HEVC Encoder Optimization and Decoding
Complexity-Aware Video Encoding

Thanuja Mallikarachchi

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

July 2017

© Thanuja Mallikarachchi 2017
Abstract

The increased demand for high quality video evidently elevates the bandwidth requirements of the communication channels being used, which in return demands for more efficient video coding algorithms within the media distribution tool chain. As such, High Efficiency Video Coding (HEVC) video coding standard is a potential solution that demonstrates a significant coding efficiency improvement over its predecessors.

HEVC constitutes an assortment of novel coding tools and features that contribute towards its superior coding performance, yet at the same time demand more computational, processing and energy resources; a crucial bottleneck, especially in the case of resource constrained Consumer Electronic (CE) devices. In this context, the first contribution in this thesis presents a novel content adaptive Coding Unit (CU) size prediction algorithm for HEVC-based low-delay video encoding. In this case, two independent content adaptive CU size selection models are introduced while adopting a moving window-based feature selection process to ensure that the framework remains robust and dynamically adapts to any varying video content. The experimental results demonstrate a consistent average encoding time reduction ranging from 55% – 58% and 57% – 61% with average Bjøntegaard Delta Bit Rate (BDBR) increases of 1.93% – 2.26% and 2.14% – 2.33% compared to the HEVC 16.0 reference software for the low delay P and low delay B configurations, respectively, across a wide range of content types and bit rates.

The video decoding complexity and the associated energy consumption are tightly coupled with the complexity of the codec as well as the content being decoded. Hence, video content adaptation is extensively considered as an application layer solution to reduce the decoding complexity and thereby the associated energy consumption. In this context, the second contribution in this thesis introduces a decoding complexity-aware video encoding algorithm for HEVC using a novel decoding complexity–rate–distortion model. The proposed algorithm demonstrates on average a 29.43% and 13.22% decoding complexity reductions for the same quality with only a 6.47% BDBR increase when using the HM 16.0 and openHEVC decoders, respectively. Moreover, decoder energy consumption analysis reveals an overall energy reduction of up to 20% for the same video quality.

Adaptive video streaming is considered as a potential solution in the state-of-the-art to cope with the uncertain fluctuations in the network bandwidth. Yet, the simultaneous consideration of both bit rate and decoding complexity for content adaptation with minimal quality impact is extremely challenging due to the dynamics of the video content. In response, the final contribution in this thesis introduces a content adaptive decoding complexity and rate controlled encoding framework for HEVC. The experimental results reveal that the proposed algorithm achieves a stable rate and decoding complexity controlling performance with an average error of only 0.4% and 1.78%, respectively. Moreover, the proposed algorithm is capable of generating HEVC bit streams that exhibit up to 20.03 %/dB decoding complexity reduction which result in up to 7.02 %/dB decoder energy reduction per 1dB Peak Signal-to-Noise Ratio (PSNR) quality loss.
Key words: Video Coding, HEVC, CU Size, Motion Classification, RD Optimization, Decoding Complexity, Decoding Energy, Decoding Complexity–Rate–Distortion Optimization, Decoding Complexity Control, Rate Control

Email: d.mallikarachchi@surrey.ac.uk

WWW: https://www.surrey.ac.uk/cvssp/people/thanuja_mallikarachchi/
Acknowledgements

I would like to express my sincere gratitude to my principle supervisor Dr. Anil Fernando, who guided me through my Ph.D. since September, 2013. He has been supportive since the beginning of this journey and has been a dedicated mentor who assisted me to expand my research potentials.

I would also take this opportunity to thank Prof. Adrian Hilton who was my co-supervisor. I thank him for the encouragements and advises during the my tenure as a Ph.D. student in Centre for Vision, Speech and Signal Processing (CVSSP) group. I must thank Dr. Hemantha Kodikara Arachchi, for being a friendly adviser in every aspect throughout this period. His indispensable advises and numerous discussions have helped my research as well as project works during this period. I’m also thankful to Dr. Dumindu Talagala, for being a supportive advisor throughout the period and guiding me through this journey to fulfill my research goals.

I’m also thankful to my lovely wife, for being with me and supporting me during these hardworking years. My love and gratitude goes to my parents and my brother who have supported and encouraged me throughout my entire life.
# Contents

List of Abbreviations ....................................................... viii

1 Introduction ...................................................................... 1
   1.1 Background and Motivation ........................................... 2
   1.1.1 Challenges in Video Compression with HEVC ............... 3
   1.1.2 HEVC Encoding Complexity: Problem Statement and Motivation 4
   1.1.3 HEVC Decoding Complexity: Problem Statement and Motivation 6
   1.2 Objectives and Methodology .......................................... 9
      1.2.1 Content Adaptive Fast Low-Delay Video Encoding in HEVC 9
      1.2.2 Decoding Complexity-Aware HEVC Encoding Using a Complexity–Rate–Distortion Model 10
      1.2.3 CTU Level Decoding Complexity and Rate Controlled Video Coding Framework for HEVC Encoding 11
   1.3 Thesis Outline ........................................................... 12

2 Overview and Related Works ............................................. 13
   2.1 Video Compression and Video Coding Standards .............. 13
   2.2 Overview of High Efficiency Video Coding ....................... 15
      2.2.1 HEVC Coding Structure ........................................ 16
      2.2.2 HEVC Prediction Modes ........................................ 19
      2.2.3 HEVC Post Processing and Parallel Processing Tools .... 20
   2.3 Fast Encoding Algorithms for HEVC ............................... 21
      2.3.1 Fast Inter-coding Techniques ................................ 22
      2.3.2 Fast Intra-coding Techniques ............................... 24
   2.4 HEVC Decoding Complexity and Energy Reduction .......... 25
   2.5 Test Video Sequences .................................................. 29
   2.6 Summary ................................................................. 30
3 Content Adaptive Fast HEVC Encoding  32
  3.1 Fast Inter Coding Techniques .......................... 32
    3.1.1 Motion Feature-Based CU Classification .............. 32
    3.1.2 RD Cost Threshold-Based CU Classification .......... 36
  3.2 Fast CU Size Selection ................................ 38
    3.2.1 Motion Feature-Based CU Size Selection .............. 38
    3.2.2 RD Cost Threshold-Based CU Size Selection .......... 39
  3.3 Proposed Fast Encoding Framework ........................ 40
    3.3.1 Joint CU Split Decision Prediction .................. 41
    3.3.2 Model Adaptation and Training ....................... 41
    3.3.3 Motion Vector Reuse in Motion Estimation .......... 45
    3.3.4 The Overall Fast Encoding Algorithm ................. 45
  3.4 Experimental Results and Discussion .................... 47
    3.4.1 Simulation Configurations and Performance Metrics .... 47
    3.4.2 Results and Performance Analysis ...................... 48
  3.5 Summary .............................................. 57

4 Decoding Complexity-Aware HEVC Video Coding  59
  4.1 Decoding Complexity Modelling .......................... 59
    4.1.1 Decoding Complexity Profiling ....................... 62
  4.2 Decoding-Complexity – Distortion Analysis and Modelling ... 64
    4.2.1 Decoding-Complexity – Distortion Analysis .......... 65
    4.2.2 Decoding Complexity-Distortion Optimization: Computing \( \lambda_c \) 67
  4.3 Decoding-Complexity – Rate – Distortion Optimized Video Encoding 71
  4.4 Experimental Results and Discussion .................... 75
    4.4.1 Simulation Environment ............................... 75
    4.4.2 Evaluation Metrics ................................. 76
    4.4.3 Performance Evaluation and Analysis .................. 76
  4.5 Summary .............................................. 88
5 Joint CTU Level Decoding Complexity, Rate Controlled Video Coding

5.1 Decoding complexity, Rate and Distortion Relationship

5.1.1 Decoding-Complexity, Rate and Distortion Space

5.1.2 The Decoding-Complexity, Rate and Distortion Behaviour

5.2 Joint Decoding Complexity and Rate Control

5.2.1 CTU level Rate and Decoding Complexity Allocation

5.2.2 Determining QP, $\bar{\lambda}_r$, and $\bar{\lambda}_c$

5.2.3 Dynamic Model Parameter Adaptation

5.3 Experimental Results and Discussion

5.3.1 Simulation Environment

5.3.2 Evaluation Metrics

5.3.3 Performance Evaluation and Analysis

5.4 Summary

6 Conclusions and Future Work

6.1 Concluding Remarks: Content Adaptive Fast CU Size Selection

6.2 Concluding Remarks: Decoding Complexity-Aware HEVC Encoding

6.3 Concluding Remarks: Decoding Complexity and Rate Controlled Video Coding

6.4 Overall Conclusion

6.4.1 Benefits to the Content Creators and Service Providers

6.4.2 Benefits to the End Users

6.5 Future Work

A List of Publications

Bibliography
List of Figures

Figure 1.1 An illustration of the compression performance and encoding complexity of HEVC and H.264 reference encoders ........................................ 5
Figure 1.2 An illustration of the decoding complexity of HEVC and H.264 encoded bit streams ................................................................. 7

Figure 2.1 A high level HEVC compatible encoder design with key processing blocks ................................................................. 16
Figure 2.2 Partitioning of a CTU into multiple CUs ........................................... 17
Figure 2.3 A representation of PU modes used in HEVC .................................. 17
Figure 2.4 A representation of residual quadtree structure within a CU ........... 18
Figure 2.5 The concept of (a) Slices, (b) Tiles and (c) WPP in HEVC ............ 21
Figure 2.5 A graphical representation of the test video sequences ............... 30

Figure 3.1 CU partitioning and motion vector distribution of a typical HEVC encoded video frame ...................................................... 33
Figure 3.2 CU categorization based on motion characteristics. ..................... 34
Figure 3.3 Distribution of normalized proportion (%) of CUs that are split across feature vector parameters ................................................. 35
Figure 3.4 Representation of CU split likelihood distribution ......................... 37
Figure 3.5 CU split thresholds and the fitted exponential models ................. 42
Figure 3.6 CU non-split thresholds and the fitted exponential models .......... 43
Figure 3.7 The relationship between motion categories $\alpha_i$ and motion vector reuse for respective PU instances ................................. 44
Figure 3.8 The proposed fast encoding algorithm for HEVC based low delay video encoding .................................................. 46
Figure 3.9 The effect of window length $W$ on the performance .................... 49
Figure 3.10 The effect of the complexity control parameter $\tau$ on the performance 49
Figure 3.11 The RD performance of the proposed and state-of-the-art algorithms. 50
Figure 3.12 The encoding time performance of the proposed and state-of-the-art algorithms. 51

Figure 4.1 A high level illustration of the relative complexity of HEVC decoding operations. 60
Figure 4.2 Modeling of estimated CPU cycles with the number of non-zero coefficients. 62
Figure 4.3 The distortion and decoding complexity variations in intra- and inter-predicted frames. 67
Figure 4.4 Typical decoding complexity vs. distortion behaviour. 68
Figure 4.5 Decoding complexity, distortion relationship averaged across a set of test sequences. 69
Figure 4.6 Decoding complexity, distortion relationship averaged across a set of test sequences (normalized). 70
Figure 4.7 The $\lambda_c$ vs QP relationship. 71
Figure 4.7 The distribution of $\Delta D$ and $\Delta R$ for different combinations of $\bar{\lambda}_r$ and $\bar{\lambda}_c$ value pairs. 74
Figure 4.8 The decoding complexity (CPU cycles)-distortion curves and rate-distortion curves for proposed algorithms. 84
Figure 4.9 The decoder energy consumption, distortion graphs. 85

Figure 5.1 The relationship among the decoding complexity ($cpp$), rate ($bpp$) and distortion (MSE) parameters for inter-predicted frames. 94
Figure 5.2 The relationship among the decoding complexity ($cpp$), rate ($bpp$) and distortion (MSE) parameters for intra-predicted frames. 95
Figure 5.3 An illustration of frame-wise percentage error between the allocated bits and actual bits. 107
Figure 5.4 An illustration of frame-wise percentage error between the allocated bits and actual bits (proposed algorithm only). 108
Figure 5.5 Frame-wise decoding complexity allocation error. 109
Figure 5.6 The variation of $\Delta C/\Delta$PSNR (i.e., $\Delta C/\Delta(\%/dB)$) for each bit rate. 110
Figure 5.7 The variation of $\Delta$PSNR for each bit rate. 111
Figure 5.8 The variation of $\Delta E/\Delta$PSNR (i.e., $\Delta E/\Delta(\%/dB)$) for each bit rate. 118

Figure 6.1 Potential benefits of the proposed research contributions. 125
List of Tables

<p>| Table 2.1 Raw video bit rates for a set of commonly used video resolutions | 14 |
| Table 2.2 8-tap DCT interpolation filter coefficients for 1/4-pel luma interpolation | 19 |
| Table 2.3 4-tap DCT interpolation filter coefficients for 1/8-pel chroma interpolation | 20 |
| Table 3.1 Average split likelihood (%) for CUs in each motion category† | 34 |
| Table 3.2 ( R^2 ) Goodness-of-Fit of the split and non-split thresholds | 38 |
| Table 3.3 Simulation setup and configurations | 48 |
| Table 3.4 Encoding time saving with respect to QP and the content | 53 |
| Table 3.5 Overall performance of the proposed algorithm (low delay P) | 54 |
| Table 3.6 Overall performance of the proposed algorithm (low delay B) | 55 |
| Table 4.1 Relative complexity levels of decoding operations for HEVC intra-predicted CUs | 63 |
| Table 4.2 Relative complexity levels of decoding operations for HEVC inter-predicted CUs | 63 |
| Table 4.3 Decoding complexity estimation performance of the proposed model | 77 |
| Table 4.4 Decoding complexity reduction performance of in the Random Access configuration | 78 |
| Table 4.5 Energy consumption† behaviour of the encoding algorithms using the openHEVC software decoder | 79 |
| Table 4.6 Energy consumption† behaviour of the proposed algorithm | 87 |
| Table 5.1 Rate controlling performance of the encoding algorithms | 103 |
| Table 5.2 Decoding complexity controlling performance of the proposed encoding algorithm | 105 |
| Table 5.3 Decoding complexity reduction performance | 112 |</p>
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Decoder energy reduction performance during a video streaming session</td>
<td>114</td>
</tr>
<tr>
<td>5.5</td>
<td>Decoding complexity and energy reduction per 1 dB quality loss for the proposed and state-of-the-art algorithms</td>
<td>115</td>
</tr>
</tbody>
</table>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMVP</td>
<td>Advanced Motion Vector Prediction</td>
</tr>
<tr>
<td>AVC</td>
<td>Advanced Video Coding</td>
</tr>
<tr>
<td>AVS</td>
<td>Audio Video Standard</td>
</tr>
<tr>
<td>BDBR</td>
<td>Bjøntegaard Delta Bit Rate</td>
</tr>
<tr>
<td>C-DVFS</td>
<td>Codec Dynamic Voltage Frequency Scaling</td>
</tr>
<tr>
<td>CABAC</td>
<td>Context Adaptive Binary Arithmetic Coding</td>
</tr>
<tr>
<td>CBF</td>
<td>Coded Block Flag</td>
</tr>
<tr>
<td>CE</td>
<td>Consumer Electronic</td>
</tr>
<tr>
<td>CFM</td>
<td>CBF Fast Mode</td>
</tr>
<tr>
<td>CIF</td>
<td>Common Intermediate Format</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal-Oxide-Semiconductor</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathod Ray Tube</td>
</tr>
<tr>
<td>CTU</td>
<td>Coding Tree Unit</td>
</tr>
<tr>
<td>CU</td>
<td>Coding Unit</td>
</tr>
<tr>
<td>DASH</td>
<td>Dynamic Adaptive Streaming over HTTP</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DPM</td>
<td>Dynamic Power Management</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processors</td>
</tr>
<tr>
<td>DST</td>
<td>Discrete Sine Transform</td>
</tr>
<tr>
<td>DVB</td>
<td>Digital Video Broadcasting</td>
</tr>
<tr>
<td>DVFS</td>
<td>Dynamic Voltage and Frequency Scaling</td>
</tr>
<tr>
<td>ESD</td>
<td>Early SKIP Detection</td>
</tr>
<tr>
<td>fps</td>
<td>Frames Per Second</td>
</tr>
<tr>
<td>FTTH</td>
<td>Fibre-To-The-Home</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of Picture</td>
</tr>
<tr>
<td>GPP</td>
<td>General Purpose Processor</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>HDR</td>
<td>High Dynamic Range</td>
</tr>
<tr>
<td>HEVC</td>
<td>High Efficiency Video Coding</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>HVS</td>
<td>Human Visual System</td>
</tr>
<tr>
<td>IDCT</td>
<td>Inverse Discrete Cosine Transform</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>JCT-VC</td>
<td>Joint Collaborative Team on Video Coding</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Square</td>
</tr>
<tr>
<td>LTE</td>
<td>Long-Term Evolution</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Pictures Expert Groups</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>MVC</td>
<td>Motion Vector Competition</td>
</tr>
<tr>
<td>MVD</td>
<td>Motion Vector Difference</td>
</tr>
<tr>
<td>MVM</td>
<td>Motion Vector Merging</td>
</tr>
<tr>
<td>NAL</td>
<td>Network Abstraction Layer</td>
</tr>
<tr>
<td>OTT</td>
<td>Over-the-top content</td>
</tr>
<tr>
<td>PMD</td>
<td>Pyramid Motion Divergence</td>
</tr>
<tr>
<td>POC</td>
<td>Picture Order Count</td>
</tr>
<tr>
<td>PPS</td>
<td>Picture Parameter Set</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>PU</td>
<td>Prediction Unit</td>
</tr>
<tr>
<td>RD</td>
<td>Rate-Distortion</td>
</tr>
<tr>
<td>RMD</td>
<td>Rough Mode Decision</td>
</tr>
<tr>
<td>RQT</td>
<td>Residual Quadtree</td>
</tr>
<tr>
<td>SAO</td>
<td>Sample Adaptive Offset</td>
</tr>
<tr>
<td>SD</td>
<td>Standard Definition</td>
</tr>
<tr>
<td>SEI</td>
<td>Supplementary Enhancement Information</td>
</tr>
<tr>
<td>SMD</td>
<td>Skip Mode Detection</td>
</tr>
<tr>
<td>SVC</td>
<td>Scalable Video Coding</td>
</tr>
<tr>
<td>TU</td>
<td>Transform Unit</td>
</tr>
<tr>
<td>UHD</td>
<td>Ultra High Definition</td>
</tr>
<tr>
<td>VCEG</td>
<td>Video Coding Experts Group</td>
</tr>
<tr>
<td>VCL</td>
<td>Video Coding Layer</td>
</tr>
<tr>
<td>WMV</td>
<td>Windows Media Video</td>
</tr>
<tr>
<td>WPP</td>
<td>Wavefront Parallel Processing</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Visual content has been an integral part of human lives since the inventions of the photographic film, image capturing technologies, video recording devices and television systems in the early 20th century. Video consumption that once dominated by the analog cameras, mechanical and Cathod Ray Tube (CRT) Television systems filled with low resolution interlaced video contents has now been shifted to High Definition (HD)[1] and Ultra High Definition (UHD)[2] video/image capturing devices, high resolution video displays with increased dynamic range and colour gamut that capture and display near-perfect video contents. In essence, the numerous technological advancements coupled with the evolving user habits on video consumption have dramatically changed the media landscape that we experience today. Yet, the transformation from analogue video signals to digital domain in 1980s has laid the foundation for the exponential growth in the field of video/image processing.

Migrating video signals from analog to digital has countless benefits. The digital signals are less prone to errors and provide a scalable and flexible mode to represent and process data. However, digital formats demand more bandwidth to transmit video signals; a scarce resource that is carefully managed by the network operators. Despite the recent advancements in communication engineering, the available channel capacity is inevitably inadequate to cater for the high resolution raw video contents. On the other hand, the continuous evolution of multimedia technologies have transformed how users consume video contents today. For example, the proliferation of mobile devices and the advancements of their display and media processing technologies have ultimately increased the mobile data traffic which has grown 18-fold over the last 5 years [3]. Hence, the progression and commercial aspects of digital video is tightly coupled with the digital video compression; a crucial element in modern video distribution tool chain. The continuous evolution of multimedia technologies which are driven by both technical and commercial aspects and ever-changing user habits of video consumption demand continuous improvements in video compression technologies. Hence, the dynamic nature of the video contents, complex video formats (i.e., UHD and High Dynamic Range (HDR) video formats) and media capturing and processing technologies make video compression a vibrant and predominant research area in the field of modern signal processing and video communication.
1.1. Background and Motivation

The main research focus of the video compression and video coding standards over the last two decades has been on the improvements relating to the compression efficiency. However, the increased amount of raw and compressed video data that require real-time processing at both content creation and consumption ends of the media distribution chain coupled with the emergence of complex video coding standards demand the video compression algorithms to be computational and energy efficient; an area where the modern video coding standards need further improvements. Therefore, reducing the computational complexity and energy consumption of video encoding and decoding devices, while keeping the coding efficiency intact has become a paramount research domain in the field of video compression and signal processing.

1.1 Background and Motivation

Digital video today has become a ubiquitous entity in numerous aspects. From traditional TV broadcasting to Over-the-top content (OTT) [4] streaming, video on-demand, video communication and teleconferencing have made digital video a global phenomenon that encapsulates both businesses and homes alike. For instance, the increasing popularity of video sharing services such as YouTube (with more than 400 hours of content being uploaded every minute with over 1 billion hours of content is watched everyday [5]), video streaming services such as Netflix (with over 98 million subscribers worldwide [6]) exemplify the prominent role that digital video is playing in our daily lives. Moreover, the recent developments in the Consumer Electronic (CE) technologies, content capturing capabilities of CE devices, proliferation of mobile video consumption and the popularity of HD, UHD video contents have made video data the most frequently exchanged type of content over the modern communication networks (e.g., mobile data traffic statistics suggest that video data is forecast to reach three-fourth of the overall mobile data traffic in 2019 [3]).

High resolution high frame rate video contents demand excessive bit rates that are unrealistic to be accommodated in current communication infrastructures; especially in wireless communication channels. Hence, the field of video communication and video compression is often pressured with both academic as well as industry oriented researches on finding mechanisms to further improve the compression efficiency. Thus, video compression has been improving continuously since early 1990s to cater for the upcoming video demands. Having said that, with the introduction of HD technologies and adoption of Digital Video Broadcasting (DVB), the H.264/Moving Pictures Expert Groups (MPEG)-4 Advanced Video Coding (AVC) video coding standard which was introduced in early 2003 has been the dominant and popular video compression standard as of today. However, it is imperative that the compression efficiency of H.264/AVC is becoming inadequate to cater for the exponentially growing video traffic that will occupy the future communication infrastructures. Thus, the continuous improvement of compression efficiency is paramount to cope with the ever increasing video demands.

In this context, early 2013 witnessed the emergence of a next generation video codec; High Efficiency Video Coding (HEVC) also known as H.265 and MPEG-H part 2 [7]. HEVC was designed specifically to cater for the upcoming high resolution video contents
and to effectively utilize the state-of-the-art parallel processing architectures available with modern General Purpose Processor (GPP)s as well as Digital Signal Processors (DSP)s [8]. HEVC was developed by the Joint Collaborative Team on Video Coding (JCT-VC) organization, a collaboration of two prominent standardization organizations; ISO/IEC MPEG and ITU-T Video Coding Experts Group (VCEG), and is estimated to achieve a 40-50% compression efficiency compared to the H.264/AVC [9]. However, the state-of-the-art constitutes multiple video coding standards introduced by various organizations and standardizing groups. For example, Windows Media Video (WMV) by Microsoft [10, 11], Theora by Xiph.org [12], Audio Video Standard (AVS) [13] and more recent VP8 and VP9 codecs by Google [14] are demonstrating a considerable market penetration. Yet, HEVC as the successor to the most popular video coding standard in the CE market (i.e., H.264 / AVC), has a greater potential to reach a wider audience and has already penetrated the CE market on a larger scale. However, despite the advancements shown in the compression efficiency, due to its assortment of coding modes and features, HEVC demonstrates numerous challenges on its usage with average CE devices with limited processing and energy resources, which are summarized as follows.

1.1.1 Challenges in Video Compression with HEVC

- The complexity incurred in the encoder when generating HEVC bit streams is identified as a conspicuous challenge in the HEVC compatible encoders. For example, the increased number of coding modes and features, superior prediction modes and advanced post processing operations make HEVC encoding complexity a crucial challenge for both mobile hand-held CE devices (i.e., smartphones, camcorders) as well as encoding servers.

- The decoding complexity and the associated energy consumption of the video playback devices are tightly correlated with the complexity of the content as well as the underlying codec. Thus, the comprehensive features and coding modes in HEVC results in a complex bit stream requiring addition processing and energy resources from the video decoders for a seamless real-time video playback. Therefore, the substantial increase of the decoding complexity of a HEVC bit stream has become evermore challenging for the resource constrained mobile hand-held devices.

- The video contents distributed via wireless communication channels are prone to numerous bit errors and packet losses. Like any other compressed bit stream, HEVC bit streams are also prone to transmission errors which result in a drastic video quality degradation at the user terminals due to the increased amount of spatial and temporal redundancies eliminated during the encoding process. Thus, improving error resilience techniques at the encoder and error concealment methods at the decoder while keeping the coding efficiency intact is a compelling challenge associated with the HEVC encoding architecture.

- HEVC constitutes numerous coding tools that support parallelization. For example, picture partitioning schemes such as Slices and Tiles allow sections of
the frame to be encoded/decoded independently enabling the multicore processing architectures to process each Tile/Slice independently as separate processes/threads. Yet, partitioning a video frame causes prediction breaks resulting in coding efficiency losses. Moreover, the determination of Tile/Slice size also requires a comprehensive analysis of the Central Processing Unit (CPU) usage, memory utilization and inter-process communication aspects etc. Thus, effective picture partitioning scheme that utilizes the full potentials of HEVC parallelization tools while keeping the coding efficiency intact is a crucial challenge associated with the HEVC video coding.

These compression challenges associated with HEVC require comprehensive research and engineering solutions as they directly impact both operational and maintenance costs of the content creators/distributions and end user’s quality of experiences. In this context, this thesis focuses on two such crucial challenges in HEVC; the increased computational complexity of the HEVC compatible encoders and the complexity and the associated energy consumption during the decoding process of HEVC encoded video bit streams. The following subsections provide a summarized background on these issues and introduce the motivation for the proposed work in this thesis.

1.1.2 HEVC Encoding Complexity: Problem Statement and Motivation

HEVC fulfills its primary objective through its vastly superior compression performance exhibited over its predecessor H.264/AVC. Experimental results report on average a 40-50% bit rate reduction in HEVC for the same video quality compared to H.264/AVC (coding efficiency improvements for certain video contents exceed 60%) [9]. HEVC constitutes an assortment of novel coding modes and coding features within its encoding architecture that facilitates it to achieve this compression performance compared to its predecessors. However, the increased complexity of these features in the HEVC architecture significantly increase the demand for computational time and energy [15]; a non-trivial bottleneck for resource-constrained CE devices such as smart phones and camcorders. The impact of HEVC’s improved coding features and coding modes on the coding efficiency and encoding complexity are graphically illustrated in the Figs. 1.1(a) and 1.1(b), respectively. For example, the Fig. 1.1(a) indicates that the HEVC demonstrates a considerable reduction in the bit rate for a given Peak Signal-to-Noise Ratio (PSNR) (dB) quality level. However, at the same time, a substantial increase in the encoding time is observed in HEVC HM 16.0 reference encoder, compared to the H.264/AVC JM 18.5 reference encoder implementation (ref. Fig. 1.1(b)). Therefore, efficient encoder designs that expedite the encoding process are crucially important for the realization of high frame rate and real-time video communication applications in CE devices.

Although HEVC is essentially based on a hybrid coding architecture similar to that of H.264/AVC, it is accompanied by an assortment of novel coding features such as efficient prediction modes, filtering modes, parallelization tools, and flexible coding structures (e.g., Coding Unit (CU), Prediction Unit (PU), Transform Unit (TU), etc.)
1.1. Background and Motivation

Figure 1.1: An illustration of the compression performance and encoding complexity of HEVC (HM 16.0) and H.264 (JM 18.5) reference encoders for 30 frames of “Kimono 1080p” HD sequence encoded at QPs 22, 27, 32, and 37 using I-B-P-B-P configuration. (a) The PSNR vs. bit rate variation. (b) The encoding time vs. bit rate variation.

[8]. The wide range of block sizes and combinations (i.e., $8 \times 8$ to $64 \times 64$) that it entails is one of the most important contributors towards the encoder’s improved efficiency, yet at the same time, is also a major source of the complexity within the HEVC architecture [16, 17]. This is mainly due to the brute force Rate-Distortion (RD) optimization required for the combinations of coding modes that determines the best coding configuration for a particular video content. For example, an average encoding time increase of 43% is reported in [17], due to a simple increase of the maximum CU size from $16 \times 16$ to $64 \times 64$. Therefore, the recent literature has predominantly proposed numerous mechanisms to reduce the complexity of the RD optimization that selects the best coding structure. In this context, the state-of-the-art fast encoding solutions generally utilize the depth correlation of spatial and temporal blocks, RD cost statistics of the CUs and the Inter $2N \times 2N$ prediction mode, feature-based offline
and online training approaches, etc. [18, 19], to determine the optimum CU size. Hence, the selection of the CU size now becomes a prediction, whose effectiveness will determine the output quality and the bit rate of the encoded content. However, the vast differences in video characteristics, and the availability of additional information in the encoding chain itself, have not been fully investigated nor have they been exploited in these prediction approaches in order to realize a consistent encoding time saving across a wide range of content types and quality settings. Thus, the potential exists to develop implementation-friendly encoding algorithms that can effectively trade-off the coding efficiency in order to gain a reduction of the computational complexity.

1.1.3 HEVC Decoding Complexity: Problem Statement and Motivation

The amount of video data processed by a CE device has shown a dramatic increase causing media entertainment a significant contributor in the overall energy consumed by the household CE devices [20]. Moreover, the vast amount of data being processed in real-time makes high resolution video playback on resource constrained mobile handheld CE devices (e.g., smart phones, tablets etc.) [21] increasingly challenging. The limited growth witnessed in the energy capacity of lithium ion batteries that power mobile CE devices are unable to cope with the ever-increasing energy demands requested by the resource intensive video processing applications. Moreover, the amount of resources consumed during a video playback is tightly coupled to the complexity of the video content as well as the compression format; thus, the proliferation of high resolution content and complex video coding algorithms can substantially affect the energy usage of a device to a larger extent. In this case, the state-of-the-art HEVC standard [8] although bandwidth efficient for high resolution content, demands significant computational resources (and therefore energy) [15] for complex HD and UHD video contents. For example, as graphically illustrated in the Fig. 1.2(a), the decoding complexity of HEVC encoded bit streams in terms of CPU cycles (measured using Valgrind/Callgrind [22]) shows an enormous increase compared to the decoding complexity of H.264 bit streams in JM 18.5 reference decoder. Moreover, the amount of decoding cycles consumed for a particular video quality level is extremely high with HEVC compared to the H.264 bit streams (Fig. 1.2(b)). In addition, Fig. 1.2(c) shows the decoding complexity measured in terms of CPU cycles consumed, when the respective bit streams are decoded using optimized ffmpeg based decoding tools [23]. Thus, reducing the decoding complexity of HEVC encoded video bit streams is crucially important to reduce the associated energy consumption in the decoding process. In this context, modifying the HEVC video bit stream generation process to minimize the decoding complexity is a potential solution to the problem that can achieve the expected quality while reducing the energy use [21]. In addition, algorithms that work to these ends, and that operate exclusively in the application layer, are highly desirable for their ability to seamlessly integrate with existing online and offline playback mechanisms and devices; hence, the modelling and controlling of the decoding complexity of a bit stream at the encoder becomes crucially important to achieve the goal of reducing a device’s energy consumption during video playback.

Traditionally, energy reductions in the video decoding devices are achieved by either im-
1.1. Background and Motivation

Figure 1.2: An illustration of the decoding complexity of HEVC (HM 16.0), H.264 (JM 18.5) reference decoders and ffmpeg decoding tools for 30 frames of “Kimono 1080p” HD sequence encoded at QPs 22, 27, 32, and 37 using I-B-P-B-P configuration. (a) The CPU cycles vs. bit rate variation. (b) The PSNR vs. CPU cycles variation. (c) The PSNR vs. ffmpeg CPU cycles variation.
1.1. Background and Motivation

proving the efficiency of the radio receiver interface (e.g., changes to hardware drivers), modifying the decoder architecture and decoding operations (e.g., changes to the video player’s architecture) or by modifying the media content to reduce the complexity of the decoding process (e.g., changes to the input video bit stream) [21]. The latter being in the domain of video coding algorithms, consists of simplistic approaches that alter the basic coding parameters such as the Quantization Parameter (QP), frame resolution, frame rate, etc. [24, 25], but have a significant impact on the perceived video quality as well as the coding efficiency. More state-of-the-art solutions manipulate the motion compensation filters and the in-loop filtering operations introduced in more recent coding standards such as H.264 and HEVC to reduce the decoding complexity [26, 27], or adopt Dynamic Voltage and Frequency Scaling (DVFS) [28, 29, 30, 31] techniques to reduce the decoder’s power consumption at the hardware level (if and where possible). However, in general, the state-of-the-art methods in the literature do not exploit the variations of the computational complexity that exist between different decoding operations to determine the optimum coding parameters at the encoder itself to reduce the decoder’s energy consumption. In addition, the rate–distortion–complexity models that have been studied in the past (although not applicable to state-of-the-art compression standards) rely on generating temporally or spatially scalable video bit streams [32, 33] to improve the decoder’s resource efficiency in tandem with the coding efficiency, but have also received limited attention.

On the content adaptation front, Hypertext Transfer Protocol (HTTP) based adaptive streaming has received prominent attention in the recent past. For example, the recent developments in HTTP based adaptive streaming solutions such as MPEG-Dynamic Adaptive Streaming over HTTP (DASH) is shown to be effective to deal with the uncertainties in the network bandwidth during a video streaming session. However, the continuous drain of the device’s energy capacity remains a crucial bottleneck for a seamless video playback affecting both video streaming as well as video communication applications. Following a similar approach to the adaptive streaming, the recent literature shows energy-aware video streaming solutions that have the ability to fetch less complex video segments based on the device’s remaining energy capacity [25, 24, 34, 35, 27]. However, the prevailing content adaptation logics typically follow indirect approaches such as reducing the bit rate, increasing QP, and down scaling spatial resolution etc., as attempts to reduce the device’s energy consumption, resulting in a poor visual quality. Hence, an encoding algorithm that can generate a video bit stream with a given bit rate and a decoding complexity constraint is seen as a potential solution that can overcome these issues and facilitate an efficient network bandwidth and device’s energy capacity adaptive video streaming sessions; a crucial area that has been overlooked in the recent literature. To this end, simultaneous control of both bit rate and decoding complexity during the video encoding phase require further investigations to develop implementation friendly, scalable engineering solutions for content preparation that enable much discussed green multimedia consumption in CE devices.
1.2 Objectives and Methodology

As illustrated in the Sec. 1.1 HEVC is on a trajectory to be the dominant video coding standard in the near future. However, complications that surround HEVC from being utilized more effectively with modern CE devices inevitably hinders its rate of adoption and affects capital and operational expenses of many businesses. For example, higher the complexity of the encoder, more the computational and processing power required in the encoding servers. Similarly, higher the complexity of the encoded bit streams, higher the resource and energy requirements of the decoding CE devices. Thus, research and engineering solutions to address these issues inevitably make a huge impact in both industry and academia.

In this context, the main objective of this thesis is to propose a set of algorithms and methodologies to expedite the encoding process of HEVC and provide an encoding platform to consider the decoder resource constraints during the encoding process to produce decoding complexity-aware HEVC bit stream. To this end, the following section defines the three main research objectives and a summary of the proposed methodology utilized in achieving them.

1.2.1 Content Adaptive Fast Low-Delay Video Encoding in HEVC

The first objective of this thesis is to introduce a fast low-delay encoding algorithm for HEVC encoders. In this context, this work presents a novel content adaptive CU size prediction mechanism for HEVC-based low-delay video encoding. The research contributions presented under this objective include,

- An algorithm that utilizes two independent content adaptive decision making models, which exploit the features extracted from within the encoding chain for preceding video frames, to enable the prediction of the optimal CU size for subsequent video frames. The proposed method first generates a dynamic motion feature-based CU classification model by initially evaluating the Inter $N \times N$ mode, which turns out to be more pertinent in terms of motion characterization [36][19] than the commonly used Inter $2N \times 2N$ mode for a CU classification involving both motion as well as RD cost characteristics. This model, together with a heuristic RD cost threshold-based model of the CU split decision, is used to predict the optimal CU size and thereby limit the brute force evaluation of the RD cost function; thus, reducing the encoding time and by extension the computational complexity.

- A window-based feature selection process is adopted in order to ensure that the framework remains robust and dynamically adapts to any varying content such as scene changes in a sequence.

- A methodology to utilize the information gathered during the Inter $N \times N$ mode evaluation, for motion estimation in order to further expedite the encoding process.

- A set of complexity control parameters within the algorithm that allow more flexibility in trading-off the complexity and coding efficiency for diverse applications.
1.2. Objectives and Methodology

Summary of Achievements

- The proposed encoding algorithm is capable of achieving an encoding complexity reduction of 55% – 58% and 57% – 61% with average Bjøntegaard Delta Bit Rate (BDBR) increases of 1.93% – 2.26% and 2.14% – 2.33% compared to the HEVC 16.0 reference encoder for the low delay P and low delay B configurations, respectively.

- The encoding complexity reduction achieved by the proposed encoding algorithm is observed to be consistent across multiple content types and quality levels; a crucial advantage over the state-of-the-art algorithms.

- The online content adaptive nature of the proposed algorithm makes the coding decisions attained during the encoding process are relevant to the content being encoded.

- The proposed algorithm is flexible in tuning its design parameters to effectively trade-off the coding efficiency against the encoding complexity depending on the application requirements.

1.2.2 Decoding Complexity-Aware HEVC Encoding Using a Complexity–Rate–Distortion Model

The second objective of this thesis is to introduce a decoding complexity-aware encoding algorithm for HEVC. This includes proposing a novel decoding complexity–rate–distortion model to facilitate the encoding process which considers the decoding complexity as an additional parameter in the mode selection cost function. In this context, the research contributions introduced in this thesis are,

- A novel CU level decoding complexity estimation model that models the complexity of the different decoding operations for both intra- and inter-predicted CUs with respect to a known decoder architecture.

- A comprehensive analysis of the relationship between the decoding complexity, rate and distortion to derive trade-off coefficients for the rate, and the decoding complexity with respect to the distortion for a given QP.

- An encoding framework which jointly utilizes the decoding complexity, rate and distortion to generate HEVC encoded bit streams that reduce the decoding complexity, and thereby the energy consumption, with minimal impact on the coding efficiency.

Summary of Achievements

- The decoding complexity estimation models proposed under this contribution is capable of predicting the CU level decoding complexity at the encoder with an average error of only 0.81% and 4.22% for intra, and inter-prediction, respectively.
1.2. Objectives and Methodology

• The proposed decoding complexity-aware video encoding algorithm is capable of assessing the impact of each coding mode on bit rate, decoding complexity and distortion prior to selecting the best coding mode for a particular content.

• The proposed encoding algorithm achieves a decoding complexity reduction of 29.43% and 13.22% on average when using HM16.0 and openHEVC decoder implementations, respectively, for a similar video quality to that of HM16.0 encoded bit streams. The average BDBR increase of the generated bit streams is measured to be only 6.47% compared to the HM16.0 encoded bit streams.

• The decoding complexity reduction achieved by the proposed encoding algorithm results in an overall energy reduction in the decoder of up to 20% for a similar video quality to that of HM16.0 encoded bit streams.

1.2.3 CTU Level Decoding Complexity and Rate Controlled Video Coding Framework for HEVC Encoding

The third and final objective of this thesis is to introduce an encoding framework that allows the content creators to generate HEVC encoded video bit streams that adhere to a given decoding complexity and a bit rate requirement. The research contributions under this objective can be summarized as follows.

• A decoding complexity, bit rate, and distortion model along with a mode selection cost function that involve both bit rate and decoding complexity constraints.

• An algorithm to derive appropriate coding parameters and bit rate and decoding complexity trade-off factors to meet a given bit rate and decoding complexity requirement while minimizing the resultant distortion.

• A content adaptive Coding Tree Unit (CTU) level rate, decoding complexity controlled video coding algorithm to generate HEVC compliant video bit streams that correspond to multiple bit rate and decoding complexity levels.

Summary of Achievements

• The proposed encoding algorithm is capable of controlling the bit rate of a video sequence with an average error of only 0.4% compared to the target bit rate.

• The decoding complexity controlling performance of the proposed algorithm results in only 1.78% average error compared to the target decoding complexity level for a particular video sequence.

• The experimental results reveal a 20.03(%/dB) decoding complexity reduction per each 1dB quality loss for the proposed encoding algorithm; a crucial advantage over the state-of-the-art approaches.

• The corresponding decoder energy reduction per each 1 dB quality loss is measured to be 7.02(%/dB) on average for the proposed encoding algorithm.
1.3 Thesis Outline

The proposed algorithm is capable preparing HEVC bit streams that can achieve multiple decoding complexity levels for a particular bit rate; a significant advancement compared to the state-of-the-art content preparation approaches to achieve decoding complexity-aware video streaming to resource constraint mobile devices.

1.3 Thesis Outline

The subsequent Chapters of this thesis provide detailed descriptions of each research objective and corresponding research contributions followed up by the experimental results for the underlying implementations. In this context, the remainder of this thesis is organized as follows.

- **Chapter 2:** This Chapter provides an overview of the HEVC and describes in detail the relevant coding modes and features that are mainly considered within the scope of the research objectives. Moreover, the Chapter 2 also illustrates the state-of-the-art related works available in the recent literature to overcome the two crucial challenges that the proposed works in this thesis attempt to address.

- **Chapter 3:** Next, Chapter 3 provides the details of the algorithmic derivations and implementation details for the research contributions proposed under the first objective; content adaptive fast low-delay video encoding in HEVC. These are followed up by the experimental results and a discussion on the performance of the proposed algorithms for numerous use cases and application scenarios.

- **Chapter 4:** This Chapter elaborates the details of the proposed algorithms, experimental setup, implementation details and experimental results and discussion for the proposed research contributions under the second objective; decoding complexity-aware HEVC video encoding.

- **Chapter 5:** The implementation details and experimental results presented for the final objective; decoding complexity and rate controlled video coding framework for HEVC encoding, are presented in Chapter 5.

- **Chapter 6:** Finally, Chapter 6 provides concluding remarks for the work presented under each research objective and potential future work for the proposed research contributions in this thesis.
Chapter 2

Overview and Related Works

This chapter first aims to provide an overview of the video compression and prevailing video coding standards; in particular the HEVC standard. The significance of the HEVC coding features to the coding efficiency and their impact on the encoding complexity are discussed followed up by a detailed description of the state-of-the-art methods available in the literature to expedite the HEVC encoding process. Thereafter, the complexity of the HEVC coded bit streams and its impact on the energy consumption in resource constrained video playback devices are described. Then, the discussion continues to elaborate the methodologies that are present in the recent literature to reduce the complexity of the HEVC decoding process and by extension the energy consumption of the HEVC compatible video playback devices. Finally, the drawbacks and crucial areas that have been overlooked in the recent literature are addressed to further elaborate the scope, significance and research potentials of the work that is proposed in this thesis.

2.1 Video Compression and Video Coding Standards

Video coding is the process of compressing a digital video content and converting it to a content representation format that is suitable for storage or transmission over a communication network. Despite the large capacities in the prevailing storage systems and transmission channels, compressing a video stream to reduce the amount of information exchanged is pivotal and has become evermore challenging. For example, the shift in video consumption from the low resolution video contents such as Common Intermediate Format (CIF), Standard Definition (SD) etc., 25-30 Frames Per Second (fps) video formats to very high resolution HD, UHD etc., 50-120 fps contents typically demand bandwidth capabilities that are far-fetched even with the state-of-the-art transmission technologies such as 4G Long-Term Evolution (LTE), Fibre-To-The-Home (FTTH) etc. The Table 2.1 for example, illustrates the raw video bit rates required for 8-bit video contents for a set of commonly used resolutions and formats. Therefore, it is apparent that the prevailing network infrastructures are not equipped to handle the raw video bit streams for real-time applications, nor does the capacities of commercial storage devices for long term content storage. In this context, the continuing growth of video
2.1. Video Compression and Video Coding Standards

Table 2.1: Raw video bit rates for a set of commonly used video resolutions

<table>
<thead>
<tr>
<th>Video resolution (W × H)</th>
<th>fps</th>
<th>Raw bit rate (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIF</td>
<td>352x288</td>
<td>25</td>
</tr>
<tr>
<td>HD 720p</td>
<td>1280x720</td>
<td>30</td>
</tr>
<tr>
<td>HD 1080p</td>
<td>1920x1080</td>
<td>30</td>
</tr>
<tr>
<td>4K UHD 2160p</td>
<td>3840x2160</td>
<td>60</td>
</tr>
<tr>
<td>8K UHD 4320p</td>
<td>7680x4320</td>
<td>60</td>
</tr>
</tbody>
</table>

resolutions, popularity of high frame rate video contents, emergence of advanced video formats such as HDR make video compression an essential element for both present and future multimedia services and applications.

A video signal, is essentially an information carrying signal which can be compressed by removing its statistical (spatial, temporal inter-pixel redundancies and coding redundancies) and psycho-visual redundancies (by exploiting the perception of visual information in the Human Visual System (HVS)) [37]. Therefore, the state-of-the-art video compression algorithms achieve a lossy or lossless compression of a video signal by exploiting these correlations within the spatial and temporal domains. However, the compression ratio of a lossless video coding algorithm is relatively small, thus in practice the videos are compressed using lossy video coding algorithms that make use of the properties of the human visual perception of visual content. In this case, the video compression becomes a challenging task to increase the compression ratio while minimizing the distortion introduced to the video signal during the compression process. Therefore, the field of video compression has been an active and dynamic research area for the past 20-25 years and numerous algorithms and methodologies have been introduced to the video compression work flows. However, the international organizations such as MPEG and VCEG which is a part of International Telecommunication Union (ITU) undertake the role of maintaining and standardizing the formats of the compressed bit streams and the decoding process ensuring that the content produced by numerous entities are playable across all platforms.

Since its inception in 1990 with H.261, the video coding standards have been evolved and incorporated with an assortment of coding features and tools to cater for the emerging and continuously changing multimedia requirements. In this context, the H.264/MPEG-4 AVC, which the first version was standardized in 2003, is the most prominent and widely adopted video coding standard for Internet streaming applications, HDTV broadcasting and digital cinema [38]. However, the increasing popularity of high resolution video contents (including HD, and UHD videos), the rapid growth in the mobile video consumption (which is expected to reach over 80% by 2019 [3]), advancements in CE technologies and capturing equipment that generate novel content formats, and emergence of interactive media applications (such as ACTION-TV[39]) have imposed severe strains on the existing communication networks demanding more efficient compression technologies that go beyond the capabilities of H.264/MPEG-4 AVC. Thus, the first version of HEVC or H.265/MPEG-H Part 2 was introduced and standardized in early 2013 by JCT-VC; a joint venture of the two prominent video coding standardization organizations, namely, MPEG and VCEG [8]. Moreover, the
version 2 of HEVC was finalized in early 2014 with extended capabilities that constitute multiview extension, range extension and scalability extension [40]. The encoding architecture and novel coding tools in HEVC are designed to achieve a 40-50% coding efficiency compared to its predecessor (H.264) to cater for the high resolution video demands, and efficiently utilize the parallel processing architectures available within the modern general purpose processors as well as Graphics Processing Unit (GPU) chips [8].

Although HEVC was introduced only a few years ago, it has been well adopted in the CE market. It can be observed that most of the modern set-top boxes include HEVC support, and commonly used video players such as VLC[41], DivX[42] and web browsers support HEVC encoded video contents. Moreover, the general purpose processors in modern personal computers are embedded with GPU assisted integrated HEVC decoding capabilities to support the novel coding standard (e.g., Intel Skylake microarchitecture [43]). In addition, the DVB has finalized its standardization process for UHD broadcasting with HEVC as the codec specification [44]. Therefore, it is evident that HEVC will surpass H.264/MPEG-4 AVC as the most prominent video coding standard in the near future, thus it is imperative that technologies and media processing work flows facilitate HEVC compatible content preparation. In this context, it is crucial that both the academia and industry focus on the challenges that HEVC entails with its vastly superior coding performance, that would expedite the use of HEVC as the standardized codec in the emerging media applications. To this end, the next section will summarize an overview of the HEVC coding architecture and describe the challenges that it entails while achieving a higher coding efficiency.

### 2.2 Overview of High Efficiency Video Coding

The HEVC standard now constitutes two versions; the HEVC version 1 which was standardized in early 2013 [7] [8], and HEVC version 2 (RExt of HEVC) [45] which was approved in 2014. However, the architectural design of the HEVC version 2 extension is designed to have minimal divergence from the initial HEVC version 1 architecture, thus the legacy decoders that only support version 1 can still process base layers of the HEVC version 2 encoded bit streams [40]. This flexibility in the HEVC design architecture inherently allows the usage of the proposed algorithms in this thesis with the RExt of HEVC as well. However, the core design, analytical and experimental analysis of the proposed algorithms are based on the version 1 of the HEVC standard, thus, the remainder of this section mainly focuses on the MPEG-H Part 2 - International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 23008-2 / ITU-T Recommendation H.265 specification of the HEVC standard [7].

One of the major goals in HEVC is to cater for the emerging high resolution video formats (i.e., 4k x 2k or 8k x 4k) such as UHD. Hence, the coding structure for a given content is made more flexible and content adaptive compared to its predecessor H.264/AVC. In addition, HEVC design architecture also considers the proliferation of parallel processing architectures, thus, a considerable amount of parallel processing tools have been incorporated within the HEVC coding tools. Therefore, while adopting
a block based coding architecture similar to its predecessors, HEVC introduces an assortment of novel coding tools and modes that improve its coding efficiency.

Following an approach that goes back to the early 1990s, the Video Coding Layer (VCL) in HEVC employs a hybrid coding architecture that constitute inter/intra prediction followed by a 2D transform coding. For example, the Fig. 2.1 depicts the key building blocks and their inter-connections of an HEVC compatible video encoder. A brief overview of the key features and operational aspects of each block in relation to the proposed work in this thesis are described next.

### 2.2.1 HEVC Coding Structure

Similar to H.264/AVC, block based prediction and compression is the baseline for HEVC. For example, each picture is split into block shaped regions and each block encapsulates a set of content dependent coding modes that are conveyed to the decoder. However, unlike H.264, HEVC supports a wider range of block sizes and possesses a more flexible quad-tree like partitioning structure to retain the prediction and residual information.

A picture is partitioned into CTUs having the size of $64 \times 64$ which is analogues to the $16 \times 16$ macroblocks in H.264/AVC. In the main profile of HEVC, a CTU is further partitioned into multiple CUs which the sizes range from $8 \times 8$ to $64 \times 64$ [17]. For example, Fig. 2.2(a) depicts the formation of a CTU and CU structures within a typical HEVC encoded video frame, and Fig. 2.2(b) illustrates the concept of coding tree, coding tree depth level and CUs. As illustrated in the Fig. 2.2, a CU can occupy the whole CTU or split into four sub-CUs with equal size, and are indicated as leaf nodes of the coding tree. Here, the numerical symbol “1” identifies that a CU is split, whereas “0” represents a leaf node (i.e, a non-split CU). In the case of HEVC, each CU
2.2. Overview of High Efficiency Video Coding

Figure 2.2: (a) CTU partitioning of a typical HEVC encoded video frame and quad-tree partitioning of CUs within a CTU. (b) A visualization of subdivision of a CTU into CUs within the quad-tree partitioning structure. The splitting of each CU into sub-CUs is indicated by the solid lines and the leaf nodes indicate the final CUs within the CTU.

Figure 2.3: A representation of PU modes used in HEVC. Here, the intra prediction only uses the PU modes identified by $2N \times 2N$, and $N \times N$. Moreover, the latter is only used with the smallest CU size, i.e., $8 \times 8$.

maintains the details of the prediction scheme (inter or intra prediction) that is used for the block. Experimental results demonstrated with respect to coding performance reveal that the flexible coding tree structure that adapts to various local characteristics of the content is a major source of its superior coding performance [17] compared to its predecessors.

A CU can constitute multiple PUs and TUs, which are used to maintain prediction and transform information, respectively. PUs encapsulate prediction information. For example, a PU of an inter-predicted CU holds the specific information relating to the inter-prediction while a PU within an intra-predicted CU holds similar information relating to intra prediction [17]. However, HEVC supports multiple PU sizes that directly influence the prediction efficiency. Having differently shaped PUs within a CU, enables the encoder to capture local characteristics of the content at a more finer level. Moreover, this flexibility allows the encoder to retain larger CUs with multiple predictions which suites the content and motion; a crucial development towards increasing the coding efficiency compared to H.264/AVC coding standard. In this context, Fig. 2.3
2.2. Overview of High Efficiency Video Coding

Figure 2.4: A representation of residual quadtree structure within a CU.

shows the PU sizes that are being used in HEVC for both intra and inter prediction\(^1\).

Once a CU has been predicted, the resulting residual is transformed from pixel domain to frequency domain using Discrete Cosine Transform (DCT)\(^2\) based integer transforms. In this context, the HEVC introduces the concept of Residual Quadtree (RQT) that allows the transform blocks to be recursively partitioned into TUs. This flexible transform block architecture allows the encoder to capture the residual variation characteristics within a CU, which in return retains a fair amount of necessary information within the block after the quantization process [46]. Hence, the quality of the reconstructed blocks are increased compared to the fixed size transforms that are being used in H.264/AVC. HEVC standard specifies core transform matrices with sizes ranging from \(4 \times 4\) to \(32 \times 32\). Similar to that of the CU quad-tree partitioning, the TU structure is also constructed with multiple depth levels. A TU having the size of \(M \times M\) can be decided to split further into four equal sized TUs having the size of \(M/2 \times M/2\) to improve the coding efficiency. For example, Fig. 2.4 illustrates the TU partitioning within a CU for a sample HEVC encoded video frame. In addition to the RQT, HEVC core transform design incorporates an assortment of improvements compared to H.264/AVC that results in its superior performance. These include, having almost orthogonal basis vectors that are much closer to Inverse Discrete Cosine Transform (IDCT), symmetry properties with smaller transforms are embedded within larger transform matrices, 8-bit representation of transform matrix elements [46]. Moreover, the transform coefficient coding has been improved with more content adaptive scanning patterns [47] and an entropy coding mechanism with high throughput Context Adaptive Binary Arithmetic Coding (CABAC) [48].

In summary, the combination of CUs, PUs and TUs that reside within a \(64 \times 64\) pixel block, also known as a CTU forms the coding structure in HEVC. The highly flexible nature and the content adaptability that it entails is a crucial development in HEVC that contributes to its superior coding performance over the prevailing encoding algorithms, yet, at the same time is a major source of complexity identified within the

---

\(^1\)It should be noted that the HEVC standard itself imposes certain constraints on PU modes applied on certain CU size in order to maintain a good trade-off between the coding efficiency and the encoding complexity [17].

\(^2\)Although, an alternate integer transform derived from the Discrete Sine Transform (DST) is used for the intra-predicted luma residuals when using \(4 \times 4\) transforms.
Table 2.2: 8-tap DCT interpolation filter coefficients for 1/4-pel luma interpolation

<table>
<thead>
<tr>
<th>Position</th>
<th>Filter coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>{-1, -4, -10, 58, 17, -5, 1}</td>
</tr>
<tr>
<td>2/4</td>
<td>{-1, -4, -11, 40, 40, -11, 4, -1}</td>
</tr>
<tr>
<td>3/4</td>
<td>{-1, -5, 17, 58, -10, 4, -1}</td>
</tr>
</tbody>
</table>

HEVC encoding process [15]. That being said, HEVC also comes with an assortment of novel prediction modes for both intra- and inter-picture prediction, which are summarized next.

### 2.2.2 HEVC Prediction Modes

When a CU is using intra-prediction, it follows a prediction based on the TU size. Thus, the boundary samples from the previously decoded neighboring TUs are used as the reference samples for the current TU. The HEVC intra-prediction architecture constitutes the quadtree based coding structure, 33 angular prediction directions, planar and DC prediction modes, adaptive smoothing for reference samples, filtering of the boundary samples, prediction mode dependent residual transform and coefficient scanning methods and efficient mode coding mechanism to facilitate the large number of coding modes [49] [8].

When the CU is inter-predicted, the CU under consideration is predicted from previously coded pictures exploiting the temporal redundancies in a sequence of frames. An inter-predicted CU in HEVC can utilize a wider range of PU sizes as illustrated in Fig. 2.3. For each inter-PU, the prediction mode can take either inter, skip or merge mode. In the case of inter mode, HEVC introduces Advanced Motion Vector Prediction (AMVP) which follows a Motion Vector Competition (MVC) scheme where a predicted motion vector for the PU is selected from a set of spatial and temporal motion vector candidates. Next, the reference picture list information (HEVC supports two reconstructed reference picture lists; list 0 and list 1), reference picture indexes, motion candidate indexes and Motion Vector Difference (MVD)s and prediction residuals are transmitted to the decoder. However, in the case of merge mode, motion information from a neighboring inter-coded PU are inferred as the motion details for the current PU. Hence, only the index of the selected neighboring PU and the residual information are transmitted. Finally, in skip mode, only the index of the selected neighboring PU is transmitted, thus the residual signal is excluded from the transmitted information [50][51].

The predicted block for a PU is obtained using the horizontal and vertical components of the motion vector determined via motion estimation. Similar to the process in H.264/AVC, fractional interpolation is used to generate pixel values for the non-integer positions in the prediction block. HEVC proposes to use 8-tap DCT based interpolation filter for 2/4 precision luma samples and a 7-tap separable DCT based interpolation filter is utilized for 1/4 precision luma samples and are illustrated in the
2.2. Overview of High Efficiency Video Coding

Table 2.3: 4-tap DCT interpolation filter coefficients for 1/8-pel chroma interpolation

<table>
<thead>
<tr>
<th>Position</th>
<th>Filter coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>{-2, 58, 10, -2}</td>
</tr>
<tr>
<td>2/8</td>
<td>{-4, 54, 16, -2}</td>
</tr>
<tr>
<td>3/8</td>
<td>{-6, 46, 28, -4}</td>
</tr>
<tr>
<td>4/8</td>
<td>{-4, 36, 36, -4}</td>
</tr>
<tr>
<td>5/8</td>
<td>{-4, 28, 46, -6}</td>
</tr>
<tr>
<td>6/8</td>
<td>{-2, 16, 54, -4}</td>
</tr>
<tr>
<td>7/8</td>
<td>{-2, 10, 58, -2}</td>
</tr>
</tbody>
</table>

Table 2.2. Moreover, a 4-tap separable DCT-based interpolation filter is utilized for chroma samples as depicted in the Table 2.3.

In summary, the both intra- and inter-prediction operations in HEVC coding standard have been improved extensively compared to its predecessor while retaining the same hybrid, block based coding approach. However, HEVC’s superior coding efficiency is also the result of additional post processing techniques that have been incorporated, namely Sample Adaptive Offset (SAO) filter and de-blocking filter which are described next.

2.2.3 HEVC Post Processing and Parallel Processing Tools

HEVC introduces two processing stages after the encoding procedure: de-blocking filter and SAO filter [8]. The objective of the de-blocking filter is to reduce the blocking artifacts and is operated on the pixels that lie on the block boundaries [52]. SAO filter on the other hand improves the accuracy of reconstructed samples by applying it adaptively to all the samples that satisfy a set of predefined criteria [53].

These coding tools and processing stages within the HEVC standard evidently improve the coding efficiency by a larger margin compared to its predecessors. Hence, these coding tools and features are crucial when considering the increasing popularity of HD, UHD video data and associated media applications. Having said that, another crucial design goal in HEVC is to facilitate the support for parallel processing architectures that are becoming increasingly popular [8]. In this context, HEVC introduces the concept of Slices and Tiles for picture partitioning and Wavefront Parallel Processing (WPP) as a parallel processing tool [8] [54][55]. Slices are a sequence of CTUs that are independently decodable from other slices and are non-trivial for error resilient video coding. Tiles on the other hand are rectangular regions within a frame which can be encoded and decoded independently and have shown to be crucial for multicore processor architectures where each Tile can be processed by a single core. WPP, is presented as a coding approach where each row of CTU in a video frame is considered as a slice and is independently encoded/decoded. As illustrated in the Fig. 2.5, WPP is scalable and has shown extremely useful for multithreaded/multicore processor architectures where each thread/CPU core processes a single row of CTUs.
2.3. Fast Encoding Algorithms for HEVC

The coding tools and features in HEVC summarized above directly impact the compression performance in HEVC on numerous levels. The level of usage of these tools boils down to an engineering decision based on the application requirements. Amongst them, the trade-off between the coding complexity and coding efficiency becomes crucial when considering the resource requirements in content preparation and distribution channels. Moreover, the increasing usage of resource constrained mobile devices for video consumption raises additional concerns for content preparation to cater for the diverse decoding complexity/energy requirements of the decoding devices. Hence, the as specified in Chapter 1, reducing the encoding complexity of HEVC encoders and implementing content adaptation methods within the encoding phase to reduce the decoding complexity are crucially important for the sustainable growth and expansion of the usage of the HEVC standard. Therefore, the following sections illustrate the state-of-the-art approaches that can be found in the recent literature to reduce the encoding complexity and state-of-the-art HEVC decoding complexity reduction methods, that fall within scope of the proposed research.

2.3 Fast Encoding Algorithms for HEVC

Expediting the encoding process in HEVC can be attempted at various operational points within the encoding cycle. For example, numerous optimization techniques are attempted at motion estimation and compensation process, prediction mode selection,
2.3. Fast Encoding Algorithms for HEVC

entropy coding operations, in-loop filtering etc. However, it is the overall process of selecting the best coding structure (i.e., CU, PU and TU sizes) for a given content that has the detrimental impact on the coding complexity. In this context, the following subsection summarizes the state-of-the-art approaches followed in the literature to improve the encoding speed in inter- and intra-prediction processes within the HEVC encoder.

2.3.1 Fast Inter-coding Techniques

Reducing the computational complexity associated with inter-prediction in the HEVC architecture can be attempted for a range of operations, such as motion estimation, coding structure determination, filtering, etc. However, the complexity analysis presented in [56] suggests that optimizing the motion estimation alone could only lead to a minimal reduction of the encoding time. Moreover, the application of these optimization techniques are not mutually exclusive, and could be utilized in conjunction with coding structure determination methods, which lead to more significant reductions of the encoding time. Hence, the remainder of this sub-section discusses the state-of-the-art methods relating to the complexity reductions associated with the efficient determination of the coding structures (the CU size in particular), for inter-prediction in HEVC.

Optimizing the PU level mode decisions can be described as one of the most popular methods used to reduce the complexity of HEVC. For example, the CBF Fast Mode (CFM) [18] method by Gweon and Lee utilizes the Coded Block Flag (CBF) information of a PU to skip the evaluation of remaining PU modes, while Vanne et al. [57] optimize the partitioning mode decisions between symmetric and asymmetric motion partitions. Other approaches, such as those from Shen et al. [58] and Sampaio et al. [36], employ the use of inter-level and spatiotemporal correlations to determine inter-mode or Motion Vector Merging (MVM) to determine the best PU mode for a given CU. RD cost prediction and dynamic encoder parameter selection (search range, bi-prediction refinement, Hadamard motion estimation, etc.) were proposed to achieve similar objectives by Jung et al. [59] and Correa et al. [60], respectively. Furthermore, in the recent past, the use of the SKIP mode decision of a particular CU to terminate the PU mode evaluation has become a popular PU level optimization technique. This has shown greater improvements in the encoding time reductions, especially in the case of low complex video sequences with low bit rate requirements [61]. One implementation of this method, the Early SKIP Detection (ESD) [62] method introduced within HEVC [63], shows a considerable encoding time reduction with negligible impact on the coding efficiency. Moreover, the Skip Mode Detection (SMD) algorithm proposed by Lee et al. [61], optimizes this process further and complements the fast CU size selection algorithms proposed therein. Shen et al. [64] and Ahn et al. [65] also make use of a similar SKIP mode detection algorithm to terminate the CU evaluation process, thereby reducing the encoding time. However, the common theme in these works is the fact that they focus on the PU mode decision and do not consider the CU size decision (it should be noted that they can still be used in conjunction with a CU size selection algorithm); thus, every CU depth level must be evaluated in order to determine the optimum CU size. Moreover, it is evident from the experimental results [61] that the
2.3. Fast Encoding Algorithms for HEVC

potential exists for further reduction of the computational complexity than what can be achieved from the application of these techniques alone.

Therefore, it is crucially important that attention is paid to the efficient selection of the CU size in the coding hierarchy. In this context, several offline classification-based methods and data mining techniques with multiple offline training stages have been proposed by Shen et al. [66, 67] and Correa et al. [68], respectively. However, the threshold estimations carried out via offline training eventually led to less efficient decisions, especially in the case of video sequences with complex motion characteristics. A number of approaches exist in the literature that utilize neighboring and co-located CU information to determine the unnecessary depth levels. One such method by Shen et al. [64] employs spatially adjacent coding blocks and threshold estimation for depth range and RD cost calculation. This however exhibits degraded performance in regions with the complex motion, due to less efficient decisions being obtained from neighboring CUs. Therefore, Hsu et al. [69] suggested encoding intermediate frames using the traditional RD optimization, which reduces the coding losses and the likelihood of propagating errors into subsequent CUs, albeit at the expense of increasing the encoding time. Yet other methods, such as the use of Pyramid Motion Divergence (PMD) by Xiong et al. [70], assess the motion homogeneity to determine if a CU should be split. However, once again, the optical flow calculation that extracts the motion characteristics can be both time consuming and resource intensive [65].

CU size prediction methods based on the statistical analysis of the RD cost have also been proposed. For example, Lee et al. [61] employ the RD cost statistics of the Inter $2N \times 2N$ and merge modes to determine the appropriate CU size for the coding block. Although this approach is effective in its decision making, the evaluation of the current depth level becomes futile in the event that the CU requires further splits in the quadtree hierarchy, which becomes more likely in the case of high quality encoding of textured sequences with complex motion. A similar drawback is observed in the fast block partitioning algorithm proposed by Lu et al. [71], which makes use of the statistics gathered during the Inter $2N \times 2N$ mode evaluation, together with texture information, to determine when to terminate the CU size selection process.

In this context, it becomes evident that the traditional CU size selection techniques that rely on the statistics gathered from the initial evaluation of the Inter $2N \times 2N$, SKIP, or other PU modes can be enhanced for further performance gains. This is especially true in the case of textured sequences with complex motion where high quality is required; an area where existing implementations show some deficiency, and a scenario that must be addressed by encoder implementations that operate in a wide range of bit rates. Moreover, a content adaptive operation is crucially important to cater for the diversity of the available content; thus, the capacity for dynamic training and content-specific feature extraction become necessities. Finally, the possibility of an initial evaluation of the Inter $N \times N$ mode, which leads to the extraction of motion characteristics and RD cost statistics of a CU requires an in-depth analysis [19], such that a content adaptive CU size selection framework can be developed to achieve greater encoding time savings while minimizing the impact on the RD performance.
2.3.2 Fast Intra-coding Techniques

The intra-prediction in HEVC has evolved significantly compared to its predecessor in numerous ways. For example, the hierarchical quad-tree based partitioning structure coupled with the 35 prediction modes (i.e., 33 angular modes, DC mode and planar mode) contribute significantly towards the coding efficiency in I-frames [49]. However, both these attributes in HEVC adds up to the non-trivial complexity increase that it demonstrates. Thus, the complexity reduction techniques introduced in the literature can be mainly grouped into two categories. They are the algorithms that focus on determining the best prediction mode for a PU and the algorithms that attempt to determine the best coding structure for a given CTU. The relevant state-of-the-art algorithms that fall into these two categories are described next.

The brute force evaluation of the full range of prediction modes for a given PU, to determine the best mode through RD optimization is a time consuming process for a HEVC compatible encoder. Therefore, the HM reference encoder [63] adopts a Rough Mode Decision (RMD) process introduced by Lainema et al. [49] to select a smaller subset from the set of all 35 modes to be evaluated through the Rate-Distortion (RD) optimization to select the best mode. Following a similar approach, numerous methods exist in the recent literature that attempt to improve the mode selection process. For example, the methods introduced by Fang et al. [72] and Chung et al. [73], propose to reduce the number of prediction mode candidates evaluated using RD optimization and RMD process using pixel energy distribution and gradient direction information, respectively. Moreover, Ismail et al. [74] propose to employ prediction information of the upper CU layers and neighboring PUs to determine the prediction mode for the current PU. However, the common drawback in these methods is that the complexity reduction that can be achieved is trivial in order to realize real-time encoding algorithms. The increased number of CU depth levels require the evaluation of these modes at each CU level, thus, the complexity reduction that can be achieved is shown to be less.

During the encoding process, a HEVC compatible encoder evaluates combinations of CU partitions and prediction modes to determine the best CU size and prediction mode for a given CTU. As a result, increased number of partition block sizes increases the complexity of the RD optimization process. Thus, the brute force evaluation of the coding structure is considered as the most crucial process in terms of the coding efficiency as well as the coding complexity. Hence, the recent literature reports numerous methods to simplify the coding structure selection process by reducing the number of candidates used in the RD optimization. These algorithms in the state-of-the-art generally utilize diverse characteristics and features that are encountered along the encoding process, hence can be categorized into two main groups. These constitute, algorithms that utilize statistical information available within the encoding chain (RD costs, prediction residuals, prediction modes of the neighboring blocks etc.), and algorithms that use the texture characteristics of the CTU to pre-determine the CU split, non-split decision of the current CU. For example, Kim et al. [75] utilize the statistical distribution of RD costs to determine the split or non-split decision of CU. In this case, if the RD cost of the CU becomes less than a predefined threshold, the remaining depth levels of the CTU are not evaluated (i.e., the CU is decided as non-split). However, the remaining
depths of the CU are further evaluated when the RD cost becomes greater than the threshold. This early-termination algorithm has achieved a 24% time saving compared to HM5.2rc1 software. The algorithm proposed in [76] uses the information available in the spatially adjacent CUs to determine the depth level of the current CU. This method has achieved a 21% computational complexity saving on average. Moreover, the variation of mode costs calculated within RMD process is utilized by Zhang et al. [77] to make the CU splitting decisions. For example, if the cost exceeds a predefined threshold, the CU is determined to split to the next depth level without processing the current level. This technique has demonstrated about 32% encoding time saving for all intra main configuration in HM7.0 software. However, the complete evaluation of the current CU depth level (brute force evaluation of all the prediction modes and PU modes within the CU) to make a final decision on further splitting of the CU is seen as a common approach in these state-of-the-art algorithms. Hence, the demonstrated encoding complexity reduction typically falls within the range of 20-30% compared to the HM reference encoder implementation [78]. Therefore, an encoding algorithm that determines the coding structure for a CTU prior to the evaluation process is crucial to further expedite the encoding process of HEVC intra-coded frames.

Next, when considering the second category of algorithms, it can be observed that there is a clear tendency to split less homogeneous areas into smaller blocks while maintaining the homogeneous regions as large blocks. For example, the proposed methods in [79] and [80] utilize the texture complexity estimated using the gradient information to determine the coding structure for a CU. In this case, Lokkuju et al. [79] propose to utilize the horizontal and vertical sobel kernels to calculate the block gradient. Thereafter, the split decision for the current CU is obtained based on the standard deviations of the gradient values of the four constituting CUs within the current CU. Once the decision is made, a fine tuning process is carried out to determine whether partitioned coding units can be merged together. This is achieved by comparing the sum of RD costs of constituent blocks with the cost of the current CU. Similarly, Zhang et al. [80] calculate the texture complexity from the directional gradients of the CU, and is compared with a QP and PU size based adaptive threshold to determine the CU’s split decision. Moreover, the algorithm also possesses a mechanism to merge the less complex CUs to generate the final partitioning structure for the CTU. In addition, Tian et al. [81] propose a similar content adaptive threshold based CU size determination mechanism to expedite the encoding process for intra frames in HEVC. However, the kernel based gradient calculation is a time consuming operation, thus the encoding complexity reduction achieved for high resolution videos falls within the range of 30-40% compared to HM reference encoder [78]. Hence, a low complex texture analysis method which does not exceed the computational cost of the RD optimization cycle, is required for the analysis of a CTU at an early stage of the encoding cycle to predetermine its coding structure that minimizes the RD cost.

2.4 HEVC Decoding Complexity and Energy Reduction

The decoding complexity and its associated energy consumption are tightly coupled to the complexity of the content as well as the codec [82, 83, 84, 85]. Therefore, the
increased resolution of the popular video contents (HD, UHD, etc.,) and the complexity of the state-of-the-art video coding standards evidently increase the decoding complexity of a typical video sequence. Thus, the increased complexity of the HEVC encoded bit streams plays a significant role in the energy consumption of the decoding devices. Hence, the high complex and high resolution HEVC video streams indeed have become a crucial bottleneck for the video playback in resource constrained mobile hand-held devices such as smartphones and tables etc.

The relationship between a device’s energy consumption and the many factors that can affect it (e.g., the complexity of the media content, the video coding algorithms, communication protocols and technologies, the hardware architecture, etc.) have resulted in research activity focused on reducing power consumption in all layers of the IP stack [21]. Yet, they can be mainly categorized into two areas [21]; solutions that operate on the physical and link layer protocols, and those that operate in the application layer. The former attempt to reduce the energy consumed in communication activities by managing the wireless network interface and performing energy aware traffic scheduling [86][87] [88], whereas the latter attempt to reduce the complexity of the content being consumed. The main focus of this work is on the application layer (i.e., adapting the content during the encoding process) and thereby reducing the decoding complexity of an HEVC coded video bit stream, and by extension, its energy consumption. Hence, the following discussion focuses on the state-of-the-art associated with the second category.

In this context, the application layer solutions available in the state-of-the-art can be categorized into three main groups. These include algorithms that propose modifications to the decoder operations, DVFS algorithms that reduce both hardware clock frequency and supply voltage, and content adaptation algorithms that modify the video stream that is received and decoded by the mobile device. Some of these algorithms which are related to the proposed works are presented and discussed next.

The power consumption in Complementary Metal-Oxide-Semiconductor (CMOS) circuits exhibit a linear relationship with the CPU’s clock frequency [89]. Therefore, exploiting the relationship between computational complexity, execution time and the clock frequency, the energy consumed per operation is mapped to a quadratic relationship to the operation’s computational complexity [89]. Hence, simplifying the decoder operations, and thereby reducing the device’s energy consumption has been attempted in numerous occasions in the recent literature. Green MPEG is one such recent initiative made by the MPEG attempting to standardize green meta data which can be transferred as Supplementary Enhancement Information (SEI) to the decoder [90] [91]. The green meta data information is then expected to be utilized to reduce the decoding complexity as well as alter the display parameters to reduce the energy consumption [91]. In addition, most recent decoder developments employ the structural improvements and data level as well as task level parallelization enabled decoder implementations to support real-time decoding of high resolution, high frame rate HEVC bit streams [92]. However, rigorous changes required to the decoder architectures make these solutions less flexible to adapt to the existing implementations. Following a similar approach, Nogues et al. [93] propose to simplify two of the most complex decoding operations in the HEVC decoder; the in-loop filtering and the interpolation filters used
2.4. HEVC Decoding Complexity and Energy Reduction

During motion compensation, the in-loop filtering is skipped in certain frames to suit the desired level of complexity, while the 7- and 8-tap luma and the 4-tap chroma interpolation filters are reduced to 3-tap luma and 1-tap chroma filters, respectively. Although the arithmetic level approximate computing applied for energy efficient HEVC decoding [94], reduces decoding complexity, the quality is severely compromised due to the modified interpolation filters operating with unmodified PU residuals.

DVFS technologies have been used extensively in the recent past to reduce the decoding complexity, where a DVFS algorithm balances the energy use against the video quality [29, 28, 30, 26]. In general, these methods estimate the operating frequency of the processor for the next frame based on the complexities and the decoding times of the previous frames that were decoded. These methods operate on a principle similar to that of Linux ondemand governor [95], but with a frequency selection which is more focused on the decoder operations. However, these approaches have shown drawbacks such as frame drops and impacts to overall system performance which adversely affect the user’s quality of experience, especially in the case of high frame rate, high quality content [28]. The poor estimation of the complexity of subsequent frame/video segment that affect the selection of appropriate operational CPU frequency and voltage level, is identified as a predominant issue associated with the DVFS based solutions. Hence, Green-MPEG specification suggests to use Codec Dynamic Voltage Frequency Scaling (C-DVFS) metadata embedded within the bit stream to accurately estimate the decoding complexity [85]. However, determining an exact operational voltage/frequency pair is still a challenging task which could be simplified by having properly decoding complexity and rate controlled video bit streams being sent to a DVFS enabled decoder.

Content adaptation algorithms on the other hand often attempt to exploit the correlation between the decoding complexity and the energy consumption to further reduce the energy consumed during video playback. For example, Scalable Video Coding (SVC) architectures that use proxy servers [96], media transcoding solutions [97] and dynamic adaptive streaming technologies [34, 35, 27] have become popular choices for reducing the complexity of the decoder and thereby reducing the device’s energy consumption. The reduced processing latency facilitated by such algorithms improves the slack-time in real-time decoders enabling the DVFS and Dynamic Power Management (DPM) algorithms to trigger less operating frequencies and voltage levels leading to crucial energy reductions. However, these, as well as more device oriented [24] and battery-aware [25] adaptive multimedia delivery schemes, are generally restricted to manipulating basic coding parameters such as the QP, the spatial resolution and the frame rate to adapt the video content and achieve energy savings. Although MPEG-DASH based energy-aware HEVC streaming solutions [27] do exist, decoding energy is only considered in the PU mode decision and during motion vector selection (i.e., integer-pel vs fractional-pel); thus, the decrease in the energy consumption is marginal with respect to the simpler approaches. Furthermore, the network and receiver aware bit stream adaptation algorithms and the complexity rate-distortion models [32, 98, 33] which have been introduced based on the previous coding standards typically focus on the spatial, temporal and quality scalable bit streams. However, the lack of comprehensive analysis on decoding complexity, rate and distortion trade-offs, and the diverse coding features available in the modern coding standards make these approaches not
directly applicable to the modern coding architectures.

The increasing popularity of HTTP-based video streaming services that adopt adaptive video segment prefetching to cope up with the unforeseen fluctuations of the network bandwidth have shown the ability to incorporate decoder energy consumption to the content prefetch logics [27]. Typically, the adaptation logic monitors the device’s remaining energy level and determines the next segment that matches the estimated energy level. However, the state-of-the-art methods available in the literature today typically consider altering bit rate, QP, and spatial resolution to cater for the degrading energy capacity [25, 24, 34, 35]. Considering the encoding algorithms available in the literature, it is evident that this is mainly due to the lack of an effective content generation algorithm that is capable of creating a video bit stream that matches a certain decoding complexity requirement. Thus, without an appropriate encoding algorithm to consider both rate as well as decoding complexity, the content generation with multiple decoding complexities is attempted indirectly by changing the basic encoding parameters (i.e., QP, spatial resolution etc.). Therefore, it is evident that an encoding algorithm that can generate video bit streams that meet specific bit rate as well as decoding complexity requirements is pertinent to realize decoding energy/complexity aware video streaming solutions with minimal quality degradation; a crucial element that is missing in the state-of-the-art.

However, creating network bandwidth and decoding complexity aware video bit streams requires the encoder to be aware of the bit rates as well as the decoding complexities associated for a particular coding mode. Existing encoding procedure however, reveals the number of bits that results by using a particular coding mode on the content being encoded, yet, the determination of the respective decoding complexity for the coding mode is still challenging without an accurate complexity estimation model. Therefore, it is crucial that a HEVC compatible encoder to be aware of the decoding complexity of a CU for all coding parameter combinations. Thus, detailed and accurate modelling of the decoding operation complexity becomes crucially important. In this context, the state-of-the-art techniques have exploited high-level complexity analysis of the decoding operations [99], energy estimation based on the decoding time [100], and mapping of decoding energy to the content and QP [101] for modelling purposes. More comprehensive studies have also modelled the intra- and inter-predicted HEVC decoding energies in [100] and [102], respectively. However, in general, the level of detail in these models is inadequate for accurate CU-level complexity estimation; an essential component for decoding complexity-rate-distortion optimized encoding. Moreover, the use of such a model within the RD optimized encoding cycle requires an in-depth analysis of the behaviors of rate, distortion and decoding complexity, and is a crucial missing element in the literature on the modern video coding standards. In addition, the algorithms that can jointly control both the rate and decoding complexity are hardly available in the literature, thus, a wider scope and research potential remains in this domain to investigate the content adaptation methods that can be used to reduce the energy consumption of the video playback devices.
2.5 Test Video Sequences

Finally, this section illustrates the HD and UHD video sequences utilized in the works presented in this thesis. These sequences possess diverse spatial and temporal characteristics and span across both natural and synthetic contents. The Fig. 2.5 graphically illustrates the textural properties of the respective sequences for a particular frame.
2.6 Summary

This Chapter provides an overview of the video coding and the prominent role that it plays within the media processing and distribution tool chain. Next, HEVC encoding architecture is discussed focusing on the novel coding modes and features. Detailed descriptions are provided on the HEVC’s coding structure which constitutes CU, PU, and TU, quadtree architecture. Furthermore, improvements in intra- and inter prediction modes in HEVC compared to the H.264/AVC standard are summarized followed up with a brief illustration of the post processing and parallel processing tools in HEVC.

The next section of this Chapter describes the state-of-the-art works available in the recent literature to expedite the HEVC encoding process. In this case, the performance level of each algorithm is discussed with potential drawbacks. Then, the state-of-the-art approaches to reduce the decoding complexity and the associated energy consumption are discussed in detail with a key focus on the application layer content adaption.
2.6. Summary

approaches. Finally, the Chapter concludes with a brief overview and a graphical illustration of the test sequences which are used for the experiments presented in the subsequent chapters.
Chapter 3

Content Adaptive Fast HEVC Encoding

One of the crucial bottlenecks in HEVC (as illustrated in Sec. 2.3) is the complexity incurred in the HEVC compatible encoders when producing a HEVC compliant bit stream with a higher coding efficiency. In this context, reducing the complexity in the RD optimization process, is seen as a potential solution considering the increased number of coding modes and options, available within HEVC. To this end, expediting the process of selecting the optimum coding structure for given content is considered as a compelling challenge as it encompass non-trivial portion of the overall encoding time (ref. Sec. 2.3) [15]. Thus, this Chapter introduces the proposed algorithms and encoding methodologies for HEVC encoders to generate HEVC bit streams with less complexity compared to the state-of-the-art methods while achieving a similar coding performance compared to HM reference encoder.

3.1 Fast Inter Coding Techniques

The leaf node CUs in a quad-tree structure within a CTU (ref. Sec. 2.2) resulting from the splitting decisions determined through the RD optimization are tightly coupled with nature and the complexity of the content. Thus, predicting the CU size beforehand, using a set of pre-determined features, becomes challenging due to this dynamic nature of the problem. In this context, it becomes evident that the early determination of the CU size requires a modelling of the CU split likelihoods of that particular content using a set of content-specific features. The selection of appropriate features that accurately model the CU split decision, and are also easy to extract from the encoding chain, is therefore crucially important. Two dynamic content-specific techniques that can be used for this purpose are described next.

3.1.1 Motion Feature-Based CU Classification

The first model, which is based on the motion feature-based CU classification approach [19][103], attempts to represent the CU split likelihood as a function of three parameters...
3.1. Fast Inter Coding Techniques

Figure 3.1: CU partitioning and motion vector distribution of a typical HEVC encoded video frame.

given by

\[ f(F) := f(\alpha, \beta, \omega), \]

where \( \alpha, \beta, \) and \( \omega \) represent a motion classification, an Inter \( N \times N \) RD cost category and the CU size, respectively, and \( f \) denotes a probabilistic model that determines the outcome of the CU split decision described in subsection 3.2.1. The relevance and impact of each parameter and the process of forming the feature vector \( F \) are described in detail below.

First, observing the partitioning behavior of CUs during inter-prediction, it can be observed that blocks with similar motion tend to utilize larger CUs, whereas blocks with complex motion tend to utilize smaller CUs [36, 70, 104]. For example, the Fig. 3.1 illustrates CU partitioning with corresponding motion vectors of a typical HEVC encoded video frame. This suggests that the motion characteristics of a CU possess an exploitable correlation with respect to the optimal CU size. Therefore, the identification and classification of the motion characteristics of a CU could potentially support optimal CU size selection process, yet at the same time could become a non-trivial overhead with the usage of complex algorithms such as optical flow analysis [70]. Attempting to classify these characteristics, from the information available within the encoding chain itself, therefore becomes attractive due to both its simplicity and its minimal impact on the complexity (for example, the PU modes evaluated for a particular CU depth level retains the prediction information, including the motion details for the CU, that could be considered to determine the required motion complexity). To this end, this thesis proposes an initial Inter \( N \times N \) mode evaluation (skipping the traditional PU evaluation order [63]) to collect the necessary motion information for each CU. Since the Inter \( N \times N \) mode evaluation performs the motion estimation for the four constituent blocks of the CU, the motion of the CU is classified based on the similarity\(^1\) of the motion vectors of the constituent blocks [19]. This results in a CU

\(^1\)Two motion vectors are considered to be equal when each others horizontal and vertical components
3.1. Fast Inter Coding Techniques

being classified into one of nine categories depicted in Fig. 3.2 and denoted by \( \alpha_i \), \( i \in \{0, 1, ..., 8\} \). Here, the motion vectors of the \( k \in \{0, 1, 2, 3\} \) constituent blocks are given by,

\[
MV_k = (mv_k, rPOC_k),
\]

where \( rPOC_k \) is the reference Picture Order Count (POC) number of the reference frame of the motion vector \( mv_k \).

Next, analyzing the split likelihood (i.e., the ratio between the number of CUs that are split and the total number of CUs) of a CU classified as described above, it can be seen are equal and point to the same reference picture.
Figure 3.3: Distribution of normalized proportion (%) of CUs that are split across $\beta$, motion category ($\alpha$) and CU size ($\omega$) for 200 frames in “City (720p)” video sequence when encoded with QP=27 using low delay P main configuration in HM 16.0. Results depict the CU split likelihoods for the different combinations of the parameters $\alpha$, $\beta$, and $\omega$ in the feature vector $F$. 

(a) Normalized proportion of CUs that are split in each motion category ($\alpha_i$) for each $\beta$. (Averaged over CU sizes 64, 32 and 16). 

(b) Normalized proportion of CUs that are split in each CU size ($\omega$) for each motion category ($\alpha_i$). (Averaged over entire range of $\beta$).

(c) Normalized proportion of CUs that are split in each CU size ($\omega$) for each $\beta$. (Averaged over all motion categories).
3.1. Fast Inter Coding Techniques

that textural diversity impacts it as seen in Table 3.1. This suggests that the motion
category of a CU alone does not sufficiently model the split likelihood of a CU; thus,
additional features need to be considered in order to realize a more robust model.

The Inter $N \times N$ RD cost $\gamma$ (commonly used for CU size prediction and also obtained
from the Inter $N \times N$ mode evaluation) therefore presents itself as a natural second
parameter that can be used to describe the split likelihood. The RD cost is a parameter
which is commonly used during the split decision modeling, thus, the distribution of
the RD cost ($\gamma$) values of the Inter $N \times N$ mode is further investigated as a potential
candidate for the feature vector $F$. In practice however the range of $\gamma$ is quite large,
making the statistical analysis of individual RD costs less useful, especially in the case
of very rare, large $\gamma$ values. Therefore, the square-root of the RD cost is adopted
instead, and is consolidate into one of 200 bins with a bin size $\Delta$ of 5. This results in
a RD cost category $\beta$ given by,

$$\beta = \begin{cases} \left\lfloor \sqrt{\frac{\gamma}{\Delta}} + \frac{1}{2} \right\rfloor & \sqrt{\gamma} \leq 200 \Delta \\ 200 & \text{otherwise.} \end{cases}$$ (3.3)

Finally, the CU size itself can be considered as a third parameter $\omega$ that describes
the CU split likelihood. The relationships between $\alpha$, $\beta$ and $\omega$, and the resulting
motion-based feature $F$ in (3.1), are however far from straightforward and are content
dependent. A pair-wise selection of these features is illustrated in Fig. 3.3 to demonstratethe complex relationships that exist within even a single sequence. For example,
it can be observed that each $\alpha_i$ exhibits diverse CU split likelihoods for different $\beta$ in
Fig. 3.3(a) and $\omega$ in Fig. 3.3(b), while larger $\beta$ favours CUs with larger $\omega$ to be split as
seen in Fig. 3.3(c). Thus, it becomes apparent that the motion feature-based approach
to classifying CUs possesses the necessary flexibility to model the dynamic nature of
the content, which, if exploited, could lead to more robust predictions of the optimal
CU size.

3.1.2 RD Cost Threshold-Based CU Classification

In contrast to the motion feature-based approach, the CU split likelihood can also be
modelled in a partly heuristic fashion. This second model investigates its relationship
with respect to a general distribution of the Inter $N \times N$ mode RD cost $\gamma$, the CU size
$\omega$ and the Quantization Parameter $QP$.

First, analyzing the results of multiple video sequences reveals that the CU splitting
behaviour can be modelled by two Gaussian distributions; a CU split and non-split
likelihood distribution [61, 105]. Fig. 3.4(a) illustrates an example of these with respect
to $\gamma$ for a particular CU size and $QP$. Crucially, these distributions reveal the existence
of three regions within the range of $\gamma$ that demarcate the CUs that are split, the CUs
that are not-split and a third region where the decision is ambiguous. The generalized
behaviour of these distributions is illustrated in Fig. 3.4(b), and can be used to classify
the CUs based on two RD cost thresholds $\gamma \geq HT_{h_{spt}}$ and $\gamma \leq HT_{h_{nspt}}$, where
$HT_{h_{spt}}$ and $HT_{h_{nspt}}$ are the CU split and non-split thresholds, respectively (how to
3.1. Fast Inter Coding Techniques

Figure 3.4: (a) CU split and non-split likelihood distributions for $32 \times 32$ CU sizes of the “Parkscene HD” video sequence for QP=32 using the low delay P configuration in HM 16.0. (b) A representation of the $HTh_{spt}$ and $HTh_{nspt}$ RD cost thresholds that identifies the CU split, CU non-split and ambiguous regions.

adaptively determine these thresholds using the mean $\gamma$ values of the CU split and non-split Gaussian distributions is described in detail in Sec. 3.3.2). Therefore, modelling $HTh_{spt}$ and $HTh_{nspt}$ in terms of $\gamma$, $\omega$ and QP represents another approach to model the splitting behaviour of a CU. In this context, the behaviors of the $HTh_{spt}$ and $HTh_{nspt}$ (which in this case are determined as the mean values of the distribution of $\gamma$ for CU split and non-split likelihoods) are further analyzed for multiple video contents to investigate the possibility of developing generalized values for the respective thresholds.

To this end, the distribution of $HTh_{spt}$ and $HTh_{nspt}$ is analyzed for sequences encoded using the low delay P configuration and $QP \in \{22, 27, 32, 37\}$ in HM16.0. Despite some variations, from the results observed in Figs. 3.5 and 3.6, it is evident that an exponential curve can parameterize the behaviour of these two RD cost thresholds$^2$.

$^2$The split likelihood distribution of inter-predicted CUs for low delay B and random access configurations is observed to be similar for a given QP and CU size.
3.2 Fast CU Size Selection

Table 3.2: \( R^2 \) Goodness-of-Fit of the split and non-split thresholds

<table>
<thead>
<tr>
<th>CU Size</th>
<th>( HTh_{spt} )</th>
<th>( HTh_{nspt} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.827</td>
<td>0.782</td>
</tr>
<tr>
<td>32</td>
<td>0.703</td>
<td>0.906</td>
</tr>
<tr>
<td>16</td>
<td>0.780</td>
<td>0.882</td>
</tr>
</tbody>
</table>

Thus, generalized RD cost thresholds can be obtained, which are given by,

\[
HTh_{spt} = \begin{cases} 
2347 \times e^{0.1248 \times \text{QP}}, & \omega = 64 \\
851.2 \times e^{0.1228 \times \text{QP}}, & \omega = 32 \\
279.9 \times e^{0.1227 \times \text{QP}}, & \omega = 16 
\end{cases} \quad (3.4)
\]

\[
HTh_{nspt} = \begin{cases} 
736.2 \times e^{0.1378 \times \text{QP}}, & \omega = 64 \\
225.4 \times e^{0.1468 \times \text{QP}}, & \omega = 32 \\
57.72 \times e^{0.1607 \times \text{QP}}, & \omega = 16 
\end{cases} \quad (3.5)
\]

Table 3.2 summarizes the R-squared measure of the goodness-of-fit obtained for \( HTh_{spt} \) and \( HTh_{nspt} \) in (3.4) and (3.5). The modelled curves and the actual data illustrated in Figs. 3.5 and 3.6 suggest that the proposed models for \( HTh_{spt} \) and \( HTh_{nspt} \) are good representations in general, yet, are not particularly accurate, especially at higher QP values given the content specificities. Obtaining these content specific thresholds is however crucially important for an accurate optimal CU size prediction. In essence, threshold values in (3.4) and (3.5) could be used as initialization parameters during the preliminary stages of the encoding, and can later be adapted with content specific data as described in Sec. 3.3. Thus, the RD cost threshold-based approach can be thought of as an independent second mechanism that predicts the split likelihood of a CU.

3.2 Fast CU Size Selection

This section describes how the two independent CU split likelihood models in Sec. 3.1.1 and 3.1.2 can be used in a complementary fashion, to determine if a CU should be split or not, all the while adapting to the specific content being encoded.

3.2.1 Motion Feature-Based CU Size Selection

Applying the feature-based CU classification model, the split probability of a CU in the \( n^{th} \) frame can be defined as

\[
P_{s,n}(F) = \frac{D_{act}^1(F)|_n}{D_{act}^1(F)|_n + D_{act}^0(F)|_n}, \quad (3.6)
\]
3.2. Fast CU Size Selection

where $D_{\text{act}}^\eta(F)|_n$ is the number of CUs with a feature vector $F$, within the given frame that are either split ($\eta = 1$) or not-split ($\eta = 0$), based on the actual split decision obtained for the CUs by the RD optimization. Thus, $P_s,n$ can be considered a frame-wise statistic of the optimal split decision computed from the statistics accumulated in $D_{\text{act}}^\eta(F)|_n$ that is obtained for each feature vector during the training phases described in Sec. 3.3.2. However, since this statistic can vary over time due to changes in the underlying content, a snapshot of the actual split probability is obtained through a windowed averaging process. Mathematically, this can be expressed as

$$P_s(F)|_n = \frac{1}{W} \sum_{t=0}^{n-1} P_s,t(F) H(t - n), \quad (3.7)$$

where the window function $H(t)$ is given by

$$H(t) = \begin{cases} 1 + \frac{t}{W} & 0 \geq t \geq -W \\ 0 & \text{otherwise} \end{cases}, \quad (3.8)$$

and $\tilde{W}$ represents the area under the curve of $H(t)$. The existence of the window function and its effective length $W$ is crucial in terms of the content adaptability, as it enables the predicted CU split decision to be biased to the most recent $W$ frames. Therefore, by appropriately selecting $W$, the predicted decision becomes less susceptible to the dynamics of scene changes, making the algorithm content adaptive.

The outcome of the decision of whether to split or not-split a CU is now obtained by comparing (3.7) with an empirically determined threshold $T$, such that the decision for the $n^{\text{th}}$ frame is given by

$$D_{fs}|_n = \begin{cases} 1 & P_s(F)|_n \geq T \\ 0 & \text{otherwise} \end{cases}. \quad (3.9)$$

The threshold $T$ therefore acts as a switch that either splits or does not split the CUs. Empirical observations reveal that the value of $T$ impacts both the bit rate and quality, where a smaller value of $T$ generally results in more CUs being split, while a larger $T$ results in less splitting. In this context, $T$ and the window length $W$ can be considered as design parameters that need to be empirically determined and preset for a desired trade-off of the quality and the bit rate.

3.2.2 RD Cost Threshold-Based CU Size Selection

In the RD cost threshold-based CU classification approach, the Inter $N \times N$ RD cost $\gamma$ is compared to the $HT h_{spt}$ and $HT h_{nsppt}$ thresholds for the $\omega$ and $QP$ relevant to each

---

3It should be noted that, in order to gather the statistics of the actual splitting behaviour of the content being encoded, the RD optimization can not be completely bypassed. Thus, the outcome of the split decision must be evaluated using an appropriate balance of either the two models presented in this paper or the traditional RD optimization approach. Precisely how this is implemented to achieve the fast coding objective, is described in Sec. 3.3.2.
3.3 Proposed Fast Encoding Framework

In this section, computing the ultimate CU split decision, using the two independent decisions in the previous section, is described.

CU. The resulting split decision $D_{hs}$ of the $n^{th}$ frame can therefore be expressed as

$$D_{hs}|_n = \begin{cases} 1 & \gamma \geq HT_{spt} \\ 0 & \gamma \leq HT_{nspt} \end{cases} \quad (3.10)$$

However, as described in the previous section, $HT_{spt}$ and $HT_{nspt}$ need to be made adaptive in order to become content-aware, and a heuristic process can be used to adapt the generic $HT_{spt}$ and $HT_{nspt}$ thresholds in (3.4) and (3.5).

First, in order to adapt the split and not-split thresholds to the content, the RD cost statistics of the actual split decision (similar to the feature-based model in (3.6)) are analyzed in terms of the mean and standard deviation of these thresholds for a particular CU size $\omega$ and Quantization Parameter $QP$. Adopting a window-based approach to maintain the content adaptability as before, the mean and standard deviation statistics during the $n^{th}$ frame for the split RD cost threshold can be expressed as

$$\mu_{spt}|_{n,\omega,QP} = \frac{1}{W} \sum_{t=0}^{W-1} E\left[\gamma_{spt}(n-t)|_{\omega,QP}\right] \quad (3.11)$$

and

$$\sigma_{spt}|_{n,\omega,QP} = \sqrt{\frac{1}{W} \sum_{t=0}^{W-1} E\left[\left(\gamma_{spt}(n-t)|_{\omega,QP} - \mu_{spt}\right)^2\right]} \quad (3.12)$$

respectively. $E(\cdot)$ represents the expectation operating on the applicable CUs in that frame, and $\gamma_{spt}(\cdot)|_{\omega,QP}$ represents the RD cost of the CUs that are split with a CU size of $\omega$ and a Quantization Parameter $QP$. The RD cost threshold statistics for the not-split scenario, $\mu_{nspt}|_{n,\omega,QP}$ and $\sigma_{nspt}|_{n,\omega,QP}$, can be obtained in a similar fashion.

Thus, the two thresholds themselves are made content adaptive by applying the following:

$$HT_{spt}|_{n,\omega,QP} = \{\mu_{spt} + \tau \times \sigma_{spt}\}|_{n,\omega,QP} \quad (3.13)$$

and

$$HT_{nspt}|_{n,\omega,QP} = \{\mu_{nspt} - \tau \times \sigma_{nspt}\}|_{n,\omega,QP} \quad (3.14)$$

The parameter $\tau$ acts as a governor that controls the adaptation of the model via the adaptation and training process described in Section 3.3.2, and is an empirical design parameter that can be used to trade-off the computational complexity for the coding efficiency in the proposed encoding algorithm.

3.3 Proposed Fast Encoding Framework

In this section, computing the ultimate CU split decision, using the two independent decisions in the previous section, is described.
3.3. Proposed Fast Encoding Framework

3.3.1 Joint CU Split Decision Prediction

The approach to using the two independent decisions described in Sec. 3.2 forms the basis of the proposed fast encoding algorithm. Thus, when obtaining the joint decision, two distinct categories exist; one where both independent split decisions concur, and another where they differ. Hence, for the former category, the joint split decision during the $n^{th}$ frame predicted by the encoding framework can be expressed as

$$CU_{\eta}|_{n} = \begin{cases} 
1 & D_{fs}|_{n} = 1 \land D_{hs}|_{n} = 1 \\
0 & D_{fs}|_{n} = 0 \land D_{hs}|_{n} = 0,
\end{cases} \quad (3.15)$$

where $D_{fs}|_{n}$ and $D_{hs}|_{n}$ are the two independent decisions obtained for the applicable $F, \gamma, \omega$ and $QP$ of each CU. The second category of decisions, where the models differ, can now be used to initiate the adaptation of the framework to enhance its robustness to different contents.

3.3.2 Model Adaptation and Training

Following from the discussion of the joint split decision prediction, the decisions that differ for the two models can be used to ensure that the framework remains both content-adaptive and efficient. In this context, it is crucial that the models are able to adapt; thus, some RD evaluation becomes essential to calculate the actual CU split statistics in Sec. 3.1.1 and 3.1.2

Initial Training

Both split likelihood models described in Sec. 3.1 require some initial training to gather content specific data at the beginning of a video sequence. In this work, the first four frames, i.e., $n = 1, \ldots, 4$, are used for this statistical information gathering. During this phase, the CU split decisions are obtained via the traditional RD optimization; thus, the content specific attributes can be extracted and the two models are initialized with sufficient data. However, these models may diverge from the actual content due to changes in the scene, making the accumulated statistics less relevant with time. Hence, the models must be continually refreshed, via intermediate training, as described next.

Intermediate Training

Intermediate training of the proposed framework via RD optimization can be split into three categories; training where no data exists for the features associated with a CU, training where the two models’ decisions differ, and training for modelling efficiency improvements.

The first type of intermediate training is triggered when the actual statistical split decision information required to compute $P_{s}(F)|_{n}$ does not exist (e.g., a situation where

---

4The RD cost threshold-based model can still utilize the generic values of the $HT_{h_{opt}}$ and $HT_{v_{opt}}$ thresholds in (3.4) and (3.5), respectively, until such content specific statistics are accumulated.
Figure 3.5: CU split thresholds and the fitted exponential curves with respect to QP for 6 different HD video sequences. The y-axis depicts each threshold with respect to the $\text{Inter } N \times N$ RD cost, $(\gamma)$. 

(a) $\omega = 64$

(b) $\omega = 32$

(c) $\omega = 16$
3.3. Proposed Fast Encoding Framework

Figure 3.6: CU non-split thresholds and the fitted exponential curves with respect to QP for 6 different HD video sequences. The y-axis depicts each threshold with respect to the Inter $N \times N$ RD cost, ($\gamma$).
Figure 3.7: The relationship between motion categories $\alpha_i$ and motion vector reuse for respective PU instances. Similar motion vectors in the Inter $N\times N$ mode are identified by the same color and pattern. The PU instances denoted by “ME”, and the PU modes that are not indicated, follow the traditional motion estimation process to determine the motion vectors.
the feature $F$ has not been encountered within the window length $W$). In this case, the decision is obtained by the RD optimization, thereby avoiding incorrect decisions being made for rarely occurring features. Similarly, the RD optimization is followed in the event that $D_{fs|n}$ in (3.9) and $D_{hs|n}$ in (3.10) contradict each other. In this case, the additional information obtained regarding the actual splitting behaviour will result in the refinement of both split likelihood models, thereby improving the accuracy of the subsequent decisions of similar CUs.

The third and most important intermediate training phase is triggered by RD costs where $HT_{haspt} < \gamma < HT_{spt}$ (see Fig. 3.4(b)), and the CU split decision of the RD cost threshold-based model $D_{hs|n}$ is undefined as per (3.10). Controlling the size of this region will result in a trade-off of the output quality for the reduction of the computational complexity. Hence, in addition to facilitating the statistics gathering, the complexity control parameter $\tau$ (introduced for this purpose in (3.13) and (3.14)) affords a degree of flexibility when implementing the overall algorithm. Here, a larger $\tau$ will expand this region, allowing more decisions to be taken by the RD optimization, resulting in better quality. A smaller $\tau$ on the other hand will reduce the number of the RD optimizations and will result in improvements to the encoding time performance.

### 3.3.3 Motion Vector Reuse in Motion Estimation

This subsection describes how the motion vectors computed during the Inter $N \times N$ mode evaluation can be reused to further expedite the encoding process during the subsequent PU level evaluations that repeatedly occur for each CU.

Consider the four constituent motion vectors and reference frames returned by the initial Inter $N \times N$ mode evaluation in (3.2). These motion vectors identify the motion category of a CU, $\alpha_i$, which interestingly has a structural relationship with the PU modes as illustrated in the Fig. 3.7. This relationship can be exploited to skip the motion estimation when a particular PU mode is being evaluated for a CU. For example, when a CU possesses the motion category $\alpha_0$ (i.e., all four motion vectors are equal and point to the same reference picture), the motion vector $MV_0$ available in (3.2) can be reused for the Inter $2N \times 2N$ mode, thereby skipping the motion estimation phase for that PU mode. However, not all PUs can be identified in this fashion; thus, some PUs require motion estimation, e.g., the PUs denoted by “ME” and the PU modes that are not illustrated in Fig. 3.7 will require the usual motion estimation.

Hence, this capability to reuse the motion information extracted during the CU size prediction emerges as a direct secondary benefit of the initial Inter $N \times N$ mode evaluation. As a result, the proposed framework is supplemented with this feature to further expedite the encoding process.

### 3.3.4 The Overall Fast Encoding Algorithm

The performance improvements of the fast encoding algorithm proposed in this Chapter can be described as a result of two distinct operations; the content-adaptive CU size prediction in Sec. 3.2 and the motion reuse operation in Sec. 3.3.3, that both exploit
Figure 3.8: The proposed fast encoding algorithm for HEVC based low delay video encoding. The flowchart describes the process of making the CU split/non-split decision during the compression phase. The model adaptation will take place during the encoding of the CTU with the selected CU structure.
the initial \( \text{Inter } N \times N \) mode evaluation. A high level flow diagram of the resulting algorithm, identifying the major decision making components and the operations of the individual blocks, is summarized in Fig. 3.8.

At an implementation level, if the CU split decision is in the negative, the CU is encoded at the selected depth level. However, if it is positive, the encoding cycle evaluates the next depth level of the CU. During the first \( N \) frames for the sequence, and whenever the CU split decision cannot be predicted, the traditional RD optimization is triggered via the initial and intermediate training processes described in Sec. 3.3.2. The statistics calculated during these periods are simultaneously used to update and refine the split likelihood models as described earlier. The shaded area in the Fig. 3.8 depicts the PU mode selection operation. Here, the available PU modes of the CU are evaluated in the traditional evaluation order \[63\], and the best PU mode is selected using an RD optimization. However, the motion estimation phase for a subset of PUs is skipped and the \( \text{Inter } N \times N \) motion vectors are reused where appropriate as described in Sec. 3.3.3.

### 3.4 Experimental Results and Discussion

The following section presents the experimental results of the proposed content-adaptive fast CU size selection and encoding algorithm for low delay HEVC video encoding. The experimental setup and evaluation conditions are described first and are followed by a detailed discussion of the results and the performance implications afterwards. The RD and encoding time performance of the proposed algorithm are compared with several state-of-the-art algorithms in the literature. These include the HM 16.0 \[63\] reference implementation, the CU size selection method proposed by Shen et al. \[64\], the fast encoding algorithms proposed by Lee et al. \[61\], the fast block partitioning algorithm proposed by Lu et al. \[71\], a PU mode decision algorithm proposed by Vanne et al. \[57\], and the offline data mining approach to CU early termination proposed by Correa et al. \[68\]. The selected state-of-the-art algorithms are implemented within HM 16.0 reference encoder and are evaluated together with the proposed algorithm in the same simulation environment to compare the performance with that of the proposed algorithm.

#### 3.4.1 Simulation Configurations and Performance Metrics

The algorithms are evaluated for a range of HD and UHD video sequences composed of both natural and synthetic content. The test sequences have been selected such that they span from simple to highly complex motion with diverse spatial and temporal characteristics. Table 3.3 summarizes the experiment setup and encoding configurations of the simulations\(^5\).

The impact on the RD performance is evaluated using the Bjøntegaard Delta Bit Rate (BDBR) \[106\] and the average percentage encoding time saving, \( \Delta T \), evaluated for

---

\(^5\)The CU classification models proposed in this Chapter are based on the motion features extracted from the preceding video frames. Adopting the same approach for the Random Access configuration must also consider the impact of future frames and is therefore outside the scope of this work.
Table 3.3: Simulation setup and configurations

<table>
<thead>
<tr>
<th>Configuration Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPs</td>
<td>22, 27, 32, 37</td>
</tr>
<tr>
<td>Encoding Config.</td>
<td>Low Delay P Main</td>
</tr>
<tr>
<td></td>
<td>Low Delay B Main</td>
</tr>
<tr>
<td>HEVC software version</td>
<td>16.0</td>
</tr>
<tr>
<td>Video sequence types</td>
<td>HD, UHD</td>
</tr>
<tr>
<td>Frame rates</td>
<td>HD: 30 fps, UHD: 50 fps</td>
</tr>
<tr>
<td>Number of frames</td>
<td>200</td>
</tr>
<tr>
<td>Machine</td>
<td>Intel Core i5 with 8GB RAM Ubuntu 14.04 LTS</td>
</tr>
</tbody>
</table>

the proposed and state-of-the-art algorithms by comparing the implementations of the respective algorithms with the HM 16.0 reference software [63]. In this context, $\Delta T$ is given by,

$$\Delta T = 100 \times \frac{T_{HM} - T_\rho}{T_{HM}},$$  

where $T_{HM}$, is the encoding time of HM16.0 and $T_\rho$ is the encoding time required for each fast encoding approach.

3.4.2 Results and Performance Analysis

The window length $W$ and the complexity control $\tau$ parameters defined in Sec. 3.2.1 and 3.2.2 will naturally affect the performance of the proposed algorithm. This section first discusses how they can be selected and their the impact, and is followed by an analysis of the overall performance and implications of using the proposed fast coding framework.

Complexity Control Parameter and Window Length Selection

The introduction of the window of length $W$ in the CU split likelihood model adaptations and their split decision predictions ensure that the both algorithms remain adaptive to changing content. $W$ is therefore essentially a temporal control on the adaptability of each CU split decision model. However, since shorter $W$ can trigger more intermediate training and longer $W$ can reduce content-adaptability, the selection of an optimal $W$ is critical to achieving good performance.

The experimental results illustrate the performance impact of $W$ on the encoding time reduction and BDBR in Fig. 3.9 for HD sequences. Here, the value of the complexity control parameter is set as $\tau = 0$, which results in $HTh_{spt}$ and $HTh_{nspt}$ being exactly similar to the mean $\gamma$ of the CU split, and non-split Gaussian distributions (as seen in
3.4. Experimental Results and Discussion

Fig. 3.9: The effect of the window length $W$ (when $\tau = 0$) on the coding efficiency and the encoding time reduction of the proposed encoding algorithm.

Fig. 3.10: The effect of the complexity control parameter $\tau$ (when $W = 30$) on the coding efficiency and the encoding time reduction of the proposed encoding algorithm.

Fig. 3.4 the mean values are already good approximations for these thresholds. It can be observed that the coding efficiency (BDBR) improves with smaller window sizes, while the encoding time performance ($\Delta T$) increases with increasing window size and vice-versa. Intuitively, this is due to less training being required for long $W$, yet longer window lengths also suggest less adaptability and sub-optimal quality. This is in contrast to the greater variability present when $W$ is shorter, where a smaller window size eventually results in a decrease in $\Delta T$ as well as BDBR decreases. An empirically determined window size of $W = 30$ that both provides comparable average BDBR increases to that of the state-of-the-art algorithms and also facilitates the adequate accumulation of statistical data (in general, an average of approximately 25 training occurrences are observed for a typical feature vector), is used in the performance analysis in the remainder of this discussion.

Unlike the window length that primarily affects the content-adaptiveness, the complexity control parameter $\tau$ mainly impacts the complexity and the coding efficiency. As described in Sec. 3.2.2, varying $\tau$ results in different $H_{Th_{sp}}$ and $H_{Th_{nsp}}$ thresholds, which alter the size of the middle region (cf. Fig. 3.4(b)) that requires the RD optimization. As was the case with the window length $W$, the experimental results illustrated in
3.4. Experimental Results and Discussion

Figure 3.11: The RD performance of the proposed and state-of-the-art algorithms compared to the HEVC 16.0 reference software.

Fig. 3.10 for different $\tau$ with HD sequences show that $\Delta T$ tends to increase with a corresponding increase of the BDBR. This is due to the fact that when the middle region is smaller, split decision of more CUs become incorrect due to fewer RD optimization occurrences, which negatively impacts BDBR but improves $\Delta T$. Naturally, when the region becomes larger the opposite is true, and is reflected in the performance in Fig. 3.10. Hence, the encoding performance results of the proposed algorithm discussed in the remaining sections use $\tau = 0$, which together with the previously selected $W$, corresponds to BDBR increases comparable to the state-of-the-art solutions.
3.4. Experimental Results and Discussion

Both of these parameters eventually pave way to trade-off the complexity and the coding efficiency of the proposed encoding algorithm to better suit the user or application requirements. Therefore, the derivation of $W$, and $\tau$ for a given requirement (i.e., encoding time saving) can be realized through the analysis depicted in Fig. 3.9 and Fig. 3.10, respectively. In addition, as discussed in Sec. 3.2.1, the threshold value $T$ in (3.9) decides the portion of CUs that will be decided to split, for a given feature vector $F$. For example, a larger $T$ results in less CUs being split, thus an increase in
3.4. Experimental Results and Discussion

$\Delta T\%$ and BDBR $\%$ can be observed. On the other hand, a smaller $T$ increases the number of CUs that are split which ultimately results in smaller $\Delta T\%$ and BDBR $\%$ increases. In this context, an empirically determined value of $T$ ($T = 0.6$) is used as the threshold, by the considering it’s impact on the RD efficiency.

Overall Performance of the Proposed Algorithm

The performance of the proposed algorithm is presented in the Tables 3.4 - 3.6 for the low delay $P$ and low delay $B$ configurations. Moreover, the Figs. 3.11 and 3.12 visually depict the RD performance and the average encoding time performance for 3 selected HD sequences which are encoded under low delay $P$ configuration, respectively. The encoding time reduction and the loss in coding efficiency achieved by the proposed fast coding framework is shown in two parts. First, the impact of only the CU size selection aspect (described in Sec. 3.2) denoted by SI is assessed. Then in SII the impact of including the motion vector reuse from the Inter $N \times N$ mode evaluation (described in Sec. 3.3.3) is evaluated. Both are compared with similar state-of-the-art algorithms.

The following discussion further analyses these results in terms of the variations seen for different content types, Quantization Parameters and other relevant attributes.

Examining the encoding time performance in Fig. 3.12 and the encoding time saving results illustrated in the Table 3.4 for a subset of sequences representing diverse content types and QPs ($QP = 22, 27, 32, 37$), a variation in $\Delta T$ with the corresponding bit rate of the video sequences can be observed. For example, $\Delta T$ tends to increase with the decreasing bit rates (i.e., increasing QP) for the proposed as well as the state-of-the-art algorithms. In general, this behaviour can be explained as follows. Typically, when encoding a CU at larger QPs, larger CUs and prediction modes such as SKIP and merge modes are favoured; thus, the algorithms that early terminate a CU at smaller depths and early detect SKIP/Merge modes, demonstrate an increased $\Delta T$ compared to smaller QPs that yield smaller CUs and fewer SKIP mode PUs.

However, interestingly, the variation of $\Delta T$ with QP is relatively large for the methods proposed by both Lee et al. [61], Shen et al. [64] as well as Correa et al. [68]. For example, the state-of-the-art algorithms demonstrate drastically varying $\Delta T$ when the $QP$ changes from 22 to 37 in Table 3.4. This is due to the evaluation of the CU Early Termination (ECUT) and CU Skip Estimation (CUSE) conditions [61] that require the encoder to evaluate the current CU (i.e., evaluate PU modes available for the current CU) prior to determining whether the CU should be split further. Similarly, the verification of the decision trees introduced in Correa et al. [68] is performed as the final operation at a particular depth level, and results in a similar behaviour. In these cases, video sequences that tend to use smaller CUs when using a lower QP will result in the algorithm unnecessarily evaluating the upper CU depth levels before early termination. Moreover, the effect is visible in the increased tendency to select SKIP or Merge modes in the algorithms by Lee et al. [61] and Vanne et al. [57], especially in the case of less complex, less textured contents for large QPs. In contrast, the proposed algorithm predicts the CU split decision prior to the encoding of a CU; thus, the unnecessary evaluations of larger CUs (i.e., PU mode evaluation for each depth level) are avoided. This leads far less performance variation between QPs, and suggests
Table 3.4: Encoding time saving with respect to QP and the content

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quantization Parameter (QP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>22 27 32 37 22 27 32 37 22 27 32 37 22 27 32 37 22 27 32 37</td>
</tr>
<tr>
<td></td>
<td>Kimono AM, HT (ΔT%)</td>
</tr>
<tr>
<td>Proposed SI</td>
<td>53 52 53 52</td>
</tr>
<tr>
<td>Proposed SII</td>
<td>55 57 55 56</td>
</tr>
<tr>
<td>Lee et al.[61]</td>
<td>27 35 44 53</td>
</tr>
<tr>
<td>Shen et al.[64]</td>
<td>39 41 43 46</td>
</tr>
<tr>
<td>Lu et al.[71]</td>
<td>29 29 27 26</td>
</tr>
<tr>
<td>Vanne et al.[57]</td>
<td>25 33 40 46</td>
</tr>
<tr>
<td>Correa et al.[68]</td>
<td>24 48 60 62</td>
</tr>
</tbody>
</table>

The sequence categories (i.e., LM, AM, HM, LT, HT) are defined as follows. LM: Low Motion, AM: Average Motion, HM: High Motion, LT: Low Texture, HT: High Texture.
Table 3.5: Overall performance of the proposed algorithm (low delay P)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed SI</th>
<th>Proposed SII</th>
<th>Lu et al. [71]</th>
<th>Lee et al. [61]</th>
<th>Shen et al. [64]</th>
<th>Vanne et al. [57]</th>
<th>Correa et al. [68]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta T$ (%)</td>
<td>BDBR (%)</td>
<td>$\Delta T$ (%)</td>
<td>BDBR (%)</td>
<td>$\Delta T$ (%)</td>
<td>BDBR (%)</td>
<td>$\Delta T$ (%)</td>
</tr>
<tr>
<td>Musicians 1080p</td>
<td>51</td>
<td>2.60</td>
<td>55</td>
<td>2.90</td>
<td>32</td>
<td>2.13</td>
<td>43</td>
</tr>
<tr>
<td>Band 1080p</td>
<td>56</td>
<td>1.78</td>
<td>59</td>
<td>1.82</td>
<td>40</td>
<td>0.68</td>
<td>47</td>
</tr>
<tr>
<td>Kimono 1080p</td>
<td>52</td>
<td>1.27</td>
<td>56</td>
<td>1.37</td>
<td>28</td>
<td>0.47</td>
<td>40</td>
</tr>
<tr>
<td>Parkscene 1080p</td>
<td>49</td>
<td>3.00</td>
<td>53</td>
<td>3.10</td>
<td>26</td>
<td>2.33</td>
<td>45</td>
</tr>
<tr>
<td>Dancer 1080p</td>
<td>52</td>
<td>1.32</td>
<td>55</td>
<td>1.69</td>
<td>29</td>
<td>3.90</td>
<td>49</td>
</tr>
<tr>
<td>GT Fly 1080p</td>
<td>52</td>
<td>1.68</td>
<td>54</td>
<td>1.74</td>
<td>34</td>
<td>3.65</td>
<td>50</td>
</tr>
<tr>
<td>Beergarden 1080p</td>
<td>55</td>
<td>1.00</td>
<td>58</td>
<td>1.01</td>
<td>16</td>
<td>0.54</td>
<td>48</td>
</tr>
<tr>
<td>Poznan 1088p</td>
<td>59</td>
<td>1.05</td>
<td>63</td>
<td>1.20</td>
<td>31</td>
<td>0.73</td>
<td>59</td>
</tr>
<tr>
<td>City 720p</td>
<td>54</td>
<td>1.76</td>
<td>58</td>
<td>2.38</td>
<td>21</td>
<td>1.11</td>
<td>52</td>
</tr>
<tr>
<td>Traffic 1600p</td>
<td>54</td>
<td>4.02</td>
<td>57</td>
<td>4.20</td>
<td>23</td>
<td>4.11</td>
<td>49</td>
</tr>
<tr>
<td>Men-Plants 2160p</td>
<td>59</td>
<td>2.08</td>
<td>62</td>
<td>2.48</td>
<td>41</td>
<td>2.03</td>
<td>51</td>
</tr>
<tr>
<td>Park-Buildings 2160p</td>
<td>60</td>
<td>2.59</td>
<td>62</td>
<td>2.60</td>
<td>29</td>
<td>1.10</td>
<td>55</td>
</tr>
<tr>
<td>Men-calendar 2160p</td>
<td>60</td>
<td>1.08</td>
<td>63</td>
<td>2.89</td>
<td>50</td>
<td>3.30</td>
<td>54</td>
</tr>
<tr>
<td>Average</td>
<td>55</td>
<td>1.93</td>
<td>58</td>
<td>2.26</td>
<td>30</td>
<td>2.00</td>
<td>49</td>
</tr>
</tbody>
</table>
Table 3.6: Overall performance of the proposed algorithm (low delay B)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed SI</th>
<th>Proposed SII</th>
<th>Lu et al. [71]</th>
<th>Lee et al. [61]</th>
<th>Shen et al. [64]</th>
<th>Vanne et al. [57]</th>
<th>Correa et al. [68]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta T$ (%)</td>
<td>BDBR (%)</td>
<td>$\Delta T$ (%)</td>
<td>BDBR (%)</td>
<td>$\Delta T$ (%)</td>
<td>BDBR (%)</td>
<td>$\Delta T$ (%)</td>
</tr>
<tr>
<td>Musicians 1080p</td>
<td>53</td>
<td>1.96</td>
<td>57</td>
<td>2.03</td>
<td>29</td>
<td>0.90</td>
<td>43</td>
</tr>
<tr>
<td>Band 1080p</td>
<td>57</td>
<td>2.08</td>
<td>61</td>
<td>2.20</td>
<td>37</td>
<td>1.10</td>
<td>49</td>
</tr>
<tr>
<td>Kimono 1080p</td>
<td>55</td>
<td>2.73</td>
<td>59</td>
<td>2.81</td>
<td>24</td>
<td>0.96</td>
<td>40</td>
</tr>
<tr>
<td>Parkscene 1080p</td>
<td>52</td>
<td>2.34</td>
<td>55</td>
<td>2.48</td>
<td>26</td>
<td>0.67</td>
<td>46</td>
</tr>
<tr>
<td>Dancer 1080p</td>
<td>54</td>
<td>1.87</td>
<td>57</td>
<td>2.13</td>
<td>29</td>
<td>3.56</td>
<td>50</td>
</tr>
<tr>
<td>GT Fly 1080p</td>
<td>53</td>
<td>1.35</td>
<td>57</td>
<td>1.72</td>
<td>37</td>
<td>5.48</td>
<td>46</td>
</tr>
<tr>
<td>Beergarden 1080p</td>
<td>56</td>
<td>1.71</td>
<td>61</td>
<td>1.18</td>
<td>12</td>
<td>0.23</td>
<td>48</td>
</tr>
<tr>
<td>Poznan 1088p</td>
<td>62</td>
<td>1.18</td>
<td>67</td>
<td>1.27</td>
<td>30</td>
<td>1.01</td>
<td>60</td>
</tr>
<tr>
<td>City 720p</td>
<td>52</td>
<td>1.44</td>
<td>56</td>
<td>2.16</td>
<td>20</td>
<td>1.00</td>
<td>48</td>
</tr>
<tr>
<td>Traffic 1600p</td>
<td>57</td>
<td>3.50</td>
<td>60</td>
<td>3.87</td>
<td>22</td>
<td>1.00</td>
<td>50</td>
</tr>
<tr>
<td>Men-Plants 2160p</td>
<td>62</td>
<td>3.00</td>
<td>66</td>
<td>3.10</td>
<td>32</td>
<td>2.01</td>
<td>54</td>
</tr>
<tr>
<td>Park-Buildings 2160p</td>
<td>63</td>
<td>1.49</td>
<td>66</td>
<td>2.02</td>
<td>21</td>
<td>1.16</td>
<td>59</td>
</tr>
<tr>
<td>Men-calendar 2160p</td>
<td>63</td>
<td>3.17</td>
<td>67</td>
<td>3.40</td>
<td>43</td>
<td>4.93</td>
<td>57</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>57</strong></td>
<td><strong>2.14</strong></td>
<td><strong>61</strong></td>
<td><strong>2.33</strong></td>
<td><strong>28</strong></td>
<td><strong>1.84</strong></td>
<td><strong>50</strong></td>
</tr>
</tbody>
</table>
that the proposed algorithm is suitable for encoding videos across a wider range of bit rates unlike the existing approaches.

With respect to different types of content, further analysis reveals that the performance of the proposed and state-of-the-art algorithms vary considerably. It can be observed that all the state-of-the-art algorithms demonstrate a relatively large encoding time reduction for less textured sequences such as “Poznan Street” (Table 3.4). This encoding time reduction is mainly due to the skipping of rarely used CU depth levels [64] and the early termination methods that have been employed [61, 71]. Hence, the less complex the content (less textured and simple motions), the more likely it becomes that they will be encoded with larger CUs, causing the early termination of the CU size evaluation to exhibit an increased encoding time saving. In addition, static or content with simple motions can exploit the CU depth range estimation algorithms in [64] to skip the unnecessary CU depth levels.

However, in the case of more textured sequences with average or low motions (e.g., “Kimono”, “Musicians” etc.) much smaller CU sizes are required; thus, the early CU termination at upper depth levels become ineffectual, leading to a much lower encoding time saving. Moreover, the skipping of the rarely used CU depth levels in [64] eventually leads to relatively high coding losses, especially in the case of sequences that exhibit multiple localized motions (i.e., “Poznan Street”). The depth range estimations in this case often become less accurate, and the errors in these are propagated across the frame to further deteriorate the coding efficiency.

That said, the method proposed by Shen et al. performs reasonably well even with sequences that generally exhibit uniform motion across the frame (e.g., “Kimono”), to the detriment of the encoding time performance. In contrast to this approach, the proposed framework exhibits a performance that varies much less with the content. It has achieved average encoding time reductions within the range of 50 – 60% across all the test sequences (including the synthetic ones such as “Dancer”, and more complex sequences such as “Traffic”) with average BDBR increases in the range of 1.8 – 2.3%. The effect of the content’s complexity therefore appears trivial to the proposed algorithm, due to its early prediction of the CU split decision prior to the actual encoding of the CU. Furthermore, the proposed algorithm only evaluates the selected CU depth level, thus, the encoding time consumed for unnecessary CU depth level evaluation is avoided. Therefore, it is evident that the proposed algorithm is suitable for encoding a wider range of video sequences ranging from low to highly complex texture and motion characteristics, providing an edge over the prevailing state-of-the-art solutions.

The overall performance of the proposed method as well as state-of-the-art methods are averaged over the evaluated QPs (i.e., 22, 27, 32, 37) and are summarized for a wider range of test sequences in the Tables 3.5 and 3.6, for Low Delay P and Low Delay B configurations, respectively. The content-wise performance variance noticed in the experimental results presented in the Table 3.4, is also visible in the averaged encoding performance results. For example, encoding time performance of the proposed algorithms on average show a consistent performance for all the test sequence, whereas the state-of-the-art algorithms show a drastically varying performance. Moreover, despite the proposed method’s slight increase in the BDBR, the algorithms are still comparable with the average performances of the state-of-the-art methods.
However, for sequences such as “Traffic”, which has fast moving objects throughout the sequence, the proposed method and the state-of-the-art algorithms demonstrate a relatively large BDBR in the range of 4%. This suggests that the complex motions and rapidly changing content has caused the decision making algorithms to make fewer efficient decisions compared to the sequences with average motion complexity. One of the solutions envisioned in this case is to reduce the window size $W$ to increase the content adaptability of the proposed algorithm. In this context, the proposed algorithm is sufficiently flexible in its design parameters to cater for the diversities of the video sequences. In addition to the joint utilization of the two independent models, and the complexity control parameter coupled with the intermediate training phases provide a more attractive training solution for the proposed framework compared to the use of fully RD optimized training frames at pre-defined intervals as proposed by Lee et al. in [61]. The content adaptability and the effectiveness of the online training in the proposed algorithm are further corroborated when comparing its performance with that of the algorithm proposed by Correa et al. [68], whose BDBR exceeds 10% for some sequences due to its fixed thresholds and offline-trained decision trees based on a selected set of training sequences. Thus, despite good encoding time reductions significant coding efficiency losses may be incurred when encoding previously unseen content.

The effect on the proposed fast CU selection method when supplemented by the motion vector reuse mechanisms which are used to optimize the PU level motion estimation, is presented under “Proposed SII” in Tables 3.5 and 3.6. In this case, the experimental results demonstrate an additional average encoding time saving of 3% and 4% for the Low Delay P and Low Delay B configurations, respectively. However, BDBR increases of 0.37% and 0.19% for the respective configurations are exhibited in comparison to the “Proposed SI” solution due to the skipping of certain motion estimation instances.

Finally, the computational cost of the training phases and the decision making stages are all included in the encoding performance results that are reported in this work. Therefore, it is evident that the additional complexities introduced by the proposed algorithms are negligible in comparison with the significant time saving that can be achieved by incorporating these algorithms into the encoding cycle. Moreover, the objective quality assessments made with PSNR suggest that the visual quality impact is trivial in the videos encoded using the proposed algorithm compared to the same sequences encoded using the state-of-the-art approaches and HM 16.0 encoder.

### 3.5 Summary

This Chapter introduces a content adaptive fast CU size selection algorithm for HEVC low delay video encoding. In this context, two CU classification methods are first introduced to model the split likelihood of a particular CU with a given feature set. Secondly, a moving window based feature selection approach is introduced to ensure that the statistical data considered to the CU split decision are content adaptive. Finally, an implementation friendly, fast encoding algorithm is introduced with the flexibility to effectively trade-off the encoding complexity to the coding efficiency.
The experimental results reveal a consistent encoding time reduction of 58% and 61% for the low delay P and low delay B configurations, respectively, across a range of video content types and quality levels. The content adaptive nature of the feature selection and decision making process of the proposed algorithm maintains the impact to the coding efficiency to a minimum (i.e., a consistent average 2.29 % BDBR increase) compared to the HM16.0 reference encoder implementation.

In conclusion, it is evident that achieving a consistent encoding time reduction across diverse content types and quality level is challenging due to the dynamics of the video contents. Thus, the simultaneous use of two CU split likelihood models facilitate the pre-determination of the CU sizes as well as an attractive mechanism to train the models online during the encoding process. In addition, the complexity control and window size design parameters introduce a flexible approach to effectively trade-off the coding efficiency to the encoding complexity, depending on the requirement. Thus, the mathematical derivations and algorithmic implementations presented herein, make a pivotal contribution to the video coding to make the HEVC encoding architecture less complex for average CE devices.
Chapter 4

Decoding Complexity-Aware HEVC Video Coding

The decoding complexity and the associated energy consumption during a decoding process is highly correlated with the codec as well as the video content being decoded (ref. Sec. 2.4). Therefore, reducing the complexity of the HEVC encoded bit streams is seen as a potential application layer solution to reduce the decoding complexity and its associated energy consumption. In this context, this Chapter introduces the proposed algorithms to generate decoding complexity-aware HEVC encoded bit streams using a decoding complexity–rate–distortion model.

Generating HEVC decoding complexity-aware bit streams require the encoder to be aware of the associated decoding complexities of the assortment of HEVC coding modes and features for a given content. Thus, this Chapter first introduces a detailed and an accurate HEVC decoding complexity model for intra- and inter- prediction. Thereafter, a comprehensive analysis is carried out to analyze the behaviour of decoding complexity, rate, and distortion parameters for a given content. Next, an algorithm is proposed to encode decoding complexity-aware HEVC bit streams that minimizes the impact on both rate and distortion.

4.1 Decoding Complexity Modelling

In general, the energy consumption of a decoder depends on a number of factors that are both architecture and implementation dependent (instruction set, memory management, CPU load balancing, voltage and frequency levels, etc.). Yet, with respect to a given architecture, the energy consumed when decoding the video stream is tightly coupled to the computational complexity [29, 30, 31, 89] of the decoding operations; hence, this coupling can be used to indirectly reduce the energy consumed to decode a bit stream during the bit stream creation process itself. In order to create an encoder that also considers the complexity of decoding the bit stream it generates requires that it is made aware of the complexity of decoding operations. This section proposes and briefly illustrates one such decoding complexity model combining intra- and inter-frame decoding in a HEVC bit stream.
4.1. Decoding Complexity Modelling

Decoding complexity is tightly coupled to the HEVC’s coding features used in a bit stream. For illustration purposes, Fig. 4.1 indicates the variations in relative complexity of the major decoding operations for each type of CU. It can be seen that the decoding operations can be separated into two basic phases; the decoding phase and the decompression phase. The decoding phase consists of entropy decoding of the transformed residual coefficients and syntax elements, whereas the decompression phase involves the complexities associated with the prediction and reconstruction operations.

The complexity of the decoding phase $C_{\text{dec}}$ can be defined as

$$C_{\text{dec}} = C_{\text{pMode}} + C_{\text{pSize}} + C_{\text{pInfo}} + C_{\text{coeff}}, \quad (4.1)$$

where $C_{\text{pMode}}$, $C_{\text{pSize}}$, $C_{\text{pInfo}}$ and $C_{\text{coeff}}$ are the complexities required to decode the prediction mode, PU size, prediction information (i.e., luma and chroma prediction modes for intra-predicted CUs and motion vector information for inter-predicted CUs) and the transform coefficients, respectively. However, considering the percentage complexity of individual processes, $C_{\text{dec}}$ can be further simplified as

$$C_{\text{dec}} \approx C_{\text{coeff}} \triangleq f(N_{\text{coeff}}), \quad (4.2)$$

Figure 4.1: A high level illustration of the relative complexity of decoding operations obtained for the “Kimono 1080p” sequence encoded at $QP = 22$ using the Random Access configuration for the HM 16.0 reference decoder.
4.1. Decoding Complexity Modelling

where $C_{\text{coeff}}$ is a linear function of $N_{\text{coeff}}$; the number of non-zero coefficients within the block [107].

The decompression complexity of the decoder $C_{\text{dci}}$ for an intra-predicted CU can be expressed as

$$
C_{\text{dci}} = \sum_{n=1}^{N} \left\{ C_{\text{rf}}^w(n) + C_{\text{lm}}^w(n) + 2C_{\text{ch}}^w(n) + C_{\text{it}}^w(n) \right\},
$$

where $C_{\text{rf}}, C_{\text{lm}}, C_{\text{ch}}$ and $C_{\text{it}}$ denote the complexity requirements for handling the reference samples, luma and chroma prediction, and the inverse transform operations, respectively [107]. Here, $n = 1, \ldots, N$ represent the individual TUs that reside within the CU with the TU size $w \in \{4, 8, 16, 32\}$ and $\text{ln} \in \{0, \ldots, 34\}$ and $\text{ch} \in \{0, \ldots, 4\}$, define the luma and chroma prediction modes, respectively. In this case, it should be noted that $C_{\text{it}}$ becomes a non-zero only for the TUs in which the CBF is also non-zero.

In this context, for the general case, the total decoder computational complexity for the $i$th intra-predicted CU can be expressed as the summation of coefficient decoding and decompression complexities and the resulting overhead components. Thus, the $C_{\text{Intra}}^{\text{CU}}(i)$ is given by,

$$
C_{\text{Intra}}^{\text{CU}}(i) = C_{\text{dec}} + C_{\text{dci}} + \sum_{k=1}^{K} C_{\alpha}(k) \times \frac{\xi(k)}{\chi(k)}.
$$

Here, $k = 1, \ldots, K$ correspond to the number of TU sizes that reside within the $i$th CU and $C_{\alpha}(k)$ encompasses the overhead due to the TU quadtree structure for the $k$th TU [107] and is scaled with the ratio between the actual number of TUs with the $k$th TU size ($\xi(k)$) and the total number of TUs (of similar size to the $k$th TU size) that could exist within a CTU ($\chi(k)$) [107]. For example, if $8 \times 8$ TUs exist in the $j$th CU, $\chi(k)$ is 64 and $\xi(k)$ is the number of $8 \times 8$ TUs that actually exist within the $j$th CU.

Similarly, the decompression complexity $C_{\text{dc}}$ for an inter-predicted CU can be expressed as

$$
C_{\text{dc}} = \sum_{m=1}^{M} \left\{ C_{\text{fc}}^{u\eta}(m) + C_{\text{lf}}^{u}(m) + 2C_{\text{cf}}^{u}(m) + C_{\text{wp}}^{u}(m) \right\} + \sum_{n=1}^{N} C_{\text{it}}^w(n) + C_{T},
$$

where $C_{\text{fc}}, C_{\text{lf}}, C_{\text{cf}}, C_{\text{wp}}, C_{\text{it}}$ and $C_{T}$ correspond to the decoding complexities for the filter copying (i.e., for integer-pel motion vectors), luma filtering, chroma filtering, weighted average prediction, inverse transform and reconstruction operations, respectively. $M$ and $N$ denote the total number of PUs and TUs that make up the CU, respectively.

Furthermore, $u$ indicate the PU size whereas $\eta = 1$ and $\eta = 2$ indicate the presence of uni- and bi-directional prediction, respectively.

In addition, it should also be noted that $C_{T}$ and $C_{\text{cf}}$ should be doubled whenever both vertical and horizontal filtering operations are required in a PU. Moreover, $C_{\text{it}}$ becomes non-zero only when a TU consists of non-zero coefficients. Thus, the total decoding complexity for the $j$th inter-predicted CU becomes

$$
C_{\text{Inter}}^{\text{CU}}(j) = C_{\text{dec}} + C_{\text{dc}} + \frac{C_{\alpha}^{u}(j)}{\zeta(j)}.
$$
4.1. Decoding Complexity Modelling

$C_0$ encompasses the overhead generated due to the quadtree structure of the CTU that results from the $j^{th}$ CU for the selected PU mode. $\zeta$ represents the total number of CUs (of the same size as the $j^{th}$ CU) that could exist within a CTU.

The decoding complexity for the CTU $C_{CTU}$ therefore becomes the summation of (4.4) and (4.6) given by

$$C_{CTU} = \sum_{i=1}^{I} C_{C_U}^{Intra}(i) + \sum_{j=1}^{J} C_{C_U}^{Inter}(j), \quad (4.7)$$

where $I$ and $J$ denote the total number of intra- and inter-predicted CUs that constitute the CTU.

4.1.1 Decoding Complexity Profiling

Identifying the relative complexity level of the decoder operations and model parameters identified in Sec. 4.1 is crucial in making the proposed decoding complexity model robust and reliable. Thus, a detailed complexity profiling that captures the intricate decoding complexities of each coding mode and feature needs to be carried out for multiple video contents and quality settings. In this case, estimating the energy consumption/decoding complexity level of the decoder operations is attempted using both hardware- and software-based setups. For example, Herglotz et al. [102][108] utilize a hardware setup that measures the voltage and ampere readings to determine the energy requirements within the decoder. However, the level of details captured is shown to be inadequate when considering the intricate CU level coding modes and features and their decoding complexities. Instruction level profiling of the software codecs using commonly used tools such as Valgrind [22] on the other hand is proven to be a useful in numerous occasions [109][110]. For instance, an instruction level profiling of a software decoder provides an insight into the intricate relative complexities associated with the assortment of coding modes and features available within the HEVC standard.

Analyzing the decoding complexity levels using estimated CPU cycles is extremely crucial as the CPU cycles consumed for a particular coding mode/feature is a non-trivial

Figure 4.2: Modeling of estimated CPU cycles with the number of non-zero coefficients identified within a CTU.

Analyzing the decoding complexity levels using estimated CPU cycles is extremely crucial as the CPU cycles consumed for a particular coding mode/feature is a non-trivial
Table 4.1: Relative complexity levels of decoding operations for HEVC intra-predicted CUs

<table>
<thead>
<tr>
<th>TU size</th>
<th>Prediction Process ($C_{im}$)$\dagger$</th>
<th>DC</th>
<th>Planar</th>
<th>Vertical</th>
<th>Horizontal</th>
<th>Integer Angles</th>
<th>Fractions</th>
<th>$C_{cf}$</th>
<th>$C_{it}$</th>
<th>$C_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(w)</td>
<td></td>
<td>1</td>
<td>0</td>
<td>26</td>
<td>10</td>
<td>2</td>
<td>18</td>
<td>34</td>
<td>Hor.</td>
<td>Ver.</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>13707</td>
<td>44097</td>
<td>24332</td>
<td>42211</td>
<td>40612</td>
<td>24332</td>
<td>22733</td>
<td>47510</td>
<td>29632</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>3835</td>
<td>11857</td>
<td>8294</td>
<td>12892</td>
<td>11192</td>
<td>8293</td>
<td>6593</td>
<td>12971</td>
<td>8376</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>1203</td>
<td>3417</td>
<td>2894</td>
<td>4166</td>
<td>3392</td>
<td>2893</td>
<td>2169</td>
<td>3858</td>
<td>2636</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>463</td>
<td>1117</td>
<td>1201</td>
<td>1552</td>
<td>1552</td>
<td>1201</td>
<td>869</td>
<td>1346</td>
<td>995</td>
</tr>
</tbody>
</table>

$\dagger$ The relative decoding complexity for chroma prediction ($C_{ch}$) is calculated based on the per pixel complexity levels estimated from the corresponding luma complexity.

Table 4.2: Relative complexity levels of decoding operations for HEVC inter-predicted CUs

<table>
<thead>
<tr>
<th>$C_{fc}$ (per 8×8 PU)$\dagger$</th>
<th>$C_{lf}$ (per 8×8 PU)$\dagger$</th>
<th>$C_{cf}$ (per 8×8 PU)$\dagger$</th>
<th>$C_{wp}$ (per 8×8 PU)$\dagger$</th>
<th>$C_{r}$ (per 8×8 CU)$\dagger$</th>
<th>$C_{it}$ (per 4×4 TU)$\dagger$</th>
<th>$C_o$ (per 8×8 CU)$\dagger$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta = 1$</td>
<td>$\eta = 2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>626</td>
<td>1389</td>
<td>14069</td>
<td>1957</td>
<td>16365</td>
<td>11552</td>
<td>6029</td>
</tr>
</tbody>
</table>

$\dagger$ The relative decoding complexity for an arbitrary sized CU, TU or PU is calculated based on the per pixel numerical values estimated from the presented complexity levels.
factor that determines the amount of energy is consumed during the decoding process. For example, the energy consumption in CMOS circuits exhibit a linear relationship with the CPU’s clock frequency \([89]\) given by,

\[
E = C_{\text{EFF}} \times V^2 \times C,
\]

(4.8)

where \(C_{\text{EFF}}, V, C\) and \(E\) are the effective switched capacitance, supply voltage, CPU cycles, and energy consumption, respectively \([89]\). Therefore, exploiting the relationship between computational complexity, execution time and the clock frequency, the energy consumed per decoding operation can be mapped to a quadratic relationship to the operation’s computational complexity \([89]\) for a given decoder architecture. Therefore, simplifying the decoding operations, and thereby reducing the device’s energy consumption in this manner, has been attempted on numerous occasions in the recent literature (ref. Sec. 2.4). In this context, number of CPU execution cycles consumed by the decoder’s operations for a particular CPU architecture and a software decoder can be considered as a substitute for the energy it consumes. Furthermore, the proposed algorithms in this thesis utilizes the commonly used Valgrind \([22]\) based instruction level profiling tools to profile the decoder operations identified in Sec. 4.1 and defines relative complexity levels which are used throughout the course of this work. For example, the Tables 4.1 and 4.2 summarize numerically the relative complexity of these parameters for the HM 16.0 reference HEVC decoder implementation in the x86 architecture obtained in \([101]\) and \([107]\). Moreover, Fig. 4.2 illustrates the linear relationship that exist between the number of non-zero coefficients and CPU cycles defined in (4.2). In this case, the proposed method utilizes the relationship defined by,

\[
C_{\text{coeff}} = 818.2 \times N_{\text{coeff}} + (1.039 \times 10^5),
\]

(4.9)

to derive the estimated CPU cycles for a given number of non-zero coefficients.

Once the decoding complexity model and corresponding relative complexity levels are obtained after the profiling stage, the proposed model can be incorporated into the HEVC encoder such that the encoder is aware of the estimated decoding complexity in terms of CPU cycles with respect to the HM 16.0 decoder for a Intel x86 CPU architecture. However, the complex relationship that exist among the three parameters (i.e., the decoding complexity, rate and distortion) must be thoroughly examined and modelled in order to determine the impact each parameter has on the others. Thus, the following sections analyze these relationships and proposes a decoding-complexity-aware encoding framework for HEVC bit stream generation.

### 4.2 Decoding-Complexity – Distortion Analysis and Modelling

The selection of a coding mode and a structure that is appropriate in terms of the decoding complexity and the coding efficiency requires an in-depth analysis of the impact of various coding parameters in a range of situations. Empirical analysis performed on a set of test sequences for a range of QPs and coding configurations as specified in \([16]\) reveals a complex relationship between the video distortion, decoding complexity
4.2. Decoding-Complexity – Distortion Analysis and Modelling

and bit rate\(^1\). In this context, the investigations carried out in \([111]\) and \([112]\) define the relationship between the bit rate and the distortion, and introduces the Lagrangian multiplier \(\lambda_r\) that trades-off the two parameters during the coding mode selection. However, an additional parameter to trade-off the decoding complexity with respect to the rate and distortion is also required if the bit stream’s decoding complexity is considered as a third criterion for the coding mode selection. To this end, this section introduces an approach to analyze the relationship between the distortion and the decoding complexity such that a preliminary trade-off parameter for distortion and decoding complexity can be derived. This will be further extended in Sec. 4.3 for the three parameter Lagrangian optimization in conjunction with the impact of the bit rate.

4.2.1 Decoding-Complexity – Distortion Analysis

In order to analyze the relationship between the distortion and the corresponding decoding complexity, the achievable range of both parameters must first be observed. To this end, the optimization function of the encoder can be modified to seek the minimum of each parameter, separately, before collating the resulting distortion and decoding complexity for further analysis.

These optimization functions can be expressed as

\[
\min_{p \in \mathcal{P}} D(p) \quad (4.10)
\]

and

\[
\min_{p \in \mathcal{P}} C(p), \quad (4.11)
\]

where \(p\) is a particular coding parameter combination in the set of all possible coding parameters \(\mathcal{P}\), and \(D(p)\) and \(C(p)\) are the distortion and decoding complexity associated with \(p\), respectively. In this case, \((4.10)\) only considers the distortion associated with a coding mode; thus, results in a bit stream with the minimum distortion \(D_{\min}\). Similarly, \((4.11)\) only considers the decoding complexity of a coding mode, which produces the encoded bit stream with the minimum decoding complexity \(C_{\min}\). It should be noted that the decoding complexity model in Sec. 4.1 enables the estimation of the decoding complexity for all \(p\); hence, the minimization of \(C(p)\) as per \((4.11)\) becomes a reliable approach to minimize the decoding complexity.

\(^1\)For this analysis, 50 frames of each test sequence are encoded at QP values ranging from 0 – 51 capturing both inter- and intra-predicted frames.
4.2. Decoding-Complexity – Distortion Analysis and Modelling

(a) Intra frames (min $D(p)$)

(b) Inter frames (min $D(p)$)

(c) Intra frames (min $C(p)$)
4.2 Decoding-Complexity – Distortion Analysis and Modelling

(d) Inter frames (min $C(p)$)

Figure 4.3: The distortion and decoding complexity variations in intra- and inter-predicted frames for QP $\in \{1, \ldots, 51\}$ of the “Basketball Pass” sequence. The Figs. 4.3(a) and 4.3(b) correspond to distortion minimization in (4.10), and the Figs. 4.3(c) and 4.3(d) correspond to decoding complexity minimization in (4.11).

An example of the decoding complexity and distortion behaviour observed for these two scenarios in (4.10) and (4.11) are illustrated in Fig. 4.3 for the “basketball pass” sequence for both intra- and inter-predicted frames. In all four cases, it can be observed that the distortion tends to monotonically increase with decreasing decoding complexity. The relationship between the two parameters can therefore be characterized as

$$D = \alpha C^{-\beta}, \quad (4.12)$$

where $\alpha$ and $\beta$ are content dependent parameters. Thus, for a particular content and coding scenario, the trade-off between the distortion and the decoding complexity can be expressed in terms of the slope of the distortion-complexity curve, given by

$$\lambda_c \triangleq \frac{\partial D}{\partial C} = \alpha \beta C^{-\beta}. \quad (4.13)$$

Expressing the decoding-complexity-distortion trade-off as a Lagrangian optimization problem, an optimization function can therefore be formulated as

$$\min_{p \in P} J_{CD} \quad | \quad J_{CD} \triangleq D(p) + \lambda_c C(p), \quad (4.14)$$

where $\lambda_c$ represents the Lagrangian parameter. For the general case however, where the relationship in (4.12) is unknown, a generic $\lambda_c$ is computed as described next.

4.2.2 Decoding Complexity-Distortion Optimization: Computing $\lambda_c$

The Lagrangian parameter $\lambda_c$ in (4.13) is both content and QP dependent. A range of QPs and contents must therefore be evaluated when computing an optimum $\lambda_c$ as described below.
4.2. Decoding-Complexity – Distortion Analysis and Modelling

Figure 4.4: (a) The typical decoding-complexity-distortion behaviour observed between the two points corresponding to $D_{\text{min}}$ and $C_{\text{min}}$ for a particular QP. (b) A representation of the normalized decoding complexity and distortion for a particular QP. The depicted tangent to the curve in (b) indicates the $\lambda_c$ which results in equal trade-off between the decoding complexity and distortion.

Observing the decoding complexity vs. distortion behaviors in Fig. 4.3, it is clear that each pair of $D_{\text{min}}$ and $C_{\text{min}}$ exhibit a relationship similar to that illustrated in Fig. 4.4(a) for a particular QP.
4.2. Decoding-Complexity – Distortion Analysis and Modelling

(a) Intra frames

(b) Inter frames

Figure 4.5: $D$ and $C$ relationship averaged for the set of sequences at QP = 22. The Figs. 4.5(a) and 4.5(b) correspond to the actual averaged $D$ and $C$ values for intra and inter frames, respectively.

Here, $D_{\text{min}}$ corresponds to $\lambda_c = 0$ and $C_{\text{min}}$ corresponds to $\lambda_c \to \infty$ for the optimization function in (4.14). The $\lambda_c \in [0, \infty)$ region between these two points therefore represents a space where the distortion and the decoding complexity can be traded-off against each other. In order to make the computation of an optimum $\lambda_c$ tractable, its impact is investigated in the $\lambda_c$ range of $[0, \lambda_{\text{max}}^c]$ for $QP \in \{0, \ldots, 51\}$. In this context, considering the empirical observations of corresponding $\lambda_c$ values which results in a decoding complexity $C \approx C_{\min}(QP)$, $\lambda_{\text{max}}^c$ is determined as,

$$\lambda_{\text{max}}^c(QP) = 2 \times \left\{ \frac{\bar{D}(QP) - D_{\min}(QP)}{\bar{C}(QP) - C_{\min}(QP)} \right\}, \quad (4.15)$$

where $\bar{D}$ is the distortion corresponding to $C_{\min}$, and $\bar{C}$ is the decoding complexity corresponding to $D_{\min}$.

Next, the distortion and decoding complexity data is collected from multiple experiments using $\lambda_c \in [0, \lambda_{\text{max}}^c]$ in the optimization function in (4.14), and are averaged across different test sequences (six test sequences in CIF and HD resolution) for a par-
4.2. Decoding-Complexity – Distortion Analysis and Modelling

(a) Intra frames

(b) Inter frames

Figure 4.6: $D$ and $C$ relationship averaged for the set of sequences at QP = 22. The Figs. 4.6(a) and 4.6(b) correspond to the averaged normalized $D$ and $C$ values for intra and inter frames, respectively.

A similar behavior is exhibited for all QP values ranging from 0–51.

The Figs. 4.5 and 4.6 illustrate the averaged and normalized $D$ and $C$ distributions, respectively at QP = 22 for intra- and inter-predicted frames, and Fig. 4.7 illustrates the behavior of the optimum $\lambda_c$ for different values of QP. The optimum $\lambda_c$ values for a decoding-complexity and distortion optimized encoding as in (4.14) is given by

$$\lambda_c = \begin{cases} 
8.739 \times 10^{-5} \cdot e^{0.1327 \cdot QP} & \text{Intra frames} \\
0.001393 \cdot e^{0.0023 \cdot QP} & \text{Inter frames}
\end{cases} \quad (4.16)$$

The decoding-complexity and distortion trade-off modelled above can now be used together with the bit rate to develop a decoding complexity–rate–distortion optimized coding mode selection algorithm as described in the following section.

\(^2\)A similar behavior is exhibited for all QP values ranging from 0–51.
4.3 Decoding-Complexity – Rate – Distortion Optimized Video Encoding

The relationship between the distortion $D$ and the bit rate $R$ has been extensively studied and a QP dependent trade-off parameter $\lambda_r$ has been defined in [111, 112] such that the cost function for coding mode selection in RD optimization for a given QP is given by

$$\min_{p \in P_{\text{opt}}} J_{\text{RD}} \mid J_{\text{RD}} \triangleq D(p) + \lambda_r R(p).$$  \hspace{1cm} (4.17)

Similarly, the relationship between distortion $D$ and the decoding complexity $C$, presented in Sec. 4.2, defines a trade-off parameter $\lambda_c$ for the decoding complexity-distortion (CD) optimization cost function given in (4.14). However, having all three parameters in the optimization cost function when determining the optimum coding mode requires an in depth analysis of the impact of the individual parameters. The Lagrangian cost function for this case can be expressed as,

$$\min_{p \in P_{\text{opt}}} J_{\text{CRD}} \mid J_{\text{CRD}} \triangleq D(p) + \bar{\lambda}_r R(p) + \bar{\lambda}_c C(p),$$  \hspace{1cm} (4.18)

where $\bar{\lambda}_r$ and $\bar{\lambda}_c$ are the bit rate and decoding complexity trade-off parameters, respectively. However, obtaining optimal values for $\bar{\lambda}_r$ and $\bar{\lambda}_c$ directly from (4.18) is not
4.3. Decoding-Complexity – Rate – Distortion Optimized Video Encoding

straightforward, and the experimental approach described next is adopted to this end.

First, the general behaviour of the cost function $J_{CRD}$ is investigated for the range of the two complexity parameters, where $\lambda_r \triangleq \alpha \lambda_r$ and $\lambda_c \triangleq \delta \lambda_c$ for $\alpha \in \{0, \ldots, \infty\}$ and $\delta \in \{0, \ldots, \infty\}$. Note that $\lambda_r$ and $\lambda_c$ in (4.17) and (4.14) respectively are used for this purpose. In this study, 50 frames of each test sequence are encoded in this manner, and the results are used to infer the general behaviour of $J_{CRD}$ in the full parameter space of $\lambda_r$ and $\lambda_c$.

Next, in order to determine a suitable operating point in this parameter space, the rate, distortion and complexity at each operating point is compared with the respective values for each obtained when using the Lagrangian cost function in (4.17). To facilitate this, the percentage differences of each parameter, i.e., $\Delta R$, $\Delta D$ and $\Delta C$, given by

$$\Delta \Gamma = 100 \times \frac{\Gamma_{RD} - \Gamma_{CRD}}{\Gamma_{RD}},$$

is used. Here, $\Gamma$ represents the distortion $D$, bit rate $R$, and decoding complexity $C$, while $\Gamma_{RD}$ and $\Gamma_{CRD}$ correspond to the scenarios where the cost functions in (4.17) and (4.18), respectively, are applied.

Fig. 4.7 illustrates the distribution of $\Delta R$, and $\Delta D$ for a single test sequence (Kimono 1080p) at a selected set of QPs for both inter and intra predicted frames. Each data point corresponds to the deviation of an operating point in Rate–Distortion–Decoder-Complexity space (when using (4.18) as the mode selection cost function) for a particular $\lambda_r$ and $\lambda_c$ pair from the bit rate and distortion of the RD optimised operating point (when using (4.17) as the mode selection cost function) for the respective QP. Here, the differences in the behaviour for the different frame types and $(\lambda_r, \lambda_c)$ pairings can be observed. Moreover, it can also be observed that distortion for example deviates significantly from that of the RD optimised value for certain $(\lambda_r, \lambda_c)$ combinations. Therefore, selecting an appropriate $(\lambda_r, \lambda_c)$ combination boils down to an engineering design decision. Hence, the design approach followed in this work is detailed as follows.

One of the objectives of this work is to minimize the impact on bit rate as well as distortion when attempting to reduce the decoding complexity. Therefore, a design constrained is enforced as follows to obtain the appropriate scaling $(\alpha, \delta)$ parameters.

In this context, an operating point in the subspace of the parameter space of $\alpha$ and $\delta$ is derived as the operating point which gives minimum $\Delta D$ with the constraint $\Delta R \leq 1$. (The selected point in each QP for the kimono sequence is highlighted in Fig. 4.7). Thereafter, a generic set of scaling factors for $(\alpha_0, \delta_0)$ which are utilized for the remainder of this work, are obtained as the average of the parameters obtained for the individual test sequences, and is given by

$$(\alpha_0, \delta_0) = \begin{cases} 0.95, 0.29 & \text{Intra frames} \\ 1.01, 0.25 & \text{Inter frames} \end{cases} \quad (4.20)$$

Thus, the rate and complexity trade-off factors in (4.18) are now become $\lambda_r = \alpha_0 \lambda_r$ and $\lambda_c = \delta_0 \lambda_c$, respectively. Here, the QP dependent $\lambda_r$ and $\lambda_c$ trade-off factors are derived from (4.17) and (4.14), respectively.
4.4. Experimental Results and Discussion

(a) Intra frames, $QP = 20$

(b) Intra frames, $QP = 30$

(c) Intra frames, $QP = 40$
4.4. Experimental Results and Discussion

Figure 4.7: The distribution of $\Delta D$ and $\Delta R$ for different combinations of $\bar{\lambda}_r$ and $\bar{\lambda}_c$ value pairs for “kimono 1080p” sequence at three sample QP values. Each point represents the deviation of rate, and distortion of the proposed algorithm from that of the RD optimized mode selection. The “green” points represent the subset of operational points that satisfy the criteria $\Delta R \leq 1$, whereas the “red” highlighted point corresponds to the selected operational point minimum $\Delta D$. 
4.4 Experimental Results and Discussion

This section presents the performance of the proposed decoding complexity–rate–distortion model based video encoding algorithm, in terms of the decoding complexity, power consumption and the impact on the coding efficiency. First, the experimental results that correspond to the validation of the proposed decoding complexity-estimation models are described. The decoder’s complexity reductions and coding efficiency performances are investigated next, and are compared with respect to two state-of-the-art algorithms in the literature. Thereafter, experimental results for the power consumption are presented when decoding the bit streams using two popular CPU frequency governing methods as well as software and hardware decoders that simulate an off-line video playback scenario. Next, simulating an online video streaming scenario, the power consumed when decoding the bit streams generated by the proposed algorithm during a video streaming session is analyzed and presented. Finally, the experimental results and observations are discussed in detail for the different scenarios described above.

4.4.1 Simulation Environment

The proposed algorithm is implemented in the HM 16.0 reference software, where the complexity models described in Sec. 4.1 perform the decoding complexity estimations and the proposed Lagrangian cost function determines the coding modes for both inter- and intra-prediction. The resulting bit streams are decoded using the HM 16.0 [63] and openHEVC [113] software decoders on a x86 Intel Core i7-6500U system running Ubuntu 16.04, and the inbuilt hardware decoder of the system’s Intel HD520 GPU. The algorithm’s performance is compared with two state-of-the-art approaches; a tunable HEVC decoder proposed by Nogues et al. [93] and the power-aware encoding algorithm proposed by He et al. [27]. The CIF and HD video sequences used in the experiments are encoded using random access configuration with QPs 22, 27, 32 and 37, as defined in the common test configuration [16]. The complexities of the decoding processes are measured using the commonly used instruction level analysis tools Callgrind/Valgrind [22]. Finally, the decoder’s energy consumption is determined by measuring the energy dissipated by the system during playback. In this context, the first test setup corresponds to an offline content viewing use case where the bit streams encoded by the respective algorithms are decoded and displayed on screen while operating on the device’s battery capacity. The overall energy consumed for the process is measured and presented to illustrate the significance of the proposed algorithm for similar applications. Next, the bit streams encoded using the proposed algorithms are streamed over a wireless network to the client device using a streaming server and are decoded using openHEVC software decoder [113] to investigate the impact of the proposed algorithms when utilized on a video streaming application. For additional context, the energy consumption of the bit streams generated by the proposed method (using the ondemand Linux frequency governor) is also compared to another state-of-the-art energy-efficiency enhancement approach, DVFS [31].
4.4.2 Evaluation Metrics

Verification of the proposed decoder estimation model is performed using a set of validation sequences which are independent from the training sequences. The decoding complexity (cumulative number of estimated CPU cycles) which is estimated during the encoding process for the test sequences are thereafter compared with the actual CPU cycles measured from the Valgrind/Callgrind decoder profiling tools. Thus, the average prediction error is given by

\[ P_e = 100 \times \frac{|C_a - C_p|}{C_a}, \]

where \(C_a\) and \(C_p\) are the actual [22] and predicted CPU cycles from the proposed decoding complexity estimation model, respectively.

The performance of the proposed and state-of-the-art algorithms are then evaluated by measuring the complexity reduction achieved by the different bit streams at the decoder. To this end, the percentage decoding complexity reduction achieved for the same quality given by BD-C is evaluated by utilizing the BD-Rate calculation specified in [114], and by considering the area under the decoder-complexity, distortion curve [115][116]. In this case, the decoding complexity of each algorithm is measured using the Valgrind/Callgrind complexity assessment tools [22]. Similarly, device’s percentage energy consumption reduction for the same quality given by BD-E is evaluated by utilizing the energy dissipated when decoding the bit streams generated by the HM reference encoder and any other algorithm, and the PSNR as the quality metric. Finally, the impact on the coding efficiency is measured in BD-BR [114] which illustrates the impact on the bit rate for a given video quality level.

4.4.3 Performance Evaluation and Analysis

As described in Sec. 2.4, mainly, two approaches have been applied to reduce the energy consumption of video decoders in the literature. These are indirect decoding complexity reduction techniques (which generally either dynamically alter the decoding process or generate energy efficient bit streams), and direct hardware-level voltage-frequency scaling approaches. This section first discusses validity of the proposed decoding complexity estimation models followed up by the complexity reduction performance of the proposed method with respect to the state-of-the-art decoding complexity reduction techniques, and thereafter investigates its potential energy savings with respect to the voltage-frequency scaling approaches thereafter.

Verification of the Decoding Complexity Estimation Model

The decoding complexity models integrated within the HM 16.0 encoder provides the estimated decoding complexity of an encoded bit stream in CPU cycles which are compared against the actual CPU cycles consumed when decoding the bit streams using HM 16.0 decoder. The percentage error between the two is reported in the Table.
4.4. Experimental Results and Discussion

Table 4.3: Decoding complexity estimation performance of the proposed model

<table>
<thead>
<tr>
<th></th>
<th>Intra frames</th>
<th>Inter frames</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training Set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bridge-far</td>
<td>1.47%</td>
<td>4.58%</td>
</tr>
<tr>
<td>Waterfall</td>
<td>0.88%</td>
<td>3.60%</td>
</tr>
<tr>
<td>Band HD</td>
<td>1.85%</td>
<td>4.33%</td>
</tr>
<tr>
<td>Kimono HD</td>
<td>0.67%</td>
<td>4.25%</td>
</tr>
<tr>
<td>GT Fly</td>
<td>0.94%</td>
<td>2.94%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>1.16%</strong></td>
<td><strong>3.94%</strong></td>
</tr>
<tr>
<td><strong>Validation Set</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coastguard</td>
<td>0.59%</td>
<td>2.43%</td>
</tr>
<tr>
<td>Container</td>
<td>0.56%</td>
<td>1.36%</td>
</tr>
<tr>
<td>Beergarden HD</td>
<td>0.32%</td>
<td>4.55%</td>
</tr>
<tr>
<td>Dancer HD</td>
<td>0.44%</td>
<td>4.32%</td>
</tr>
<tr>
<td>Cafe HD</td>
<td>0.84%</td>
<td>5.88%</td>
</tr>
<tr>
<td>Musicians HD</td>
<td>0.10%</td>
<td>8.47%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.47%</strong></td>
<td><strong>4.50%</strong></td>
</tr>
</tbody>
</table>

4.3 for a set of test sequences. It can be observed that the prediction error for Intra-predicted frame is 0.815 % on average whereas the prediction error is 4.22% for the inter-predicted frames. It becomes evident that despite the variations of the prediction error observed (which is mainly due to the dynamics of the video contents), the overall prediction error for the decoding complexity estimation within the encoder is < 5%. This makes the proposed model an accurate estimator of the decoding complexity, which can be used for analyzing and defining decoding complexity–rate–distortion models to be used for decoding complexity-aware video encoding solutions. Thus, the level of details considered within these models and the accuracy of the prediction lays the foundation for the superior performance of the proposed encoding algorithm over the state-of-the-art.

Comparison with Dynamic Decoding Implementations

Modifying the decoding operations at run-time is often seen as a flexible approach to reduce the decoder’s complexity, and by extension its energy consumption. Modification of the motion compensation filters in the decoder (MC) and the intermittent skipping of the loop filter (LF), proposed by Nogues et al. [93], contributes significantly to reduce the complexity of the decoding operations. This is evident from the BD-C results in Table 4.4 as well as the decoding complexity, distortion curves presented in Fig. 4.8 for a subset of sequences. However, this impacts visual quality considerably due to the
Table 4.4: Decoding complexity reduction performance of in the Random Access configuration

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed (Model only)</th>
<th>Proposed (Model + LF [93])</th>
<th>He et al. [27] (PUM + DBLK)</th>
<th>Nogues et al. [93] (MC+LF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BD-C† (%)</td>
<td>BD-C‡ (%)</td>
<td>BD-BR (%)</td>
<td>BD-C† (%)</td>
</tr>
<tr>
<td>Akiyo</td>
<td>-27.3</td>
<td>-9.8</td>
<td>5.5</td>
<td>-38.0</td>
</tr>
<tr>
<td>Waterfall</td>
<td>-19.6</td>
<td>-12.6</td>
<td>5.6</td>
<td>-25.1</td>
</tr>
<tr>
<td>Container</td>
<td>-21.7</td>
<td>-8.7</td>
<td>2.7</td>
<td>-29.0</td>
</tr>
<tr>
<td>Coastguard</td>
<td>-17.2</td>
<td>-9.9</td>
<td>4.5</td>
<td>-23.3</td>
</tr>
<tr>
<td>Band</td>
<td>-34.0</td>
<td>-12.3</td>
<td>7.7</td>
<td>-43.3</td>
</tr>
<tr>
<td>Beergarden</td>
<td>-25.8</td>
<td>-11.4</td>
<td>2.5</td>
<td>-35.7</td>
</tr>
<tr>
<td>Cafe</td>
<td>-36.6</td>
<td>-12.2</td>
<td>5.8</td>
<td>-46.6</td>
</tr>
<tr>
<td>Dancer</td>
<td>-34.9</td>
<td>-17.9</td>
<td>9.7</td>
<td>-43.7</td>
</tr>
<tr>
<td>GTFly</td>
<td>-40.0</td>
<td>-18.5</td>
<td>9.1</td>
<td>-51.1</td>
</tr>
<tr>
<td>Kimono</td>
<td>-38.6</td>
<td>-20.3</td>
<td>6.7</td>
<td>-45.4</td>
</tr>
<tr>
<td>Musicians</td>
<td>-34.5</td>
<td>-16.5</td>
<td>9.9</td>
<td>-44.1</td>
</tr>
<tr>
<td>Parkscene</td>
<td>-31.4</td>
<td>-17.3</td>
<td>7.3</td>
<td>-40.0</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>-32.6</td>
<td>-12.4</td>
<td>2.7</td>
<td>-33.7</td>
</tr>
<tr>
<td>BasketDrill</td>
<td>-26.7</td>
<td>-10.7</td>
<td>9.1</td>
<td>-37.1</td>
</tr>
<tr>
<td>BasketPass</td>
<td>-20.6</td>
<td>-7.9</td>
<td>8.3</td>
<td>-31.4</td>
</tr>
<tr>
<td>Average</td>
<td><strong>-29.43</strong></td>
<td><strong>-13.22</strong></td>
<td><strong>6.47</strong></td>
<td><strong>-37.83</strong></td>
</tr>
</tbody>
</table>

† BD-C (%) achieved using the HM 16.0 reference decoder.
‡ BD-C (%) achieved using the openHEVC decoder.
Table 4.5: Energy consumption† behaviour of the encoding algorithms using the openHEVC software decoder.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
</tr>
<tr>
<td>Band</td>
<td>-9.2</td>
<td>-10.3</td>
<td>-15.4</td>
<td>-13.0</td>
<td>-22.0</td>
<td>-17.3</td>
<td>-4.7</td>
</tr>
<tr>
<td>Beergarden</td>
<td>-5.1</td>
<td>-12.5</td>
<td>-15.5</td>
<td>-8.5</td>
<td>-22.6</td>
<td>-18.7</td>
<td>-3.7</td>
</tr>
<tr>
<td>Cafe</td>
<td>-4.5</td>
<td>-5.9</td>
<td>-12.8</td>
<td>-8.0</td>
<td>-17.0</td>
<td>-14.0</td>
<td>-1.6</td>
</tr>
<tr>
<td>Dancer</td>
<td>-3.5</td>
<td>-9.7</td>
<td>-13.5</td>
<td>-8.5</td>
<td>-21.0</td>
<td>-16.0</td>
<td>-1.7</td>
</tr>
<tr>
<td>GT Fly</td>
<td>-2.7</td>
<td>-8.4</td>
<td>-11.6</td>
<td>-8.0</td>
<td>-19.1</td>
<td>-12.6</td>
<td>-1.3</td>
</tr>
<tr>
<td>Kimono</td>
<td>-5.3</td>
<td>-7.1</td>
<td>-13.8</td>
<td>-7.0</td>
<td>-21.2</td>
<td>-16.7</td>
<td>-1.9</td>
</tr>
<tr>
<td>Musicians</td>
<td>-3.8</td>
<td>-8.8</td>
<td>-16.0</td>
<td>-7.8</td>
<td>-20.7</td>
<td>-14.5</td>
<td>-2.1</td>
</tr>
<tr>
<td>Park scene</td>
<td>-7.1</td>
<td>-10.4</td>
<td>-17.8</td>
<td>-14.1</td>
<td>-22.3</td>
<td>-18.4</td>
<td>-4.0</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>-2.3</td>
<td>-8.7</td>
<td>-10.7</td>
<td>-7.7</td>
<td>-18.2</td>
<td>-10.3</td>
<td>-4.0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-4.83</strong></td>
<td><strong>-9.08</strong></td>
<td><strong>-14.12</strong></td>
<td><strong>-9.17</strong></td>
<td><strong>-20.45</strong></td>
<td><strong>-15.38</strong></td>
<td><strong>-2.77</strong></td>
</tr>
</tbody>
</table>

† BD-E is expressed with respect to the energy consumed to decode the HM 16.0 reference encoder’s bit stream using the Linux ondemand governor.
distortions introduced by the modified motion compensation filtering operations. For example, the algorithm in [93] reduces the luma and chroma filter sizes from 7-tap to 3-tap and 4-tap to 1-tap, respectively. This results in a different predicted image to that used by the encoder to calculate the motion vectors for the PU. Hence, this partially filtered predicted PU now gets compensated with a somewhat incorrect residual, which in turn distorts the reconstructed PU. Furthermore, the propagation of these error to future frames further impacts the visual quality of the reconstructed video as a whole. However, the intra-refresh in the random access configuration marginally limits the impact of error propagation, yet, the distortions incurred in an increased BD-BR %. A similar trend is visible in the RD curves presented in Fig. 4.8, which demands an increased bit rate during encoding to achieve a similar quality level to that of HM encoded bit streams when using a modified decoder such as [93] to decode the HEVC bit streams.

Moreover, intuition suggests that the impact on quality would be content dependent when the decoding operations are altered in this fashion, especially since distortions would only be significant in the complex video sequences with high motion and textured content (e.g., “musicians” and “coastguard” vs. “container”, and “poznan street”). This can be observed in Table 4.4, but the proposed method by contrast, shows a negligible change in BD-BR compared to the method proposed by Nogues et al. in [93]. This is due to the proposed algorithm operating on the encoder-side which determines the type of the motion vector (integer-pel vs. fractional-pel) based on the optimization cost function in (4.18); thus, requiring no changes to the decoding process itself.

Skipping the loop filter (LF) on the other hand, as in [93], reduces the decoding complexity with minimal impact on quality, and can also be implemented when decoding the proposed bit stream. For example, the experimental results presented in the Table 4.4 illustrate the BD-C improvements that can be achieved for the proposed algorithm in this manner. Here, the de-blocking and the Sample Adaptive Offset (SAO) filters are skipped by the decoder based on the activation level specified. In this case, performance of the proposed method can be improved, albeit for an additional BD-BR increase of 6.64% for the random access configuration. Overall, this suggests that the complexity reductions achieved using the proposed method can be further improved by optimizing the loop filtering of the decoder; a process that can easily be integrated into a decoder.

Comparison with Power-Aware Encoding Algorithms

The encoding algorithm proposed by He et al. [27] attempts to reduce the complexity of the filtering operations during motion compensation and the de-blocking performed by the decoder. In this context, the energy optimized motion vector selection algorithm (PUM) and the de-blocking filter disabling algorithm (DBLK) produce a bit stream which demonstrates a moderate complexity reduction as seen in the Table 4.4 as well as decoding complexity, distortion curves in Fig. 4.8. In comparison to [93], a much higher BD-BR loss is observed, especially for the sequences with high motion and

\[^3\text{It should be noted that the results presented in the Table 4.4 for [27] correspond to the lowest complexity level achievable by the algorithm}\]
complex texture properties. Even though the motion vector and PU mode decisions are taken at the encoder side, the selection of the trade-off factors which doesn’t consider the impact of both rate and distortion significantly impact the coding efficiency. Hence, despite the 12% and 7% BD-C reduction achieved by the algorithm, it’s applicability for video playback applications hinders due to the increased rate increase required to achieve a video quality similar to that of HM encoded streams. Furthermore, the lack of a detailed decoding complexity model coupled with the QP agnostic trade-off factor selection in [27] results in the poor BD-C reduction and the increased coding efficiency loss. Moreover, the encoding algorithm proposed by He et al. [27] requires the communication of de-blocking filter decisions to the decoder, thus requires an additional overhead as either multiple Picture Parameter Set (PPS) Network Abstraction Layer (NAL) units or metadata must be exchanged between the encoder and decoder. More importantly, the decoding complexity-efficient mode selection is only limited to the PU level motion vectors, thus, the BD-C (%) reduction achieved is relatively small when compared to the proposed algorithm and [93].

In contrast, the proposed algorithm demonstrates considerable improvements in decoding complexity reduction with minimal impact on the BD-BR due to its more comprehensive assessment and QP dependent selection of trade-off factors for both the bit rate and the decoding complexity. This limits the BD-BR increase to 6.47% on average compared to state-of-the-art methods and delivers BD-C reductions of 29.43% and 13.22 % for HM and openHEVC decoders, respectively. These results together with the RD and DC curves illustrated in Fig. 4.8 reveal that the bit streams generated by the proposed algorithm can achieve a non-trivial decoding complexity reduction for a similar video quality compared to the other decoding complexity reduction methods. This is aided by the use of a more detailed and accurate decoding complexity estimation model that is based on the HEVC coding features, which yields more accurate decoding complexity estimates for complexity-rate-distortion optimization.
4.4. Experimental Results and Discussion

(a) PSNR vs. CPU cycles (HM decoder), beergarden 1080p

(b) PSNR vs. CPU cycles (openHEVC decoder), beergarden 1080p

(c) PSNR vs. Bit rate, beergarden 1080p
4.4. Experimental Results and Discussion

(d) PSNR vs. CPU cycles (HM decoder), kimono 1080p

(e) PSNR vs. CPU cycles (openHEVC decoder), kimono 1080p

(f) PSNR vs. Bit rate, kimono 1080p
4.4. Experimental Results and Discussion

Figure 4.8: The decoding complexity (CPU cycles)-distortion curves and rate-distortion curves for proposed algorithms.
4.4. Experimental Results and Discussion

Figure 4.9: The decoder energy consumption, distortion graphs for the proposed and state-of-the-art algorithms. Here, the energy consumed to decode the streams is presented as the total reduction in the device’s battery capacity for the duration of video playback.
4.4. Experimental Results and Discussion

Energy Consumption Behaviour: offline video playback

Next, the overall energy consumed when decoding the bit streams generated by the proposed method is investigated and compared to those of the HM 16.0 encoder. To this end, the device’s energy consumption is first analyzed for the optimized software decoder openHEVC [113], followed up by an investigation on impact of using a state-of-the-art DVFS algorithm, for video decoding applications in a resource constrained mobile device. In both scenarios the video bit streams that are stored within the test device are decoded in real-time using openHEVC decoder to be displayed on the screen for 15 minutes simulating an offline video playback use case in a mobile device. The energy consumption during the whole decoding and playback process is measured in terms of the reduction in battery capacity given by the Linux kernel information. The process is repeated three times to get the average energy consumed for each algorithm under test. The energy consumed for bit streams under each QP (22, 27, 32, 37) is recorded and used against quality metric used for the decoded stream (PSNR) to calculated BD-E (%), which represents the energy consumed for a given video quality. The observed reduction in energy under these conditions using the openHEVC software decoder with different DVFS schemes and bit streams is reported in the Table 4.5. Moreover, a graphical presentation on the energy reductions achieved for the same video quality is illustrated in the Fig. 4.9 as energy, distortion curves for the proposed as well as state-of-the-art algorithms. Here, when compared to the HM 16.0 generated bit streams, the proposed algorithm demonstrates on average 4.83% decoding energy reduction when using the Linux kernel’s ondemand DVFS governor. Thus, the effectiveness of the proposed method’s complexity reduction, in terms of the impact on energy consumption, is quite apparently beneficial.

Next, the impact of using a more sophisticated DVFS scheme is investigated. In this context, the dynamic frequency selection algorithm proposed by Raffin et al. [31] has been integrated to the openHEVC decoder. Here, the operating frequency of the processor is controlled based on the estimated complexity of the subsequent frame, which is assessed using the moving weighted average of the complexities of previously decoded frames. Therefore, the selection of the CPU frequency becomes application and content specific, i.e., in this context the decoder and the current bit stream. The energy consumption behaviour shown in the Table 4.5 for the bit streams illustrate how an application specific DVFS governor can indeed outperform a generic DVFS governor. However, as was the case before, the increased complexity of the HM 16.0 encoded bit streams, limits the potential energy savings that can be achieved. In fact, the complexity reduction by the proposed algorithm’s bit streams allow the DVFS algorithm to select much lower CPU operating frequencies that lead to greater energy efficiency. Hence, as in the case of the complexity reductions discussed previously, a similar behaviour can be observed for the proposed algorithm with an improvement of -5.08% BD-E reduction when compared to the HM 16.0 bit stream with DVFS [31]. The reduction in performance can be attributed to the greater scope for control available to the DVFS algorithm when the processes are more complex, however, the improved performance of the proposed method largely remains intact. Moreover, as illustrated in the Fig. 4.9 and Table 4.4, the BD-E reduction of the proposed algorithm when utilized with a decoder that skips the loop filtering process similar to the algorithm proposed
Table 4.6: Energy consumption\(^\dagger\) behaviour of the proposed algorithm.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>software decoder (video streaming)</th>
<th>software decoder (video streaming)</th>
<th>hardware decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
<td>BD-E (%)</td>
</tr>
<tr>
<td>Band</td>
<td>-9.2</td>
<td>-5.8</td>
<td>-1.6</td>
</tr>
<tr>
<td>Beergarden</td>
<td>-5.1</td>
<td>-2.1</td>
<td>-1.6</td>
</tr>
<tr>
<td>Cafe</td>
<td>-4.5</td>
<td>-1.8</td>
<td>-1.5</td>
</tr>
<tr>
<td>Dancer</td>
<td>-3.5</td>
<td>-5.0</td>
<td>-3.2</td>
</tr>
<tr>
<td>GT Fly</td>
<td>-2.7</td>
<td>-2.5</td>
<td>-1.2</td>
</tr>
<tr>
<td>Kimono</td>
<td>-5.3</td>
<td>-4.2</td>
<td>-1.3</td>
</tr>
<tr>
<td>Musicians</td>
<td>-3.8</td>
<td>-3.4</td>
<td>-1.1</td>
</tr>
<tr>
<td>Park scene</td>
<td>-7.1</td>
<td>-2.9</td>
<td>-1.9</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>-2.3</td>
<td>-4.9</td>
<td>-3.3</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-4.83</strong></td>
<td><strong>-3.62</strong></td>
<td><strong>-1.85</strong></td>
</tr>
</tbody>
</table>

\(^\dagger\)BD-E, reduction is expressed with respect to the energy consumed to decode the HM 16.0 reference encoder’s bit stream.

in [93], is on average turns out to be -20.45\%. This indicate that the video playback devices can reduce the energy consumption by approximately 20\% by decoding the bit streams generated by the proposed algorithm and by skipping the de-blocking filters. These experimental results suggest that the decoder energy reductions that can be achieved for a similar quality of the HM encoded bit streams is significantly high for the proposed algorithm compared to the state-of-the-art approaches, when considering software based HEVC decoder implementations.

Energy Consumption Behaviour: other scenarios

In this subsection, we discuss the behaviour of the proposed algorithms for two other common video play back scenarios. This involves an analysis on the decoding energy consumption behaviour of the bit streams generated by the proposed algorithm when decoded using the Intel HD520 hardware HEVC decoder [117] followed up by an evaluation of the decoder energy consumption in a video streaming use case involving the bit streams of the proposed algorithm with openHEVC software decoder [113].

First, the decoding energy consumption behaviour for the bit streams of the proposed algorithm is analyzed and compared against the bit streams of the HM 16.0 reference encoder, when they are decoded using the Intel HD520 hardware HEVC decoder [117]. The experimental results for the offline video playback use case involving the hardware decoder is presented in the Table 4.6. In this case, CPU and GPU clock frequencies can
be maintained at their minimums while the GPU is managed by the hardware itself. Crucially, the results for the scenario under concern demonstrate that the proposed algorithm outperforms HM 16.0 in terms of the decoder’s energy consumption by -1.85% for a similar video quality. The reduced complexity of the bit streams generated by the proposed algorithm increases the GPU’s idle time, which contributes to the BD-E reduction specified in the Table 4.6.\footnote{The remaining state-of-the-art algorithms require modifications to the decoder implementations which is not feasible under a hardware implementation. Hence, the experimental results for the Intel HD520 hardware HEVC decoder is presented only for the proposed algorithm.} That being said, despite the comparable energy consumption behaviour to the software decoding scenario (indicated in the first column of the Table 4.6), the absolute power consumption will still depend on the power management policy of the GPU driver, inter-process communications, etc. However, the efficient management of these resources are outside the scope of the this work, hence, the results presented here correspond to Intel’s default GPU power management settings in the Skylake processor architecture.

The proposed algorithm doesn’t consider any rate controlling aspects during its formulation process, thus, a BD-BR increase of 6% is witnessed as an impact on its coding efficiency. Therefore, the bit streams of the proposed algorithm are subjected to a video streaming session to further investigate the impact of the bit rate increase on the decoder’s energy consumption compared to the HM encoded bit streams. To this end, the bit streams generated by the proposed algorithm for QPs 22, 27, 32, and 37 are streamed from a streaming server over a wireless connection to the device under test and are decoded using the openHEVC software decoder. The total energy consumed for the process (that incorporate communication via wireless interface, video decoding and presentation) is measured using the Linux power measurement tools. A BD-E measurement for the proposed method is calculated using the total energy consumed under each QP compared to the bit streams encoded using HM encoder. For example, the experimental results presented in the Table 4.6 showcase a -3.62% BD-E decoder energy reduction for the proposed algorithm for a similar video quality to that of HM encoded streams. Therefore, it is evident that the reduced complexity of the HEVC bit streams in the proposed algorithm, outperform the slight increase in the bit rate when it comes to overall energy consumption of a resource constrained video decoder. In this context, the proposed encoding algorithm is shown to be beneficial in the content preparation for both offline and online video playback and streaming scenarios; a crucial improvement compared to the state-of-the-art when considering green multimedia content preparation and distribution.

\section{Summary}

This Chapter proposes an encoding algorithm that produces less complex HEVC video bit streams with minimal impact on the coding efficiency for a given QP. In this context, two CU level decoding complexity estimation models are first introduced to predict the decoding complexity of inter- and intra-predicted blocks at the encoder. The proposed models are capable of capturing the intricate decoding complexities for the assortment of HEVC coding features with respect to the HM16.0 decoding architecture.
and estimate the decoding complexity of a particular coding mode/feature during the encoding phase itself. Next, the relationship between decoding complexity and distortion is modelled which is extended to introduce the proposed encoding algorithm that consider decoding complexity, rate and distortion to prepare HEVC bit streams that consume less computational resources at the decoder, with minimal impact to the distortion and bit rate.

The experimental evaluations on the proposed decoding complexity estimation models reveal that the proposed models are capable of predicting the decoding complexity at the encoder with < 5% error for both inter- and intra-predicted frames. Moreover, the complexity measurement results reveal decoding complexity reductions (BD-C) of -29.43% and -13.22% with only 6.47% BDBR increase for a similar video quality to that of HM16.0 encoded bit streams for HM16.0 decoder and openHEVC decoder, respectively. In addition, energy consumption measurements reveal an average BD-E reduction of -4.83% for the optimized openHEVC software decoder. Furthermore, DVFS and modified loop filters at the openHEVC decoder enable the proposed algorithm to achieve upto -20.45% BD-E reductions for a similar video quality to that of HM16.0 bit streams. Finally, experimental results conducted with a hardware decoder reveal BD-E reductions of -1.85% compared to HM encoded bit streams. Similarly, a -3.62% average BD-E reduction is witnessed for software decoding during a video streaming session with the bit streams generated from the proposed algorithm.

Energy consumed during video playback is tightly correlated to the complexity of the codec as well as the content being decoded. Thus, adapting the video content to reduce the decoding complexity is seen as a scalable and flexible approach to reduce the energy consumption of the decoding CE devices. In this content, preparing a video bit stream which is less complex during the encoding phase is extremely challenging due to the complexities of the standards, and dynamics of the video content. In this context, the proposed complexity estimation models provide a mechanism to make the encoder aware of the relative decoding complexity levels for the assortment of HEVC coding features. The relationship and the trade-off factors defined between the decoding complexity, rate and distortion enables the encoder to select a particular coding mode by considering the impact of all three parameters. Therefore, in summary, it can be concluded that the proposed encoding algorithm facilitates decoding complexity-aware video bit streams that could potentially improve the energy efficiency of mobile video playback devices.
Chapter 5

Joint CTU Level Decoding Complexity, Rate Controlled Video Coding

Streaming video contents to resource constrained mobile hand-held devices over wireless networks has become evermore challenging due to the limited energy resources and the dynamic nature of the communication channels. In this case, adaptive streaming of video contents over HTTP is seen as an effective solution to cope with the dynamic fluctuations observed in the network bandwidth (ref. Chapter 2). For example, predominant video streaming services such as YouTube [118], Netflix [119] etc., are already supporting MPEG-DASH to provide adaptive streaming capability between the streaming servers and playback clients. As illustrated in Sec. 2.4, recent literature has turned towards adaptive streaming by extending its capabilities to reduce the decoding complexity and the associated energy consumption of the mobile CE devices. However, the lack of an encoding algorithm in the state-of-the-art which has the capability to produce video bit streams that adhere to a given bit rate and a decoding complexity level limits the full potential of the prevailing adaptive streaming solutions to maintain a maximum perceivable video quality while catering for both network bandwidth fluctuations and diminishing energy capacity of the video playback devices. In this context, this Chapter introduces a novel approach to generate HEVC video bit streams with multiple bit rate and decoding complexity levels at the encoder that could potentially improve the existing video streaming solutions to cater for both network bandwidth and device energy capacity fluctuations while minimizing the impact on perceived video quality.

Introducing an encoding algorithm that can simultaneously perform decoding complexity and rate controlling to achieve a given target bit rate and a decoding complexity level, requires an in-depth analysis of the relationship between decoding complexity, bit rate and distortion parameters for a given content. More specifically, a dynamic and content adaptive model which describes the relationship among the three parameters is crucial to realize the ultimate goal of producing a video bit stream that satisfy the given constraints. In this case, the first step in deriving such a relational model is to
5.1 Decoding complexity, Rate and Distortion Relationship

Capture the intricate decoding complexities exhibited by the numerous coding modes and features available in the HEVC standard. Thus, a detailed and an accurate decoding complexity model becomes an essential component. In this case, HEVC decoder analysis and profiling results presented in Chapter 4, and the HEVC decoding complexity estimation models proposed therein (for both inter- and intra-predicted CU), can be integrated within the HEVC encoding process. This facilitates the encoder to be aware of the relative decoding complexities of each coding mode and feature during the mode selection phase [107][101]. Next, a comprehensive analysis of the decoding complexity, bit rate and the distortion for a wide range of decoding complexity and bit rate levels for diverse video content types is required in order to derive a model that describes the relationship among the three parameters. Hence, the remainder of this Chapter first describes the experimental process followed to analyze and formulate the relationship among the decoding complexity, rate and distortion. Next, the proposed joint decoding complexity, rate controlled video encoding framework for HEVC is introduced followed up by the experimental results and discussions on the proposed and the related state-of-the-art algorithms.

5.1 Decoding complexity, Rate and Distortion Relationship

Encoding a video sequence for a given bit rate and a decoding complexity requirement entail a number of prerequisites from the encoding perspective. First, making the encoder aware of the estimated decoding complexities for a particular coding mode or feature becomes non-trivial for such an encoding algorithm. Further to that, a comprehensive analysis on the behaviour of decoding complexity, rate and distortion for various contents and encoding configurations is pertinent when determining the coding modes for a given content. As such, a content adaptive decoding-complexity–rate–distortion model is crucial to measure the impact of a particular coding mode on a video content for the three parameters involved. Therefore, this section describes the approach used in this work to analyze the relationship among the three parameters, the process of forming a content dependent decoding-complexity–rate–distortion model and its composition within the decoding-complexity, rate, distortion space.

5.1.1 Decoding-Complexity, Rate and Distortion Space

Selecting the best coding modes for a given content that maximizes the coding efficiency (i.e., minimizing the distortion between the original and reconstructed samples while minimizing the resultant bit rate) is crucial for any video encoder. In the rate control domain, the problem of finding the optimum set of coding decisions that minimizes $D$ subjected to a maximum rate constraint enforced on the content $R_T$ is given by,

$$\min \left\{ D(p) \right\} \quad \text{s.t.} \quad R(p) \leq R_T, \quad p \in \mathcal{P}. \quad \text{(5.1)}$$

In this case, $p$ is a set of coding parameter combination from the set of all the possible coding options $\mathcal{P}$, and $D(p)$, $R(p)$ are the distortion and rate associated with the
selected set of coding parameters, respectively [111][9]. Typically, a HEVC compatible encoder utilizes a Lagrangian optimization approach using a RD cost function given by,

$$\min \left\{ D(p) + \lambda R(p) \right\}, \quad p \in \mathcal{P},$$

(5.2)
to determine the optimum coding modes and coding structure for a given content. Here, the $\lambda \geq 0$ is the Lagrange multiplier. However, each coding parameter set $p$ results in a decoding complexity different to another for a given content [107] [120] [115]. Yet, the decoding complexity cost of a coding mode or structure is not accurately captured by the cost function in (5.2). Therefore, it is imperative that the decoding complexity should be integrated to the mode selection process within the encoding chain to realize the proposed algorithms that control both rate as well as decoding complexity.

In this context, the decoding complexity, rate and distortion analysis requires a mode selection cost function that considers both rate as well as decoding complexity as constraints when minimizing the distortion. Thus, the problem of finding the coding modes that minimize $D$ with rate constraint $R_T$ and decoding complexity constraint $C_T$, is defined by,

$$\min \left\{ D(p) \right\} \quad \text{s.t.} \quad R(p) \leq R_T, \quad C(p) \leq C_T \quad \big| \quad p \in \mathcal{P}. \quad (5.3)$$

In this case, using the Lagrangian optimization principles, the joint mode selection cost function utilized in this work is given by,

$$\min_{p \in \mathcal{P}} J_{{\text{CRD}}} \mid J_{{\text{CRD}}} \triangleq D(p) + \lambda_r R(p) + \lambda_c C(p),$$

(5.4)

where $\lambda_r$ and $\lambda_c$ are the bit rate and decoding complexity trade-off parameters, respectively. In this case, the range of $\lambda_r$ and $\lambda_c$ defines the decoding-complexity–rate–distortion space spanned by the coding parameter combinations in $\mathcal{P}$. Thus, determining the $\lambda_r$ and $\lambda_c$ trade-off parameters becomes crucial when selecting the best coding modes for a given content. Therefore, the relationship among the decoding complexity, rate, distortion parameters is analyzed next to derive a model of these parameters for use in a joint decoding-complexity–rate control algorithm.

5.1.2 The Decoding-Complexity, Rate and Distortion Behaviour

In order to determine the decoding-complexity, rate and distortion behaviours, the parameter space created by (5.4) must first be determined. To this end, an experimental sweep of the space created by $\lambda_r \in [0, \infty)$ and $\lambda_c \in [0, \infty)$ was performed on six different test sequences (with representative and varying spatial and temporal characteristics) for which empirical data was collected from both inter- and intra-predicted frames and QPs ranging from 0 – 51. The resulting decoding-complexity$^1$, rate and distortion can thereafter be expressed for further analysis in terms of cycles per pixel (cpp), bits per pixel (bpp), and the mean squared error (MSE), respectively, as follows.

In this case, MSE is calculated as,

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^{N} \left( y_n - \hat{y}_n \right)^2,$$
5.2. Joint Decoding Complexity and Rate Control

\[ \text{MSE}(p, \lambda_r, \lambda_c, q) = \frac{1}{N} \sum_{i=1}^{N} \{s_i - s'_i(p, \lambda_r, \lambda_c, q)\}^2, \quad (5.5) \]

where \( s_i \) and \( s'_i \) correspond to the \( i^{th} \) original and reconstructed pixel, respectively, \( q \) is the QP, and \( N \) is the number of pixels. Similarly, the bpp and cpp are defined as,

\[ \text{bpp}(p, \lambda_r, \lambda_c, q) = \frac{R(p, \lambda_r, \lambda_c, q)}{W \times H}, \quad (5.6) \]

and

\[ \text{cpp}(p, \lambda_r, \lambda_c, q) = \frac{C(p, \lambda_r, \lambda_c, q)}{W \times H}, \quad (5.7) \]

where \( W \) and \( H \) correspond to the frame width and frame height, respectively. Further, \( R \) and \( C \), represent the total number of bits required to encode the frame and the estimated decoding-complexity of that frame once encoded.

The Figs. 5.1 and 5.2 graphically illustrate behaviour of the decoding-complexity, rate and distortion in the parameter space spanned by \( p, \lambda_r \) and \( \lambda_c \) for the ‘kimono 1080p’ sequence\(^2\), including the discrete operating points that can be achieved by the encoder (i.e., each data point in Figs. 5.1 and 5.2 represents a particular combination of \( \lambda_r \) and \( \lambda_c \), for both inter- and intra-predicted frames, respectively.). It is observed that this general behaviour can be modelled in a content-dependent manner using a 2-dimensional \( n^{th} \)-power model given by

\[ \text{MSE}(p, \lambda_r, \lambda_c, q) = a(q) \times \text{bpp}^{b(q)}(p, \lambda_r, \lambda_c, q) + c(q) \times \text{cpp}^{d(q)}(p, \lambda_r, \lambda_c, q), \quad (5.8) \]

where \( a(q), b(q), c(q), \) and \( d(q) \) are QP and content-dependent model parameters.

The relationship among the three parameters defined in (5.8) is crucial to determine the impact on each parameter for a coding mode and feature combination on a particular content. However, the model parameters in (5.8) are content dependent, which naturally implies that a content-adaptive approach is necessary to compute the appropriate model parameters dynamically, in order to determine the optimum coding structure \( p \) for a particular content. To this end, this work proposes an initial usage of a generic set of model parameters which will be dynamically adapted during the encoding loop (as described in Sec. 5.2.3), which leads to the novel joint decoding-complexity–rate control algorithm proposed which is described next.

5.2 Joint Decoding Complexity and Rate Control

As with any rate control algorithm, the joint control of both decoding-complexity and rate also requires target decoding-complexities and bit rates to be defined. In this context, this section first describes how these can be allocated at the CTU-level. This is followed by the derivation of an appropriate QP and content-adaptive decoding-complexity and rate trade-off factors in (5.8) next, and finally by an update algorithm to dynamically adapt the model parameters in (5.8).

\(^2\)The behaviours of decoding-complexity, rate and distortion remain similar across QPs and sequences, albeit with different model parameters.
5.2. Joint Decoding Complexity and Rate Control

The state-of-the-art rate control algorithms adopted in HM reference encoder [111][121] first perform a Group of Picture (GOP) level and frame level rate allocations prior to the CTU level bit distribution. Adopting a similar approach, the proposed algorithm first performs a GOP level and frame-level bit and decoding complexity targets which can be later used to derive their CTU-level allocation. In this case, the target number

Figure 5.1: The relationship among the decoding complexity ($cpp$), rate ($bpp$) and distortion (MSE) parameters for inter-predicted frames for QPs 25, 30 and 35.

5.2.1 CTU level Rate and Decoding Complexity Allocation

The state-of-the-art rate control algorithms adopted in HM reference encoder [111][121] first perform a Group of Picture (GOP) level and frame level rate allocations prior to the CTU level bit distribution. Adopting a similar approach, the proposed algorithm first performs a GOP level and frame-level bit and decoding complexity targets which can be later used to derive their CTU-level allocation. In this case, the target number
5.2. Joint Decoding Complexity and Rate Control

(a) Intra frame, QP = 25

(b) Intra frame, QP = 30

(c) Intra frame, QP = 35

Figure 5.2: The relationship among the decoding complexity (cpp), rate (bpp) and distortion (MSE) parameters for intra-predicted frames for QPs 25, 30 and 35.

of bits for the GOP can be expressed as

\[ R_{\text{GOP}} = \psi \left\{ \frac{B_l - B_a (M_l - W)}{W} \right\} , \quad (5.9) \]

where \( \psi, B_l, B_a, M_l, W \) are the GOP size, bits remaining, average bits per frame, frames remaining, and window size, respectively [121]. The, average bits per frame \( (B_a) \) in this case is determined using,

\[ B_a = \frac{B_T}{N}, \quad (5.10) \]
5.2. Joint Decoding Complexity and Rate Control

where $B_T$ is the total bits available for the sequence and $N$ is the total number of frames in the sequence to be encoded. Similarly, the target decoding-complexity for the GOP becomes

$$C_{T}^{\text{GOP}} = \psi \left\{ \frac{C_t - C_a(M_t - W)}{W} \right\}, \tag{5.11}$$

where $C_t$ and $C_a$, are the total decoding-complexity budget left over and the average complexity per frame which is calculated as,

$$C_a = \frac{C_T}{N}. \tag{5.12}$$

Here, $C_T$ represents the total decoding-complexity assigned to the sequence\(^3\). In both cases, the window size is maintained at $W = 40$, which is the default configuration in the HM 16.0 [63] encoder implementation.

Next, the GOP-level bit rate and decoding-complexity allocation in (5.9) and (5.11) must be distributed within the GOP at the frame-level. A similar approach is adopted for both quantities leading to a weighted distribution of the GOP-level allocation shown in (5.13). For the $j^{th}$ frame in the GOP, the bit rate and decoding-complexity allocation is given by,

$$X_{T}^{\text{Frame}}(j) = X_{T}^{\text{GOP}} \times \frac{\omega_{X}^{\text{Frame}}(j)}{\sum_{j=1}^{\psi} \omega_{X}^{\text{Frame}}(j)}, \tag{5.13}$$

where $X \in \{R, C\}$ and the weights are defined as

$$\omega_{X}^{\text{Frame}}(j) = \rho(j) + \eta_X(j). \tag{5.14}$$

$\rho(j)$ is set to the default weighting factors defined in HM16.0 [63, 121], whereas $\eta_X(j)$ is experimentally determined and assigned

$$j = \text{Intra-frame} : \eta_R(j) = 50, \eta_C(j) = 80,$$
$$j = \text{Inter-frame} : \eta_R(j) = 3, \eta_C(j) = 10.$$

Finally, bit rates and decoding-complexity targets are allocated to the individual CTUs based on the MSE of the previous co-located CTU, as is often done in traditional rate control [122, 123]. The decoding-complexity–rate–distortion model in (5.8) is used to these ends, and as the model parameters therein are functions of QP and content, the MSE of the $k^{th}$ CTU in the $j^{th}$ frame of the GOP is first predicted for each QP $q$ which can be expressed as,

$$\hat{\text{MSE}}_{k,j}^{\text{CTU}}(q) = a_{k,j}(q) \times \text{bpp}_{\text{avg}} + b_{k,j}(q) \times \text{cpp}_{\text{avg}} + c_{k,j}(q) \times \text{cpp}_{\text{avg}}^d_{k,j}(q), \tag{5.15}$$

where $a_{k,j}(q)$, $b_{k,j}(q)$, $c_{k,j}(q)$, and $d_{k,j}(q)$ are the appropriate model parameters for that CTU. Here, bpp$_{\text{avg}}$ and cpp$_{\text{avg}}$ are determined to be the average bpp and cpp observed for the QP $q$ from the CTUs of the preceding frames\(^4\). Next, the MSE of the co-located

\(^3\)The available decoding-complexity budget in this case is defined as the total number of CPU cycles allocated to decode a particular sequence.

\(^4\)It should be noted that the algorithm maintains actual bpp and cpp values observed for each QP $q$ after the encoding of a CTU and are utilized in (5.19) when estimating $\hat{\text{MSE}}_{k,j}^{\text{CTU}}(q)$. 

5.2. Joint Decoding Complexity and Rate Control

CTU, $\text{MSE}_{k,j,\text{co}}$, is then compared with $\hat{\text{MSE}}_{k,j}^{\text{CTU}}$ for all QPs to obtain a QP $q_0$ such that

$$\min_{q_0} \left| \text{MSE}_{k,j}^{\text{CTU}}(q) - \text{MSE}_{k,j,\text{co}}^{\text{CTU}} \right|. \tag{5.16}$$

Thus, the bpp$_{\text{avg}}$ and cpp$_{\text{avg}}$ corresponding to the QP $q_0$ are then used as the weights for both the bit rate and decoding-complexity to the CTU. Thus, the final target bit rate and decoding-complexity of the $k^{th}$ CTU in the $j^{th}$ frame in the GOP can be expressed as

$$X_T^{\text{CTU}}(k,j) = \frac{\omega_X^{\text{CTU}}(k,j)}{\sum_{\phi=k}^{\Phi} \omega_X^{\text{CTU}}(\phi,j)} \times \Delta(k,j), \tag{5.17}$$

where $\Delta(k,j) = \left\{ X_{\text{Frame}}^{\text{Frame}}(j) - \sum_{\phi=1}^{k-1} X_T^{\text{CTU}}(\phi,j) \right\}$ is the remaining bits or decoding-complexity available to the remaining CTUs in the frame and $\Phi$ is the total number of CTUs in the frame. Note that these bit rates and decoding-complexities can be expressed in terms of bpp or cpp by simply dividing $X_T^{\text{CTU}}$ by the number of pixels in the CTU.

Once the respective allocations are made, the selection of appropriate CTU coding parameters is crucial to meet the given targets. Therefore, the process followed in this work to select the QP and rate and decoding complexity trade-off factors is described next.

5.2.2 Determining QP, $\bar{\lambda}_r$, and $\bar{\lambda}_c$

The decoding complexity–rate–distortion model explained in the Sec. 5.1.2 constitutes a set of content dependent model parameters for each QP (ref. eq. (5.8)). Hence, the coding parameter selection phase of the proposed algorithm, first determines the appropriate QP which is thereafter utilized to determine $\bar{\lambda}_r$, and $\bar{\lambda}_c$ using the QP specific model parameters.

Determining QP

Once the CTU level decoding complexity and bit allocations are made, the QP selection is first performed using a similar Mean Square Error (MSE) based approach. For example, the MSE of the co-located CTU, $\text{MSE}_{k,j,\text{co}}^{\text{CTU}}$, is now compared with $\hat{\text{MSE}}_{k,j}^{\text{CTU}}$ for all QPs to obtain a QP $q_0$ such that

$$\min_{q_0} \left| \hat{\text{MSE}}_{k,j}^{\text{CTU}}(q) - \text{MSE}_{k,j,\text{co}}^{\text{CTU}} \right|. \tag{5.18}$$

In this case, the estimated MSE of the $k^{th}$ CTU in the $j^{th}$ frame of the GOP is predicted for each QP $q$ using,

$$\hat{\text{MSE}}_{k,j}^{\text{CTU}}(q) = a_{k,j}(q) \times \left\{ \frac{R_T^{\text{CTU}}(k,j)}{N} \right\}^{b_{k,j}(q)} + c_{k,j}(q) \times \left\{ \frac{C_T^{\text{CTU}}(k,j)}{N} \right\}^{d_{k,j}(q)}, \tag{5.19}$$

where $N$ is the number of pixels and $R_T^{\text{CTU}}$ and $C_T^{\text{CTU}}$ are the number of bits, and decoding complexity level allocated for the CTU, respectively.
Determine $\bar{\lambda}_r$, and $\bar{\lambda}_c$

Once the QP $\bar{q}_0$ is determined, the trade-off factors $\bar{\lambda}_r$, and $\bar{\lambda}_c$ are thereafter derived utilizing the partial derivatives of the decoding complexity–rate–distortion model (5.8) with respect to bpp and cpp, respectively. For example, in this case, from (5.8), for the $k^{th}$ CTU in the $j^{th}$ frame

$$\lambda_r(k, j, q) \triangleq \frac{\partial \text{MSE}}{\partial \text{bpp}} = a_{k,j}(q) b_{k,j}(q) \text{bpp}^{b_{k,j}(q)-1}$$

and

$$\lambda_c(k, j, q) \triangleq \frac{\partial \text{MSE}}{\partial \text{cpp}} = c_{k,j}(q) d_{k,j}(q) \text{cpp}^{d_{k,j}(q)-1}$$

Equations (5.20) and (5.21) imply that the CTU-level model parameters, together with the CTU-level bit rate and decoding-complexity allocations described in the previous subsection, completely define the optimization function in (5.4) needed to determine the optimum coding structure. Thus, from (5.17)–(5.21) for the $k^{th}$ CTU in the $j^{th}$ frame the bit rate and decoding-complexity trade-off parameters can be expressed as

$$\lambda_r(k, j, \bar{q}_0) = \alpha_{k,j}(\bar{q}_0) \times \left( \frac{R_{CTU}^T(k, j)}{N} \right)^{\beta_{k,j}(\bar{q}_0)}$$

and

$$\lambda_c(k, j, \bar{q}_0) = \rho_{k,j}(\bar{q}_0) \times \left( \frac{C_{CTU}^T(k, j)}{N} \right)^{\tau_{k,j}(\bar{q}_0)}$$

respectively, where $N$ is the number of pixels in the CTU.

It now becomes apparent that the two trade-off parameters are both content and QP dependent via the four modelling parameters in (5.8), (5.20) and (5.21). However, a content-independent generic set of parameters can also be obtained (to be used as initialization values in the adaptive model parameter computation process described in the following subsection Sec. 5.2.3) from the data collected in Sec. 5.1.2. These can be expressed as

$$\begin{bmatrix}
\alpha(q) \\
\beta(q) \\
\rho(q) \\
\tau(q)_{\text{Intra}}
\end{bmatrix}_{\text{Intra}} =
\begin{bmatrix}
6.8 \times 10^{10} \times q^{-8.745} \\
0.0671 \times q - 7.375 \\
2.28 \times 10^{-6} \times q^{7.188} \\
-4.24 \times 10^{-6} \times q^{3.51} - 1.275
\end{bmatrix}$$

and

$$\begin{bmatrix}
\alpha(q) \\
\beta(q) \\
\rho(q) \\
\tau(q)_{\text{Inter}}
\end{bmatrix} =
\begin{bmatrix}
0.000721 \times q^{2.516} \\
3.89 \times 10^{-5} \times q^{2.48} - 1.707 \\
-39.38 \times q^{4.473} + 1.76 \times 10^9 \\
-0.02157 \times q - 3.684
\end{bmatrix}$$
for the two frame types. Naturally, the bit rate and decoding-complexity achieved using (5.24) and (5.25) will be inaccurate and not adaptive to the content. Hence, a mechanism for the dynamic updating of the model parameters is necessary for joint decoding-complexity and rate controlled encoding.

### 5.2.3 Dynamic Model Parameter Adaptation

In order to derive content-dependent model parameters, the generic parameter set in (5.24) and (5.25) can be adapted using a Least Mean Square (LMS) [124] based approach. To this end, the error between the assigned and achieved bit rate and decoding-complexity must be minimized, and to do so in this case, a joint error function for the two quantities must first be defined. Note that the following derivations will omit the \(k, j,\) and \(q_0\) subscripts for notational simplicity, but the adaptation process must be applied independently to each CTU to compute their unique model parameters.

Now, let the difference between the assigned and achieved bit rate and decoding-complexity per pixel be \(\Delta R\) and \(\Delta C\), respectively. Similarly let the difference between the predicted distortion and actual distortion in terms of MSE be \(\Delta D\). Expressing the total derivative of distortion in terms of MSE as the sum of partial derivatives of the dependent variables in the model in (5.8) and the definitions in (5.20), (5.21),

\[
\frac{d(MSE)}{\partial \text{bpp}} = \frac{\partial \text{MSE}}{\partial \text{bpp}} d(\text{bpp}) + \frac{\partial \text{MSE}}{\partial \text{cpp}} d(\text{cpp}) \tag{5.26}
\]

\[
\Delta D = -\lambda_r \Delta R - \lambda_c \Delta C. \tag{5.27}
\]

Obtaining the squared term of (5.27) and rearranging the terms,

\[
\Delta D^2 - 2 \Delta R \Delta C \lambda_r \lambda_c = \lambda_r^2 \Delta R^2 + \lambda_c^2 \Delta C^2 > 0 \tag{5.28}
\]

and dividing both sides by \((\lambda_r \lambda_c)^2\)

\[
\left( \frac{\Delta D}{\lambda_r \lambda_c} \right)^2 - 2 \left( \frac{\Delta R \Delta C}{\lambda_r \lambda_c} \right) = \left( \frac{\Delta R}{\lambda_c} \right)^2 + \left( \frac{\Delta C}{\lambda_r} \right)^2. \tag{5.29}
\]

The right hand side of (5.29) can be simplified further as

\[
\Delta R^2 \left( \frac{\partial \text{cpp}}{\partial \text{MSE}} \right)^2 + \Delta C^2 \left( \frac{\partial \text{bpp}}{\partial \text{MSE}} \right)^2 \approx 2 \left( \frac{\Delta R \Delta C}{\Delta D} \right)^2. \tag{5.30}
\]

The objective of minimizing \(\Delta C\) and \(\Delta R\) simultaneously is now made possible by multiplying (5.30) and therefore (5.29) by \(\Delta D^2\). Hence, by combining (5.29) and (5.30), the joint error function to be minimized can be defined as

\[
\mathcal{F} := \left( \frac{\Delta D^2}{\lambda_r \lambda_c} \right)^2 - 2 \Delta D^2 \left( \frac{\Delta R \Delta C}{\lambda_r \lambda_c} \right). \tag{5.31}
\]
Thus, using (5.31) and a LMS adaptive filter, the updated model parameters can be expressed as

\[
\begin{bmatrix}
\alpha_n(q_0) \\
\beta_n(q_0) \\
\rho_n(q_0) \\
\tau_n(q_0)
\end{bmatrix} = \begin{bmatrix}
\alpha_{n-1}(q_0) - \vartheta_\alpha \frac{\partial F}{\partial \alpha} \\
\beta_{n-1}(q_0) - \vartheta_\beta \frac{\partial F}{\partial \beta} \\
\rho_{n-1}(q_0) - \vartheta_\rho \frac{\partial F}{\partial \rho} \\
\tau_{n-1}(q_0) - \vartheta_\tau \frac{\partial F}{\partial \tau}
\end{bmatrix},
\] (5.32)

where \(\alpha_n, \beta_n, \rho_n\) and \(\tau_n\) and \(\alpha_{n-1}, \beta_{n-1}, \rho_{n-1}\) and \(\tau_{n-1}\) are the newly computed and previous model parameters for the QP \(q_0\) being considered. Further, \(\vartheta_\alpha, \vartheta_\beta, \vartheta_\rho,\) and \(\vartheta_\tau\) are the LMS filter’s step size controlling the adaption speed and are empirically determined as \(10^{-4}, 10^{-5}, 10^{-5}\) and \(10^{-6}\) respectively. Finally, the partial derivatives of \(F\) in (5.32) with respect to the model parameter are

\[
\frac{\partial F}{\partial \alpha} = -\frac{2}{\alpha} \left( \frac{\Delta D^2}{\lambda_r \lambda_c} \right)^2 + 2 \frac{\Delta C \Delta R \Delta D^2}{\alpha \lambda_r \lambda_c},
\] (5.33)

\[
\frac{\partial F}{\partial \beta} = 2 \ln \left( \frac{R_{CTU}^\text{T}}{N} \right) \left\{ \frac{\Delta C \Delta R \Delta D^2}{\lambda_r \lambda_c} - \left( \frac{\Delta D^2}{\lambda_r \lambda_c} \right)^2 \right\},
\] (5.34)

\[
\frac{\partial F}{\partial \rho} = -\frac{2}{\rho} \left( \frac{\Delta D^2}{\lambda_r \lambda_c} \right)^2 + 2 \frac{\Delta C \Delta R \Delta D^2}{\rho \lambda_r \lambda_c},
\] (5.35)

and

\[
\frac{\partial F}{\partial \tau} = 2 \ln \left( \frac{C_{CTU}^\text{T}}{N} \right) \left\{ \frac{\Delta C \Delta R \Delta D^2}{\lambda_r \lambda_c} - \left( \frac{\Delta D^2}{\lambda_r \lambda_c} \right)^2 \right\},
\] (5.36)

where \(R_{CTU}^\text{T}, C_{CTU}^\text{T}\) are the target bit rate and decoding-complexities, respectively. Once the model parameters are updated as per (5.32) in this manner, the new parameters can be used to determine the \(\lambda_r\) and \(\lambda_c\) trade-off factors for the mode selection in the cost function in (5.4).

### 5.3 Experimental Results and Discussion

This section presents the performance of the proposed CTU level decoding complexity and rate control algorithm in terms of its decoding complexity, and rate controlling capabilities, decoding complexity and energy consumption reduction performance with an analysis on the impact to the video quality. The rate and decoding complexity controlling capabilities of the proposed algorithm are first compared with three state-of-the-art decoding complexity-aware encoding algorithms in the literature. Thereafter, experimental results for the power consumption during a video streaming session are presented when decoding the bit streams using two popular CPU frequency governing methods. Finally, the experimental results and observations are discussed in detail for the different use cases.
5.3.1 Simulation Environment

The proposed encoding algorithm is implemented in the HM 16.0 reference encoder, where the decoding complexity estimation models presented in Chapter 4 and Lagrangian cost function that determines the coding modes for both inter- and intra-prediction and the proposed decoding complexity, rate controlling algorithm are integrated to the HEVC encoding tool chain. The resultant bit streams are decoded using the openHEVC [113] software decoder on an Intel x86 Core i7-6500U system running Ubuntu 16.04 to measure the decoding complexity performance of the bit streams. In this case, the proposed algorithm’s performance is compared with three state-of-the-art approaches; the power-aware encoding algorithm proposed by He et al. [27], rate, distortion and decoding energy optimized encoding algorithm proposed by Herglotz et al. [115] and the tunable HEVC decoder proposed by Nogues et al. [93]. The HD video sequences used in the experiments are encoded in the random access configurations for 900Kbps, 1Mbps, 2Mbps and 4Mbps video bit rates (with rate control enabled), as defined in the common test configuration [16]. Moreover, decoding complexity and rate controlling performance of the proposed algorithm are evaluated using two decoding complexity levels for each aforementioned encoding bit rates. In this case, the decoding complexity levels are determined to have 30% and 40% less total CPU cycles (with reference to the HM decoder implementation [63]) compared to the HM encoded bit streams for the respective bit rates and are indicated as proposed L1 and L2, respectively. The complexity of the decoding process is measured using the commonly used instruction level analysis tools Callgrind/Valgrind [22].

Finally, the decoder’s energy consumption is determined by measuring the energy dissipated by the system during the video playback for the proposed as well as state-of-the-art algorithms. In this context, the test setup corresponds to an online video streaming scenario where the openHEVC decoder is used as the playback client. The encoded bit streams are streamed for a duration of 15 minutes and the energy capacity reduction of the lithium ion battery of the device is measured using the Linux power measurement tools. In should be noted that the energy reduction measured corresponds to the overall energy consumption of the device that includes energy consumed for the wireless communication, video decoding and video presentation. Furthermore, the impact on energy consumption for the video decoding when using an application specific DVFS algorithm [31] as oppose to the Linux ondemand frequency governor is also analyzed and presented in the experimental results.

5.3.2 Evaluation Metrics

The performance of the proposed algorithm is evaluated in multiple stages. The evaluation metrics used to measure the performance of the proposed as well as state-of-the-art algorithms are described as follows.

Decoding Complexity and Rate Controlling Performance

First, the decoding complexity, and rate controlling capabilities of the proposed algorithm are evaluated by measuring the percentage error in achieving the target decoding
5.3. Experimental Results and Discussion

complexity and rate level. In this case, percentage error in bit rate is calculated using,

\[ R_e = 100 \times \frac{(R^T - R^r)}{R^T}. \]  \hspace{1cm} (5.37)

where \( R^r \) and \( R^T \) are the achieved bit rate, and target bit rate, respectively. Similarly, the overall decoding complexity controlling performance of the proposed algorithm is measured using,

\[ C_e = 100 \times \frac{(C^T - C^r)}{C^T}. \]  \hspace{1cm} (5.38)

where \( C^T \) and \( C^r \) are the target and achieved decoding complexity levels in terms of the CPU cycles, for a particular number of frames. In this case, the target decoding complexity is given for the decoding complexity levels L1 and L2 (ref. 5.3.1) with respect to the HM reference decoder \[63\] and the decoding complexity achieved is measured using the commonly used instruction level analysis tools Callgrind/Valgrind \[22\] for the same HM16.0 reference decoder \[63\]. Moreover, the frame-wise rate, and decoding complexity control performances are measured using the percentage error between the allocated and actual number of bits and decoding complexity per frame, respectively.

Decoding Complexity, Energy Reduction Performance and Video Quality Impact

Next, the impact on video quality, decoding complexity, and the respective energy reduction achieved for a particular decoding complexity level in the proposed algorithm (e.g., decoding complexity level L2 is considered in this case) are compared against the state-of-the-art algorithms keeping HM16.0 as the reference. In this case, the average impact on video quality is measured using the impact on PSNR given by,

\[ \Delta PSNR = PSNR^\kappa - PSNR^HM, \]  \hspace{1cm} (5.39)

where \( PSNR^HM \) and \( PSNR^\kappa \) are the resultant average PSNR for the reconstructed video sequences when using HM16.0 and proposed and other state-of-the-art algorithms, respectively. Similarly, the percentage reduction in decoding complexity, and corresponding energy reduction achieved by the algorithms are measured using,

\[ \Delta C = 100 \times \frac{(C^\kappa - C^HM)}{C^HM}, \]  \hspace{1cm} (5.40)

and

\[ \Delta E = 100 \times \frac{(E^\kappa - E^HM)}{E^HM}, \]  \hspace{1cm} (5.41)

respectively. Here, the subscript \( HM \) denotes the HM encoded bit stream whereas, the subscript \( \kappa \) corresponds to the bit streams generated by the proposed and state-of-the-art algorithms. Moreover, the proposed and state-of-the-art algorithms are evaluated

\[ ^5\text{The decoding complexity controlling performance is presented only for the proposed algorithm as state-of-the-art algorithms do not support a mechanism to achieve a given decoding complexity.} \]
5.3. Experimental Results and Discussion

Table 5.1: Rate controlling performance of the encoding algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Proposed L2</th>
<th>HM 16.0 [63]</th>
<th>He et al. [27]</th>
<th>Herglotz et al. [115]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_e%$</td>
<td>$R_e%$</td>
<td>$R_e%$</td>
<td>$R_e%$</td>
</tr>
<tr>
<td>Band</td>
<td>0.350</td>
<td>0.093</td>
<td>-0.03</td>
<td>0.002</td>
</tr>
<tr>
<td>Beergarden</td>
<td>0.189</td>
<td>3.050</td>
<td>8.22</td>
<td>3.319</td>
</tr>
<tr>
<td>Cafe</td>
<td>0.010</td>
<td>1.229</td>
<td>0.12</td>
<td>0.443</td>
</tr>
<tr>
<td>Dancer</td>
<td>1.448</td>
<td>2.344</td>
<td>1.56</td>
<td>0.100</td>
</tr>
<tr>
<td>GTFly</td>
<td>0.018</td>
<td>0.067</td>
<td>0.14</td>
<td>0.129</td>
</tr>
<tr>
<td>Kimono</td>
<td>0.014</td>
<td>4.097</td>
<td>6.34</td>
<td>1.607</td>
</tr>
<tr>
<td>Musicians</td>
<td>1.125</td>
<td>0.054</td>
<td>3.78</td>
<td>0.725</td>
</tr>
<tr>
<td>Parkscene</td>
<td>0.105</td>
<td>2.352</td>
<td>4.56</td>
<td>0.386</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>0.359</td>
<td>2.548</td>
<td>3.07</td>
<td>2.579</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.40</strong></td>
<td><strong>1.75</strong></td>
<td><strong>3.08</strong></td>
<td><strong>1.02</strong></td>
</tr>
</tbody>
</table>

To measure the amount of decoding complexity and energy reduction achieved for a 1 dB PSNR loss in the video quality. In this case, $\Delta C(\%)$ per PSNR(dB) is measured as,

$$\Delta C(\% / \text{dB}) = \frac{\Delta C}{\text{PSNR}}.$$  \hfill (5.42)

Similarly, $\Delta E(\%)$ per PSNR(dB) is measured as,

$$\Delta E(\% / \text{dB}) = \frac{\Delta E}{\text{PSNR}}.$$  \hfill (5.43)

where $\Delta C$ and $\Delta E$ are calculated as in (5.40) and (5.41), respectively.

5.3.3 Performance Evaluation and Analysis

This section presents the experimental results for the proposed as well as state-of-the-art algorithms compared to the HM encoded HEVC bit streams. In this context, the decoding complexity and rate controlling performances are analyzed first. Then, the decoding complexity reductions achieved by the proposed as well as state-of-the-art algorithms and their quality impacts are discussed with respect to the experimental setup discussed in Sec. 5.3.1.

Rate Controlling Performance

The percentage deviation of the bit rate achieved after the encoding process from the target bit rate for the proposed as well as state-of-the-art algorithms are presented in
5.3. Experimental Results and Discussion

In this case, the video sequences are encoded using four different bit streams (described in Sec. 5.3.1) and the averaged percentage error in the achieved bit rates compared to the respective targets is presented in the Table 5.1.

It can be observed that the rate controlling algorithm implemented in the HM 16.0 reference encoder shows on average 1.75% deviation on from the target bit rate after the encoding. However, it is relatively trivial compared to the 3.08% deviation from the target bit rate experienced with the encoding algorithm proposed by He et al. [27]. The encoding algorithm proposed in [27] uses a rate, distortion and decoding complexity based cost function to select the PU level prediction modes and integer-pel vs. fractional-pel motion vectors during the encoding process. Moreover, the use of in-loop filters per frame is also decided based on a cost function that considers the overall impact of the operation to the distortion and decoding complexity. However, the rate and decoding complexity trade-off factors which are content and QP agnostic causes the rate controller to fail when effectively utilizing the bit budget allocated for the frame as well as CTU. However, the encoding algorithm proposed in [115] uses a QP dependent trade-off factors for both rate and decoding-complexity, thus the impact on the rate controller is significantly improved compared to the method proposed by He et al. [27].

In contrast, the proposed algorithm uses a content adaptive decoding complexity, rate, distortion model to derive QP as well as rate and decoding complexity trade-off factors to determine the optimum set of coding modes and structures that minimize the distortion while achieving a given bit and decoding complexity budget. Therefore, as illustrated in the Table 5.1, the proposed algorithm is proven to achieve the allocated bit rate with only <1% error indicating that both CTU level bit allocation as well as coding parameter selection are more accurate and content adaptive compared to the state-of-the-art method.

In addition, the frame-wise rate controlling performance of the encoding algorithms is analyzed using the % error between the allocated bits and actual bits per frame. A graphical illustration of the frame-wise % error for bits is presented in the Fig. 5.3. 6 It can be observed that the rate controlling algorithms implemented in HM 16.0 and other state-of-the-art encoding algorithms suffer from large % errors throughout the video sequence. The incorporation of a third parameter within the mode selection cost function in He et al. [27] and Herglotz et al. [115] crucially affect the rate controller in achieving the allocated number of bits for a given block. For example, both these algorithms use a RD optimization based bit allocation, QP and Lagrangian parameter determination approach [121] for the rate control while utilizing a three parameter cost function (involving rate, distortion and decoding complexity) for the coding mode selection. The correlation that exist between the three parameters which is ignored when performing the rate control, results in a large average rate controlling errors as illustrated in the Table 5.1. The rate controlling algorithm in HM16.0 which follows a R-λ based bit allocation and coding parameter selection approach also shows some deficiency in achieving the allocated bit budget for each frame. However, as illustrated in the Table 5.1, HM16.0 encoder still demonstrate a 1.75% error in its rate controlling.

6The % bit errors per frame for the proposed algorithm is also presented separately in the Fig. 5.4 for the purpose of clarity.
Table 5.2: Decoding complexity controlling performance of the proposed encoding algorithm

<table>
<thead>
<tr>
<th></th>
<th>Proposed L1</th>
<th></th>
<th>Proposed L2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_e %$</td>
<td>$C_e %$</td>
<td>$R_e %$</td>
<td>$C_e %$</td>
</tr>
<tr>
<td>Band</td>
<td>0.638</td>
<td>-1.348</td>
<td>0.350</td>
<td>3.395</td>
</tr>
<tr>
<td>Beergarden</td>
<td>0.383</td>
<td>0.733</td>
<td>0.189</td>
<td>6.716</td>
</tr>
<tr>
<td>Cafe</td>
<td>0.012</td>
<td>-0.750</td>
<td>0.010</td>
<td>4.248</td>
</tr>
<tr>
<td>Dancer</td>
<td>1.534</td>
<td>8.485</td>
<td>1.448</td>
<td>1.448</td>
</tr>
<tr>
<td>GTFly</td>
<td>0.001</td>
<td>-5.864</td>
<td>0.018</td>
<td>-1.326</td>
</tr>
<tr>
<td>Kimono</td>
<td>0.015</td>
<td>-9.019</td>
<td>0.014</td>
<td>-4.276</td>
</tr>
<tr>
<td>Musicians</td>
<td>0.992</td>
<td>-6.867</td>
<td>1.125</td>
<td>-2.950</td>
</tr>
<tr>
<td>Parkscene</td>
<td>0.208</td>
<td>-7.407</td>
<td>0.105</td>
<td>-3.340</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>1.140</td>
<td>1.973</td>
<td>0.359</td>
<td>8.265</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.54</strong></td>
<td><strong>-2.22</strong></td>
<td><strong>0.40</strong></td>
<td><strong>1.35</strong></td>
</tr>
</tbody>
</table>

function.

In contrast, the proposed algorithm uses a decoding-complexity–rate–distortion model based coding mode selection approach for the simultaneous rate and decoding complexity controlling. This enables the encoder to effectively utilize the correlation between the three parameters to perform rate, decoding complexity allocation and appropriate coding mode selection resulting in a lesser % bit error (illustrated in the Fig. 5.4 and Table 5.1). Moreover, the model parameter update process introduced in Sec. 5.2.3 keeps the algorithm content relevant, thus, minimizing the errors achieved during the encoding process.

**Decoding Complexity Controlling Performance**

Controlling a decoding complexity level allocated for a frame or a CTU is currently not available for any of the state-of-the-art encoding algorithms. Hence, the experimental results summarized in the Table 5.2 that shows the percentage deviation of the achieved decoding complexity from the target decoding complexity level (allocated CPU cycles for the video sequence), corresponds to the proposed encoding algorithm. In this context, the proposed algorithm shows on average $\approx 1.78\%$ decoding complexity error for both complexity levels considered. Hence, it is evident that the proposed algorithm is capable of generating a bit stream that adheres to a given bit rate and a decoding complexity level.

Furthermore, frame-wise decoding complexity error illustrated in the Fig. 5.5 also emphasizes that the proposed encoding algorithm is capable of maintaining a marginal
error despite the dynamic nature of the video content. In summary, the simultaneous rate and decoding complexity control capability of the proposed method verified in both numerical and graphical illustrations indicate that the proposed method is more content adaptive and is capable of achieving the respective bit and decoding complexity targets; a crucial contribution for the adaptive streaming solutions attempting to reduce the decoding complexity and energy consumption of video playback devices.

### Decoding Complexity Reduction and Video Quality Impact

Table 5.3 demonstrates the average decoding complexity reductions and the corresponding quality impact in terms of PSNR for the proposed as well as state-of-the-art algorithms.

The experimental results presented in the Table 5.3 reveal that the algorithms proposed by He et al. [27] and Herglotz et al. [115] both achieve decoding complexity reductions in the range of 10% and 20%, respectively. However, both these algorithms have shown a significant reduction in PSNR while delivering these decoding complexity reductions. For example, the encoding algorithm proposed by Herglotz et al. [115] uses a decoding complexity estimation model proposed in [102][108][100]. However, the bit rate, decoding complexity trade-off factors are selected independently, thus the impact on each other is overlooked during the coding mode selection. Even though, the bit rate trade-off factor which is based on the RD relationship defined in [63][111], is content adaptive, the decoding complexity trade-off factor remains agnostic to the dynamics of the video sequence. These crucial elements that have been overlooked in this method, ultimately results in a higher quality loss. Similarly, the method proposed in [27], uses predefined trade-off factors only for PU level mode selection, thus, the amount of sacrifice made in video quality to maintain the bit rate requirements is evidently high. However, the decoding complexity-aware coding mode selection is used only at the PU level enabling the decoding complexity reduction achieved to be less than that of the method proposed by Herglotz et al. [115]. These attributes are graphically illustrated in the ΔPSNR vs. decoding complexity graphs presented in the Fig. 5.7. Here, it can be observed that both these algorithms demonstrate a higher quality impact in a rate controlled scenario if they were to achieve a particular decoding complexity. However, the encoding algorithm proposed in Herglotz et al. has shown a slight improvement in very-low complex video sequences such as “band”, “cafe” etc., which should be noted.

Nogues et al. [26] propose an algorithm that modifies the decoding operations to reduce the decoding complexity of a video bit stream. For example, the skipping of in-loop filtering and simplifying the motion compensation operations within the decoder results in a significant complexity reduction. However, changing the motion compensation filters and thereby, applying the decoded residuals on a predicted PU which is different from that of the encoder’s, causes more distortions in the reconstructed block. Even though, the intra-frames that appear within the given interval avoid the propagation of these error, the proposed algorithm results in a much higher PSNR reduction (Ref. Fig. 5.7).

---

7It should be noted that the presented results correspond to the highest complexity reduction that can be achieved by applying the proposed decoder modifications to all frames in the bit stream.
Figure 5.3: An illustration of frame-wise percentage error between the allocated bits and actual bits for the HM 16.0, proposed and other state-of-the-art algorithms.
Figure 5.4: An illustration of frame-wise percentage error between the allocated bits and actual bits for the proposed algorithms.
5.3. Experimental Results and Discussion

Figure 5.5: An illustration of frame-wise percentage error between the allocated decoding complexity and the actual decoding complexity for the proposed algorithm for two complexity levels for a given bit rate.

(a) Parkscene 2Mbps

(b) Dancer 2Mbps

(c) Musicians 2Mbps
Figure 5.6: The variation of $\Delta C/\Delta \text{PSNR}$ (i.e., $\overline{\Delta C}(%/\text{dB})$) for each bit rate.
5.3. Experimental Results and Discussion

(a) Dancer 1088p

(b) Musicians 1080p

(c) Parkscene 1080p

Figure 5.7: The variation of $\Delta$PSNR for each bit rate.
Table 5.3: Decoding complexity reduction performance

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed L2 (Model only)</th>
<th>Proposed L2* (Model + LF [93])</th>
<th>He et al. [27] (PUM + DBLK)</th>
<th>Herglotz et al. [115]</th>
<th>Nogues et al. [93] (MC+LF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>∆PSNR</td>
<td>∆C% ‡</td>
<td>∆PSNR</td>
<td>∆C% ‡</td>
<td>∆PSNR</td>
</tr>
<tr>
<td>Band</td>
<td>-0.89</td>
<td>-9.77</td>
<td>-1.36</td>
<td>-16.74</td>
<td>-0.46</td>
</tr>
<tr>
<td>Cafe</td>
<td>-2.01</td>
<td>-8.36</td>
<td>-3.20</td>
<td>-15.35</td>
<td>-0.56</td>
</tr>
<tr>
<td>GTFly</td>
<td>-1.85</td>
<td>-16.22</td>
<td>-2.24</td>
<td>-23.28</td>
<td>-1.11</td>
</tr>
<tr>
<td>Kimono</td>
<td>-1.06</td>
<td>-17.48</td>
<td>-1.28</td>
<td>-24.62</td>
<td>-1.02</td>
</tr>
<tr>
<td>Musicians</td>
<td>-1.03</td>
<td>-16.52</td>
<td>-1.16</td>
<td>-23.47</td>
<td>-1.63</td>
</tr>
<tr>
<td>Parkscene</td>
<td>-0.96</td>
<td>-17.01</td>
<td>-1.36</td>
<td>-23.55</td>
<td>-2.03</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>-2.00</td>
<td>-5.92</td>
<td>-3.25</td>
<td>-13.03</td>
<td>-2.08</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-1.44</strong></td>
<td><strong>-12.71</strong></td>
<td><strong>-2.03</strong></td>
<td><strong>-19.58</strong></td>
<td><strong>-1.67</strong></td>
</tr>
</tbody>
</table>

‡ ∆C% achieved using the openHEVC decoder.

* Here, the bit streams for complexity level 2 are subjected to the LF algorithm.
In contrast, the proposed algorithm uses a more comprehensive and a dynamic approach to simultaneously control both decoding complexity and bit rate. First, the use of more accurate and detailed decoding complexity estimation models enables the encoder to estimate the decoding complexity requirements for a given coding mode for the content being encoded. Next, the detailed analysis and the proposed novel decoding complexity–rate–distortion model allows the encoder to determine the impact of a coding mode on all three parameters. Finally, the continuous update of the decoding complexity–rate–distortion model allows the encoder to pick the most content relevant trade-off factors making the proposed algorithm selecting the best coding modes that minimize the distortion while achieving the given rate and decoding complexity constraints. For example, the graphical illustration in the Fig. 5.7 depict that the proposed algorithm allows the encoder to generate bit streams that provide the least quality impact for a given decoding complexity. Moreover, the proposed algorithm is highly scalable and provides the capability to generate bit streams with multiple bit rate and decoding complexity levels; a crucial benefit for adaptive video streaming services that target streaming videos to mobile devices. Finally, the bit streams generated by the proposed algorithms are decoded with an openHEVC decoder that skips the in-loop filter operations. Here, it can be observed that it increases the decoding complexity reduction by \( \approx 7\% \), with only a minor impact on the video quality. Thus, it is evident that the bit streams generated by the proposed algorithm can be subjected to decoder modifications such as [93] to attain further complexity reductions.

The graphical illustration presented in the Fig. 5.6 demonstrate the amount of decoding complexity reduction that can be achieved for particular quality loss in PSNR. It can be observed that the proposed algorithm on average manages to achieve a higher \( \Delta C(\% \text{/} \text{dB}) \) across all bit rates. However, the \( \Delta C(\% \text{/} \text{dB}) \) metric returns a much higher number for the proposed algorithm at lower bit rates due to the reduced quality impact compared to the HM encoded bit streams. Thus, it is apparent that the proposed algorithm surpasses the state-of-the-art encoding algorithms in achieving an increased decoding complexity reduction for each 1 dB quality loss incurred during the process; a non-trivial advantage over the state-of-the-art to facilitate video content preparation for decoding complexity reduction and adaptive streaming solutions.

### Decoder Energy Reduction Performance

Next, the actual energy consumption performance of the proposed as well as state-of-the-art algorithms is evaluated for a video streaming use case. First, the bit streams generated by the proposed and state-of-the-art algorithms are decoded using the openHEVC video decoder with Linux ondemand as the frequency scaling governor [95]. It can be observed that both proposed as well as state-of-the-art algorithms demonstrate an energy consumption reduction in the range of \( \approx 4\% \) compared to HM 16.0 encoded video bit streams. Moreover, forcing the decoder to skip in-loop filters enable the proposed algorithm to increase the energy consumption reduction up to 5.65%.

Changing the Linux ondemand governor to a more application specific DVFS algorithm [31] that alters the CPU’s operational frequency based on the estimated complexity of the next video frame, improves the energy consumption reduction of all the algorithm.
Table 5.4: Decoder energy reduction performance during a video streaming session

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed L2 (Model only)</th>
<th>Proposed L2 (Model + LF [93])</th>
<th>He et al. [27] (PUM + DBLK)</th>
<th>Nogues et al. [93] (MC+LF)</th>
<th>Herglotz et al. [115]</th>
<th>$\Delta E%$ †</th>
<th>$\Delta E%$ ‡</th>
</tr>
</thead>
<tbody>
<tr>
<td>Band</td>
<td>-1.56</td>
<td>-4.61</td>
<td>-3.49</td>
<td>-7.71</td>
<td>-1.16</td>
<td>-3.49</td>
<td>-2.34</td>
</tr>
<tr>
<td>Beergarden</td>
<td>-2.22</td>
<td>-3.73</td>
<td>-2.78</td>
<td>-5.34</td>
<td>0.19</td>
<td>-2.53</td>
<td>-2.75</td>
</tr>
<tr>
<td>Kimono</td>
<td>-4.55</td>
<td>-11.16</td>
<td>-5.92</td>
<td>-12.61</td>
<td>-4.72</td>
<td>-6.81</td>
<td>-8.56</td>
</tr>
<tr>
<td>Musicians</td>
<td>-2.11</td>
<td>-6.24</td>
<td>-4.06</td>
<td>-6.86</td>
<td>-1.12</td>
<td>-3.64</td>
<td>-1.53</td>
</tr>
<tr>
<td>Parkscene</td>
<td>-3.82</td>
<td>-6.74</td>
<td>-4.14</td>
<td>-7.33</td>
<td>-1.69</td>
<td>-3.52</td>
<td>-5.32</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>-6.57</td>
<td>-7.41</td>
<td>-6.71</td>
<td>-7.04</td>
<td>-2.22</td>
<td>-8.36</td>
<td>-4.54</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>-4.00</strong></td>
<td><strong>-7.77</strong></td>
<td><strong>-5.65</strong></td>
<td><strong>-9.10</strong></td>
<td><strong>-2.22</strong></td>
<td><strong>-5.04</strong></td>
<td><strong>-5.15</strong></td>
</tr>
</tbody>
</table>

† $\Delta E\%$ achieved when using Linux ondemand frequency governor.

‡ $\Delta E\%$ achieved when using an application specific DVFS algorithm as the frequency governor.
Table 5.5: Decoding complexity and energy reduction per 1 dB quality loss for the proposed and state-of-the-art algorithms

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed L2 (Model only)</th>
<th>Proposed L2 (Model + LF [93])</th>
<th>He et al. [27] (PUM + DBLK)</th>
<th>Nogues et al. [93] (MC+LF)</th>
<th>Herglotz et al. [115]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta C$</td>
<td>$\Delta E^\dagger$</td>
<td>$\Delta E^\ddagger$</td>
<td>$\Delta C$</td>
<td>$\Delta E^\dagger$</td>
</tr>
<tr>
<td>Band</td>
<td>10.97</td>
<td>1.75</td>
<td>6.30</td>
<td>12.30</td>
<td>2.56</td>
</tr>
<tr>
<td>Beergarden</td>
<td>5.65</td>
<td>1.76</td>
<td>2.96</td>
<td>7.03</td>
<td>1.38</td>
</tr>
<tr>
<td>Cafe</td>
<td>4.15</td>
<td>3.12</td>
<td>6.27</td>
<td>4.79</td>
<td>3.08</td>
</tr>
<tr>
<td>Dancer</td>
<td>8.23</td>
<td>1.11</td>
<td>2.66</td>
<td>8.92</td>
<td>2.06</td>
</tr>
<tr>
<td>GTFly</td>
<td>8.76</td>
<td>3.65</td>
<td>6.08</td>
<td>10.39</td>
<td>3.92</td>
</tr>
<tr>
<td>Musicians</td>
<td>16.03</td>
<td>2.04</td>
<td>6.05</td>
<td>20.23</td>
<td>3.50</td>
</tr>
<tr>
<td>Parkscene</td>
<td>17.71</td>
<td>3.97</td>
<td>7.02</td>
<td>17.31</td>
<td>3.04</td>
</tr>
<tr>
<td>Poznan St.</td>
<td>2.96</td>
<td>3.28</td>
<td>3.70</td>
<td>4.00</td>
<td>2.06</td>
</tr>
<tr>
<td>Average</td>
<td>10.11</td>
<td>2.77</td>
<td>5.73</td>
<td>11.58</td>
<td>2.91</td>
</tr>
</tbody>
</table>

The metrics $\widetilde{\Delta C}$ (%/dB) and $\widetilde{\Delta E}$ (%/dB) are both measured in terms of the $\Delta C$(%) and $\Delta E$(%) achieved per 1 dB PSNR quality loss for the proposed and state-of-the-art algorithms.

$\dagger$ $\widetilde{\Delta E}$ (%/dB) achieved when using Linux ondemand frequency governor.

$\ddagger$ $\widetilde{\Delta E}$ (%/dB) achieved when using an application specific DVFS algorithm as the frequency governor.
In this case, the proposed algorithm has achieved 7.77% and 9.10% decoder energy consumption reduction compared to the HM encoded bit streams; a non-trivial performance with only -1.44 dB and -2.03 dB quality impacts for with and without in-loop filter operations, respectively. Moreover, the decoder energy reduction achieved per 1 dB PSNR video quality loss for proposed as well as state-of-the-art algorithms is presented in the Table 5.5. In addition the variation of $\Delta E(\%/\text{dB})$ achieved for each 1 dB quality level drop is graphically demonstrated for three different test sequences in the Fig. 5.8. These results further elaborate that the energy reduction achieve by the bit streams generated from the proposed algorithm are subjected to less quality impacts compared to the state-of-the-art approaches, thus the decoding energy consumption reduction achieved for each 1 dB PSNR loss, by reducing the decoding complexity is relatively high in the proposed encoding algorithm. Hence, the overall improvements of the proposed algorithm over the existing content generation methods is substantial and paves the foundation for a seamless decoder energy-aware adaptive video streaming solutions; a crucial development that support green multimedia consumption in CE devices.

5.4 Summary

This Chapter proposes an encoding algorithm that can prepare HEVC bit streams with arbitrary bit rate and decoding complexity levels. In this context, a decoding complexity, rate and distortion model is first introduced that models the relationship among the three parameters. Next, a decoding complexity, and rate allocation method is proposed followed up by a mechanism to determine appropriate QP, and trade-off factors for decoding complexity, and rate to meet the given complexity and bit rate constraints. Finally, a Least Mean Square (LMS) adaptive filter based model parameter update algorithm is introduced to keep the proposed model adaptive to the content dynamics.

The experimental results demonstrate stable overall and frame-wise rate and decoding complexity controlling capabilities for the proposed encoding algorithm. For example, average errors of 0.47% and 1.78% are observed for the rate controlling, and decoding complexity controlling, respectively between the allocated and actual values. Moreover, an average decoding complexity reduction performance 10.11 (%/dB) and energy reduction performance of 5.73 (%/dB) are observed for a 1dB PSNR quality drop for video streaming applications with the bit streams generated by the proposed algorithm, compared to the bit streams from HM 16.0 encoder.

Adaptive streaming is often seen as a potential solution to cater for the dynamic fluctuations of the network bandwidth. Similarly, adaptive controlling of the bit streams to cater for the dynamic resource capacities of the decoder (i.e., remaining energy capacity) has become a compelling challenge to improve the energy efficiency of the mobile video playback devices. Thus, the proposed encoding framework presents a novel approach to prepare video contents with multiple decoding complexity and bit rate levels to facilitate adaptive streaming solutions that consider both network as well as decoder resource constraints. In this context, the bit streams generated by the proposed algorithm achieve the highest decoding complexity and energy reductions for a particular
quality drop in the bit streams. Moreover, the consistent decoding complexity achieved
gives additional benefits for the DVFS algorithms to select appropriate operational
frequency level for the subsequent video frames. In summary, the capability of the pro-
posed algorithm to prepare high quality bit streams with given bit rate and decoding
complexity constraints facilitates the adaptive video streaming applications to improve
the energy efficiency of the decoders as well as end-user’s quality of experience.
Figure 5.8: The variation of $\Delta E/\Delta \text{PSNR}$ (i.e., $\Delta E(\%$/dB)) for each bit rate.
Chapter 6

Conclusions and Future Work

The research work proposed in this thesis introduces three main contributions to the HEVC based video coding. The first research contribution in this thesis introduces a novel solution to achieve a consistent encoder complexity reduction for HEVC based video coding across multiple quality levels for a wider range of video content types. Next, a novel encoding algorithm that can generate HEVC decoding complexity-aware video bit streams with minimal impact on the coding efficiency is proposed as the second research contribution. Finally, as the third research contribution, this thesis introduces the novel concept of decoding complexity and rate controlled video coding which can produce HEVC video bit streams that adhere to multiple bit rate and decoding complexity levels; a crucial contribution towards realizing green multimedia technologies. The research studies constitute detailed analysis and descriptions of algorithmic derivations, extensive simulations and experiments focusing on a wider range of use cases and application scenarios. This Chapter concludes the work presented in the thesis with a summary of research contributions, achievements and their impacts followed by potential future work and research directions.

6.1 Concluding Remarks: Content Adaptive Fast CU Size Selection

As illustrated in Chapter 1 and Chapter 2, video compression is a dynamic research area which needs continuous improvements to cater for the upcoming video demands. As a result, HEVC video coding standard was introduced as the successor to the H.264/AVC in early part of 2013, which demonstrates average bit rate reductions of 40% – 50% compared to its predecessor for the same video quality levels. However, as detailed in Chapters 2 and 3, the compression efficiency improvements in HEVC in return has resulted in a significant increase in the computational complexity of the HEVC encoders. Therefore, numerous research activities have emerged in the recent past that attempt to reduce the encoder complexity associated with RD optimization process that is followed to determine the best coding structure for a given video content. As Sec. 2.3 elaborates, these state-of-the-art approaches show deficiencies in their improvements when it comes to highly textured and complex video sequences. Moreover, it can be
observed that the complexity achieved is barely consistent across a wider range of quality levels and content types. Therefore, it becomes evident that implementation friendly, flexible encoding algorithms are crucially important that has the capability to be dynamic and content adaptive to achieve a fairly consistent encoding complexity reduction performance with a consistent marginal impact to the coding efficiency.

Hence, the first research contribution in this thesis proposes a content adaptive fast CU size selection algorithm for HEVC based low delay video encoding. In this context, two novel CU split likelihood models (based on a motion feature-based and a RD cost threshold-based CU classification approaches) are introduced to model the CU split and non-split decisions. These models are dynamically generated and are continuously adapted using initial and intermediate training phases, such that they independently predict the split decision for a given CU. Moreover, the possibility of reusing motion vectors identified during the modeling stage, for motion estimation in the remaining PU modes, is also investigated to supplement the proposed algorithm. One major conclusion to be drawn from this analysis is that the initial evaluation of the Inter $N \times N$ mode provides motion and complexity properties of the underlying CU, which can be used to classify a CU, in order to model the split likelihood. Furthermore, the use of two independent models facilitates the split decision refinement as well as the identification of when the models require training dynamically during the encoding cycle. The window based approach used in the model adaptation and decision making ensures that the resultant split decisions are content-adaptive and less susceptible to the dynamic variations such as scene changes; a non-trivial advantage over the state-of-the-art methods.

In conclusion, the simulation results for the proposed CU size selection and encoding algorithm reveal an average encoding time saving of 58% and 61% for the low delay $P$ and low delay $B$ configurations, respectively. Moreover, the experimental results reveal that the proposed encoding algorithms can achieve a relatively uniform average encoding time saving across a wide range of QPs and content ranging from low to highly complex textures and motion characteristics, due to its SKIP/merge mode agnostic early CU size prediction. The capacity of the proposed algorithm to maintain a consistent performance, in terms of both the encoding time saving as well as BDBR increase (which is 2.29 % on average), across diverse content types and QPs is especially notable when considering the performance fluctuations observed in the state-of-the-art solutions.

6.2 Concluding Remarks: Decoding Complexity-Aware HEVC Encoding

As presented in Chapter 1 and Chapter 2, ever-increasing multimedia consumption has caused multimedia devices to be amongst the largest energy consuming equipment in the CE market. In addition, as explained in Chapter 2, and 4, the increased popularity of HD and UHD video contents and the mobile video consumption requirements have made video decoding and presentation resource intensive operations for a mobile handheld device. Moreover, the tight correlation that exist between the decoder’s energy
consumption and the complexity of the codec as well as the content being decoded evidently make the high resolution complex video contents encoded with complex HEVC video coding standard to demand more computational processing and energy resources from the resource constrained devices. To this end, reducing the complexity of the encoded bit stream is seen as a potential application layer solution to reduce the complexities associated with the decoder operations. In this case, as illustrated in the Sec. 2.4, the state-of-the-art application layer approaches hardly consider the intricate decoding complexity levels associated with the assortment of coding modes and features in HEVC. Thus, the prevailing approaches typically follow brute-force approaches to reduce the complexity and the associated energy consumption of the decoders.

In this context, as the second contribution of this thesis, Chapter 4 introduces a decoding complexity-aware video encoding algorithm for HEVC using a decoding complexity–rate–distortion model. In this case, this work first introduces two decoding complexity estimation models which estimate the decoding complexity for a given intra- or inter-coded HEVC encoded video bit stream. Then, these models are utilized in the encoder to perform a decoding-complexity-distortion analysis, to introduce an optimum decoding-complexity-distortion trade-off factor for a given QP. The mode decision cost function in the encoder is thereafter extended to include the decoding complexity as a cost parameter, and to introduce the proposed decoder-complexity, rate, and distortion aware coding mode selection framework. Then, the decoding complexity and rate trade-off factors are carefully selected such that decoder complexity, rate and distortion are minimized for a given content for a particular quality level.

The experimental results of the proposed algorithm reveal decoding complexity reductions (BD-C) of -29.43%, and -13.22%, with only 6.47%, BD-BR increase for a given video quality with HM 16.0 reference and openHEVC decoders, respectively. In addition, the overall energy consumption analysis reveals that the proposed algorithm can reduce the device’s energy consumption (measured as BD-E) by -4.83% when using an optimized software decoder such as openHEVC. Moreover, utilizing an application specific DVFS governor together with a loop filter skipping algorithm in the decoder improves the energy consumption performance of the proposed algorithm further, achieving up to -20.45% BD-E reduction for a similar video quality to that of HM encoded bit streams. Furthermore, the experimental results conducted with a hardware decoder reveal a BD-E reduction of -1.85% compared to the HM encoded bit streams. Finally, the proposed algorithm demonstrate an overall BD-E reduction of -3.62% when the bit streams generated by the proposed algorithm are decoded during a video streaming session. The numerical BD-E figure illustrates the decoder energy reduction that the proposed algorithm achieves for a similar video quality to that of the HM encoded bit streams, which is significant compared to the state-of-the-art algorithms in the literature.

In conclusion, it is evident that a detailed and an accurate decoding complexity estimation model is essential to effectively utilize the decoding complexity as a parameter within the mode decision cost function. Here, the inclusion of the intricacies of the assortment of coding modes and features into the models result in a bit stream which is less complex compared to those generated by the state-of-the-art algorithms. The experimental results reveal that the reduced complexity is sufficiently large to have an
impact on the device’s energy consumption in both the software and hardware decoding regimes. In addition, the bit streams generated by the proposed algorithms can be directly subjected to modified loop filtering algorithms at the decoder to gain further reductions in the decoding complexity and energy consumption. Finally, the efficient use of a DVFS algorithm can further exploit the reduced complexity of the bit streams to achieve additional energy savings as demonstrated. The decoding-complexity-distortion trade-off factor along with the consideration of rate-distortion trade-off factor within the optimization function, limits the impact on the coding efficiency to a minimum. That being said, the in depth analysis carried out in this work on these trade-off factors also reveal the flexibility in the proposed algorithm to effectively trade-off the decoding complexity to the coding efficiency based on the application requirements. Therefore, in summary, it can be concluded that the proposed encoding framework has the potential to facilitate in creating decoding-complexity-aware video bit streams; a solution which could be utilized to improve the energy efficiency of video playback in mobile devices.

6.3 Concluding Remarks: Decoding Complexity and Rate Controlled Video Coding

The rapid growth in mobile consumption of high resolution video contents coupled with the increasing complexity of the video coding standards make video decoding a challenging task for the mobile CE devices. Moreover, the proliferation of HD and UHD video contents and the limitations that prevail in the modern communication infrastructures to support this huge amount of data eventually demand network and decoding resource adaptive video streaming solutions. In this content, state-of-the-art HTTP adaptive streaming provides a solution to cope with the bandwidth related issues. Yet, adapting the video contents at video coding level by considering both network bandwidth and device constraints remains an area which hasn’t been adequately explored. The state-of-the-art work that prevail in the recent literature do not focus directly on preparing a video content for a given decoding complexity/energy requirement.

In this context, the third contribution of this thesis introduces the novel concept of generating HEVC compliant bit streams that adhere to a given bit rate and decoding complexity requirement. The proposed algorithm provides an encoding framework that performs a content adaptive CTU level simultaneous decoding complexity and rate controlling to deliver a bit streams with a predefined bit rate and a decoding complexity. To this end, this work proposes a novel decoding complexity–rate–distortion model followed by a mode selection cost function to the encoding chain. Next, an algorithm is proposed to derive the coding parameters along with the decoding complexity and bit rate trade-off factors to meet a given bit rate and decoding complexity requirements. Finally, a novel content adaptive CTU level decoding complexity and rate controlled video coding framework is proposed to generate HEVC video bit streams that comply to multiple bit rate and decoding complexity levels.

The experimental results with respect to the rate controlling aspects suggest that the proposed algorithm is capable of achieving a given bit rate with an average error of only
0.47%. Furthermore, the complexity controlling capabilities reveal that the average error of achieving a given decoding complexity is only 1.78%. The experimental results for the two complexity levels considered reveal that the proposed method is capable of generating bit streams with multiple complexity levels for a given bit rate. Thus, the experimental results presented for a particular complexity level demonstrate an average decoder complexity reduction of -12.71% with only -1.44 dB impact to the video quality. The energy consumption analysis for the respective bit streams reveal an average overall energy consumption reduction of -4.44% compared to the HM encoded bit streams. Moreover, utilizing an application specific DVFS governor allows the energy consumption reduction to be further improved up to -7.77%. In addition, skipping the in-loop filters at the decoder increases the decoder energy reduction up to -14.26%; a significant improvement compared to the state-of-the-art. Further to that, the average decoding complexity reduction and the corresponding energy reduction achieved by the bit streams generated by the proposed algorithm per 1 dB quality loss is relatively high compared to the state-of-the-art approaches. Further analysis on the experimental results reveal that the proposed algorithm is capable of generating a video bit streams at a given decoder complexity for a particular bit rate with much less quality impact compared to the state-of-the-art approaches. That being said, the decoding complexity reduction and corresponding decoder energy reduction achieved per 1 dB PSNR loss is relatively high in the proposed algorithm compared to the state-of-the-art methods.

In summary, the capability of the proposed algorithm to generate bit streams with different combinations of bit rates and decoding complexities enables the adaptive streaming services to target mobile devices with diverse capabilities. Moreover, capability of the proposed algorithm to achieve a target bit rate a decoding complexity and it’s ability to maintain a stable quality level throughout the video sequence proves the potentials of the proposed algorithm in video streaming domain. The novel concept of adaptively controlling the decoding complexity, jointly with the bit rate allows the encoder to produce bit streams with stable decoding complexities, facilitating DVFS algorithms to effectively predict the frame segment complexities and adopt the CPU frequencies accordingly. The consideration of the accurate relationship among decoding complexity, rate and distortion and continuous adaptation of the proposed decoding complexity–rate–distortion model allow the rate and decoding complexity controllers to maintain the consumed bits and decoding complexity levels within the given limits; a crucial development compared to the state-of-the-art decoding complexity-aware encoding algorithms. This makes the proposed algorithm more scalable, flexible and implementation friendly in achieving the bit streams with highest possible quality level while adhering to the rate and decoding complexity constraints. Therefore, it is evident that the proposed algorithm shows a novel content preparation dimension for the adaptive video streaming solutions that can contribute towards a green media consumption.

6.4 Overall Conclusion

The media landscape today has undergone dramatic changes over the last few decades introducing numerous technological advancements to video capturing, processing, distributing and display technologies. Moreover, the proliferation of mobile technologies,
the popularity of high resolution video contents and video sharing applications have made video data to occupy a larger percentage of the mobile data traffic. Thus, the video compression has become a crucial element in the media distribution chain that joins the content preparation and distribution nodes.

Video coding standards have been evolving since 1990s to cater for the upcoming video demands. Thus, HEVC was introduced in early 2013 to provide a coding gain of 40-50% compared to its predecessor, H.264/AVC. However, the complexity of the HEVC compatible encoders and the complexity of HEVC compliant bit streams pose, significant challenges in terms of the resource requirements of encoding servers and energy demands of the video playback devices. Therefore, this thesis presents three crucial contributions to the HEVC based video coding to reduce the complexities incurred in the encoders and to prepare less complex video bit streams that target resource constrained mobile playback devices.

The research contributions proposed in this thesis reside within the video coding node in the media distribution tool chain (Fig. 6.1.). However, the benefits of the proposed contributions are reaped by both ends of the distribution link (i.e., content creators/service providers and end users). The importance and the impact of the proposed algorithms to these entities are summarized as follows.

### 6.4.1 Benefits to the Content Creators and Service Providers

- Reducing the complexity of the HEVC encoding process, reduces the amount of processing power required to perform video encoding for both live and on-demand use cases. Thus, it is expected to minimize the operational costs of video processing that will eventually impact the net income of the businesses.

- It is expected that the efforts to reduce the encoding complexity of HEVC will eventually increase the feasibility of using HEVC as the established standard for video coding. Thus, the end-users are expected to receive high quality video contents for less bandwidth from their service providers. Therefore deploying HEVC is envisioned to attract more customers and strengthen the customer loyalty.

- The capability to generate less complex HEVC bit streams with minimal quality impact reduces the decoder energy and processing requirements, thus, the content creators and service providers are expected to improve their customer’s user experience. Moreover, the content generating and video communication mobile applications are expected to be less complex and generate less complex video streams that consider the resource requirement aspects of the decoders. Hence, the proposed contributions pave a significant opportunity to make mobile video applications energy and resource efficient, thereby attracting more customers.

### 6.4.2 Benefits to the End Users

- The contributions presented in this thesis are expected to increase the feasibility of using HEVC as the norm for video coding in the media processing work-flows
Figure 6.1: A summary of the key benefits of the proposed research contributions to the content creators, service providers and end users. The video encoding entities where the proposed contribution reside within the video distribution tool chain are indicated in the marked boxes.
of content creators/service providers. The use of HEVC brings high quality video streams to the end users with less bandwidth requirements; thus, increases the overall user experience.

- The less complex HEVC bit streams prepared by the proposed encoding algorithms require less computational and energy resources for the offline and streaming video playback. Thus, the end users are expected consume high quality HEVC videos for a longer duration without having to recharge the mobile devices.

- The capability of the proposed contributions to prepare bit streams with arbitrary decoding complexity and bit rate levels, is envisioned to enhance the adaptive video streaming services that will facilitate seamless video streaming capabilities; eventually improving the end user’s quality of experience.

- Making the encoder aware of the available resources at the decoder allows the encoder to consider these aspects to prepare the appropriate HEVC bit streams that suits a specific decoder. Decoding resource-aware encoding coupled with the reduced complexity of the encoders are expected to improve the resource utilization aspects of the video communication and content creation mobile applications. Hence, the users are expected to appreciate the benefits of longer high quality video communication experiences.

In conclusion, the research work presented in this thesis provide a significant contribution to make state-of-the-art HEVC based video encoding less complex, while keeping the coding efficiency intact. Reducing the computational cost of HEVC encoders evidently influence the content creators to incorporate HEVC into their existing media processing work flows and increases the rate adoption of HEVC encoding with the CE devices with limited processing power. In addition, making the HEVC bit streams less complex makes a direct influence on the green multimedia consumption, contributing to the efforts of reducing the carbon footprint caused by the media entertainment sector.

### 6.5 Future Work

This section identifies potential future work and research directions which can expand the work presented in this thesis. These can be summarized as follows.

- The fast encoding algorithm for HEVC proposed in Chapter 3 is designed to predict the CU size based on the content adaptive features. In this case, a potential future work would be to utilize a similar approach to predict the entire coding structure (i.e., PUs and TUs) for a particular CTU.

- The utilization of GPUs for parallel processing is a popular method to achieve real-time performance. In this case, the proposed algorithms can be used with parallel coding tools available within HEVC (i.e., Tiles, WPP) in conjunction with GPU technologies to achieve real-time encoding capabilities with less computational processing resources.
6.5. Future Work

- The proposed decoding complexity-aware encoding algorithm in Chapter 4 allows the HEVC encoder to prepare video bit streams that are optimal in terms of bit rate, distortion as well as decoding complexity. The proposed decoding complexity estimation models therein, can be integrated with the DVFS algorithms at the decoder to develop more efficient CPU frequency selection algorithms which are more accurate than the complexity prediction mechanisms utilized in the state-of-the-art DVFS algorithms.

- The resulting bit rate, video quality, and decoding complexity levels for the HEVC encoded bit streams are highly correlated with the video content. Moreover, modern GPPs, are equipped with multicore processing architectures that facilitate parallel processing. However, the proposed algorithms in Chapter 3 and 4, are not considering dynamic partitioning of HD, and UHD picture frames using Tiles and Slices which will facilitate parallel decoding of a video frame. In this context, a potential future work would be to extend the proposed algorithms to introduce content adaptive picture partitioning algorithms to allow efficient utilization of parallel processing architectures in modern CPUs.

- The experimental results presented in Chapter 5 illustrate that accurate rate controlling for dynamic and complex video contents is extremely difficult and state-of-the-art rate controlling algorithms (including the one adopted in HM 16.0) show considerable deficiencies in this area. However, as illustrated in Chapter 5, the controlling approach followed in proposed algorithm shows non-trivial improvements in both rate and decoder complexity controlling aspects. Therefore, another potential future work would be to extend these algorithms to introduce a content adaptive and accurate rate controlling algorithms for HEVC.

- The research work proposed in this thesis enables the encoding process to be reconfigured based on the encoder capabilities, user requirements and decoding complexity requirements. However, the proposed methods do not consider the networking conditions to dynamically adopt the coding structure as well as picture partitioning structure (slices, tiles etc.,) to suite the current bandwidth constraints. Therefore, another potential future work would be to utilize the proposed techniques to extend the encoding process to consider factors such as encoder capabilities, decoding complexity constraints, quality requirements and network conditions to introduce re-configurable encoding techniques.
Appendix A

List of Publications

The research work presented in this thesis have been presented in multiple conferences and are peer-reviewed resulting in 12 publications (8 accepted, 4 are currently under review). These are listed as follows,


- R. Perera, A. Fernando, T. Mallikarachchi, H. Kodikara Arachchi (Univ of Surrey, UK), and Mahsa Pourazad (UBC, Canada), "QoE Aware Resource Allocation for Video Communications over LTE Based Mobile Networks", in International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Greece, Aug. 2014.

- T. Mallikarachchi, A. Fernando, H. Kodikara Arachchi, "Fast Coding Unit Size Selection for HEVC Inter Prediction", in IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, USA, 9-12 January 2015.


• T. Mallikarachchi, D.S. Talagala, A. Fernando, H. Kodikara Arachchi, "A Feature Based Complexity Model for Decoder Complexity Optimized HEVC Video Encoding", in *IEEE International Conference on Consumer Electronic(ICCE)*, Las Vegas, USA, 8-10 Jan, 2017.


• T. Mallikarachchi, D.S. Talagala, A. Fernando, H. Kodikara Arachchi, "Decoding Complexity, Rate, Distortion Optimized HEVC Video Encoding", in *IEEE International Conference on Consumer Electronic(ICCE)*, Las Vegas, USA, 12-14 Jan, 2018. (under review)

• T. Mallikarachchi, D.S. Talagala, A. Fernando, H. Kodikara Arachchi, "Decoding Complexity-Aware Rate Controlled Video Coding for HEVC", in *IEEE International Conference on Consumer Electronic(ICCE)*, Las Vegas, USA, 12-14 Jan, 2018. (under review)
Bibliography


[44] EBU, “Digital video broadcasting (DVB); specification for the use of video and audio coding in broadcasting applications based on the MPEG-2 transport stream,” ETSI TS 101 154 V2.3.1 (2017-02).


[113] openHEVC, https://github.com/OpenHEVC/openHEVC.


