem ser descobertas. Por fim, mesmo as referências já encontradas não definem uma comunicação com o sistema no qual o codificador está inserido.

B.2.1 Formulação do Problema e Motivação

Para se obter complexidade escalável em um sistema, é possível utilizar técnicas de controle de complexidade. Todavia, esse tipo de solução insere novos desafios quando o alvo é um sistema de codificação de vídeo. Por exemplo, a Figura B.1 apresenta a distribuição de tempo entre quadros de um mesmo vídeo e entre CTUs de um mesmo quadro.

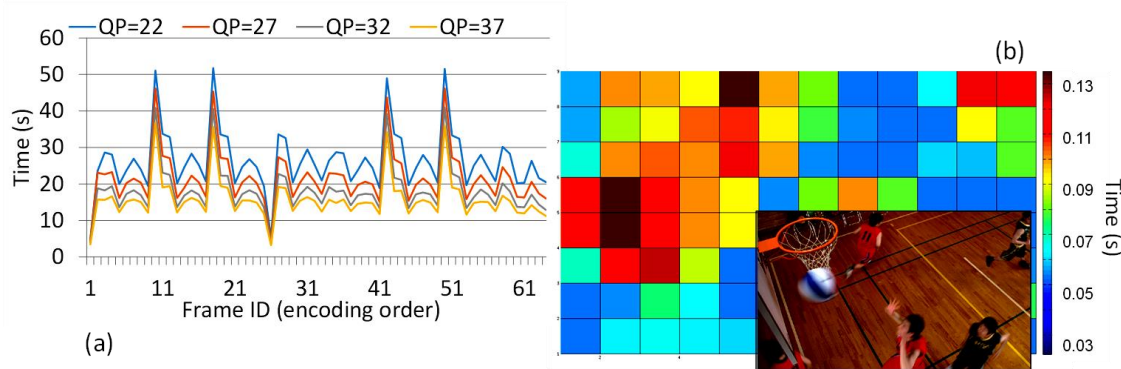


Figura B.1: Tempo de codificação entre quadros (à esquerda) e entre CTUs de um mesmo quadro (à direita) (sequência: *BasketballDrill*)

Esta análise primeiramente indica visto que a complexidade não se distribui de forma constante durante a codificação de um vídeo, portanto é importante entender e modelar a complexidade na codificação HEVC ao invés de desenvolver soluções às cegas. Em segundo lugar, desenvolver sistemas de controle de complexidade envolve tarefas que não triviais, por exemplo: (1) analisar e caracterizar a sensibilidade dos parâmetros de codificação com relação à complexidade associada a cada um; (2) desenvolver um controle que se comporte adequadamente em um sistema de codificação e estudar a adaptabilidade, a controlabilidade e a observabilidade desse sistema; (3) descobrir a melhor política para associar recursos computacionais entre quadros de uma sequência; (4) investigar a melhor forma de alocar recursos computacionais entre as CTUs de um quadro; e (5) monitorar a computação dinamicamente e reagir prontamente quando ela desvia da computação alvo.

B.2.2 Proposta e Metodologia

Esse trabalho objetiva a investigação de técnicas de controle de complexidade para codificadores HEVC. Nessa dissertação, duas contribuições podem ser apontadas:

1. Uma análise do esforço computacional associado a parâmetros-chave da codificação, utilizando métricas que foram elaboradas nesse trabalho, e;
2. Um Sistema de Controle de Complexidade (SCC) para codificadores HEVC que se adapta dinamicamente às restrições computacionais da plataforma.

O SCC proposto foi desenvolvido assumindo a existência de uma interface com o sistema no qual o codificador está inserido. Por essa interface, informações relevantes ao esforço computacional disponível são informadas, como frequência corrente de operação, disponibilidade da CPU e taxa de amostragem alvo (medida em quadros por segundo ou *frames per second* – fps). Com essa entrada, os recursos computacionais disponíveis para cada quadro do vídeo são calculados e distribuídos entre as CTUs que compõem esse quadro.

Os resultados da análise de esforço computacional foram utilizados como conhecimento base para desenvolver os componentes do Sistema de Controle de Complexidade. Todas as contribuições foram descritas em C++ e incorporadas ao software de referência do padrão, o HEVC Model (HM) (KIM, MCCANN, *et al.*, 2013), a fim de extrair os resultados, tanto de análise como resultados de impacto do

SCC na codificação. As condições de teste utilizadas foram ditadas pelo próprio grupo JCT-VC, descritas em (BOSEN, 2011).

Diversas versões do SCC foram investigadas, utilizando diferentes técnicas de alocação de recursos e controladores distintos. Para eleger a melhor configuração, resultados de eficiência de compressão e também de qualidade foram comparados. A melhor configuração de SCC foi então submetida a um conjunto extra de análises, a fim de avaliar a adaptabilidade e a controlabilidade do sistema, assim como seu desempenho comparado a trabalhos relacionados da literatura. Adicionalmente, resultados de tempo de codificação e de compressão foram avaliados, utilizando o software de referência como base de comparação.

B.3 Análise de Esforço Computacional na codificação HEVC

Essa seção descreve uma extensa análise do esforço computacional necessário para codificar vídeos utilizando o HEVC Model sob diversas configurações distintas. Partindo da configuração padrão do HM, as demais foram desenvolvidas variando-se um único parâmetro, a fim de analisar o impacto isolado de cada um na compressão e no esforço computacional. As conclusões dessa análise foram utilizadas para desenvolver o Sistema de Controle de Complexidade descrito na Seção B.4. Visto que o SCC utiliza os parâmetros de codificação para atingir diferentes níveis de esforço computacional, desejou-se conhecer os parâmetros que sejam capazes de fazer isso com o menor impacto possível em compressão.

B.3.1 Métricas de Complexidade

Para mensurar a o esforço computacional de cada teste, duas métricas foram utilizadas: a primeira é o tempo de execução tradicionalmente utilizado em trabalhos na literatura; a segunda é baseada em uma modelagem menos ligada à plataforma de execução, chamada Complexidade Aritmética (Arithmetic Complexity – AC), a qual é dada pela fórmula a seguir.

$$AC = N_{SAD} \cdot c_{SAD} + N_{SSE} \cdot c_{SSE} + N_{SATD} \cdot c_{SATD} + N_T \cdot c_T \quad (B1)$$

Em (B1), N_{op} e c_{op} representam o número de chamadas à função op e o número de ciclos gastos nessa operação respectivamente. Já que o HEVC trabalha com tamanhos de bloco variáveis, esses valores são normalizados para uma CU 64x64 (i.e., cálculos de um SAD 32x32 contam como 1/4 de uma operação de SAD). Da mesma forma, os valores de c_{op} também valem para uma CU 64x64. Os valores utilizados para o número de ciclos nesta análise são baseados em referências na literatura.

Todavia, para considerar também os valores de compressão na análise, uma segunda métrica foi criada, a qual relaciona esforço computacional (medido em AC) e bitrate, chamada Eficiência de Compressão-Complexidade (*Rate-Complexity Efficiency* – RCE), calculada pela seguinte fórmula:

$$RCE = \frac{norm(\Delta AC)}{norm(\Delta B)} \quad (B2)$$

Em (B2), $norm(x)$ é a normalização do valor x para uma escala de 1 a 10. Isso foi feito para que os valores de complexidade e de compressão fossem relacionados na mesma magnitude. Ainda em (B2), ΔAC representa a redução de complexidade com relação à configuração inicial. De forma análoga, ΔB simboliza a diferença de bitrate. O significado da métrica RCE é direto: quanto maior a redução de complexidade, maior o RCE; todavia, o valor de RCE cai com acréscimos maiores no bitrate.

B.3.2 Configurações de Teste

Visto que os valores de ciclo são parametrizáveis na métrica AC, é possível customizar a plataforma alvo para os resultados de esforço computacional da codificação. Como estudo de caso neste trabalho, pesquisou-se na literatura arquiteturas de alto desempenho para as funções SAD, SSE, SATD e transformadas. Somente dois trabalhos foram encontrados: (NALLURI, ALVES and NAVARRO, 2013) e (AHMED, SHAHID and REHMAN, 2012). O primeiro propõe uma arquitetura altamente paralela para o cálculo SAD de uma CU 64x64, levando 64 ciclos para essa função, portanto foi esse o valor atribuído a c_{SAD} ; o segundo apresenta uma proposta de arquitetura para a transformada DCT de N pontos, a qual requisita 136 ciclos para calcular uma TU 32x32. Como existem 4 TUs 32x32 em uma CU 64x64, esse

valor foi multiplicado por 4, totalizando 544 ciclos para o parâmetro c_T . Nenhuma arquitetura para SATD foi encontrada, então considerou-se que a implementação dessa função no HM executa 4 vezes mais operações aritméticas que o SAD, portanto $c_{SATD} = 4 * c_{SAD}$. O mesmo valor de c_{SATD} foi estipulado c_{SSE} , pois uma multiplicação é certamente mais custosa que uma adição/subtração. A Tabela B.2 resume as configurações utilizados nas simulações, assim como as sequências de entrada.

Tabela B.2: Configurações utilizadas na análise de esforço computacional

Plataforma de Execução*	Intel Core i7, 3.4 Ghz, 4 cores, 16 GB de memória RAM,			
Co-processador**	$c_{SAD}=64$	$c_{SATD}=256$	$c_{SSE}=256$	$c_T=544$
Classe	Resolução	Vídeos (consulte Appendix A)	Taxa de quadros	Nº de quadros
A	2560x1600	<i>PeopleOnStreet</i>	30	64
B	1920x1080	<i>Kimono</i>	24	64
C	832x480	<i>BasketballDrill</i>	50	64
D	416x240	<i>BlowingBubbles</i>	50	64
F	1024x768	<i>ChinaSpeed</i>	30	64

*utilizado para resultados de tempo **utilizado para resultados de AC

B.3.3 Parâmetros Avaliados

O HM possui muitos parâmetros de codificação, e cada um afeta um ou mais componentes do codificador. Aqueles considerados mais relevantes neste trabalho são detalhados nos parágrafos a seguir:

- **Max Partition Depth (MAXCUd):** define a profundidade máxima na árvore de CTUs. Cada novo nível de profundidade representa uma nova subdivisão das CUs. As configurações padrão atribuem o valor 4 a esse parâmetro, então começando com 64x64, as CUs podem atingir o tamanho mínimo de 8x8 (3 subdivisões). Esse parâmetro tem impacto direto não somente no esforço computacional da predição, como também na codificação residual, já que cada nodo de CU também contém uma árvore RQT. O módulo de entropia também é menos usado com profundidades menores, pois menos operações de cálculo RD são feitas.
- **Search Range (SR):** determinar a área de busca da Estimação de Movimento (*Motion Estimation* – ME). Esse parâmetro usualmente faz com que a busca termine mais rapidamente (a não ser que outra condição de terminação ocorra), então esse parâmetro pode ser usado para reduzir as computações da inter-predição.
- **Asymmetric Motion Partition (AMP):** habilita/desabilita a avaliação das partições AMP para CUs na inter-predição. Esse parâmetro teve bastante impacto nas primeiras versões do software HM, mas as versões atuais apresentam heurísticas que reduzem o número de AMP avaliadas, além de eliminar a busca para essas partições. Todavia, o custo RD ainda é calculado muitas vezes para cada quadro, então esse parâmetro não pode ser desconsiderado.
- **Hadamard ME (HadME):** habilita o cálculo de SATD durante a ME fracionária. Esse parâmetro também afeta o esforço computacional da inter-predição, pois o SATD envolve mais operações aritméticas que o SAD.
- **Max TU Depth (MaxTUD):** como o nome sugere, define a profundidade máxima na árvore de TU. Isso afeta o esforço computacional dos módulos de transformadas, quantização e entropia, já que envolve o número de partições avaliadas na decisão das TUs. O valor padrão é 4, então uma TU pode ter um tamanho máximo de 32x32 e mínimo de 4x4 amostras.
- **RDOQ:** habilita a técnica RDOQ, através da qual diferentes valores de QP são avaliados para cada coeficiente, elegendo os que atingem o melhor custo RD. A quantização e a entropia são afetadas por esse parâmetro.

Com base nesses parâmetros, diversas configurações de execução foram montadas, como mostrado na Tabela B.3.

Tabela B.3: Configurações utilizadas nas análises (c0: configuração padrão)

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
AMP	1	0	1	1	1	1	1	1	1	1	1	1	1
FME	1	1	0	1	1	1	1	1	1	1	1	1	1
Had ME	1	1	1	0	1	1	1	1	1	1	1	1	1
Max. CTUd	4	4	4	4	1	2	3	4	4	4	4	4	3
Max. RF	4	4	4	4	4	4	4	1	4	4	4	4	3
SR	64	64	64	64	64	64	64	64	16	32	8	64	64
Max TUD	3	3	3	3	3	3	3	3	3	3	3	1	2

Os valores Max. RF e FME não são parâmetros de codificação, mas podem ser manuseados alterando-se o código de referência. Max. RF representa o número máximo de quadros de referência utilizados na inter-predição, enquanto que FME corresponde à disponibilidade da ME fracionária.

Note que a configuração c0 corresponde às condições de teste inicialmente encontradas no HM. As demais configurações foram desenvolvidas com o intuito de codificar os vídeos em um tempo menor que o tempo de c0. Com base nisso, os valores ΔAC e ΔB da configuração c1 correspondem à diferença com c0, assim como nas demais. O valor de RCE de cada configuração é mostrado na Figura B.2.

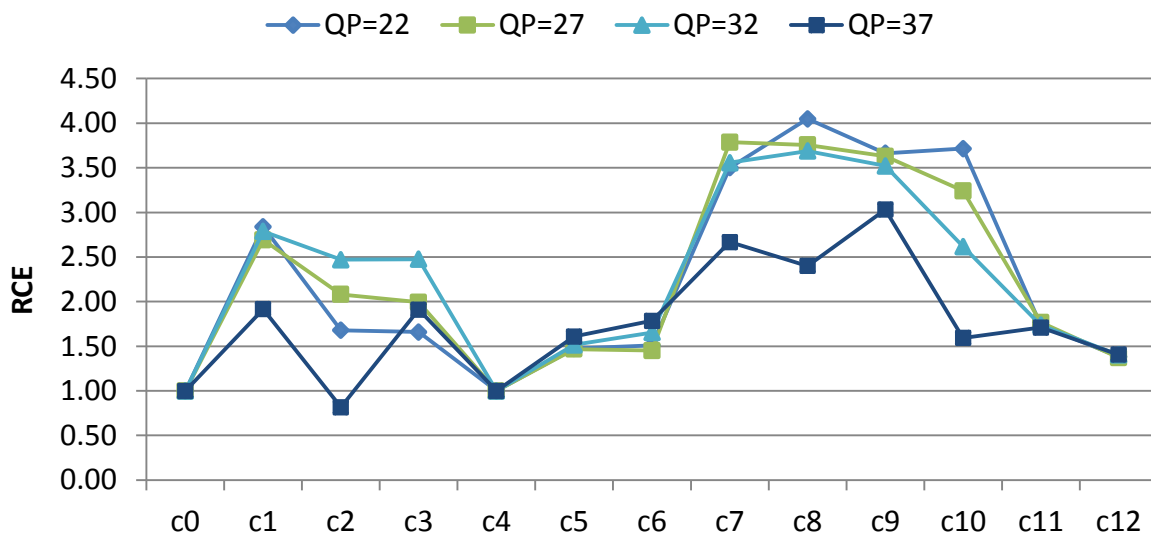


Figura B.2: Resultados de RCE para cada configuração (c0 utilizada como base de comparação)

Com base nesses resultados, os parâmetros que possuem o melhor RCE são: SR em primeiro lugar, pois o grafo atinge seu pico quando SR vale 16 (c8); restringir o número de quadros de referência (Max. RF) para 1 (c7) também se mostrou uma solução com alto RCE, colocando esse parâmetro em segundo lugar; em quarto lugar fica o parâmetro AMP (desabilitado na configuração c1), seguido por HadME (c3) ou FME (c2), dependendo do valor de QP. Note que o parâmetro MaxCUd, recorrentemente utilizado em trabalhos relacionados para reduzir a complexidade, atingiu um bom valor de RCE. Isso acontece pois, mesmo que esse parâmetro seja capaz de reduzir o esforço computacional consideravelmente, isso vem acompanhado de um alto incremento de bitrate, o que baixa o resultado da métrica.

B.4 Controle Computacional em Codificadores HEVC

Nesse trabalho, um Sistema de Controle de Complexidade (SCC) foi desenvolvido. Diferentes implementações de alguns componentes foram estudadas, de forma a garantir uma solução eficiente. O modelo do SCC é ilustrado na Figura B.3. Cada componente é descrito a seguir.

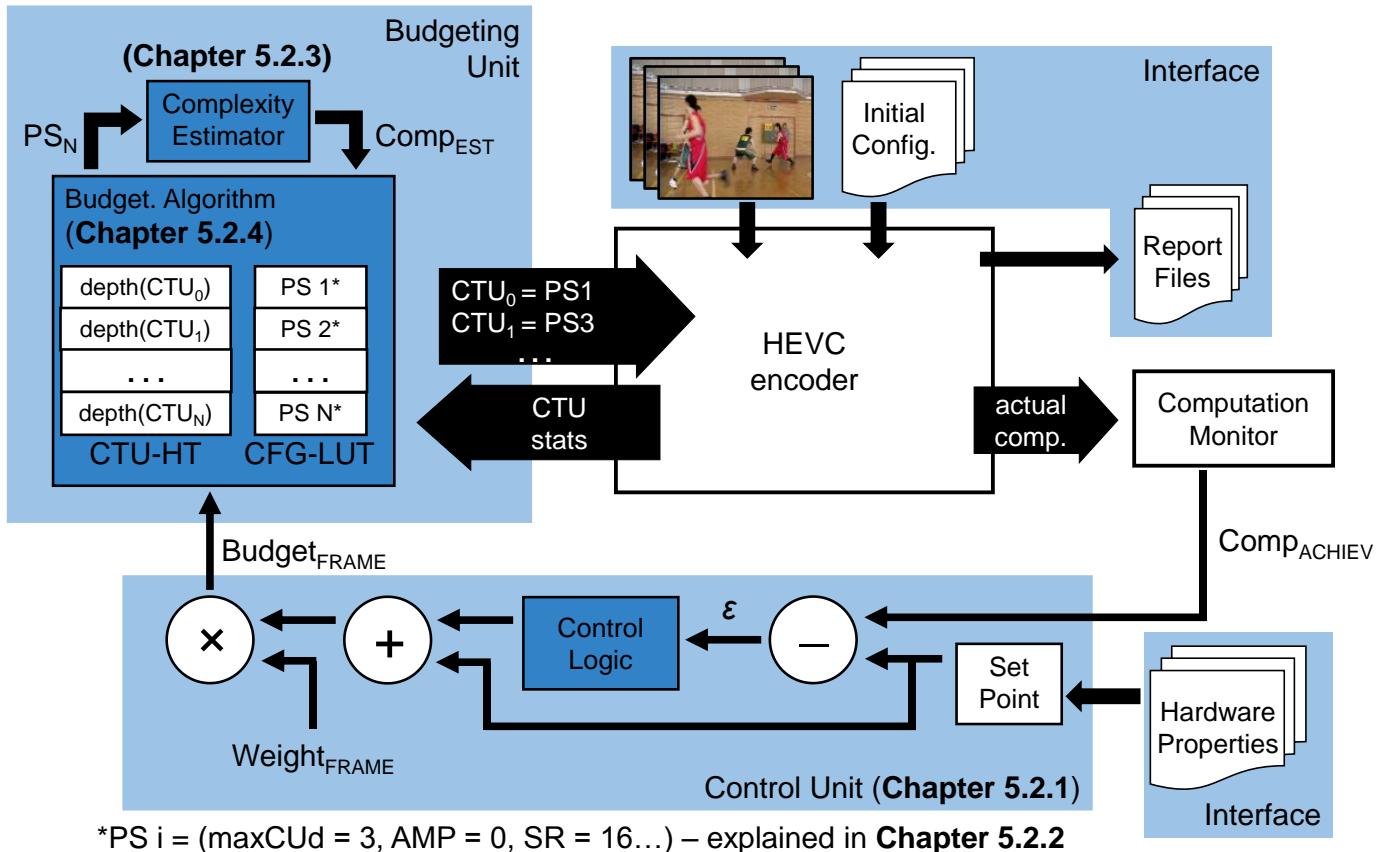


Figura B.3: Diagrama do Sistema de Controle de Complexidade utilizado neste trabalho

- **Interface:** uma interface com o sistema no qual o codificador está inserido é assumida, através da qual é possível receber informações relevantes, como frequência de operação da CPU, disponibilidade da CPU, vida útil da bateria e FPS alvo. Essa informação é usada para calcular o alvo (*Set Point* – SP). Além disso, os quadros de entrada e a configuração inicial (c0) servem de entrada para o codificador. Dados importantes são arquivados na saída.
- **Computation Monitor:** esse componente monitora quanta computação está sendo efetivamente requerida pelo codificador. Os valores são acumulados enquanto o quadro é codificado. Quando um novo quadro começa, o valor acumulado é enviado para a Unidade de Controle.
- **Control Logic:** computa o orçamento do quadro ($Budget_{FRAME}$) baseado na diferença entre a computação atingida (que vem do Comp. Monitor) e do alvo (SP). Nesse trabalho, um controlador PID foi utilizado.
- **Configuration Lookup Table (CFG-LUT):** armazena os conjuntos de parâmetros necessários para atingir diferentes graus de redução do esforço computacional (de 0% a 80% de redução). Os conjuntos foram criados variando cada parâmetro até que a redução alvo fosse atingida. A ordem seguida foi baseada naqueles com maior RCE. Nesse trabalho foram criados 5 conjuntos de parâmetros com reduções alvo de 0%, 20%, 40%, 60% e 80%.
- **CTU History Table (CTU-HT):** armazena a profundidade máxima de cada CTU no quadro anterior. A profundidade é utilizada no algoritmo de alocação (budgeting) para classificar as CTUs que precisam de mais ou menos ciclos para codificar.
- **Complexity Estimator:** esse componente estima o esforço computacional que será requerido para codificar o próximo quadro de acordo com a alocação dos conjuntos de parâmetros para cada CTU.
- **Budgeting Algorithm:** essa unidade é responsável por distribuir o $Budget_{FRAME}$ entre as CTUs do quadro. Ele é composto da CTU-HT e da CFG-LUT. Esse é um componente chave, pois garante que mais ciclos são gastos em CTUs que são mais custosas para codificar devido a intensa

movimentação ou texturas complexas. De forma análoga, menos recursos são atribuídos a CTUs localizadas em regiões de baixa complexidade no quadro, como aqueles com ausência de movimento.

As próximas subseções detalham alguns aspectos do SCC. A seguir, diferentes análises são apresentadas, incluindo a comparação com os trabalhos relacionados e com a implementação inicial do HM (correspondente à configuração c0 na seção de análise de esforço computacional).

B.4.1 Controlador PID

O controlador PID recebe como entrada a diferença entre o SP e a computação efetivamente utilizada para codificar o vídeo, o que gera um sinal de erro (e). Com isso, para cada instante de tempo t , a saída é calculada pela seguinte fórmula:

$$C_{PID}(t) = K_p \varepsilon + K_i \sum_{i=0}^t \varepsilon(i) dt + K_d \frac{(\varepsilon(t) - \varepsilon(t - \Delta t))}{\Delta t} \quad (B3)$$

Esse controlador tenta fazer com que a saída alcance o SP a uma certa velocidade, a qual depende de três constantes, K_p , K_i , e K_d . Esses coeficientes devem ser cuidadosamente ajustados, já que configurações ineficientes podem causar fortes oscilações na saída ou mesmo levar a uma divergência. Existem diferentes técnicas para realizar esse ajuste para controladores PID. Nesse trabalho, o método de Ziegler-Nichols (ZIEGLER and NICHOLS, 1993), amplamente conhecido, foi utilizado.

B.4.2 Conjuntos de Parâmetros

A análise de esforço computacional com base na métrica RCE foi bastante útil para esse trabalho, pois com base nela priorizou-se os parâmetros utilizados na geração de conjuntos (Parameter Sets – PS). Cada conjunto foi idealizado com um alvo de redução de esforço computacional. Os parâmetros foram ajustados seguindo a ordem mencionada até que o alvo fosse atingido. A Tabela B.4 mostra os parâmetros resultantes, e a Tabela B.5, os resultados de redução e esforço computacional e de compressão/qualidade atingidos.

Tabela B.4: Conjuntos de Parâmetros (PSs) utilizados no SCC

Conjunto	AMP	Had ME	Max CUd	SR	Max TUD	Max RF
PS0	1	1	4	64	3	4
PS20	1	1	4	32	3	4
PS40	0	1	4	32	1	4
PS60	0	0	3	16	1	4
PS80	0	0	3	8	1	1

Tabela B.5: Resultados de tempo e de compressão/qualidade de cada PS

	Redução Alvo	Redução em AC	Redução em Tempo	ΔBR	$\Delta PSNR$ (dB)
PS20	20%	17%	22%	0.1%	0.02
PS40	40%	38%	44%	1.1%	-0.04
PS60	60%	68%	64%	7.8%	-0.27
PS80	80%	80%	72%	10.8%	-0.33

B.4.3 Algoritmo de Budgeting

O algoritmo utilizado no SCC, chamado *Priority-Based Budgeting* (PBB) prioriza as CTUs mais complexas, alocando mais recursos (através da configuração PS0) para elas. Em contrapartida, poucos recursos (PS80) são atribuídos às CTUs avaliadas como sendo de menor complexidade. As CTUs de média complexidade foram alocadas com uma quantidade intermediária de recursos (PS40). Para classificar a

complexidade das CTUs, utilizou-se a profundidade máxima atingida pela CTU colocalizada no quadro anterior (armazenada na CTU-HT). A Figura B.4 mostra a figura do algoritmo PBB.

Algorithm 3: Priority-Based Budgeting algorithm

```

input : Frame: current frame
input :  $B_F$ : frame budget

// Starts by assigning fix PSets based on collocated CTU
depth

1 for each CTU with Depth =1 in Frame do
    // assign few computations to every low complexity CTU
2   set( $PS_{80}$ , CTU);
3    $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_{80})$ ;
4 end

5 for each CTU with Depth =4 in Frame do
    // assign more computations to high CTU depths
6   set( $PS_0$ , CTU);
7    $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_0)$ ;
8 end

9 for each CTU with Depth =2,3 in Frame do
    // assign an intermediate amount of computations to other
    CTUs
10  set( $PS_{40}$ , CTU);
11   $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_{40})$ ;
12 end

    // Promotion/Demotion refinement in case there is more/less
    budget available

13  $Demote_{Depth} \leftarrow 1$ ;
14 while  $B_F \neq Est_{Cycles}$  do
15   for each CTU with  $Demote_{Depth}$  in Frame do
16     demote(CTU);
17      $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_x)$ ;
18   end
19    $Demote_{Depth} ++$ ;
20 end
21  $Promote_{Depth} \leftarrow 3$ ;
22 while  $B_F \neq Est_{Cycles}$  do
23   for each CTU with  $Promote_{Depth}$  in Frame do
24     promote(CTU);
25      $Est_{Cycles} \leftarrow \text{updateEstimation}(PS_x)$ ;
26   end
27    $Promote_{Depth} --$ ;
28 end

```

Figura B.4: Algoritmo PBB

Note que após a alocação inicial, existe um refinamento caso o número de ciclos seja diferente do alvo. Caso haja mais ciclos para gastar no quadro, ocorre uma etapa de promoção, na qual CTUs com poucos

recursos são promovidas (se PS40 foi atribuído, passa a ser PS20 e assim por diante). Da mesma forma, as CTUs são gradativamente demovidas (PS20 para a ser PS40) caso mais ciclos sejam gastos.

Os resultados do SCC composto por um controlador PID e o algoritmo de alocação de recursos PBB são apresentados na próxima seção.

B.4.4 Resultados

A primeira análise feita com o SCC averiguou a eficiência do controlador PID em atingir o alvo SP. Para isso, 64 quadros da sequência *BasketballDrill* foram codificados. A computação atingida (medida em AC) foi então comparada com o SP, como pode ser visto na Figura B.5.

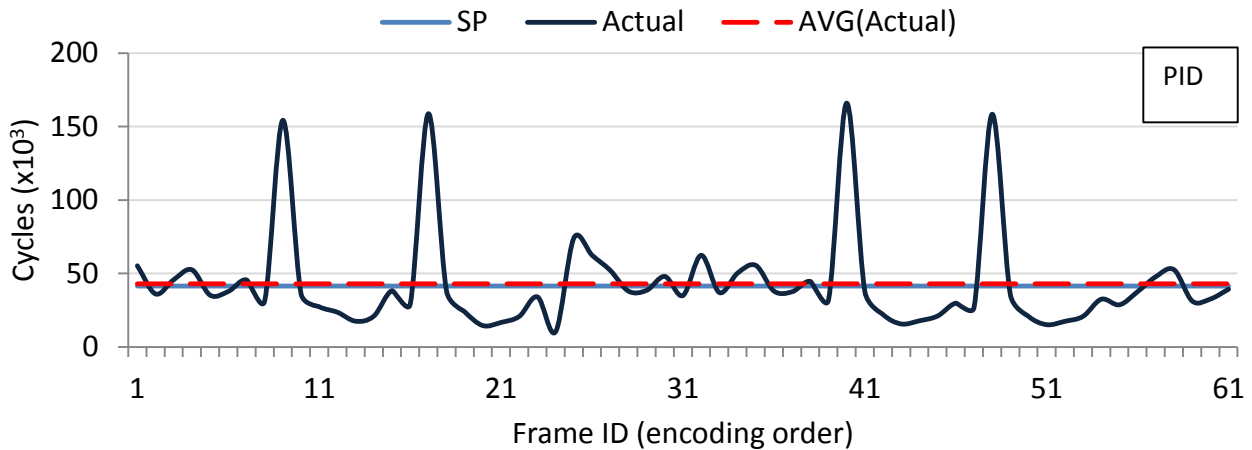


Figura B.5: Análise do controlador PID

É possível notar que a linha tracejada, representando a média do esforço computacional para os 64 quadros, está praticamente sobreposta com a linha que representa o SP (média 42.8k, SP: 41.5k), portanto conclui-se que o controlador PID atinge satisfatoriamente o alvo requisitado.

A segunda análise consistiu em codificar o vídeo *RaceHorses* (classe C) com o SCC habilitado e desabilitado em certos períodos, a fim de avaliar se o SCC se adapta rapidamente quando ele é requisitado no meio de uma codificação. Isso pode acontecer se, por exemplo, o usuário grava um vídeo no celular e de repente uma tarefa adicional é criada, dividindo os recursos computacionais do aparelho e portanto acionando o SCC. A Figura B.6 mostra o gráfico resultante.

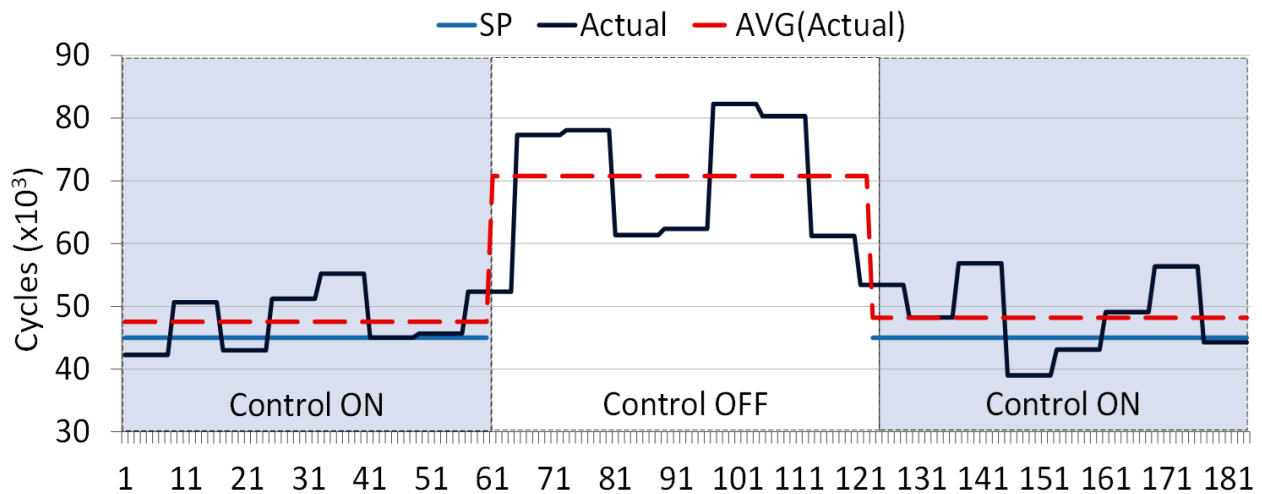


Figura B.6: Análise de adaptabilidade do SCC

Observa-se que, quando o SCC é desligado, o esforço computacional cresce em torno de 40%. Quando o SCC é novamente habilitado, o esforço computacional volta a baixar após aproximadamente dez quadros, mostrando que o sistema se adapta rapidamente quando necessário.

Para as próximas análises, diversos vídeos foram utilizados, a fim de se evitar resultados dependentes das características uma única sequência. A Tabela B.6 resume a configuração dessas simulações.

Tabela B.6: Configurações de teste para as análises do SCC

Vídeos (consulte Appendix A)	<i>Todas da classe A</i>
	<i>Todas da classe B</i>
	<i>Todas da classe C</i>
	<i>Todas da classe D</i>
	<i>Todas da classe F</i>
Contagem de Quadros	64
QP	22, 27, 32, 37
Frequência do Processador	2 GHz
Disponibilidade do Processador	50%, 60%
FPS alvo (varia de acordo com a sequência e com o QP)	Classe A: 2,4 – 3,5
	Classe B: 5 – 7
	Classe C: 23 – 30,3
	Classe D: 113 – 145,6
	Classe F: 15 – 17
Constantes PID	$K_p, K_i, K_d =$ {0,036, 0,18, 0,018}

Note que indicar que a disponibilidade da CPU é 60% é o mesmo que requisitar uma redução de 40% no esforço computacional. Isso se deve pelo fato de os valores de FPS alvo terem sido calculados com base em uma disponibilidade de 100% na configuração padrão.

Inicialmente, o SCC foi comparado com a configuração padrão do HM, a fim de avaliar se o esforço computacional é efetivamente reduzido. Além disso, é importante que se garanta a eficiência de compressão, mensurada pela métrica BD-BR. Os resultados para dois alvos de redução (40% e 50%) estão nas tabelas a seguir.

Tabela B.7: Resultados de BR-BR para todas as sequências usando as configurações RA e LB (redução alvo: 40%)

Alvo de Redução 40%	Random Access			Low Delay		
	Y	U	V	Y	U	V
Classe A	2,2%	2,2%	2,0%	--	--	--
Classe B	3,3%	3,8%	3,5%	2,7%	3,1%	3,4%
Classe C	3,3%	5,9%	5,0%	4,4%	6,8%	5,3%
Classe D	3,1%	2,1%	2,0%	7,1%	6,2%	5,8%
Classe F	6,8%	5,9%	6,4%	5,6%	3,7%	4,2%
Média	3,7%	4,0%	3,8%	4,8%	4,8%	4,6%
Redução de ciclos (%)	40%			40%		
Redução de Tempo (%)	34%			36%		

Tabela B.8: Resultados de BR-BR para todas as sequências usando as configurações RA e LB (redução alvo: 50%)

Alvo de Redução 50%	Random Access			Low Delay		
	Y	U	V	Y	U	V
Classe A	3,7%	3,8%	3,5%	--	--	--
Classe B	4,7%	5,1%	4,9%	3,6%	3,9%	4,3%
Classe C	4,5%	7,2%	6,6%	5,9%	7,9%	7,1%
Classe D	3,9%	2,8%	2,7%	8,9%	7,7%	7,7%
Classe F	11,6%	10,6%	11,1%	7,9%	6,0%	6,5%
Média	5,6%	5,9%	5,7%	6,4%	6,2%	6,3%
Redução de ciclos (%)	48%			50%		
Redução de Tempo (%)	43%			45%		

Os resultados mostram que uma economia significativa de tempo foi alcançada com um pequeno incremento na porcentagem de BD-BR (que representa acréscimo de bitrate). Na tabela Tabela B.7, os menores incrementos foram obtidos para as classes A (nos resultados para Random Access) e B (colunas Low Delay), indicando que o esforço computacional é melhor administrado para resoluções maiores. Isso se deve provavelmente ao fato de quadros HD possuírem um número maior de CTUs, permitindo uma distribuição mais fina. Os piores resultados foram para as sequências das classes F (RA) e D (LB). Os vídeos da classe F são voltados para conteúdo gerado pelo usuário (como pode ser observado na seção A.5), como jogos, videoconferências e apresentações de slides, portanto os piores resultados podem estar relacionados a essa particularidade. Resultados similares foram obtidos para redução alvo de 50%, como exposto na Tabela B.8.

A última comparação foi com trabalhos relacionados encontrados na literatura. A tabela a seguir mostra a comparação entre o HM-CMS a dois trabalhos relevantes relacionados a controle de complexidade. Já que nos dois casos os autores utilizaram tempo de codificação como métrica, esse valor foi extraído das simulações para realizar uma comparação justa. Os autores divulgaram resultados para mais de um alvo de complexidade, e os mais similares foram escolhidos para comparação com o HM-CMS.

Tabela B.9: Comparação com trabalhos relacionados (Alvo de redução: 40% 4 50%)

Trabalho	Redução Alvo	Redução de Tempo	Redução de Ciclos (AC)	ΔY -PSNR	ΔY -bitrate	BD-BR
HM-CMS	40%	35%	40%	-0,08	1,84%	4,28%
	50%	44%	49%	-0,12	2,9%	6,02%
(CORREA, ASSUNÇÃO, <i>et al.</i> , 2012a)*	40%	38%	N/A	-0,07	3,44%	6,29%
(CORREA, ASSUNÇÃO, <i>et al.</i> , 2011)*	40%	37,6%	N/A	-0,11	1,26%	N/A

* somente sequências da classe B

A comparação mostra que o HM-CMS é uma solução competitiva entre as referências encontradas, obtendo resultados de BD-BR superiores a ambas.

Para uma redução de 40% (primeira linha na Tabela B.9), o HM-CMS atingiram resultados muito similares de redução de tempo quando comparados a ambos os trabalhos. Os resultados de bitrate do HM-CMS são melhores que (CORREA, ASSUNÇÃO, *et al.*, 2012a), enquanto que os de PSNR são superiores aos de (CORREA, ASSUNÇÃO, *et al.*, 2011).

Ainda na Tabela B.9, os resultados para uma redução de 50% (segunda linha) mostram que o HM-CMS atinge uma redução de tempo maior em ambas comparações. Os resultados de bitrate também são melhores para o HM-CMS comparados aos de (CORREA, ASSUNÇÃO, *et al.*, 2012a).

É importante adicionar que o HM-CMS atingiu resultados mais acurados (com relação à proximidade ao alvo) do que os trabalhos comparados. Outra observação importante é que somente resultados para vídeos da classe B foram divulgados nas duas referências citadas, portanto os resultados para todas as sequências podem ser diferentes.

A próxima seção conclui essa dissertação com as considerações finais e as futuras investigações planejadas nessa pesquisa.

B.5 Conclusão e Trabalhos Futuros

Essa dissertação apresentou uma análise profunda do esforço computacional relacionado à codificação HEVC, seguida do desenvolvimento de um Sistema de Controle de Complexidade para codificadores HEVC. A primeira análise foi extremamente útil para entender como o esforço computacional se comporta e os parâmetros de codificação que mais impactam nesse aspecto. Essa análise utilizou métricas novas desenvolvidas nesse trabalho: a Complexidade Aritmética, e a Eficiência de Custo e Distorção. Particularmente a métrica AC se mostrou bastante conveniente, já que gera boas estimativas e pode ser customizada para diferentes plataformas de execução.

Diferentes soluções para o SCC foram estudadas, e a que atingiu os melhores resultados é composta de um controlador PID e algoritmo de alocação de recursos com prioridade (PBB). Definido isso, uma análise detalhada do sistema final foi realizada, comparando-o com o software de referência do padrão, assim como com trabalhos relacionados encontrados na literatura.

Os resultados de comparação com o software de referência HM mostraram que essa solução atinge uma redução de 45% no tempo de codificação com perdas pequenas em BD-BR (6.3% na média). Quando comparado a trabalhos relacionados, o HM-CMS se mostrou uma solução competitiva, atingindo resultados de BD-BR superiores aos dois trabalhos comparados.

Para que o CMS atinja uma redução ainda maior, é preciso investigar melhorias na implementação, especialmente no que diz respeito aos conjuntos de parâmetros (PS) criados (se um PS for capaz de reduzir o esforço computacional significativamente, reduções maiores podem ser atingidas). Isso será, no entanto, penalizado com acréscimos maiores no bitrate. Em segundo lugar, um refinamento na métrica AC também está planejado, a fim de considerar outros componentes da codificação como FME e RDOQ, obtendo estimativas cada vez mais fiéis. Além disso, implementações alternativas do controlador PID são estudadas, através da variação das constantes desse componente. Outros controladores também são considerados, como o *Model Prediction Controller*. Os algoritmos de *budgeting* também podem ser bastante explorados, utilizando conhecimento dos algoritmos de alocação de recursos estudados em Sistemas Operacionais. Por fim, soluções híbridas compostas de algoritmos de redução de esforço computacional em cada componente do codificador, juntamente com o um SCC, apontam bastantes descobertas em potencial, portanto serão devidamente pesquisadas.

REFERENCES

- AHMED, A.; SHAHID, M. U.; REHMAN, A. N-Point DCT VLSI Architecture for Emerging HEVC Standard. **Hindawi: VLSI Design**, v. 2012, p. 13, 2012.
- ASTROM, K. J.; HAGGLUND T. **PID Controllers: Theory, Design and Tuning**. 2. ed. [S.l.]: ISA: The Instrumentation, Systems, and Automation Society, 1995.
- BJONTEGAARD, G. **Calculation of Average PSNR Differences between RD-curves**. ITU-T. [S.l.]. 2001.
- BJONTEGAARD, G. **Improvements of the BD-PSNR model**. ITU-T. Berlin. 2008.
- BOSSEN, F. **Common test conditions and software reference configurations**. JCT-VC. Geneva. 2011.
- BROSS, B. et al. **High Efficiency Video Coding (HEVC) text specification Working Draft 5**. JCT-VC. [S.l.]. 2011.
- CASSA, M. B.; NACCARI, M.; PEREIRA, F. **Fast Rate Distortion Optimization for the Emerging HEVC Standard**. Picture Coding Symposium (PCS). Krakow: Institute of Electrical and Electronics Engineers (IEEE). 2012. p. 493-496.
- CHOI, K.; PARK, S.-H.; JANG, E. **Coding tree pruning based CU early termination**. JCT-VC. Torino. 2011.
- CISCO. Cisco Visual Networking Index: Forecast and Methodology. **CISCO**, 2012. Disponível em: <www.cisco.com/>. Acesso em: 19 jun. 2012.
- CORREA, G. et al. Complexity control of high efficiency video encoders for power-constrained devices. **IEEE Transactions on Consumer Electronics**, 57, n. 4, November 2011. 1866 - 1874.
- CORREA, G. et al. Complexity control of high efficiency video encoders for power-constrained devices. **IEEE Transactions on Consumer Electronics**, 57, n. 4, November 2011. 1866 - 1874.
- CORRÊA, G. et al. **Adaptive coding tree for complexity control of high efficiency video encoders**. Picture Coding Symposium (PCS). Krakow: Institute of Electrical and Electronics Engineers (IEEE). 2012. p. 425-428.
- CORREA, G. et al. Performance and Computational Complexity Assessment of High-Efficiency Video Encoders. **Circuits and Systems for Video Technology, IEEE Transactions on**, v. 22, n. 12, p. 1899 - 1909, December 2012.
- CORREA, G. et al. **Adaptive coding tree for complexity control of high efficiency video encoders**. Picture Coding Symposium (PCS). Krakow: Institute of Electrical and Electronics Engineers (IEEE). 2012a. p. 425-428.
- CORREA, G. et al. Performance and Computational Complexity Assessment of High-Efficiency Video Encoders. **Circuits and Systems for Video Technology, IEEE Transactions on**, v. 22, n. 12, p. 1899 - 1909, December 2012b.
- GRELLERT, M. et al. **An adaptive workload management scheme for HEVC encoding**. IEEE International Conference on Image Processing (ICIP). Melbourne: [s.n.]. 2013. p. 1850 - 1854.

- HUIJIBERS, E. A. M.; OZCELEBI, T.; BRIL, R. J. **Complexity scalable motion estimation control for H.264/AVC**. 2011 IEEE International Conference on Consumer Electronics (ICCE). [S.l.]: IEEE. 2011. p. 49-50.
- HUIJIBERS, E. A. M.; OZELEBI, T.; BRIL, R. J. **Complexity scalable motion estimation control for H.264/AVC**. 2011 IEEE International Conference on Consumer Electronics (ICCE), 2011. 49-50.
- INTEL. Intel® 64 and IA-32 Architectures, September 2013. Disponível em: <<http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf>>. Acesso em: January 2014.
- INTEL. Intel® VTune™ Amplifier XE 2013, 2013. Disponível em: <<http://software.intel.com/en-us/intel-vtune-amplifier-xe>>. Acesso em: January 2014.
- INTEL CORPORATION. **Intel® 64 and IA-32 Architectures**. [S.l.]. 2013.
- ITU-T. **ITU-T Recommendation H.264/AVC (05/03): advanced video coding for generic audiovisual services**. International Telecommunication Union. [S.l.]. 2003.
- ITU-T. **ITU-T Recommendation H.264/AVC: advanced video coding for generic audiovisual services**. International Telecommunication Union. [S.l.]. 2003.
- ITU-T. **ITU-T Recommendation H.265: High efficiency video coding**. ITU-T. [S.l.]. 2013.
- ITU-T. **ITU-T Recommendation H.265: High Efficiency Video Coding**. International Telecommunication Union. [S.l.]. 2013.
- JCT-VC. **Vision, Applications and Requirements for High Efficiency Video Coding (HEVC)**. JCT-VC. Daegu. 2011.
- JIANG, W.; HANJIE, M.; YAOWU, C. **Gradient Based Fast Mode Decision Algorithm for Intra Prediction in HEVC**. Consumer Electronics, Communications and Networks (CECNet). Yichang: Institute of Electrical and Electronics Engineers (IEEE). 2012. p. 1836-1840.
- JOINT COLLABORATIVE TEAM ON VIDEO CODING (JCT-VC). **Vision, Applications and Requirements for High Efficiency Video Coding (HEVC)**. JCT-VC. Daegu. 2011.
- JOO, J.; CHOI, Y.; LEE, K. **Fast sample adaptive offset encoding algorithm for HEVC based on intra prediction mode**. IEEE Third International Conference on Consumer Electronics - Berlin (ICCE-Berlin). [S.l.]: [s.n.]. 2013. p. 50 - 53.
- KANNANGARA, C. S. **Complexity Management of H.264/AVC Video Compression**, 2006. Disponível em: <<https://openair.rgu.ac.uk/bitstream/10059/643/1/Kannangara%20PhD.pdf>>. Acesso em: 2013.
- KANNANGARA, C. S. **Complexity Management of H.264/AVC Video Compression**. [S.l.]. 2006.
- KANNANGARA, C. S.; RICHARDSON, I. E.; MILLER, A. J. **Computational Complexity Management of a Real-Time H.264/AVC Encoder**. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 18, n. 9, p. 1191-1200, 2008.
- KIM, I.-K. et al. **High Efficiency Video Coding (HEVC) Test Model 10 (HM10) Encoder Description**. Joint Collaborative Team on Video Coding. [S.l.]. 2013.
- KIM, J. et al. **Fast Intra Mode Decision of HEVC based on Hierarchical Structure**. 8th International Conference on Information, Communications and Signal Processing (ICICS). Singapore: Institute of Electrical and Electronics Engineers (IEEE). 2011. p. 1-4.
- KIM, J. et al. **Early Determination of Mode Decision for HEVC**. Picture Coding Symposium (PCS). Krakow: Institute of Electrical and Electronics Engineers (IEEE). 2012. p. 449-452.

- LENG, J. et al. **Content Based Hierarchical Fast Coding Unit Decision Algorithm For HEVC**. International Conference on Multimedia and Signal Processing (CMSP). Guilin: Institute of Electrical and Electronics Engineers. 2011. p. 56-59.
- LI, B.; SULLIVAN, G. J.; XU, J. **Comparison of Compression Performance of HEVC Draft 8 with AVC High Profile and Performance of HM8.0 with Different Delay Characteristics**. JCT-VC. [S.l.]. 2012.
- MCCANN, K. et al. **High Efficiency Video Coding (HEVC) Test Model 10 (HM 10) Encoder Description**. [S.l.]. 2013.
- MIRTAR, A.; DEY, S.; RAGHUNATHAN, A. **Adaptation of video encoding to address dynamic thermal management effects**. Green Computing Conference (IGCC), 2012 International. [S.l.]: IEEE. 2012. p. 1-10.
- MIYAZAWA, K. et al. **Complexity Reduction of In-Loop Filtering for Compressed Image Restoration in HEVC**. Picture Coding Symposium (PCS). Krakow: Institute of Electrical and Electronics Engineers (IEEE). 2012.
- NALLURI, P.; ALVES, L. N.; NAVARRO, A. **A novel SAD architecture for variable block size motion estimation in HEVC video coding**. International Symposium on System on Chip (SoC). [S.l.]: [s.n.]. 2013. p. 1 - 4.
- OGATA, K. **Modern Control Engineering**. 5th Edition. ed. [S.l.]: Pearson, 2013.
- OU, Y.-F. et al. Perceptual Quality Assessment of Video Considering Both Frame Rate and Quantization Artifacts. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 21, n. 3, p. 286-298, 2011.
- PENNEBAKER, W. B.; MITCHELL, J. L. **JPEG still image data compression standard**. 3^a ed. ed. [S.l.]: Springer, 1993.
- SCHWARZ, H.; HINZ, T.; SUEHRING, K. **JMVC software model Version 8.2**. [S.l.]. 2010.
- SHEN, L.; ZHANG, Z.; AN, P. Fast CU size decision and mode decision algorithm for HEVC intra coding. **IEEE Transactions on Consumer Electronics**, v. 59, n. 1, p. 207 - 213, February 2013.
- SHEN, X.; YU, L.; CHEN, J. **Fast coding unit size selection for HEVC based on Bayesian decision rule**. Picture Coding Symposium (PCS). Krakow: Institute of Electrical and Electronics Engineers (IEEE). 2012. p. 453-456.
- SHI, Y. et al. **Local saliency detection based fast mode decision for HEVC intra coding**. IEEE 15th International Workshop on Multimedia Signal Processing (MMSP). [S.l.]: [s.n.]. September 2013. p. 429 - 433.
- TAN, Y. H. et al. **Complexity scalable rate-distortion optimization for H.264/AVC**. 2009 16th IEEE International Conference on Image Processing (ICIP). [S.l.]: IEEE. 2009. p. 3397-3400.
- TENG, S.-W.; HANG, H.-M.; CHEN, Y.-F. **Fast mode decision algorithm for Residual Quadtree coding in HEVC**. IEEE Visual Communications and Image Processing (VCIP). Tainan: Institute of Electrical and Electronics Engineers (IEEE). 2011. p. 1-4.
- VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1885 - 1898, December 2012.
- VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **Circuits and Systems for Video Technology, IEEE Transactions on**, 22, n. 12, December 2012. 1885-1898.
- WIEGAND, T. et al. Overview of the H.264/AVC Video Coding Standard. **IEEE Transaction on Circuits and Systems for Video Technology**, 13, n. 7, julho 2003. 560-576.

- XIONG, J. et al. A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence, v. 16, n. 2, p. 559 - 564, February 2014.
- YAN, S. et al. **Back to Results Group-Based Fast Mode Decision Algorithm for Intra Prediction in HEVC**. Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS). [S.l.]: [s.n.]. 2012. p. 225 - 229.
- ZHAO, L. et al. **Fast Mode Decision Algorithm for Intra Prediction in HEVC**. IEEE Visual Communications and Image Processing (VCIP). Tainan: Institute of Electrical and Electronics Engineers (IEEE). 2011. p. 1-4.
- ZIEGLER, J. G.; NICHOLS, N. B. Optimum Settings for Automatic Controllers. **Journal of Dynamic Systems, Measurement, and Control**, v. 115, n. 2B, p. 220-222, June 1993.