DECODER-COMPLEXITY-AWARE ENCODING OF MOTION COMPENSATION FOR MULTIPLE HETEROGENEOUS RECEIVERS

by

Mohsen Jamali Langroodi

B.Eng., University of Tehran, 2011

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

APPROVAL

Name:	Mohsen Jamali Langroodi
Degree:	Master of Science
Title of Thesis:	Decoder-Complexity-Aware Encoding of Motion Compensation for Multiple Heterogeneous Receivers

Examining Committee: Dr. Ted Kirkpatrick,
Associate Professor, Computing Science,
Simon Fraser University
Chair

____

Dr. Joseph Peters,
Professor, Computing Science,
Simon Fraser University
Senior Supervisor

____

Dr. Shervin Shirmohammadi,
Professor, Electrical Engineering and Computer Science,
University of Ottawa
Supervisor

____

Dr. Mohamed Hefeeda,
Professor, Computing Science,
Simon Fraser University
Internal Examiner

Date Approved: ____________ September 15 2014 ____________

Abstract

For mobile multimedia systems, advances in battery technology have been much slower than those in memory, graphics, and processing power, making power consumption a major concern in mobile systems. The computational complexity of video codecs, which consists of CPU operations and memory accesses, is one of the main factors affecting power consumption. In this thesis, we propose a method that achieves near-optimal video quality while respecting user-defined bounds on the complexity needed to decode a video. We start by formulating a scenario with a single receiver as a rate-distortion optimization problem and we develop an efficient decoder-complexity-aware video encoding method to solve it. Then we extend our approach to handle multiple heterogeneous receivers, each with a different complexity requirement. Our experimental results show that our method can achieve up to 97% and an average of 97% of the optimal solution value in single receiver and multiple receiver scenarios, respectively.

Keywords: Decoder complexity modeling, H.264/AVC decoding, H.264/SVC decoding, motion compensation

I would like to dedicate my thesis to my father for his continuous and immense support

"Always be yourself, express yourself, have faith in yourself, do not go out and look for a successful personality and duplicate it."

— Bruce Lee, *1973*

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisors Prof. Joseph Peters and Prof. Shervin Shirmohammadi for the continuous support of my M.Sc. study and research, for their patience, motivation, enthusiasm, and immense knowledge and at the same time the guidance to recover when my steps faltered. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my M.Sc. study.

I thank my fellow labmates and friends in Network System lab Group for the stimulating discussions, commenting on my views and helping me understand and enriching my ideas ,and also for all the fun we have had in the last years. Also my special thanks goes to my dear friend Erfan Sadeqi Azer for valuable discussions and helps in algorithmic part of the thesis.

Most importantly, none of this would have been possible without the love and patience of my family. My immediate family to whom this dissertation is dedicated to, has been a constant source of love, concern, support and strength all these years. I would like to express my heart-felt gratitude to my family.

Finally, I appreciate the financial support from Faculty of Applied Science at Simon Fraser University and my supervisors to make it possible for me to complete this road. My research was also supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) Grant.

Contents

Approval ii

Partial Copyright License iii

Abstract iv

Dedication v

Quotation vi

Acknowledgments vii

Contents viii

List of Tables x

List of Figures xi

1 Introduction 1

1.1 Motivation 2

1.2 Research Problem and Objective 3

1.3 Research Contributions 4

1.4 Research Publications 5

1.5 Thesis Outline 5

2 Related Work 7

2.1 Complexity Reduction on the Encoder Side 7

2.2 Complexity Reduction on the Decoder Side 8

2.3 Complexity Modeling of Decoder . . . 9
2.4 Power-Aware Video Encoding . . . 10
2.5 Power-Aware Video Decoding . . . 11

3 Background 13
3.1 H.264 Advanced Video Coding . . . 13
3.2 H.264 Scalable Video Coding . . . 16
3.3 Review of Complexity Modeling of Motion Compensation . . . 22
3.4 Algorithmic Background . . . 26

4 Methodology 30
4.1 System Model . . . 30
4.2 Formulation of Problem . . . 32
4.3 Dynamic Programming Method . . . 35
4.4 Solution Method for a Single Threshold . . . 36
4.5 Solution Method for Multiple Complexity Thresholds . . . 39

5 Implementation 43
5.1 Single Stream Scenario . . . 43
5.2 Multiple Stream Scenario . . . 44

6 Experiments and Analysis 47
6.1 Experimental Results for H.264/AVC . . . 48
6.2 Experimental Results for SVC Extension . . . 54

7 Conclusions and Future Work 61
7.1 Conclusion . . . 61
7.2 Future Work . . . 62

Bibliography 64

List of Tables

3.1 Different type of sub-pel used in the example of Figure 3.3 16

3.2 Number of basic operations for interpolation of a single pixel 23

3.3 Number of half and quad interpolations and number of memory accesses for a $m \times n$ block 24

5.1 AVC Encoder Configuration 44

5.2 SVC Encoder Configuration 45

6.1 OR improvements with β changing from 2 to 10 seconds in two different HGOP sizes 53

6.2 Average OR values over β_0 and β_1 with fixed β_2. 60

List of Figures

3.1 Different macroblock sizes in H.264 video coding standard 14

3.2 Interpolation of luma half-pel positions 15

3.3 half and quad interpolations for a 4×4 block with quad MV_x and half MV_y 17

3.4 Scalable video coding scheme in a heterogeneous environment 18

3.5 The basic types of scalability in video coding 19

3.6 hierarchical prediction structure in SVC 20

3.7 inter-layer prediction structure in SVC 21

3.8 Complexity modeling of H.264 motion compensation process 25

3.9 The graph used in sequence alignment algorithm 26

3.10 Dynamic programming approach in computing $f(i, j)$ 27

3.11 Dynamic programming approach in computing $g(i, j)$ 28

3.12 Shortest path from $V(0, 0)$ to $V(m, n)$ 28

3.13 Recursive approach in sequence alignment algorithm 29

4.1 Our system model in single receiver scenario [21] (re-used in accordance with ACM publishing agreement) 31

4.2 Our system model in multiple receiver scenario [21] (re-used in accordance with ACM publishing agreement) 31

4.3 An example of an initial solution for our heuristic method shown in a table. The red entries show the state chosen for the corresponding macroblock . . . 37

4.4 Multiple receivers scenario summarized in a flowchart with a base layer and two enhancement layers. There is an objective function in each rectangle and lower bound solution that corresponds to the constraints. Such constraints of the optimization problem for each iteration is shown in a rectangle shape . 42

6.1 Trade-offs between *CE* and *CRT* for dynamic programming method. The dashed lines show *CRT*s and bars show *CE*s. 48
6.2 Error cancellation phenomenon that occurs in complexity value rounding . . 49
6.3 State distribution over different modes and interpolation for the Parkrun sequence which shows the number of times that each state has been chosen in the solution. Different bars show different $\log_2 S$ in each state which starts from 10 at the left and ends with 17 at the right. 50
6.4 Trade-offs between *CRT* and *OR* for different HGOP lengths. 52
6.5 Impact of β on the performance of the hybrid algorithm. The dashed lines are for HGOP size 15 and the solid lines are for HGOP size 2. 54
6.6 Average OR reduction with β_1 changing from 3 to 0.6 in two different GOP sizes. 55
6.7 Optimality Ratio for different β_0, β_1, and β_2 for GOP size 5. 56
6.8 Optimality Ratio for different β_0, β_1, and β_2 for GOP size 5. 57
6.9 Optimality Ratio for different β_0, β_1, and β_2 for GOP size 9. 58
6.10 Optimality Ratio for different β_0, β_1, and β_2 for GOP size 9. 59

Chapter 1

Introduction

Mobile phones have been undergoing a tremendous evolution over the last two decades, starting from simple cell-phones with only call services towards smart devices offering modern services such as mobile Internet, maps and geo location, multimedia streaming, health-care services, and so on. There has been a booming growth in the area of video streaming in Internet daily usage. Consumers have a great interest in watching movies on-demand and viewing video content on every possible device: High definition TV, video players, PCs, laptops, netbooks, and smartphones. People enjoy the convenience of online video and the high-resolution features that are now available universally. It is estimated that video streaming of Netflix and YouTube content is responsible for 50.31 percent of overall Internet traffic in North America during the peak part of the day according to the Global Internet Phenomena Report [3]. Companies such as Netflix, Comcast, Time Warner, Hulu, Amazon, among others, are working toward satisfying this growing demand.

Video streaming is also taking another step beyond the entertainment area into education [36, 15] and health-care [5, 45], among other areas. Educators can take advantage of documentaries, movies, and instructional videos to facilitate the learning process. Students have a growing tendency to learn through moving images rather than text alone which has been expressed in mobile learning. Teachers are incorporating educational applications to improve the interactivity with the course content, increase learner engagement, and help learners remember and retain concepts more easily [15].

Health-care providers have made important advances in video usage as well. Physicians adopt video conferencing to confer with patients remotely from medical offices. Many companies such as VIDIZMO [1] and StreamingWell [2] incorporate video and digital content into their health-care solutions. One of the applications of video streaming is to help patients through mobile remote monitoring of chronic diseases in order to record their own health measures and send them electronically to physicians for health-care cost reduction. It can also be a key to inform patients about their medical care which enables them to maintain their health more efficiently [47].

Apart from the technical applications mentioned above for video usage, video conferencing and live broadcasting services can be generally used to bring people at different sites together. This technology can be simply used for a conversation between people in private places (point-to-point) or to connect several sites in large meeting rooms at multiple locations (multipoint). Besides remote video conferencing of meeting activities, allied video conferencing technologies are becoming a tool to share documents and display information on virtual whiteboards. Smart-phone customers can benefit from such services too. However, these services have a different requirement than video on demand services: they must stream the same video simultaneously to multiple receivers with different bandwidth and resource requirements. To address the different requirements in these settings, solutions such as Scalable Video Coding (SVC) [39, 49] have been proposed and deployed. SVC creates one or more subset of bitstreams, known as layers, each of which has a lower bit-rate than the original video stream, and SVC offers different types of scalability including temporal, spatial, and SNR/quality scalabilities. Each receiver can select the video layer(s) that it needs depending on its capabilities and network bandwidth.

1.1 Motivation

In the current state of the art, most smartphones are powered by lithium-ion batteries [52]. There is some new progress to produce more power-efficient batteries using nanotechnology [4], hybrid electrochemical characteristics [50], and many more, however its technology growth is generally limited and not keeping pace with the new features added to mobile devices that are loaded with a huge set of capabilities and functionalities.

A crucial problem for hand-held devices such as netbooks and smartphones is sustaining a trustworthy long battery life given the large amount of energy required for video decoding and rendering. Furthermore, as video codecs become more advanced with higher computational complexity, their power consumption increases. This can be seen in the H.264 video coding standard [22] which requires more power than its predecessor, and the HEVC specification [9] which is more complex and power-hungry than H.264.

Among various factors that affect power consumption, computational complexity, which consists of CPU operations and memory accesses, is one of the main factors. In other words, primary low level hardware operations are the most important factors that contribute to energy consumption of mobile devices. Such factors can be used to estimate actual power expenditures of hand-held devices indirectly from *time complexity*. The term ‘time complexity’ refers to the amount of time that it takes to decode the video sequence expressed as milliseconds or clock cycles. As described in [11, 10, 41, 17], in CMOS circuit design the power consumption of computational operations is proportional to the number of processor cycles per second (processor clock frequency) using dynamic voltage scaling (DVS) technology. It can be concluded from [17] that power consumption can be indirectly estimated from the time complexity of the device based on average power-dissipation per clock cycle. It should be noted that for simplicity the computational complexity of H.264 decoding is quantitatively represented based on the basic operations and time complexity in this thesis.

Despite the fact that a high quality video is desirable for mobile customers, power limitations of portable devices will force them to adopt a method for encoding video streams that optimizes the trade-off between the video quality of a stream and the decoding complexity in the receiver. Such a method would be a vital contribution for video-on-demand content providers, such as Netflix and YouTube, who could use our method to enable longer playing times for their mobile customers. In a same way, for mobile receivers in a video conferencing or live broadcasting session, we need a method that can encode a video in a way that optimizes the trade-offs between video quality and multiple decoding complexity limits based on the video layer(s) to which the different receivers subscribe.

1.2 Research Problem and Objective

In this thesis, we first address the video on demand scenario by developing a method for video encoders to maximize the video quality of a single video stream while guaranteeing that the complexity needed to decode the video does not exceed a specific user-defined threshold. We formulate this single threshold scenario as a rate-distortion optimization problem and present an efficient decoder-complexity-aware video encoding method to solve it. Our experiments with the H.264/AVC video codec show that our method can achieve up to 97% of the optimal solution value. A preliminary version of this part of our work was presented in [20].

Then we extend our approach from the single threshold scenario to the multiple receiver scenario of video conferencing and live broadcasting, taking into account the different complexity restrictions of mobile users. Our method for this scenario is a decoder-complexity-aware video encoding method that satisfies the decoding complexity requirements of multiple users while maximizing the overall video quality of all layers. Our experiments with H.264/SVC show that an average of 97% of the optimal solution value can be achieved.

In our work, we focus on the motion compensation process, including motion vector prediction and interpolation, in which macroblocks coded with different inter prediction modes have different decoding complexities. The motion compensation process accounts for as much as 38% of the decoding complexity [19] and is the single largest component of computation-based power consumption.

1.3 Research Contributions

This thesis presents a theoretical and offline approach for decoder complexity-aware encoding that uses decoding configuration and parameters to ensure the complexity needed for decoding the video on mobile device does not exceed the requested threshold.

The new contributions included in this thesis are as follows.

- Exploiting the trade-off between complexity, video quality, and bit-rate
- Formulation of the complexity-rate-distortion (C-R-D) problem for the decoder
- Proposing a memory-efficient method for solving the C-R-D problem in a feasible amount of time
- Proposing a new and fast heuristic method to solve the C-R-D problem for a GOP (group of pictures) with negligible time overhead on the encoding process
- Combining the last two methods to create a comprehensive hybrid method for addressing decoder complexity-aware encoding for the single receiver scenario
- Proposing a new recursive method to address decoder complexity-aware encoding for the multiple receivers scenario with a feasible time overhead

1.4 Research Publications

In addition to the above contributions this research has led to the scholastic achievements and publications listed below.

Journal Paper:

- Mohsen Jamali Langroodi, Joseph Peters, and Shervin Shirmohammadi. Decoder-Complexity-Aware Encoding of Motion Compensation for Multiple Heterogeneous Receivers, in *ACM Transactions on Multimedia Computing, Communications and Applications (TOMM)*, (revised version in review), 2015 [21].

Conference Publications:

- Mohsen Jamali Langroodi, Joseph Peters, and Shervin Shirmohammadi. Complexity aware encoding of the motion compensation process of the H.264/AVC video coding standard, in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, pages 103-108, ACM, New York, NY, USA, 2013 [20].
- Mehdi Semsarzadeh, Mohsen Jamali Langroodi, Mahmoud Reza Hashemi, and Shervin Shirmohammadi. Complexity modeling of the motion compensation process of the H.264/AVC video coding standard, in *Int. Conf. on Multimedia and Expo (ICME)*, pages 925-930, IEEE, 2012 [40].

1.5 Thesis Outline

The rest of this thesis is organized as follows.
Chapter 2 - Related Work: discusses existing approaches to related research topics and compares them to our approach
Chapter 3 - Background: essential multimedia and algorithmic background are reviewed
Chapter 4 - Methodology: our methodology and approaches are described in detail
Chapter 5 - Implementation: experimental setup and implementation details are explained
Chapter 6 - Experiments and Analysis: presents our experimental results and performance evaluations

Chapter 7 - Conclusions and Future Work: summarizes our work and discusses future research possibilities

Chapter 2

Related Work

In the following sections, we will discuss the related work in each category of complexity related video coding and decoding. Then, we explain the drawbacks associated with the discussed related work and also compare them to our approach to clarify how our proposed approach differs from them.

Power consumption restrictions have recently generated a major interest in complexity reduction, modeling, and control in video coding applications. Some of the previous research efforts are dedicated to low-complexity video encoding on the transmitter side of H.264/AVC standard components [32, 53, 51, 12] while some others have focused on the same concept on the receiver side [46, 13, 23].

2.1 Complexity Reduction on the Encoder Side

This first category is about decreasing the computational complexity of the encoding process while maintaining the same video coding performance.

In [53], a weighted-window complexity model based on QP (quantization parameter) decisions and a MAD (Mean Absolute Deviation) prediction model is proposed to decrease the computational complexity of Macroblock-layer (MB-layer) rate control. The relationship among QP, MAD, and the total number of coded bits is analyzed to decide the size of the QP window. The final QP is computed using the MB-layer bits allocation and the previous macroblock's QP with consideration of the remaining available bits.

In [51], the conventional H.264 rate distortion optimization algorithm is modified for error-prone video transmission environments. The rate-distortion (RD) cost function for motion estimation is improved by means of lost packet concealment to show better PSNR (Peak signal-to-noise ratio) on video output. The authors have analyzed the complexity of their proposed rate-distortion optimization and have shown that the actual computational complexity is noticeably decreased.

A new fast and optimized algorithm to perform motion estimation and mode decision for the encoding of depth sequences in free viewpoint video is proposed in [12]. Free viewpoint video uses a high number of viewpoints of a video which are used to reconstruct a 3-D scene. Each viewpoint is coded as a traditional sequence called texture together with a gray-scale depth that shows the 3-D characteristic of the video. In this work, the authors have analyzed the motion characteristics of video plus depth and exploited the redundancy that exists among textures since they are representing the same scene captured by a traditional camera. They took advantage of this redundant information to considerably reduce the computational burden on the encoder.

2.2 Complexity Reduction on the Decoder Side

The second category of related research focuses on the decoder side complexity burden. In fact, many heuristic ideas are proposed to facilitate receiver side computations. In [46], the authors have proposed a low complexity deblocking filter algorithm for multi-view video coding by investigating view correlations in the motion skip coding mode. They claimed that if the boundary strength (BS) value, which is a one of criteria for smoothing the edges in deblocking filter component can be directly copied from the associated reference macroblock in a neighbor view, then a significant amount of deblocking filter computations can be eliminated with negligible quality degradation.

Another deblocking filter heuristic idea is proposed in [13] to decrease the amount of unnecessary BS decisions leading to a computational complexity reduction for this component. The authors exploit the number of operations involved in each BS type in the deblocking filter process and also apply a newly designed filter which is much simpler than the conventional filter to each BS type. A fast prediction algorithm is also proposed to determine the BS of successive Lines of Pixels (LOP) using the BS of the first LOP since they possess similar characteristics and amounts of blocking artifacts.

In the same area but with a different platform, a fast smoothing decision algorithm is proposed in [23] to reduce the computational complexity of intra prediction in an HEVC decoder hardware architecture. More precisely, the proposed architecture employs a fast smoothing decision algorithm to reduce the complexity of intra prediction hardware. This proposed piece of hardware is a shared operation unit that shares adders for computing common operations of smoothing equations to eliminate computational redundancy and reduce the number of the execution cycles of the component.

2.3 Complexity Modeling of Decoder

There have been numerous attempts to create comprehensive modeling schemes for decoders to pave the way of establishing a decoder complexity controller infrastructure. In [29], the authors modeled the whole H.264/AVC video decoding process by determining the fundamental operation unit, called Complexity Unit (CU), in each decoding module. Each module, based on its individual properties has its own CU such as bit parsing, macroblock data structure initialization unit, half-pel interpolation, and so on. Such CUs are determined empirically for a fixed implementation and then a product of the average number of cycles required by each CU over a frame or a GOP times the number of CUs reported in a decoder module as the time complexity. To validate, the proposed complexity model is tested on both Intel and ARM hardware platforms to decode H.264/AVC bitstreams.

In a similar way, a complexity model of the motion compensation component of a video decoder based on the number of cache misses, the number of y-direction interpolation filters, the number of x-direction interpolation filters, and the number of MVs (Motion Vector) per MB was presented in [25]. It is worth mentioning that this work is similar to the model proposed in [40]; in both models, there is a weight associated with each number of complexity units calculated in a training phase. Firstly, the number of each CU is extracted in a training pool of video sequences during the encoding phase, and then each bitstream is decoded on the encoder side to find the best fitting weights for each CU to establish an estimation model.

The authors of [25] have also performed a complexity analysis of the entropy decoder module for both context-based adaptive variable length coding (CAVLC) and universal length coding (UVLC) which is supported in all profiles of the H.264/AVC codec. The model has been proposed as a function of the number of CAVLC executions, the number of trailing ones, the number of remaining non-zero quantized transformed coefficients, and the number of run executions. The H.264/AVC encoder integrated with the proposed complexity control scheme can generate a bit stream that is suitable for a receiver platform with a power/hardware constraint [26]. The main difference between the work in [26] and our proposed method is the components that have been studied. In addition, in contrast with the models proposed in [26] and [25], the model used in this thesis, based on [40], can be utilized in any desired implementation, both hardware and software.

Recently, an analytical energy consumption and decoding performance model similar to the model in [40] was presented in [7]. The model achieves a balance between the abstract high level and detailed low level based on the video bit-rate and clock frequency. Parameters are extracted and evaluated to determine their relationships to decoding speed and thereby to energy consumption. Nevertheless, the authors in [7] have only focused on complexity modelling of the H.264/AVC standard, whereas in this thesis a generic decoder complexity-aware encoding has been developed based on the proposed complexity model in [40].

2.4 Power-Aware Video Encoding

The main concern in power-aware video encoding is the power shortage in encoding devices such as portable cameras and sensors that capture and record long video sequences.

One of the most recent efforts concentrated on power-aware video encoding while maximizing video quality under specific encoder constraints. In [27], the authors tackled the power consumption of surveillance cameras without power lines by proposing an optimal video coding configuration to enhance video quality based on the remaining charge in the battery. Stochastic event characteristics and the nonlinear discharge of the battery are considered to change the behavior of the power expenditure. To be more exact, whenever a suspicious event occurs within the camera's scope making the sum of the absolute difference (SAD) between the captured image and background image larger than a predefined threshold value, the camera will capture the video with higher complexity encoding leading to higher video quality. The proposed scheme also estimates the remaining event active duration to set the IDR (Instantaneous Decoding Refresh) period to increase the video quality as much as possible given the battery energy constraints.

In [14] and [28], the conventional rate-distortion model was extended to a new model for controlling power while optimizing the bit rate and visual quality of a video. In [14], the prediction stage was optimized for energy-constrained compression using a new framework called RDE (Rate-Distortion-Energy) optimization by adding a new energy dimension. This dimension is derived from a wide range of encoding parameters, such as number of iterations, and data size, which influence the demanded energy while some others do not. Simply put, the central idea of this paper is that the demanded energy and encoder cost might be differently affected by different parameters. The authors have investigated this issue by measuring energy and encoder cost on comprehensive training data sets which cover all possible parameters.

In [28], a new delay-power-rate-distortion model was proposed based on encoding parameters such as macroblock coding mode, search range, number of reference frames, and quantization parameter. The purpose of this model is to exploit the relationships among video encoding time, power, bit rate, and distortion. The authors have done so by determining encoding time of the above mentioned parameters independently with experimental results, and then computing the correlated function theoretically. An encoding time and power model is also provided in their formulation. At the end, the finalized model is empirically verified and validated to show its accuracy. The main difference between this research and ours is that we focus on the power consumption of the decoder instead of the encoder.

2.5 Power-Aware Video Decoding

Power-aware video decoding takes into account the computational complexity consumed during the decoding phase in portable devices.

In this regard, a power-aware H.264 video decoding method to satisfy decoder real-time requirements was recently proposed in [43]. This method selects appropriate temporal and quality layers based on their computational complexities to satisfy decoder real-time constraints. In contrast, the goal of our method is to satisfy real-time constraints on both the encoder and decoder sides while ensuring that decoding complexities do not exceed thresholds requested by user devices. In addition, instead of choosing layers, the method that we propose for the scenario of multiple complexity constraints adapts the complexity of each layer to its corresponding user request and ensures that the decoding complexities do not exceed specified thresholds.

One of the most recent works, [16], has focused on complexity-aware adaptive scalable video coding similar to our multiple receiver approach. The method in [16] uses a dynamic transition for Region-of-Interest (ROI) that adaptively changes encoding and decoding complexity using various preprocessing parameters such as standard deviation, kernel matrix size, and number of applied filters. The method presented in [16] is not as flexible as our approach in adjusting the video complexity, since it only considers the Region-of-Interest properties. As a matter of fact, the video encoder is forced to use the ROI feature during the encoding phase to perform complexity-aware video coding, whereas in our approach there is not such limitations.

Chapter 3

Background

In this chapter, we explain some necessary background about the H.264 video codec, including the scalable video coding standard, and also essential algorithmic background needed in our proposed method.

3.1 H.264 Advanced Video Coding

The content of this chapter is derived from [34] therefore for more information the reader is referred to this book. In general, multimedia data, including video, do not follow a uniform pattern. Level of activity, detail, motion speed, and direction are factors that impact the encoding process, hence each frame in the video is partitioned into smaller pieces called macroblocks. In the motion estimation process, using smaller blocks leads to more compression and higher detail. H.264 allows the encoder to do motion estimation on 16×16, 16×8, 8×16, and 8×8 blocks and these blocks can be further partitioned into blocks as small as 4×4 for even better effectiveness if necessary as shown in Figure 3.1. A motion vector is associated with each block to transfer the motion data of the picture to the decoder. The error in motion prediction is called residual and it is defined as the difference between the reconstructed picture and the original one. Motion vectors accompanied by residuals are compressed and sent to the decoder so that it would be able to recover the exact video block information on the receiver side.

The improved flexibility in H.264/AVC in comparison to former video coding standards is one of the strong points of this newly proposed codec. This flexibility has been well studied at the macroblock level, and is supported by various intra prediction modes as

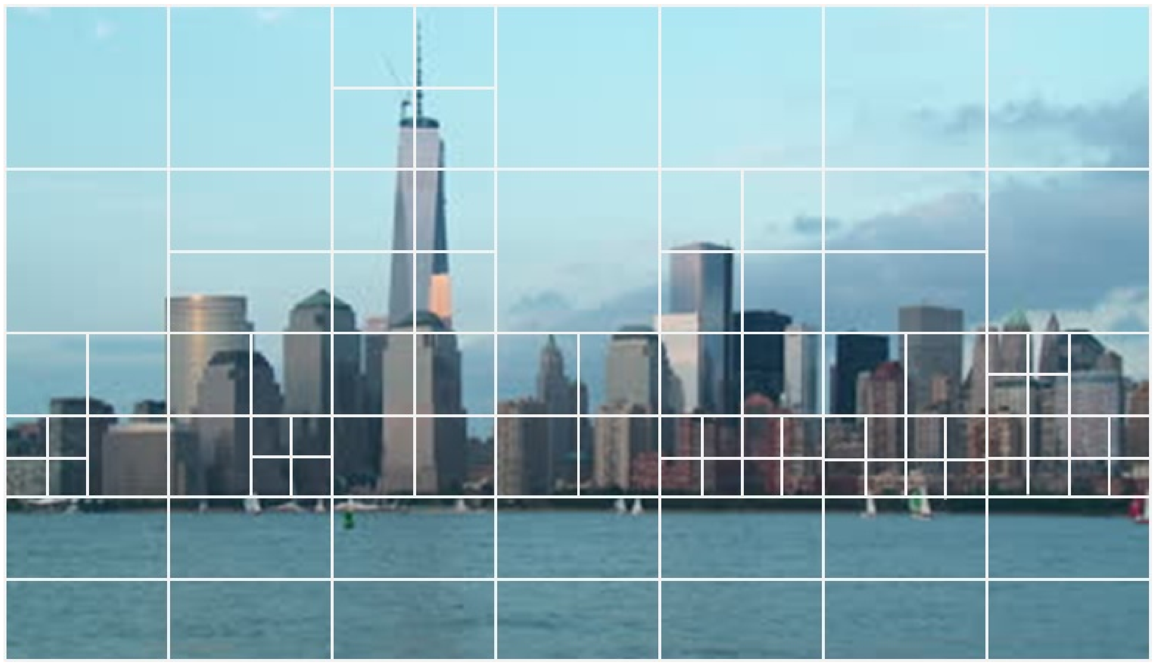


Figure 3.1: Different macroblock sizes in H.264 video coding standard

well as fine-grain macroblock partitioning options for motion-compensated prediction which did not exist in older video coding standards. However, a significant amount of progress has been made in H.264/AVC for flexible prediction methods at a picture/sequence level. In contrast to older video coding standards, for the sake of coding efficiency, the coding technique and display order of pictures is completely changed. Furthermore, a picture can use any arbitrary preceding pictures as a reference frame for motion-compensated prediction independent of the coding type of the corresponding slices.

When sub-pel mode is enabled in the motion estimation process, motion vectors are reported at quarter precision of the distance between luminance samples of the picture. In a case where the motion vector points to an integer-sample position, the prediction signal will be retrieved from the reference picture directly; otherwise the corresponding sample is computed using a specific interpolation method to generate non-integer positions. Luma and chroma components are interpolated separately. The prediction values at half-sample positions are calculated by applying a one-dimensional 6-tap FIR filter horizontally and vertically using the formula

$$b = round((E - 5F + 20G + 20H - 5I + J)/32)) \tag{3.1}$$

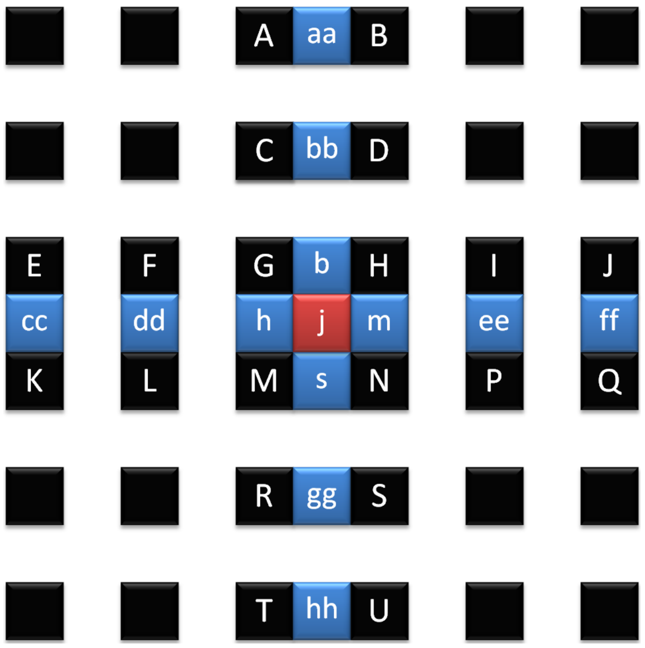


Figure 3.2: Interpolation of luma half-pel positions

where b is the half-pel interpolated pixels, and, F, G, H, I and J are the neighboring integer pixels as demonstrated in Figure 3.2. The same calculation occurs for h, m, and s half-pel pixels, however h and m use vertical neighbor integer pixels. Eventually, pixel j is interpolated from neighbor half-pel pixels (cc, dd, h, m, ee, and ff). The quad-pel pixels located at positions that are a quarter of unit, are generated by averaging samples at integer and half-pel positions. For chrominance interpolation, there is no difference between quad and half pixels. In fact, each interpolated sample is generated by engaging a 4-tap filter on four neighboring chrominance pixels with integer precision, using the following formula.

$$a = round([(8 - d_x) \cdot (8 - d_y)A + d_x \cdot (8 - d_y)B + (8 - d_x).d_yC + d_x.d_yD]/64) \qquad (3.2)$$

Here A, B, C and D are the neighboring chrominance pixels in integer precision, a is the interpolated sample, and d_x and d_y are the distance between a and A, in pixels.

Figure 3.3 illustrates an example of fractional sample luma interpolation when the x- and y-components of the motion vector are in quad and half precision respectively. Black pixels represent integer pixels, and for other pixels their resolution in the horizontal and vertical direction are shown with I, H and Q symbols which represent Integer, Half and Quad precision, respectively. Pixel symbols in Figure 3.3 are interpreted in Table 3.1.

	Precision in X-axis	Precision in Y-axis
HI	Half	Integer
QH	Quad	Half
HH	Half	Half
IH	Integer	Half

Table 3.1: Different type of sub-pel used in the example of Figure 3.3

According to Figure 3.3, in order to obtain a QH sub-pixel, an IH and an HH pixel are required. Each IH pixel is generated by using the 6-tap filter of Equation (3.1) on its 6 vertical surrounding integer pixels. In the same way, each HH pixel is produced by using the 6-tap filter on its 6 vertical surrounding HI pixels. In the same way but in a different direction, each HI pixel is extracted by applying the 6-tap filter to its 6 horizontal surrounding integer pixels [34].

3.2 H.264 Scalable Video Coding

The Scalable Video Coding amendment (SVC) of the H.264/AVC standard provides network-friendly scalability at a bit stream level with a modest increase in decoder complexity relative to single-layer H.264/AVC. The output of the video encoding process is a bit-stream consisting of the concatenation of bit-streams for several layers; the goals are to adapt the video stream according to bandwidth constraints and to be a solution for congested networks. In addition to bit rate adaptation, SVC can fit the video stream into a specific format, perform power adaptation, provide graceful video degradation in error-prone environments, and perform lossless conversion of quality-scalable SVC bit streams to single-layer H.264/AVC bit streams. These functionalities have created a tremendous improvement in transmission and storage applications. SVC has achieved these improvements in coding efficiency with a wider range of supported scalabilities than the ones introduced in prior video coding standards. This section is based on the content of an invited paper [39].

The motivation for scalable video coding originates from the increasing capabilities of electronic devices and the expanding usage of networks and distributing systems that have unreliable connection quality. In particular, the most important applications of video encoding and decoding are in the Internet and wireless networks. Video transmission in such

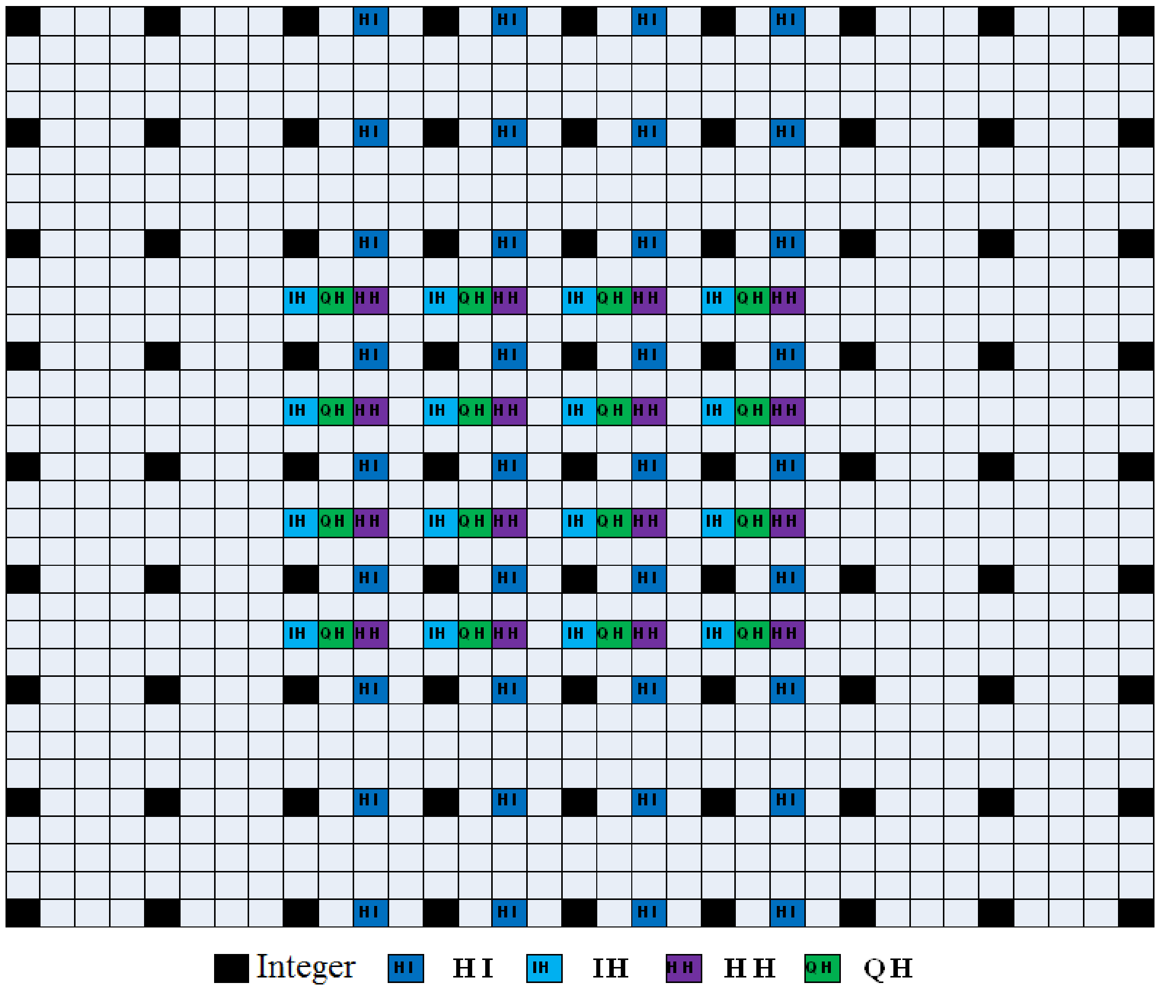


Figure 3.3: half and quad interpolations for a 4×4 block with quad MV_x and half MV_y

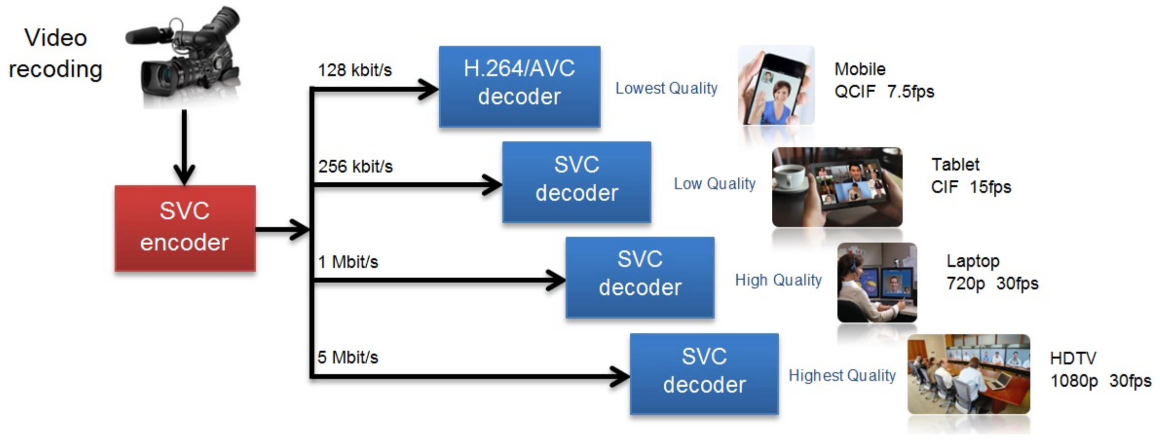


Figure 3.4: Scalable video coding scheme in a heterogeneous environment

systems, which are not error-free environments, can be dealt with using scalability features. Furthermore, in some applications video content is sent to a variety of decoding receivers with heterogeneous display and computational capacities (see Figure 3.4). One of the primary goals of such systems in these diverse environments is flexible adaptation of the content that only has to be encoded once, while enabling the integration of the encoder and decoder products from different sources.

SVC standardizes the encoding of a high-quality video bitstream that consists of one or more subset bitstreams. Each subset can be decoded hierarchically with a reconstruction quality similar to the original version encoded by the existing H.264/MPEG-4 AVC design with the minimum data usage overhead. The subset bitstream is derived by dropping packets from the larger bitstream. A subset bitstream can represent the following types of lower resolution and perceptual visual quality.

- Temporal scalability (frame rate): in this scalability, the frame rate of the video will change depending on the layer that the user chooses. It can be done by dropping a complete frame from the bitstream.

- Spatial scalability (picture size): the video is coded at multiple picture sizes. Macroblocks for pictures in higher resolutions can be predicted from previous reference frames of the same resolution and can also use lower resolution frames in the same temporal layer in order to reduce the bit rate and improve coding efficiency.

- SNR/Fidelity scalability (Quality): the video is coded at a certain spatial resolution

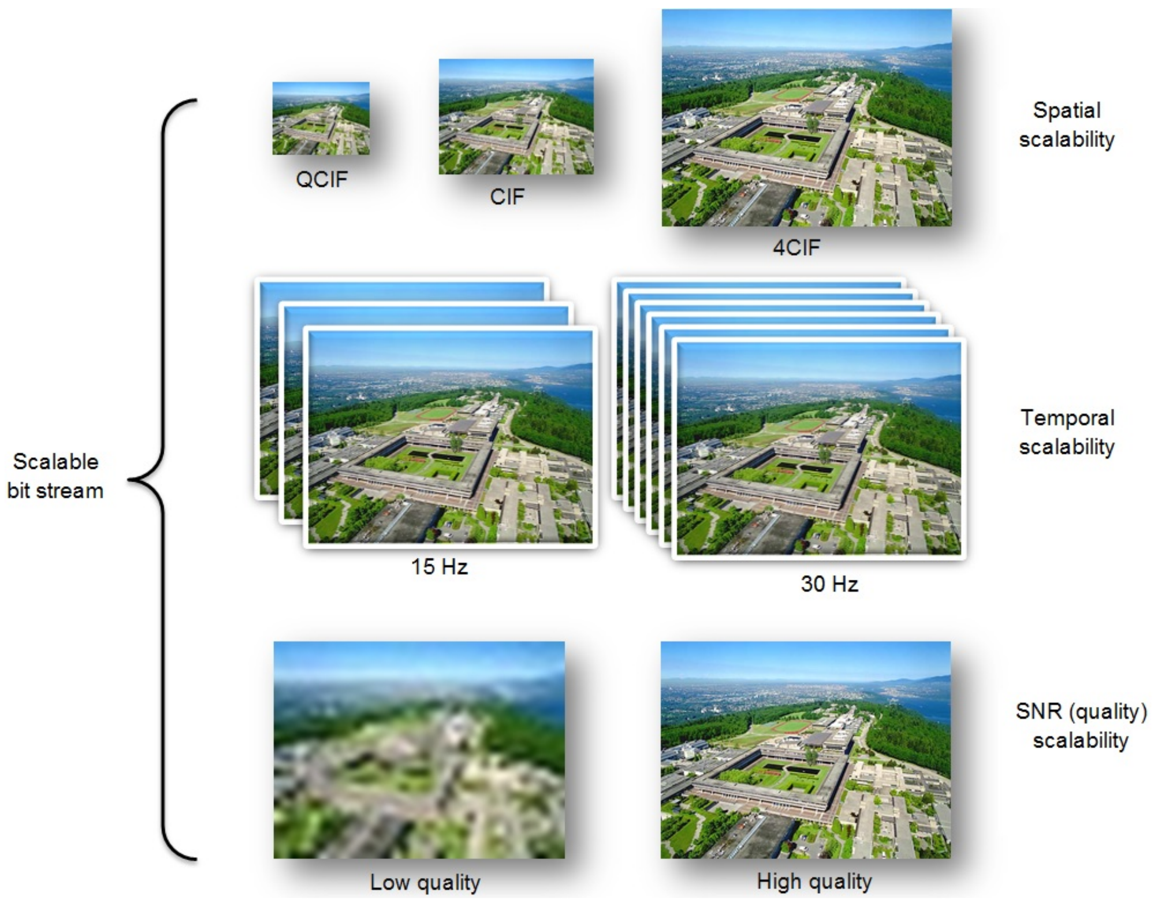


Figure 3.5: The basic types of scalability in video coding

but with different qualities. SNR scalability can be considered to be a special type of spatial scalability with identical picture sizes. In this modality, different qualities can be achieve by requantizing the residual texture signal in the enhancement layer and modifying the quantization step size.

- Combined scalability: a combination of the 3 modalities described above.

All type of scalabilities are depicted in Figure 3.5.

A hierarchical prediction structure with four layers is depicted in Figure 3.6. The first picture of a video sequence is intra-coded as the IDR picture shown as a black bar in Figure 3.6. A picture is referred to as a key picture when all previously coded pictures antecede this picture in display order. As illustrated in Figure 3.6, all pictures that are temporally located between the two consecutive key pictures are considered to be a group of

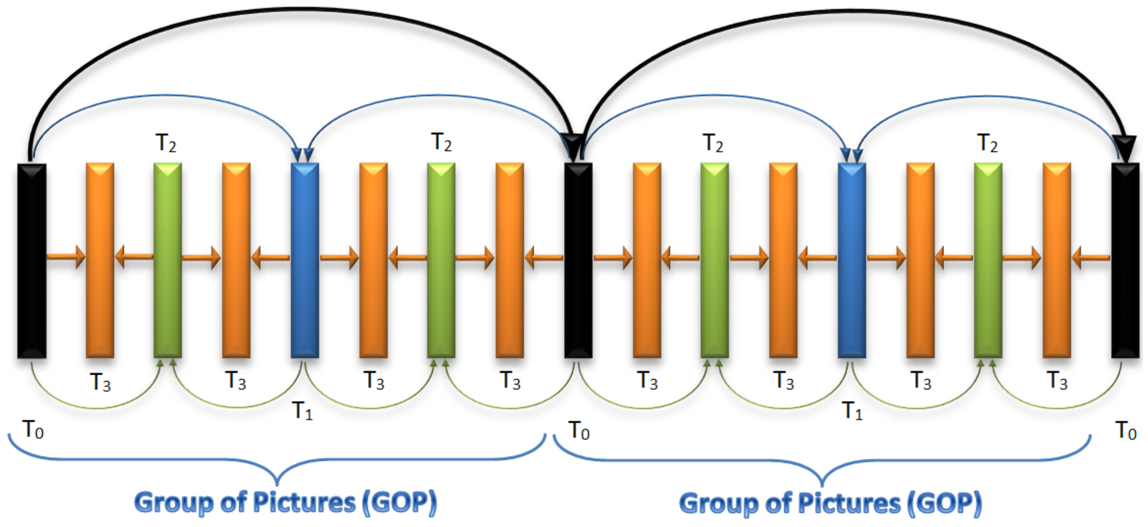


Figure 3.6: hierarchical prediction structure in SVC

pictures. The key pictures are not necessarily intra-coded; they can be coded dependently using previous (key) pictures as references for motion-compensated prediction (inter-coded). The remaining pictures of a GOP are recursively predicted as illustrated in Figure 3.6. One of the innovative ideas in SVC is the hierarchical B or P picture structure that will enable us to efficiently implement temporal scalability with dyadic temporal enhancement layers.

In order to create a temporal scalability feature, a set of corresponding temporal access units should be partitioned into a base layer and one or more enhancement layers with the following property. In Figure 3.6, the frames labelled T_0 represents temporal frames for the base layer, and T_n, $n \geq 1$ shows the n_{th} enhancement layer. Then the bit stream that is achieved by removing all the access units of T_k, $k > m$, will create temporal layer m for the decoder.

Spatial and quality scalable coding are also supported in SVC through the conventional approach of multi-layer coding which has been utilized before in previous standards. In each spatial layer, in addition to motion-compensation prediction and intra prediction as it is employed for single-layer coding, SVC provides another prediction method called inter-layer prediction (see Figure 3.7). In this method, statistical dependencies and redundancy between the layers is taken into account to improve the efficiency of the motion compensation process in enhancement layers.

Basically inter-layer prediction targets the reconstructed samples of the lower layer signal

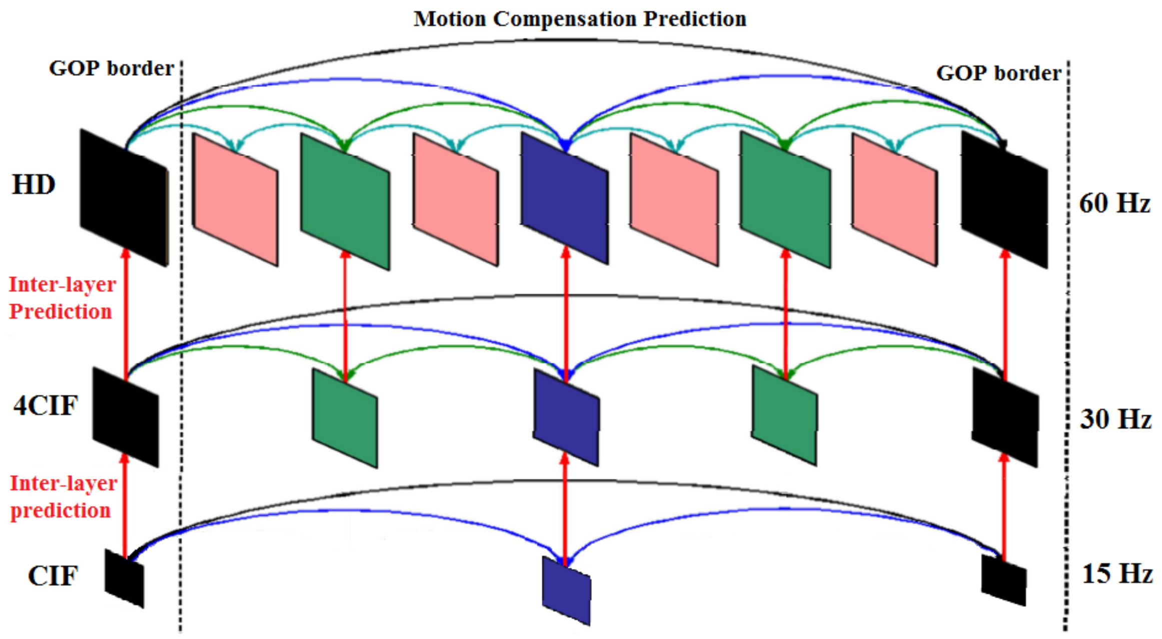


Figure 3.7: inter-layer prediction structure in SVC

with an appropriate resolution to use for upper layers. There are two options for obtaining prediction signals; either they are formed by applying motion estimation to the up-sampled version of the reconstructed lower layer signal which could be the base layer or an enhancement layer, or by averaging such an up-sampled signal with the signal predicted from the same spatial layer. It seems that the reconstructed lower layer signals represent a perfect sample of lower layer data, but in fact they are not necessarily the most suitable reference for prediction. Usually, the inter-layer predictor should consider both temporal and inter-layer prediction to make a logical decision in terms of bit rate and coding effectiveness. Such dependency on temporal layers will be exacerbated for sequences with slow motion and high spatial detail structure; the temporal prediction signal generally exhibits a better approximation of the original data than the upsampled lower layer reconstruction. A few more spatial scalable coding techniques exist in the standard which are more advanced in this regard such as prediction of macroblock modes with associated motion parameters, and prediction of the residual signal. The reader is referred to [39] for more detailed information.

3.3 Review of Complexity Modeling of Motion Compensation

In this section, we describe the complexity modeling scheme that we have utilized in this thesis to predict decoder processing time accurately. In the motion compensation process, in addition to the reconstruction process of pixels, the interpolation accounts for a significant share of the computational complexity of the motion compensation process. Macroblock size and sub-pel precision are the factors that make the bit stream more complicated to decode on the receiver side. The computational complexity is directly related to the amount of interpolation performed on the decoder side. In our previous work presented in [40], a generic complexity model based on an algorithmic analysis of the motion compensation process is proposed which considers the influence of mode decisions and interpolation on the computational complexity of the decoder.

The motion compensation process of the H.264/AVC standard is performed for each inter coded macroblock. The MC module gets residual data and its corresponding motion vector as input. For each motion vector value, the corresponding reference block is read from the frame memory. An interpolation operation may be performed on this block if the motion vector has half or quad precision. Finally, the residuals are added to the interpolated block to reconstruct the corresponding block.

For the luma pixels, the interpolation is applied by engaging a specific Finite Impulse Response filter on neighboring pixels as shown in Equation (3.1). If the motion vectors have quad precision, then the quad interpolation will be computed after half-pel interpolation from the resulting half pixels using a linear interpolation operation showed in Equation (3.2).

Essentially, the basic operations in motion compensation such as addition, multiplication, shift, and memory access contribute to the complexity of the hardware and software implementation. In each filter operation, the number of basic operations involved depends on the type of interpolation and the size of the macroblock. The Computational Complexity (CC) is therefore equal to the total numbers of basic operations multiplied by their corresponding weights:

$$CC = \sum_{i=1}^{n} P_i \cdot W_i \tag{3.3}$$

where P_i represents the number of i^{th} basic operation and W_i is its corresponding weight. P_i parameters can be defined according to the algorithm and W_i is adjusted based on the implementation and platform specification. Total Computational Complexity of a video sequence (TCC) can be divided into time complexity and space complexity categories as follows.

$$\begin{aligned} TCC &= TimeComplexity + SpaceComplexity \\ TimeComplexity &= \alpha \cdot N_{Sum} + \beta \cdot N_{Mul} + \gamma \cdot N_{Shift} \\ SpaceComplexity &= \mu \cdot N_{Mem} \end{aligned} \tag{3.4}$$

Here α, β, γ, and μ are the weight parameters, while N_{Sum}, N_{Mul}, N_{Shift}, and N_{Mem} are total numbers of additions, multiplications, shifts, and memory accesses, respectively. These weights are not pre-defined values, rather they are trained with a set of pre-encoded bit streams which will be explained later. The numbers of basic operations (i.e. N_{Sum}, N_{Mul}, N_{Shift}, and N_{Mem}) are determined based on macroblock mode, size and the precision of its motion vectors. According to the MC algorithm of H.264/AVC, the interpolation operation for each macroblock is applied on its $m \times n$ blocks of size 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 separately. By analyzing (3.1), (3.2) and the MC algorithm, we can find the number of each basic operation shown in Table 3.2.

$TimeComplexity$		N_{Sum}	N_{Mul}	N_{Shift}
Luminance	Half-pel Interpolation Complexity	5	4	1
	Quad-pel Interpolation Complexity	1	0	1
Chrominance	Interpolation Complexity	7	8	1

Table 3.2: Number of basic operations for interpolation of a single pixel

It should be noted that the total number of memory accesses (N_{Mem}) is computed based on the block size and the precision of its motion vectors, hence a pixel level formula cannot be defined for this operation. In order to find the time complexity at block level for N_{Mem} as well, we need to determine the required number of interpolations for a block based on its size and motion vector precision. For an $m \times n$ block, the number of half and quad interpolation operations as well as the number of required memory accesses for luminance and chrominance are presented in Table 3.3.

Using the information of Tables 3.2 and 3.3, the number of basic operations for time and

MV_x	MV_y	N^L_{Half}	N^L_{Quad}	N^L_{Mem}	N_{Ch}	N^{Ch}_{Mem}
I	I	0	0	$m \cdot n$		
I	H	$m \cdot n$	0	$m \cdot (n+5)$		
I	Q	$m \cdot n$	$m \cdot n$	$m \cdot (n+5)$		
H	I	$m \cdot n$	0	$n \cdot (m+5)$		
H	H	$(2n+5) \cdot m$	0	$(m+5) \cdot (n+5)$	$m.n/4$	
H	Q	$(3n+5) \cdot m$	$m \cdot n$	$(m+5) \cdot (n+5)$		
Q	I	$m \cdot n$	$m \cdot n$	$n \cdot (m+5)$		
Q	H	$(3m+5) \cdot n$	$m \cdot n$	$(m+5) \cdot (n+5)$		
Q	Q	$2m \cdot n$	$m \cdot n$	$m \cdot n + 5(m+n)$		

Table 3.3: Number of half and quad interpolations and number of memory accesses for a $m \times n$ block

space complexity can be computed:

$$\begin{aligned} N_{Sum} &= 5 \cdot N^L_{Half} + N^L_{Quad} + 7 \cdot N_{Ch} \\ N_{Mul} &= 4 \cdot N^L_{Half} + 8 \cdot N_{Ch} \\ N_{Shift} &= N^L_{Half} + N^L_{Quad} + N_{Ch} \\ N_{Mem} &= N^L_{Mem} + N^{Ch}_{Mem} \end{aligned} \tag{3.5}$$

In order to obtain the total complexity of a sequence on a specific platform and implementation, we need to adjust the weight parameters. An initial encoding is necessary to extract the number of each basic operation using (3.5) on a training set of sequences. The encoded streams will be sent to the decoder and the empirical TCC ($TCC_{empirical}$) will be extracted to help us tune the weight coefficients. When the number of each basic operation and the TCC values are found, we can determine the model weight parameters denoted as α, β, γ, and μ in (3.4) by using the constrained-least-squared method for curve fitting. Once the weights are initialized and tuned, the model is established and can be used to estimate the TCC of motion compensation for sequences outside of the training set which is called $TCC_{estimate}$. The whole complexity modeling process stated above is summarized in a flowchart shown in Figure 3.8. For evaluating the simulation results and more detailed information, we refer readers to [40].

In this thesis, our goal is to control the computational complexity involved in motion compensation at the decoder. Thus, a generic complexity model is needed to monitor motion compensation and ensure that it does not exceed pre-defined complexity thresholds for both

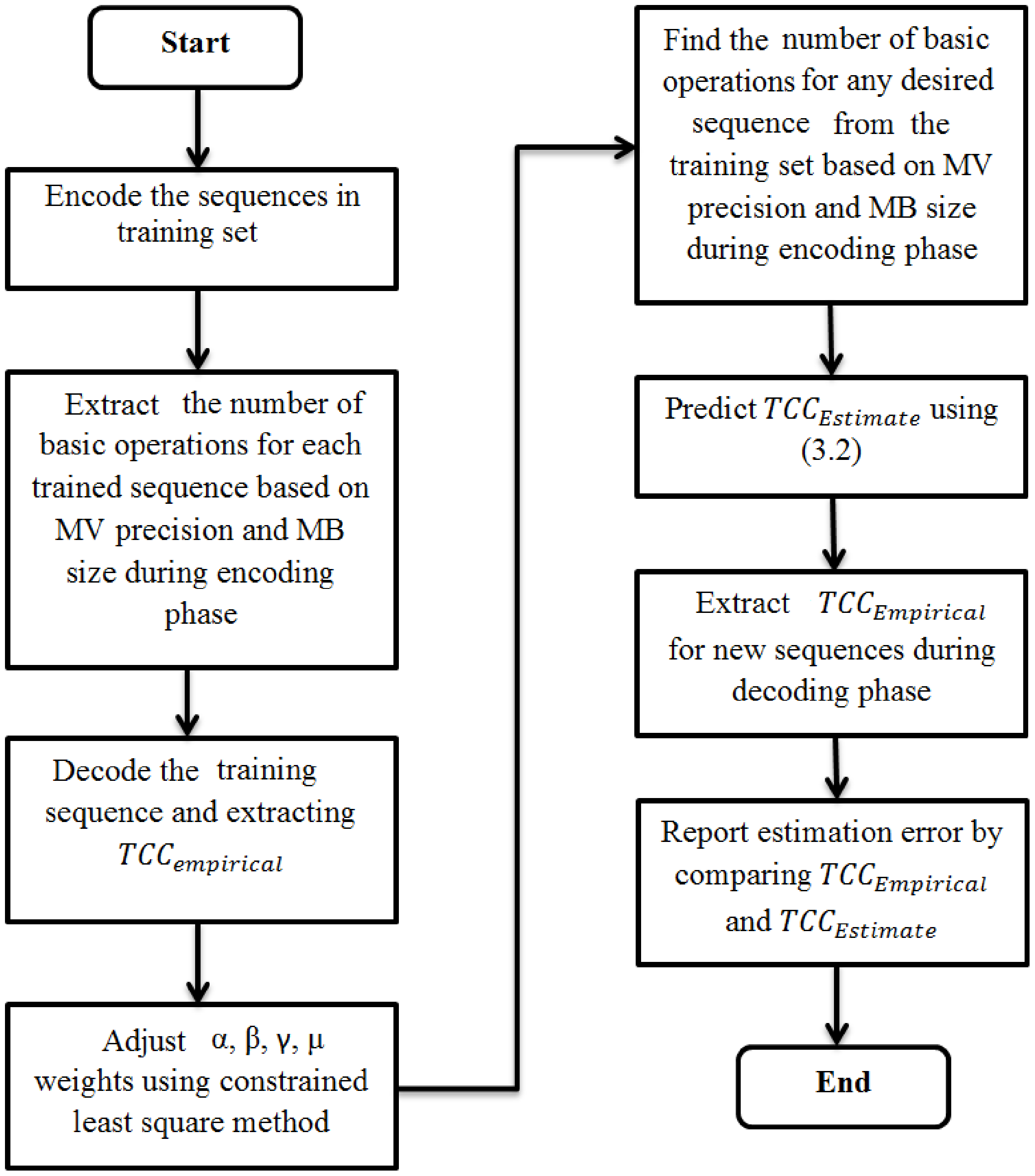

Start
Encode the sequences in training set
Extract the number of basic operations for each trained sequence based on MV precision and MB size during encoding phase
Decode the training sequence and extracting $TCC_{empirical}$
Adjust α, β, γ, μ weights using constrained least square method
Find the number of basic operations for any desired sequence from the training set based on MV precision and MB size during encoding phase
Predict $TCC_{Estimate}$ using (3.2)
Extract $TCC_{Empirical}$ for new sequences during decoding phase
Report estimation error by comparing $TCC_{Empirical}$ and $TCC_{Estimate}$
End

Figure 3.8: Complexity modeling of H.264 motion compensation process

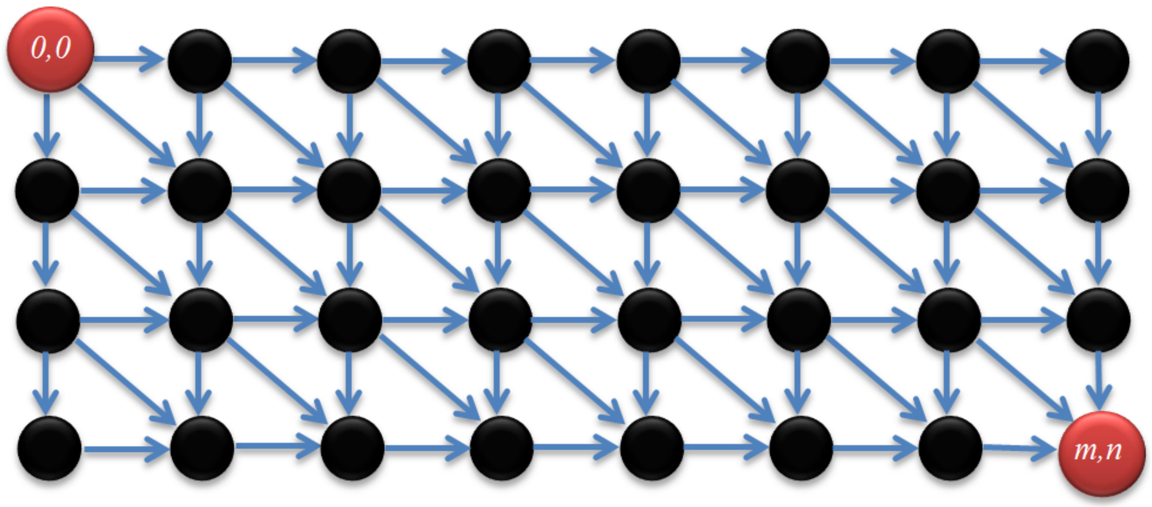


Figure 3.9: The graph used in sequence alignment algorithm

the single and multiple receiver scenarios. We have used [40] as our generic model, and our proposed solution, described in Chapter 4, depends on it.

3.4 Algorithmic Background

In this section, we describe the sequence alignment algorithm from [24] which is based on the shortest path in an edge-weighted matrix graph as shown in Figure 3.9. This method is very efficient in terms of memory and running time. Given a directed graph $G = (V, E)$, $V(i, j) \in V$ is a vertex in row i and column j, and $E_1(i, j)$, $E_2(i, j)$, $E_3(i, j)$ are the weights on the edges entering to $V(i, j)$ horizontally, diagonally, and vertically (if available) respectively (see Figure 3.10).

The goal is to find the shortest path from $V(0, 0)$ to $V(m, n)$. Let $f(i, j)$ be the length of the shortest path from $V(0, 0)$ to $V(i, j)$ which can be computed using dynamic programming in $O(mn)$ running time. The recurrence relation for the dynamic programming formulation of finding the shortest path is shown in Equation (3.6)(also see Figure 3.10).

$$\begin{aligned} &f(i, j) = \min(f(i, j-1) + E_1(i, j), f(i-1, j-1) + E_2(i, j), f(i-1, j) + E_3(i, j)) \\ &f(0, 0) = 0 \end{aligned} \tag{3.6}$$

Similarly, let $g(i, j)$ be the length of the shortest path from $V(m, n)$ to $V(i, j)$ which

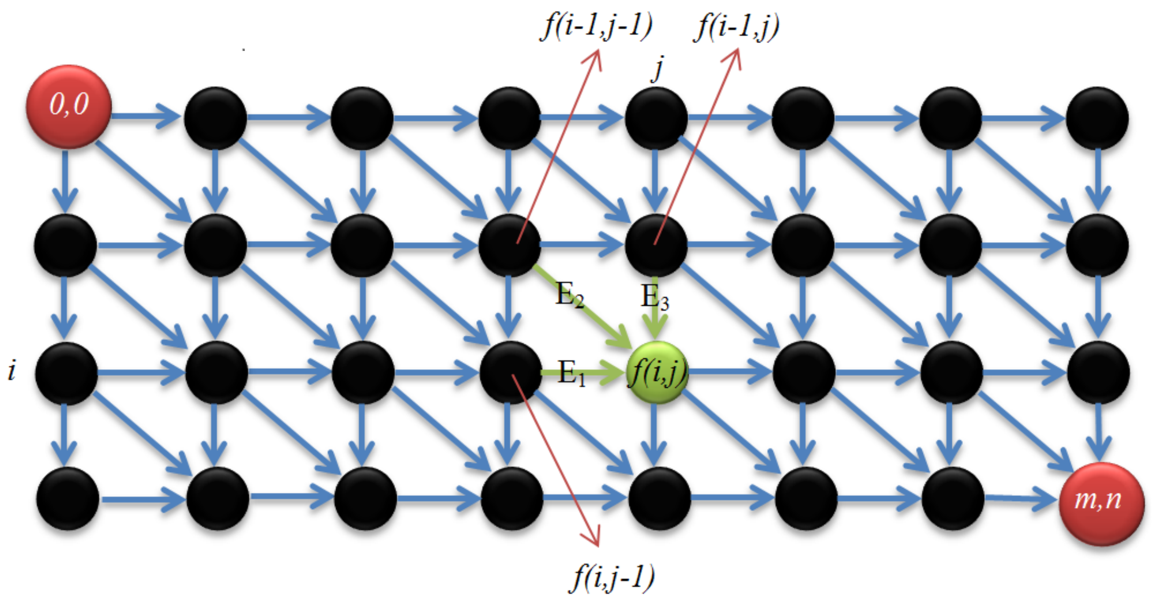


Figure 3.10: Dynamic programming approach in computing $f(i, j)$

can be computed by reversing the edge orientations and inverting the roles of $V(0, 0)$ and $V(m, n)$ (see Figure 3.11). Both $f(i, j)$ and $g(i, j)$ can be solved by the following divide and conquer method.

- Divide: find $f(i, n/2)$ and $g(i, n/2)$ for all $0 \leq i \leq m$
- Conquer: find index $0 \leq q \leq m$ that minimizes $f(q, n/2) + g(q, n/2)$

Once index q has been found, $V(q, n/2)$ will be a part of shortest path (see Figure 3.12). Thie method is then applied recursively to two smaller instances to find other vertices of shortest path (see Figure 3.13).

In terms of implementation, two rows of the matrix need to be stored for each instance of problem which implies $O(m)$ space complexity. Let $T(m, n)$ be the running time of this algorithm on a graph with size of m rows and n columns. Then we have

$$T(m, n) \leq 2T(m, n/2) + O(mn) \Rightarrow T(m, n) = O(m \cdot n \cdot logn). \quad (3.7)$$

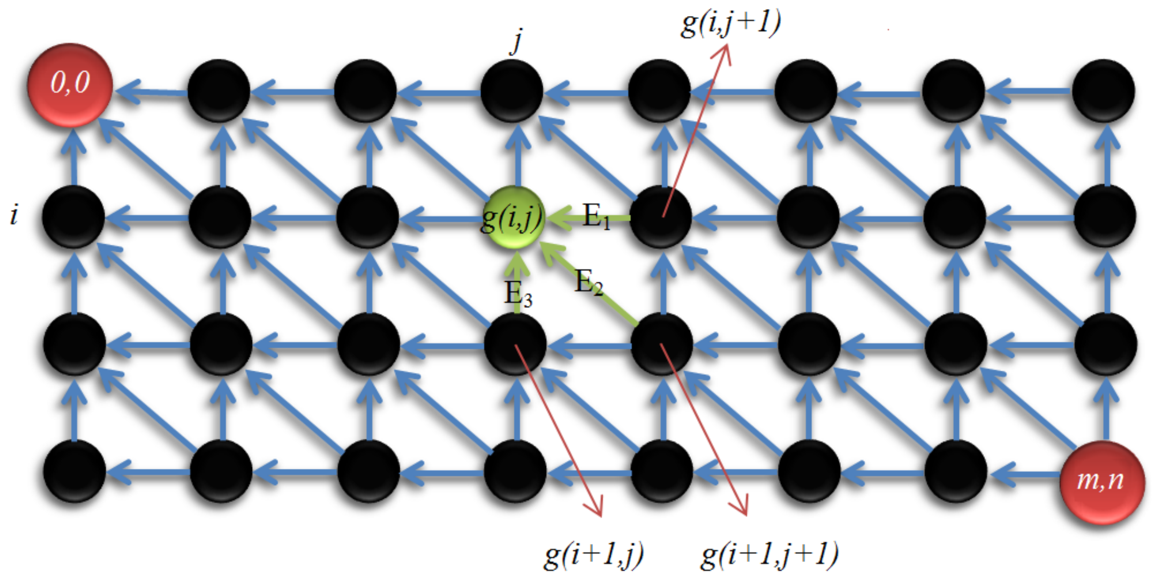


Figure 3.11: Dynamic programming approach in computing $g(i, j)$

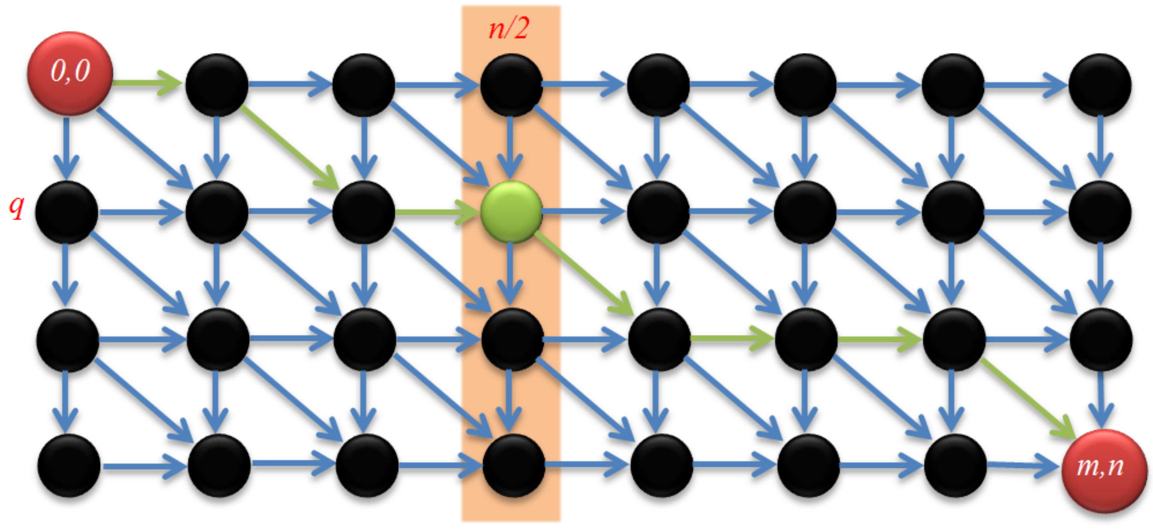


Figure 3.12: Shortest path from $V(0, 0)$ to $V(m, n)$

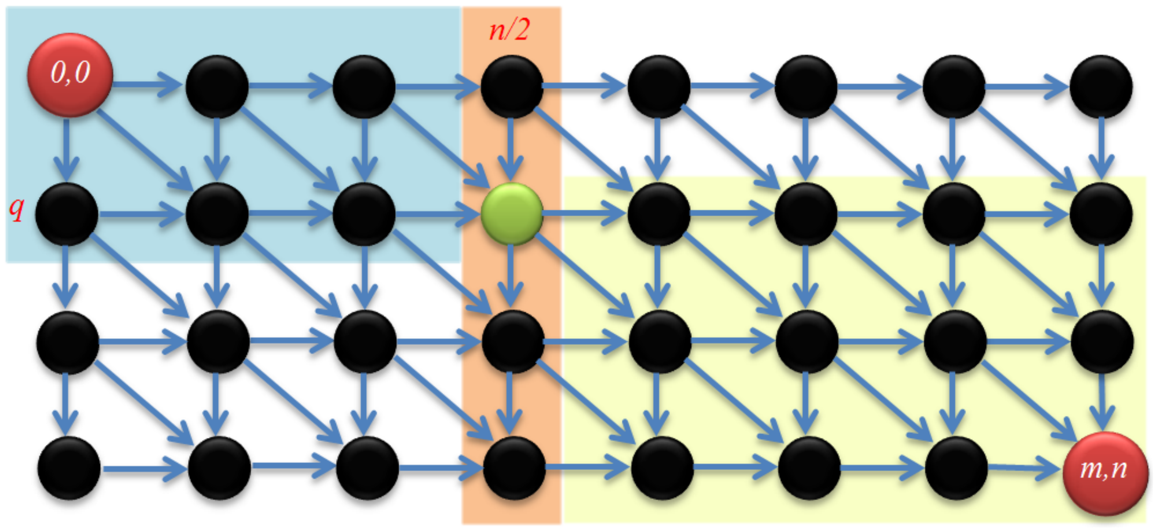


Figure 3.13: Recursive approach in sequence alignment algorithm

Chapter 4

Methodology

In this section, we define the optimization problems of complexity-aware encoding for both the single and multiple receiver scenarios, and we develop the methods that we will use to solve the problems. Our approach for the single receiver scenario is based on the H.264/AVC standard and our approach for the multiple receiver scenario is based on the SVC extension of H.264. But First, the system model that we used is introduced.

4.1 System Model

Our model is based on a client-server system shown in Figure 4.1. The encoder is located in the server side and the decoder receives the video data and is able to send information about its resource requirements. Client information is translated into complexity demands once it is received by the encoder. The encoder processes the video sequence to extract the encoding configuration and customize it according to the complexity bound specified by the client. Then, the video is encoded with the new encoding configuration that will satisfy the input complexity bound and sent to the user(s).

For the multiple receiver scenario, the same client-server system is considered with a wide range of receivers that can select appropriate layers according to their bandwidth and power limitations as shown in Figure 4.2. In this scenario, the solution includes the encoding configuration for each layer.

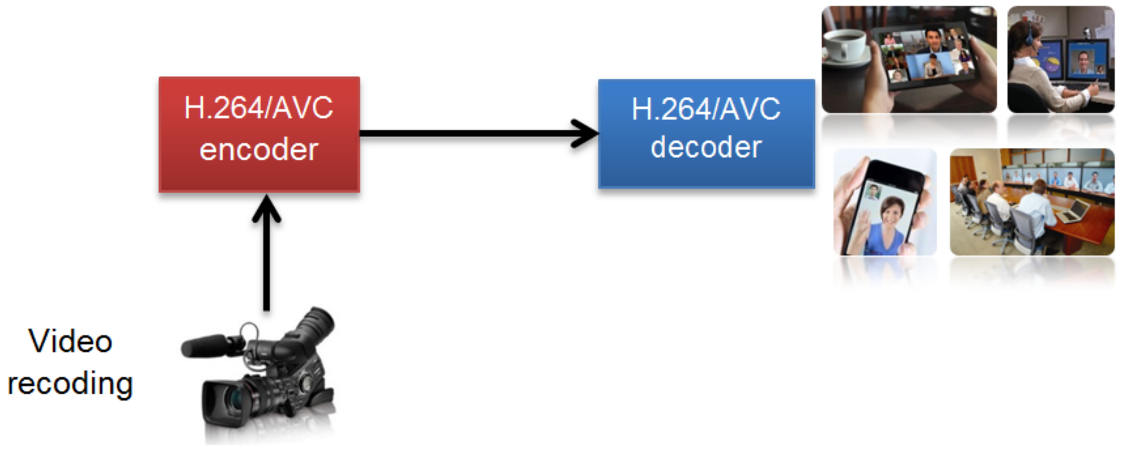


Figure 4.1: Our system model in single receiver scenario [21] (re-used in accordance with ACM publishing agreement)

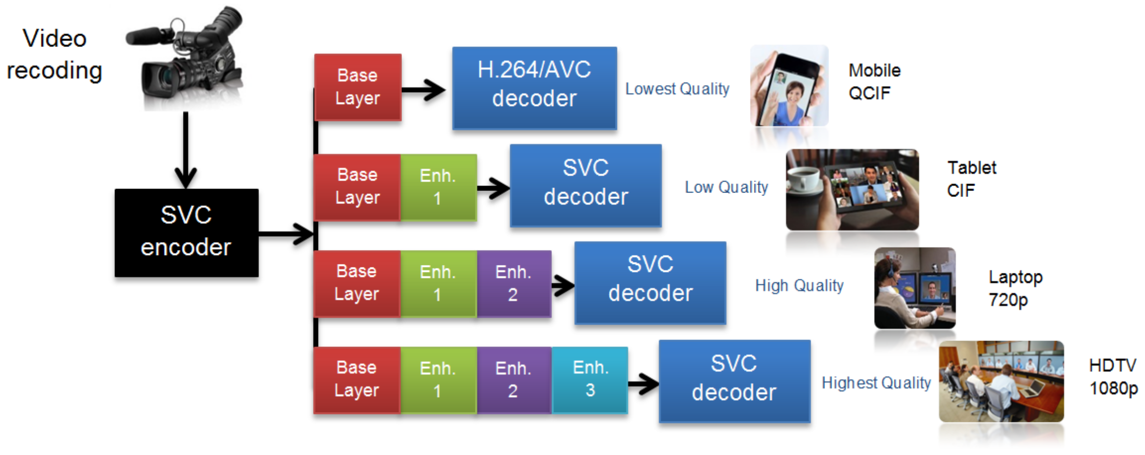


Figure 4.2: Our system model in multiple receiver scenario [21] (re-used in accordance with ACM publishing agreement)

4.2 Formulation of Problem

In H.264/AVC, the default decision for the encoding process is to choose the result that yields the highest quality output image. However, the choices that the encoder makes during the motion estimation process might require more bits to deliver a relatively high quality benefit. Therefore, it cannot necessarily be claimed that the more complexity a mode has, the better bit-rate and quality it can provide for the bit-stream. The motion estimation process solves this issue by "*rate-distortion*" optimization of the aforementioned problem based on the following two metrics.

- Distortion: The deviation from the source material is usually measured as the mean squared error to maximize the peak signal-to-noise ratio (PSNR) video quality metric.
- Bit-rate: The is the bit cost for a particular decision outcome.

There is a trade-off among the bit-rate of the video (R_{tot}), quality of the video (D_{tot}), and the computational complexity of decoding the motion compensation (C_{tot}). Motion vector precision (MVP) and mode decisions are the source coding parameters that have a direct influence on this trade-off. In our proposed optimization problem, the computational complexity of the decoding process acts as a constraint on conventional rate-distortion optimization. So we formulate the problems as functions of source coding parameters plus the computational complexity constraint(s). The formulation for H.264/AVC is

$$\begin{aligned}&\text{minimize } D_{tot}(MVP, mode) \text{ given } R_{tot}(MVP, mode)\\&\text{subject to } C_{tot}(MVP, mode) \leq \beta\end{aligned} \tag{4.1}$$

where β is a complexity threshold received from the user's device.

A combination of MVP and intra or inter modes is called a *state* in this thesis. Each macroblock is in one possible state that includes a specific precision and a particular mode after the motion compensation process in the encoder. We formulate Problem (4.1) as an optimization problem.

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n}\sum_{j=1}^{m} RDC_{ij} \cdot X_{ij} \\ &\text{subject to } \begin{cases} \sum_{i,j} CC_{ij} \cdot X_{ij} \leq \beta \\ CC_{ij} = \sum_k P_{ijk} W_k \\ \sum_j X_{ij} = 1; \quad \forall 1 \leq i \leq n \\ X_{ij} \in \{0,1\}; \quad 1 \leq i \leq n; \quad 1 \leq j \leq m \end{cases} \end{aligned} \tag{4.2}$$

RDC_{ij} and CC_{ij} are the rate-distortion cost and computational complexity, respectively, when state j is chosen for macroblock i. W_k is the weight corresponding to the k^{th} type of basic operation and P_{ijk} is the number of type k basic operations when macroblock i is in state j. The numbers of macroblocks and states are n and m respectively [40].

Problem (4.2) is a multiple-choice knapsack problem. Each (macroblock,state) pair (i, j) is an object that can be placed into the knapsack, and the total computational complexity of the objects in the knapsack cannot exceed the threshold β. Furthermore, exactly one state j must be chosen for each macroblock i, so exactly one pair from each *macroblock group* $G_i = \{(i, j)|1 \leq j \leq m\}$ is placed into the knapsack.

In the SVC extension of H.264, there can be multiple layers. There is a base layer (which we call layer 0) that is similar to the baseline profile of the H.264/AVC standard, and there can be one or more optional enhancement layers (layers $1, 2, \ldots$), each of which depends on the base layer and the enhancement layers below it. Thus, the layers must be decoded in order starting with the base layer, and the formulation becomes

$$\begin{aligned} &\text{minimize } R_{tot}(MVP, mode) \text{ and } D_{tot}(MVP, mode) \\ &\text{subject to } \begin{cases} C_{0,tot}(MVP, mode) \leq \beta_0 \\ C_{0,tot}(MVP, mode) + C_{1,tot}(MVP, mode) \leq \beta_1 \\ \quad\quad\vdots \\ C_{0,tot}(MVP, mode) + C_{1,tot}(MVP, mode) + \cdots + C_{l,tot}(MVP, mode) \leq \beta_l \end{cases} \end{aligned} \tag{4.3}$$

where l is the maximum number of enhancement layers that any user device will use, $C_{h,tot}$ is the computational complexity of decoding the motion compensation in layer h, and $\beta_0, \beta_1, \ldots, \beta_l$ are complexity thresholds received from user devices that will use $0, 1, \ldots, l$

enhancement layers, respectively. The extension of Problem (4.2) to SVC is

$$\text{minimize} \sum_{h=0}^{l}\sum_{i=1}^{n_h}\sum_{j=1}^{m} RDC_{hij} \cdot X_{hij}$$
$$\text{subject to} \begin{cases} \sum_{i,j} CC_{0ij} \cdot X_{0ij} \le \beta_0 \\ \sum_{i,j} CC_{0ij} \cdot X_{0ij} + \sum_{i,j} CC_{1ij} \cdot X_{1ij} \le \beta_1 \\ \qquad\qquad \vdots \\ \sum_{h,i,j} CC_{hij} \cdot X_{hij} \le \beta_l \\ CC_{hij} = \sum_k P_{hijk} W_k \\ \sum_j X_{hij} = 1; \quad \forall\, 0 \le h \le l; \quad 1 \le i \le n_h \\ X_{hij} \in \{0,1\}; \quad 0 \le h \le l; \quad 1 \le i \le n_h; \quad 1 \le j \le m \end{cases} \tag{4.4}$$

In (4.4), n_h is the number of macroblocks in layer h, and the first subscript of other variables refers to layers; CC_{hij} is the computational complexity when state j is chosen for macroblock i in layer h, and so on. As for Problem (4.2), each (macroblock,state) pair (i, j) is an object that can be placed into the knapsack and exactly one pair from each *macroblock group* $G_i = \{(i, j)|1 \le j \le m\}$ is placed into the knapsack. Similar to (4.2), the total computational complexity of the objects (from all layers) in the knapsack cannot exceed the threshold β_l. The main difference is that Problem (4.4) has $l + 1$ complexity constraints instead of one, so Problem (4.4) is a multi-dimensional multiple-choice knapsack problem.

Our approaches to solving Problems (4.2) and (4.4) are both based on dynamic programming with greedy heuristics. Our dynamic programming method has been designed for a single constraint and is common to both approaches; the greedy heuristics and the organizations of the two approaches are different. We start by describing our dynamic programming method in the context of H.264/AVC which has a single constraint. Then we use the method, together with (different) greedy heuristics to solve Problems (4.2) and (4.4).

4.3 Dynamic Programming Method

The recurrence relations for a dynamic programming formulation of (4.2) are [6]:

$$\begin{aligned} & f(0,0) = 0; \quad f(0,b) = 0; \quad f(i,0) = \infty \\ & f(i,b) = \min\{f(i,b-1), \\ & \qquad \min_j\{f(i-1,b-CC_{ij}) + RDC_{ij} \mid CC_{ij} \le b\}\} \\ & 1 \le i \le n; \quad 1 \le b \le \beta; \quad 1 \le j \le m \end{aligned} \tag{4.5}$$

A conventional dynamic programming approach to solving (4.5) uses an $(n+1)\times(\beta+1)$ table and results in a pseudo-polynomial time algorithm with $O(n\beta)$ space complexity and $O(mn\beta)$ time complexity. In our context, this algorithm is both CPU- and memory-bound because β is very large (over one million nanoseconds), so it is only practical for solving small instances of the problem. To reduce the resource requirements, we apply two techniques. The first technique is to scale the CC_{ij} values by a factor S and then round to the nearest integer (denoted $\lfloor\ \rceil$) to reduce the problem size. The resulting optimization problem is

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n}\sum_{j=1}^{m} RDC_{ij}\cdot X_{ij} \\ &\text{subject to } \begin{cases} \sum_{i,j}\lfloor \frac{CC_{ij}}{S} \rceil \cdot X_{ij} \le \lfloor \frac{\beta}{S} \rceil \\ CC_{ij} = \sum_k P_{ijk}W_k \\ \sum_j X_{ij} = 1; \quad \forall\, 1 \le i \le n \\ X_{ij} \in \{0,1\}; \quad 1 \le i \le n; \quad 1 \le j \le m \end{cases} \end{aligned} \tag{4.6}$$

and the space and time complexities of the table-driven pseudo-polynomial algorithm are reduced to $O(n\lfloor\frac{\beta}{S}\rceil)$ and $O(mn\lfloor\frac{\beta}{S}\rceil)$ respectively. Note that this approximation technique is different from the usual scaling approaches for pseudo-polynomial time algorithms which scale the objective function. Instead, we are scaling the constraints. This is possible because the CC_{ij} values are sparsely distributed in a large range, so the errors introduced by careful scaling are small.

The second technique that we apply to our dynamic programming algorithm is a space reduction technique explained in Section 3.4. We have redefined this method for our own purpose. To do so, the dynamic programming table is modelled as a graph in which each node(i,b) corresponds to table entry (i,b), $0 \le i \le n$, $0 \le b \le \beta$, and the weighted edges are as follows: there is an edge(i,b,j) connecting node(i,b) and node$(i-1,b-CC_{ij})$ with weight

RDC_{ij} if $b \geq CC_{ij}$ for $1 \leq i \leq n$, $1 \leq b \leq \beta$, $1 \leq j \leq m$, and an edge(i,b,0) connecting node(i,b) and node(i,$b-1$) with weight 0 for $0 \leq i \leq n$, $1 \leq b \leq \beta$.

The shortest path between node(0,0) and node(n,β) corresponds to an optimal solution. Such a path must pass through a node($\frac{n}{2}$,b) for some $1 \leq b \leq \beta$. The idea is to apply divide-and-conquer to recursively solve two sub-problems: find the shortest paths from node(0,0) to each node($\frac{n}{2}$,b), $1 \leq b \leq \beta$; find the shortest paths from each node($\frac{n}{2}$,b) to node(n,β); and then merge the sub-problem solutions. This technique reduces the space complexity by a factor of $O(n)$ at the expense of an $O(\log_2 n)$ increase in the time complexity. Combining the two technique results in space and time complexities $O(\lfloor\frac{\beta}{S}\rceil)$ and $O(mn \log_2 n \lfloor\frac{\beta}{S}\rceil)$ respectively [18].

4.4 Solution Method for a Single Threshold

Our method to solve Problem (4.2) takes advantage of similarities among consecutive frames by considering solutions for previous frames as potential solutions for the current frame. Our method consists of three steps; first we search through all potential solutions to find the best initial solution for the current frame. The second and third steps iteratively improve the initial solution by local exchanges. We discuss the three steps in detail.

Finding a good initial solution is the most important step of our method. We consider all previous frames with computational complexity no more than 1.2β to be candidates for the initial solution. We have set the complexity threshold in this step to be higher than the actual threshold β to increase the number of candidates. The constant 1.2 has been tuned experimentally to give a good trade-off between efficiency and accuracy. We choose the most recent of the candidate frames as the basis for our initial solution because the frames closest to the current frame are usually the most similar. If the chosen basis frame E is the frame F that immediately precedes the current frame, then the initial solution is $I = E = F$. Referring to (4.2), we set $X^I_{ij} = X^E_{ij}$, $1 \leq i \leq n$, $1 \leq j \leq m$. If E and F are different, then we integrate them into a single initial solution I by resolving the differences between the states of their macroblocks. We initialize $I = E$ and then apply the following pseudo-code for each macroblock i, $1 \leq i \leq n$.

$$\textbf{if} \sum_{j=1}^{m} RDC_{ij} \cdot X^F_{ij} > \sum_{j=1}^{m} RDC_{ij} \cdot X^E_{ij} \textbf{ then } X^I_{ij} = X^F_{ij} \text{ for each } 1 \leq j \leq m$$

The idea behind these substitutions is to modify the basis solution E to get an initial solution

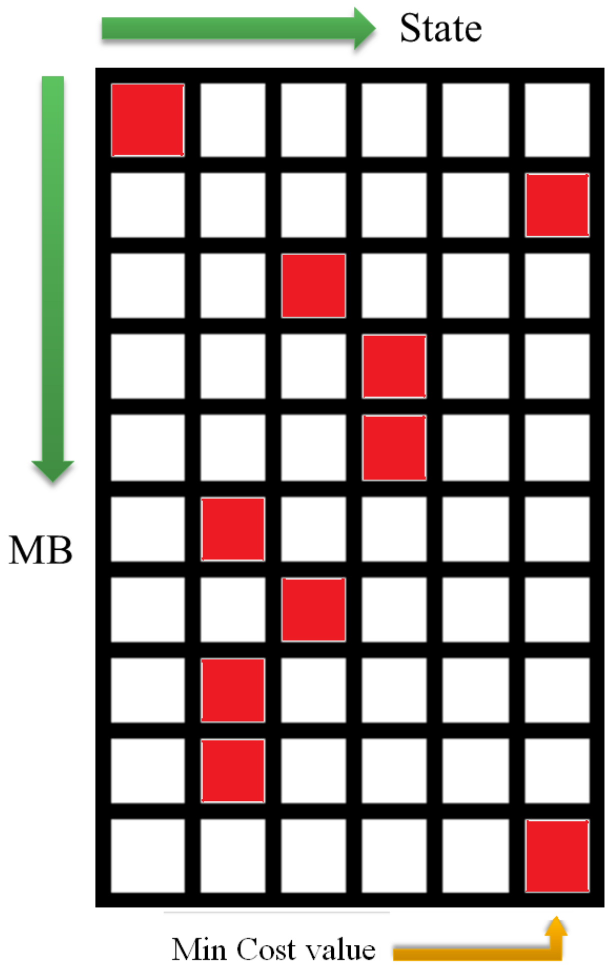


Figure 4.3: An example of an initial solution for our heuristic method shown in a table. The red entries show the state chosen for the corresponding macroblock

that is closer to frame F. However, we do not perform substitutions for Macroblocks that would decrease the total RDC while potentially increasing the solution weight.

The second step is to improve the initial solution. This step is quite similar to solving a dynamic programming problem with an initial solution. First we place the items (i, j) in each macroblock group G_i into a table sorted by RDC value. Figure 4.3 presents an example of a solution in a dynamic programming table in which the macroblocks correspond to rows and the states correspond to columns. We sort the states in each row (MB) based on the rate-distortion value in a way that the minimum values are located on the right side of the table. Then, the initial solution consists of the states indicated in red. Our ultimate goal is to push the initial solution to the right side of the table as far as the complexity threshold allows. We achieve this by incrementally reducing the rate-distortion using a local search around the initial solution. To be more exact, we repeatedly exchange one of the items in the current solution with another one in the same macroblock group to decrease the sum of the RDCs according to the following policy. Let v_i^c and w_i^c be the solution value (sum of the RDCs) and weight (computational complexity) of a candidate item in G_i, and let v_i^s and w_i^s be the value and weight of the current solution in G_i. We define the *Resource Requirement* (RR) of the candidate item to be $RR_i = w_i^c - w_i^s$ and its *Value Update* to be $VU_i = \frac{v_i^c - v_i^s}{w_i^c - w_i^s}$.

Given two candidate items (i, j_1) and (i, j_2) in G_i with RR values RR_1, RR_2 and VU values VU_1, VU_2, we use the following rules for the exchange:

• If $RR_1 \leq 0$ or $RR_2 \leq 0$, the item that minimizes the solution value the most without increasing the weight is chosen.

• If both $RR_1 > 0$ and $RR_2 > 0$, the item with the largest negative VU value (if one exists) is chosen.

The goal of these exchanges is to find the best exchange that does not increase the total weight of the current solution. If the first rule does not apply, then we choose the exchange that gives the largest decrease in RDC per unit of increase in computational complexity.

In this second step, we redefine feasibility from 1.2β that we used in the first step to $\max(\sum_{i,j} CC_{ij} \cdot X_{ij}, \beta)$ so that the solution converges towards the target feasibility value of β. We continue exchanging items until no eligible item exists for exchange.

In the third step, we consider pairs of groups looking for a pair of exchanges, one in each group, which when done together will decrease the total RDC value while maintaining feasibility which in this step means that the total computational complexity does not exceed β.

Concerning the time complexity of the algorithm, we first sort all items in each group in time $O(nm \log_2 m)$. We then perform at most $n(m-1)$ exchanges to find the best solution and each exchange searches through all groups for a total of $O(n^2 m)$. The third step checks $O(n^2)$ possible pairs of exchanges. The total time complexity of our polynomial algorithm is $O(n^2 m + nm \log_2 m)$.

The dynamic programming method in Section 4.3 provides good solutions, but even with our modifications that permit the solution of large instances, the CPU and memory requirements are large. The greedy method that we have developed in this section is much more efficient, but it requires an initial solution that is reasonably close to the optimal solution, and errors propagate if it is used to find solutions for a sequence of consecutive frames. Our proposed method for controlling computation-based power consumption is a combination of the dynamic programming and greedy methods that attempts to minimize these shortcomings. We partition the sequence of frames into *hybrid groups of pictures* (HGOPs). The dynamic programming method is used to find an accurate solution for the first frame of each HGOP and to provide a good initial solution for the greedy solutions of the remaining frames. Our results in Section 6 show that this method provides a good trade-off between efficiency and accuracy.

4.5 Solution Method for Multiple Complexity Thresholds

Our method for the SVC extension of H.264/AVC is also based on the dynamic programming method of Section 4.3. The primary goal is to minimize the total rate-distortion for all layers while satisfying multiple user complexity constraints. The dynamic programming method is designed for a single constraint, but we can use it recursively to handle the multiple complexity constraints in Problem (4.4).

The general idea of our approach is to solve a relaxation of Problem (4.4) to obtain an initial solution, which provides a lower bound on the optimal solution, and then converge the initial solution towards the optimal solution by reinforcing the original constraints.

The relaxed problem is obtained by removing all constraints except the last one to obtain Problem (4.7) below. The solution for (4.7) provides a lower bound on the optimal solution for Problem (4.4), but it is not necessarily a feasible solution for (4.4) because some of the omitted complexity constraints might not be satisfied.

$$\begin{aligned} &\text{minimize } \sum_{h=0}^{l}\sum_{i=1}^{n_h}\sum_{j=1}^{m} RDC_{hij}\cdot X_{hij} \\ &\text{subject to } \begin{cases} \sum_{h,i,j} CC_{hij}\cdot X_{hij} \le \beta_l \\ CC_{hij} = \sum_k P_{hijk}W_k \\ \sum_j X_{hij} = 1; \quad \forall\, 0\le h\le l; \quad 1\le i\le n_h \\ X_{hij}\in\{0,1\}; \quad 0\le h\le l; \quad 1\le i\le n_h; \quad 1\le j\le m \end{cases} \end{aligned} \tag{4.7}$$

If the initial solution for (4.7) is a feasible solution for Problem (4.4), then it is the final and optimal solution. Otherwise, one or more constraints are not satisfied, and we converge the initial solution for (4.7) towards a solution for (4.4). First, we calculate a fair distribution of the available complexity budget β_l among the macroblocks of all layers. In particular, each layer h will be given a fraction of the complexity budget proportional to the ratio $\frac{n_h}{n_f}$ of the number of macroblocks in layer h to the number of macroblocks in the first layer. The reason is due to the fact that the complexity of each macroblock for a specific state follows a unique and constant pattern, so it is realistic to allocate a fixed amount of budget to each macroblock. Let α_f denote the amount of the budget that will be allocated to layer f. We solve for α_f as follows.

$$\begin{aligned} \alpha_f &\leq \beta_f \\ \alpha_f + \left(\frac{n_{f+1}}{n_f}\right)\alpha_f &\leq \beta_{f+1} \\ &\vdots \\ \alpha_f + \left(\frac{n_{f+1}}{n_f}\right)\alpha_f + \cdots + \left(\frac{n_l}{n_f}\right)\alpha_f &\leq \beta_l \end{aligned} \tag{4.8}$$

Rearranging (4.8), we obtain

$$\begin{aligned} \alpha_f &\leq \beta_f \\ \left(\frac{n_f + n_{f+1}}{n_f}\right)\alpha_f &\leq \beta_{f+1} \\ &\vdots \\ \left(\frac{\sum_{h=f}^{l} n_h}{n_f}\right)\alpha_f &\leq \beta_l \end{aligned} \tag{4.9}$$

The solution of (4.8) gives the value of α_f:

$$\alpha_f = \min\left(\beta_f,\ \frac{\beta_{f+1} n_f}{\sum_{h=f}^{f+1} n_h}, \ldots, \frac{\beta_l n_f}{\sum_{h=f}^{l} n_h}\right) \tag{4.10}$$

Next, we allocate budget α_f to the first layer and solve it independent of the enhancement layers using the dynamic programming method shown in (4.11).

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{n_f}\sum_{j=1}^{m} RDC_{fij}\cdot X_{fij} \\ &\text{subject to} \begin{cases} \sum_{i,j} CC_{fij}\cdot X_{fij} \le \alpha_f \\ CC_{fij} = \sum_k P_{fijk}W_k \\ \sum_j X_{fij} = 1; \quad 1\le i\le n_f \\ X_{fij}\in\{0,1\}; \quad 1\le i\le n_f; \quad 1\le j\le m \end{cases} \end{aligned} \tag{4.11}$$

f is initialized to zero for the first iteration so that the problem can start from the base layer. The base layer and its budget are then removed from (4.4) to obtain the following problem which we solve recursively for each enhancement layer.

$$\begin{aligned} &\text{minimize} \sum_{h=1}^{l}\sum_{i=1}^{n_h}\sum_{j=1}^{m} RDC_{hij}\cdot X_{hij} \\ &\text{subject to} \begin{cases} \sum_{i,j} CC_{1ij}\cdot X_{1ij} \le \beta_1-\alpha_0 \\ \sum_{i,j} CC_{1ij}\cdot X_{1ij} + \sum_{i,j} CC_{2ij}\cdot X_{2ij} \le \beta_2-\alpha_0 \\ \qquad \vdots \\ \sum_{h,i,j} CC_{hij}\cdot X_{hij} \le \beta_l-\alpha_0 \\ CC_{hij} = \sum_k P_{hijk}W_k \\ \sum_j X_{hij} = 1; \quad \forall 1\le h\le l; \quad 1\le i\le n_h \\ X_{hij}\in\{0,1\}; \quad 1\le h\le l; \quad 1\le i\le n_h; \quad 1\le j\le m \end{cases} \end{aligned} \tag{4.12}$$

This algorithm uses the same space complexity described in Section 4.3 for the repetitive execution of the dynamic programming algorithm. In the worst-case scenario for running time, $l-1$ instances of the single constraint problem (4.11) have to be solved and l lower bounds have to be computed. As mentioned in Section 4.3, it takes $O(\beta n \log_2 n)$ running time to solve problem (4.11) when the number of states and scaling factor are constant. Since $\beta \le \beta_l$ and $n \le \sum_{h=0}^{l} n_h$ in each iteration, the running time of our approximation algorithm is $O(\beta_l \sum_{h=0}^{l} n_h \log_2 \sum_{h=0}^{l} n_h)$.

For illustration, the proposed method is depicted as a flowchart for solving the multiple receivers scenario with three layers in Figure 4.4. To increase the readability, only the constraints of the problem are shown in each phase and a new parameter is defined:

$$Z_h = \sum_{i,j} CC_{hij}\cdot X_{hij}.$$

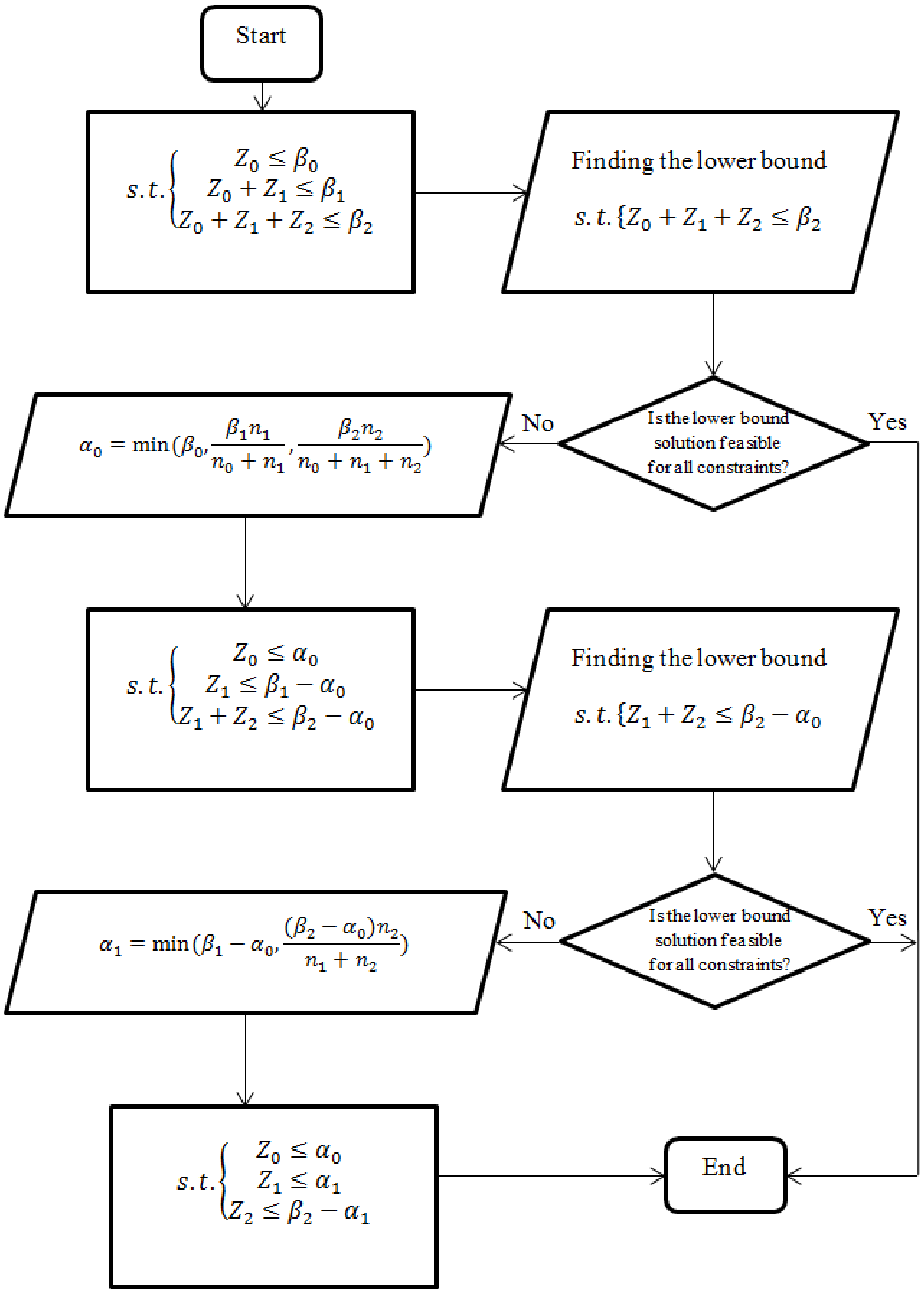

Start
$s.t.\begin{cases} Z_0 \le \beta_0 \\ Z_0 + Z_1 \le \beta_1 \\ Z_0 + Z_1 + Z_2 \le \beta_2 \end{cases}$
Finding the lower bound
$s.t.\{Z_0 + Z_1 + Z_2 \le \beta_2$
Is the lower bound solution feasible for all constraints?
No
Yes
$\alpha_0 = \min(\beta_0, \frac{\beta_1 n_1}{n_0 + n_1}, \frac{\beta_2 n_2}{n_0 + n_1 + n_2})$
$s.t.\begin{cases} Z_0 \le \alpha_0 \\ Z_1 \le \beta_1 - \alpha_0 \\ Z_1 + Z_2 \le \beta_2 - \alpha_0 \end{cases}$
Finding the lower bound
$s.t.\{Z_1 + Z_2 \le \beta_2 - \alpha_0$
Is the lower bound solution feasible for all constraints?
No
Yes
$\alpha_1 = \min(\beta_1 - \alpha_0, \frac{(\beta_2 - \alpha_0) n_2}{n_1 + n_2})$
$s.t.\begin{cases} Z_0 \le \alpha_0 \\ Z_1 \le \alpha_1 \\ Z_2 \le \beta_2 - \alpha_1 \end{cases}$
End

Figure 4.4: Multiple receivers scenario summarized in a flowchart with a base layer and two enhancement layers. There is an objective function in each rectangle and lower bound solution that corresponds to the constraints. Such constraints of the optimization problem for each iteration is shown in a rectangle shape

Chapter 5

Implementation

In this chapter the experimental setup and implementation details are explained. Our framework is designed to be platform-independent as mentioned in Section 3.3. For simplicity, we implemented our approaches in only one platform. We evaluated our decoder-complexity-aware encoding methods on a PC platform with an 8-core Intel 3.40 GHz Core i7 CPU and 8 GB of RAM, running the 64-bit Windows 7 Enterprise operating system. Four video sequences, each with 45 frames, were used for testing purposes: “Parkrun”, “Stockholm”, “Shields”, and “Bluesky”. The motion compensation component of H.264 consists of interpolation and mode decisions. The 12 possible states considered in our general testbed for each macroblock are five inter modes with interpolation, five inter modes without interpolation, and two intra modes which have zero complexity.

5.1 Single Stream Scenario

We tested our single stream method using the JM implementation of the H.264/AVC codec [42]. The experiments were performed using an H.264 baseline profile, which has only I and P frames, and the configuration shown in Table 5.1. It should be noted that the quantization parameter has been fixed to a constant value because changes to this parameter do not affect the performance of our algorithms.

The implementation procedure consisted of two phases. In the first phase, we initialized the weights W_k using pre-encoded bit streams in a training pool for our specific platform according to the method explained in Section 3.3. The sequences were then fed into the JM reference software to determine RDC and CC values, and a lower bound for β.

Parameter	Setting	Parameter	Setting
Level	4	Number of encoded frames	45
Frame rate	30	GOP size	25
Intra period	25	Number of reference frames	1
Search Mode	Full search	Search range	32
Size	HD	Quantization parameter	36

Table 5.1: AVC Encoder Configuration

In the second phase, CC values are scaled with an appropriate scaling factor S shown in equation (4.6) which is measured in an empirical way. Then our hybrid method including dynamic programming and greedy heuristic technique is applied to decrease the complexity of the decoder to the user-defined threshold β.

We used the dynamic programming method without the approximation and space reduction techniques to find the solution of equation (4.2) for each frame. This produced an optimal solution that we used as a benchmark to evaluate the performance of our hybrid method. We used the following three performance metrics to evaluate our methods.

- The *optimality ratio (OR)* measures the quality of a solution X relative to a benchmark solution X^B:

$$OR = \left(1 - \frac{\sum_{i,j} RDC_{ij} \cdot X_{ij} - \sum_{i,j} RDC_{ij} \cdot X^B_{ij}}{\sum_{i,j} RDC_{ij} \cdot X^B_{ij}}\right)$$

- The *computation running time (CRT)* is the average execution time of the algorithm per frame.

- The *complexity error (CE)* is the relative difference between the complexity of a solution X and β:

$$CE = \left(\frac{\sum_{i,j} CC_{ij} \cdot X_{ij} - \beta}{\beta}\right)$$

5.2 Multiple Stream Scenario

We tested our method for multiple complexity thresholds using the conventional SVC reference software JSVM 9.19.15 [44] and the same four test sequences that we used in the previous section for H.264/AVC. We used three layers and the configuration show in Table 5.2 for our experiments. The profile has frames of types I, B, and P in contrast to the profile for H.264/AVC which had only types I and P. We tested several different temporal and spatial scalabilities. Since SNR scalability does not affect our experiments, we fixed the value of the quantization parameter to 36. We tested two GOP sizes (5 and 9) for layer 2; the size of the GOP for layer 2 determines the sizes for layers 0 and 1.

Quantization Parameter		36
Search Mode		Block search
Intra period		8
Fast search		OFF
Inter-layer prediction		ON
Search range	Base layer	16
	Enhancement Layers	8
GOP size	Base layer (Layer 0)	2 3
	Enhancement Layer 1	3 5
	Enhancement Layer 2	5 9
Spatial Resolution	Base layer (Layer 0)	CIF
	Enhancement Layer 1	4CIF
	Enhancement Layer 2	HD
Frame rate	Base layer (Layer 0)	15
	Enhancement Layer 1	30
	Enhancement Layer 2	60

Table 5.2: SVC Encoder Configuration

Similar to the single stream scenario, the JSVM software is customized to report *RDC* and *CC* values independently for the base and each enhancement layer with the encoder configuration mentioned in Table 5.2. Then, our heuristic method for multiple streams is adopted to produce the solutions for each layer considering the complexity thresholds applied to each one of them. In this phase, the single stream method will be called if there is a single constraint as in Problem (4.7).

Moreover, we used a lower bound on the optimal solution for Problem (4.4) (obtained by solving Problem (4.7)) as the benchmark for evaluating our heuristic algorithm. This lower bound solution for Problem (4.4) is not necessarily feasible and the real optimal solution for Problem (4.4) might have a higher *RDC*. Thus, the actual performance of our algorithm might be better than the values reported here. The optimality ratio is calculated using the following formula.

$$OR = \left(1 - \frac{\sum_{h=1}^{l}\sum_{i=1}^{n_h}\sum_{j=1}^{m} RDC_{hij} \cdot X_{hij} - \sum_{h=1}^{l}\sum_{i=1}^{n_h}\sum_{j=1}^{m} RDC_{hij} \cdot X_{hij}^{B}}{\sum_{h=1}^{l}\sum_{i=1}^{n_h}\sum_{j=1}^{m} RDC_{hij} \cdot X_{hij}^{B}}\right)$$

Chapter 6

Experiments and Analysis

In this chapter the experimental results and evaluation are presented. First we evaluate the dynamic programming method which is used in both the single and multiple streams scenarios. Then the experimental results for both methods are presented.

We tested the dynamic programming method (with the approximation and space reduction techniques) for a wide range of scaling factors to determine the best one to use in the hybrid method. Figure 6.1 shows the trade-offs between complexity error (CE) and computation running time (CRT) for four video sequences. The *CE*s for the Parkrun, Stockholm, Shields, and Bluesky sequences are only 0.9%, 0.7%, 1%, and 1%, respectively, using scaling factor $S = 2^{13}$. Furthermore, with $S = 2^{13}$ the running time overheads are trivial compared to the encoding time and the optimality ratios are close to 1 indicating that the solutions are close to optimal. Therefore, we used $S = 2^{13}$ when testing the hybrid method.

The reason that $S = 2^{15}$ is not chosen as the optimal solution is due to the error cancellation that occurs in this scale. As a matter of fact, the error that is reported in graph 6.1 is the absolute total deviation error in complexity. The rounded computational complexity values introduce negative and positive errors which can cancel each other when presented as "absolute error". The amount of negative and positive error is depicted in Figure 6.2 for all sequences. When $S = 2^{15}$ is used, total positive and negative errors cancel each other which results in an artificial zero total error.

Figure 6.3 the distribution of states chosen in the solution for each approximation scale. The graph shows that the solution follows a unique pattern up to point $S = 2^{13}$ and starts deviating from $S = 2^{14}$ as the number of intra modes is reduced from 1953 to 1719 in the 16×16 size and from 802 to 686 in the 4×4 size. The same deviation occurs for $S = 2^{15}$

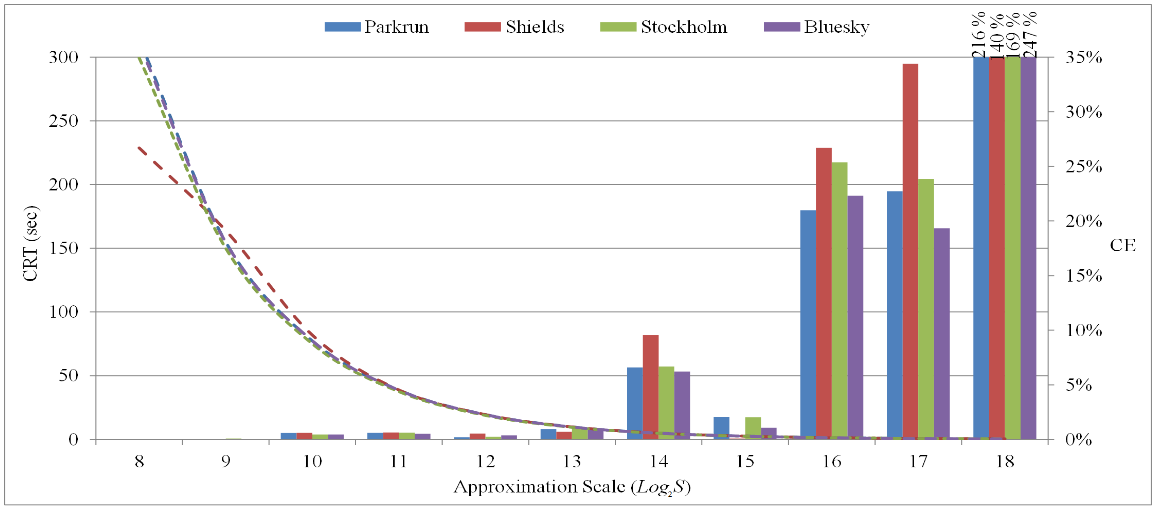


Figure 6.1: Trade-offs between *CE* and *CRT* for dynamic programming method. The dashed lines show *CRT*s and bars show *CE*s.

compared to $S = 2^{13}$ which emphasizes the fact that the solutions produced by $S = 2^{14}$ and $S = 2^{15}$ are not reliable.

6.1 Experimental Results for H.264/AVC

To evaluate our hybrid method, the frequency of utilizing the greedy heuristic method is analyzed and its effect on algorithm performance and timing overhead is tested. Figure 6.4 shows the trade-offs among *OR*, *CRT*, and the length of the HGOPs for the hybrid method for the four 45-frame sequences. We varied the length of the HGOPs between 2 and 30 and used $\beta = 2$ for all experiments. In the figure, *CRTS* is the average time per frame taken by the hybrid method to solve all 45 frames sequentially. *CRTP* refers to a parallel implementation that will be discussed later.

We can conclude from Figure 6.4 that the accuracy achieved by our hybrid method increases with decreasing HGOP length. When the HGOP length is 2, the *OR* (expressed as a percentage) is greater than 92% and it decreases gradually to the 82%-91% range as the HGOP length increases. The *CRTS* decreases gradually with increasing HGOP length and is between 0.34 and 3 seconds per frame for most HGOP lengths.

Even better real-time encoding times can be achieved for H.264 using fine- and coarse-grain parallelism [38, 35, 54]. Our method can be easily adapted to take advantage of

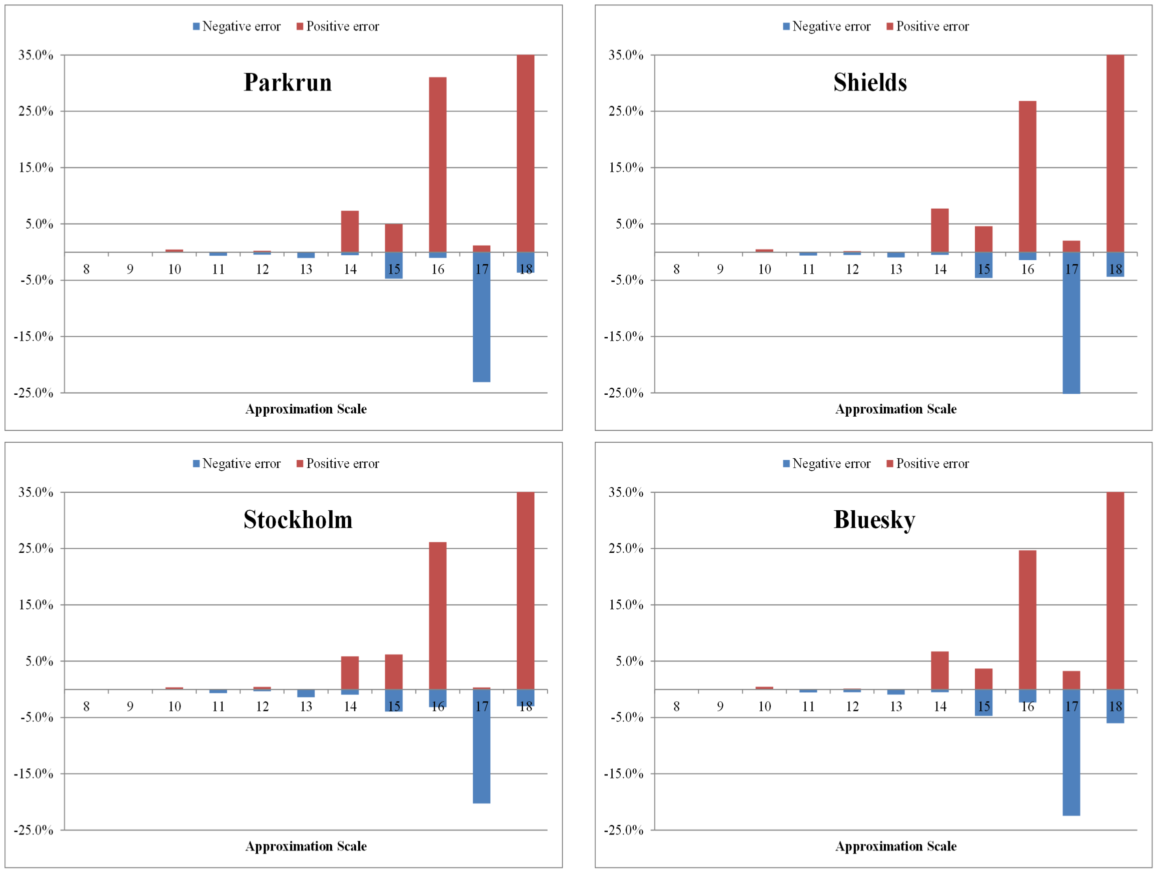


Figure 6.2: Error cancellation phenomenon that occurs in complexity value rounding

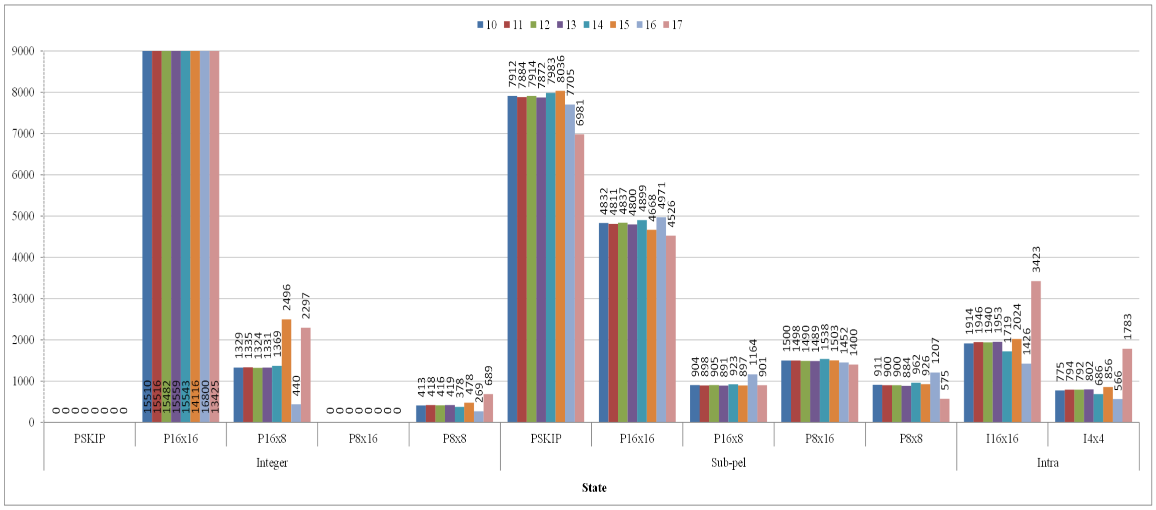
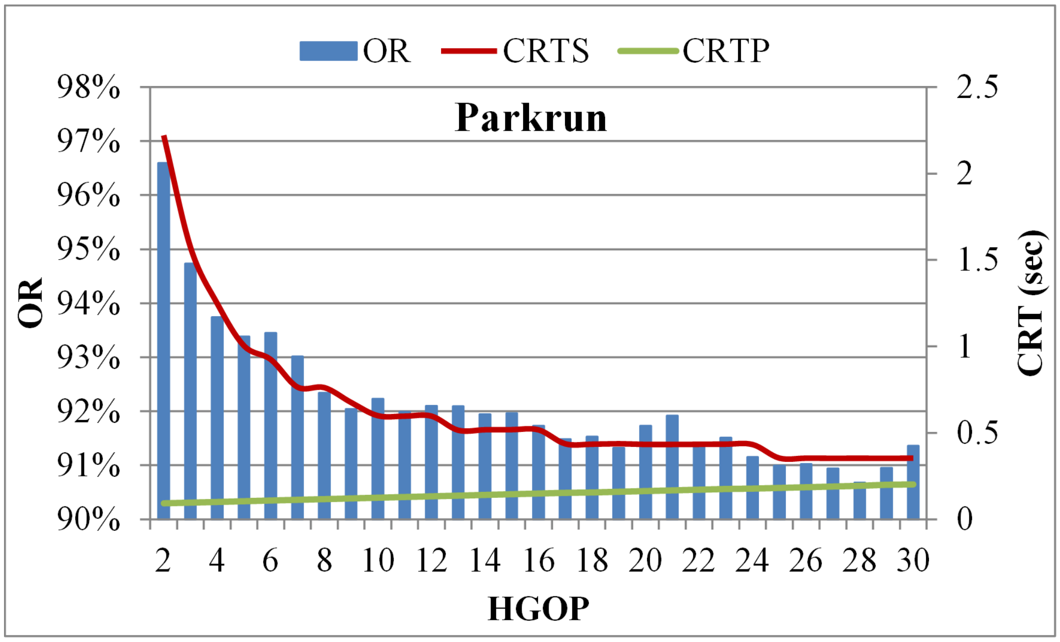


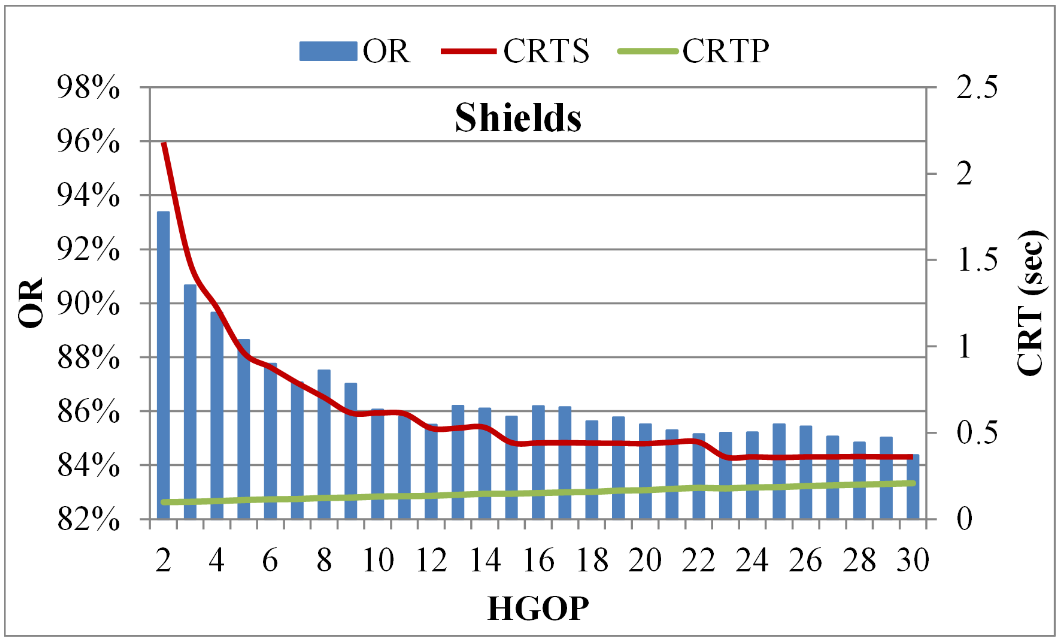
Figure 6.3: State distribution over different modes and interpolation for the Parkrun sequence which shows the number of times that each state has been chosen in the solution. Different bars show different $\log_2 S$ in each state which starts from 10 at the left and ends with 17 at the right.

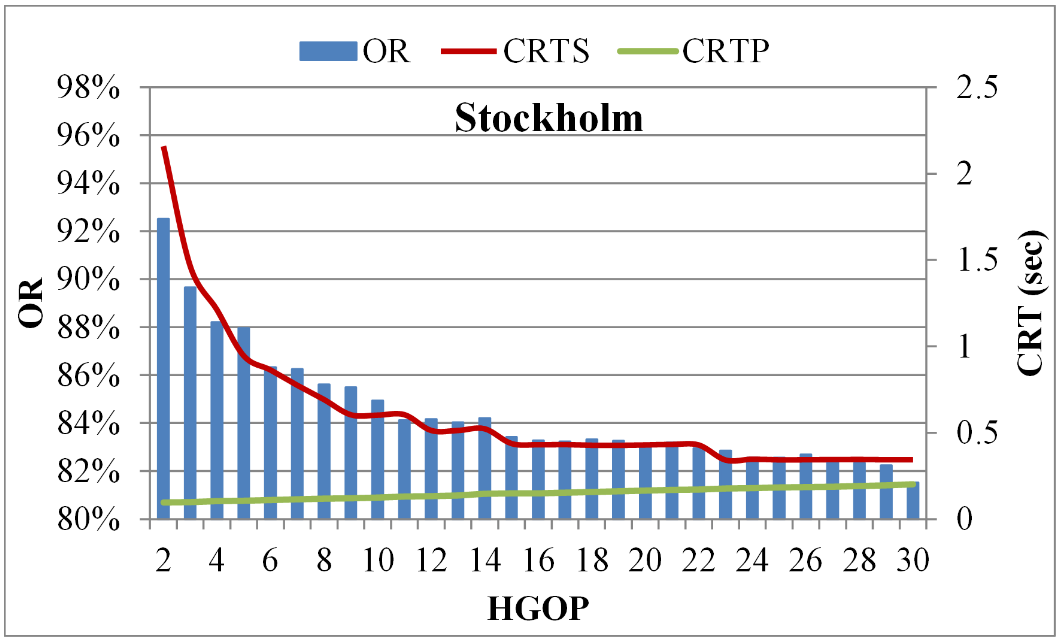
additional hardware resources because it can take advantage of GOP level parallelism to process the HGOPs independently in parallel. In Figure 6.4, *CRTP* is the maximum of the *CRT*s for the HGOPs in contrast to *CRTS* which is the average of the *CRT*s. Both *CRTP* and *OR* improve when the HGOP length decreases, but more hardware is needed to achieve these gains.

Figure 6.5 shows the impact of the user-defined threshold β on the performance of our algorithms. We tested our hybrid method for all video sequences for HGOP size 2 and HGOP size 15 with $S = 2^{13}$ and β ranging from 2 to 10. Values of $\beta > 10$ do not result in significant performance improvements for any of the sequences. Variations in β have no impact on our dynamic programming method but it has a large influence on the initial solution chosen by the greedy method. The greedy method examines all previous frames with computational complexity less than 1.2β for initial solution candidacy. If β is increased, then more of the recent frames will be examined. This increases the chance of choosing an initial solution with a high rate-distortion value and improves the convergence of our hybrid method to an optimal solution. The improvements when β is increased from 2 to 10 for different sequences are shown in Table 6.1

In a media streaming context, the term *real-time* means delivering frames at the same


OR
CRTS
CRTP
Parkrun
OR
CRT (sec)
HGOP
98%
97%
96%
95%
94%
93%
92%
91%
90%
2.5
2
1.5
1
0.5
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30


OR
CRTS
CRTP
Shields
OR
CRT (sec)
HGOP
98%
96%
94%
92%
90%
88%
86%
84%
82%
2.5
2
1.5
1
0.5
0
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30



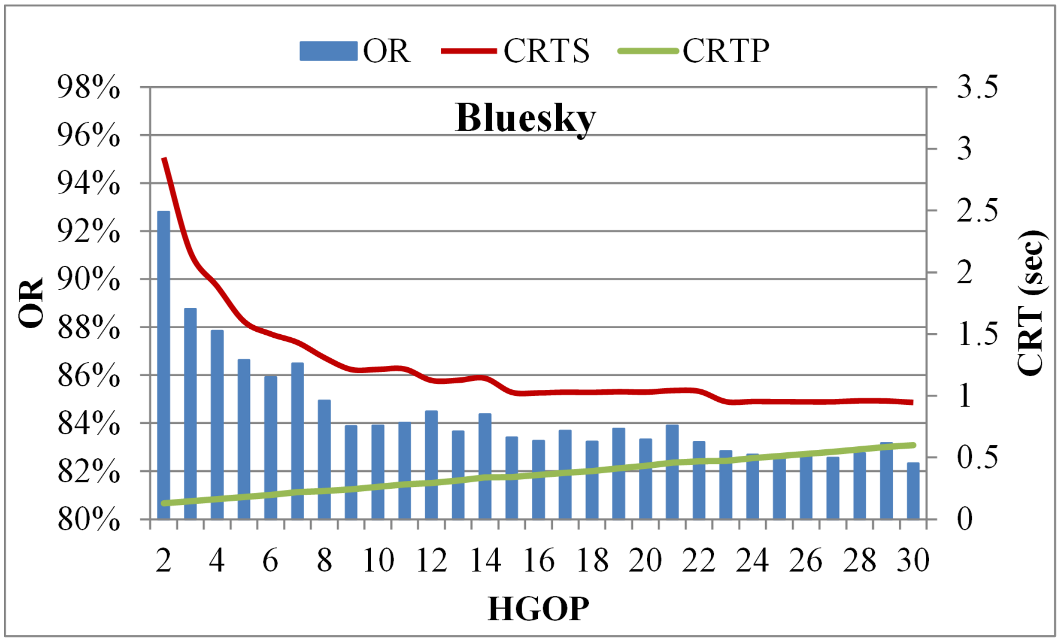


Figure 6.4: Trade-offs between *CRT* and *OR* for different HGOP lengths.

Sequence	HGOP size	
	2	15
Parkrun	3%	6.7%
Shields	5.5%	12.3%
Stockholm	8.2%	16.5%
Bluesky	7.2%	16.5%

Table 6.1: OR improvements with β changing from 2 to 10 seconds in two different HGOP sizes

rate that the client decodes them. In [30], a parallel hardware architecture for the 0/1 knapsack problem is presented. The proposed architecture uses $\theta(n+p(\beta+W_{max}))$ memory and the running time is $\theta(n\beta/p + n\log(n/p))$, where n is the number of objects, β is the knapsack capacity, p is the number of processors for parallel execution, and W_{max} is the maximum weight. Utilizing such a hardware implementation to solve knapsack problems in our approach without any parallelism will result in $\theta(n+\beta)$ memory usage and $\theta(n\beta)$ running time (W_{max} is relatively small value in our application). By increasing the memory usage by an additive factor corresponding to the number of objects (n), they managed to reduce the running time by a logarithmic factor ($\log n$) compared to our approach. With parallel processing, even more time can be saved at the expense of allocating more memory space and processing units to the algorithm. According to the experimental results of [30] which evaluate the running time of the algorithm based on knapsack capacity (β) and the number of objects (n), it takes approximately 50 seconds using 32 processors when $n = 250000$ and $\beta = 50000$. In our case, $n = 3600$ and β varies from 1 second to 10 seconds which translates to 12000 to 120000 in our formulation when the approximation scale is set to 2^{13}. Since the proposed algorithm in [30] is proportional to the number of objects, the running time can be normalized to $n = 3600$ with a resulting time of 720 ms.

A hardware implementation of dynamic programming for the multi-dimensional knapsack problem is presented in [8]. The authors implemented the dynamic programming by parallelizing the for loops in the algorithm on Graphics Processing Units (GPU). Their experimental results show that the multi-dimensional knapsack problem can be solved in 900 ms with 10000 objects (n) and knapsack capacity of 4989314 (β). The results in [8] cannot be directly compared to our scenario with $n = 3600$ and and β ranging from 12000 to 120000, but application of this method to our scenario would certainly result in a running

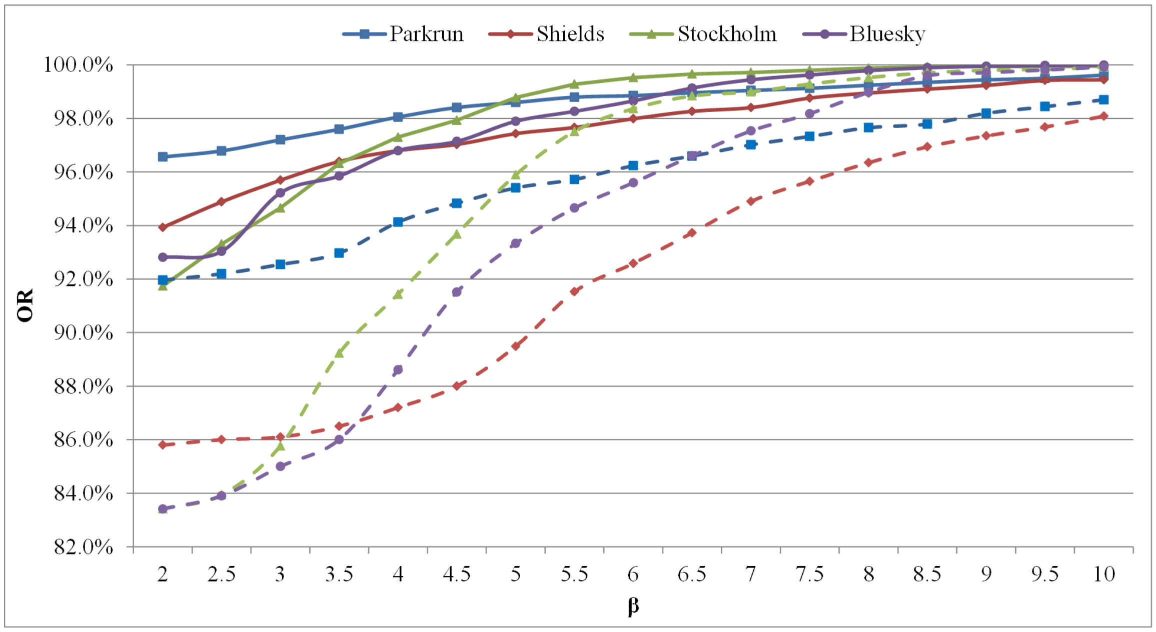


Figure 6.5: Impact of β on the performance of the hybrid algorithm. The dashed lines are for HGOP size 15 and the solid lines are for HGOP size 2.

time less than 900 ms.

In summary, the two implementations in [30] and [8] are promising hardware approaches that we might be able to use to make our methods real-time.

6.2 Experimental Results for SVC Extension

We conducted extensive tests of our method with different combinations of complexity thresholds (β_0, β_1, and β_2) for the three layers for GOP sizes 5 and 9. Each of the graphs in Figures 6.7-6.10 shows 150 data points. The values chosen for each β range between 0 and the *saturation point* for the corresponding layer. The saturation point for a layer is the complexity consumption of the layer in the lower bound solution. Allocating complexity to a layer beyond its saturation point will not deliver any performance improvements whatsoever.

Each curve in Figures 6.7-6.10 corresponds to a value of β_2 as indicated in the legend at the top of each graph. There are five columns of curves in each graph corresponding to different values of β_1 according to the legend at the bottom of each graph. Each curve contains five points corresponding to different values of β_0 ranging from 0.1 to 0.02 as

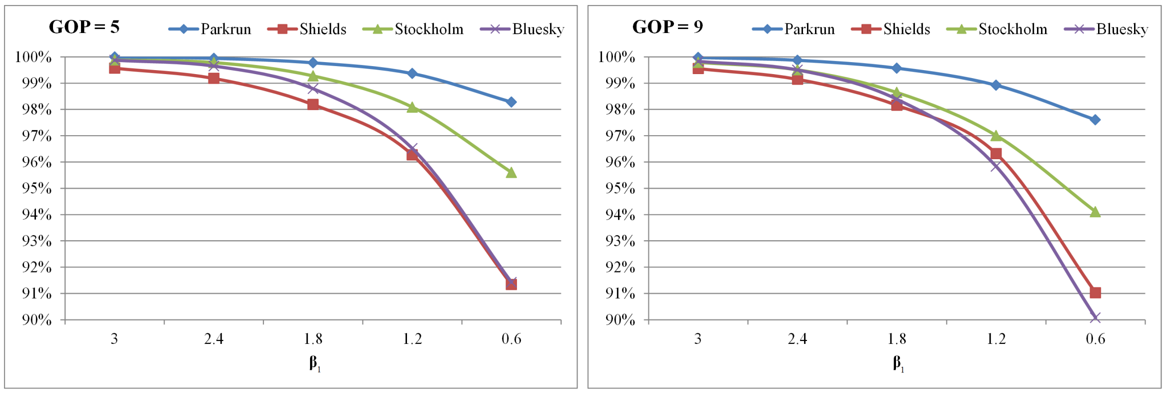


Figure 6.6: Average OR reduction with β_1 changing from 3 to 0.6 in two different GOP sizes.

indicated near the bottom of each graph above the legend for β_1.

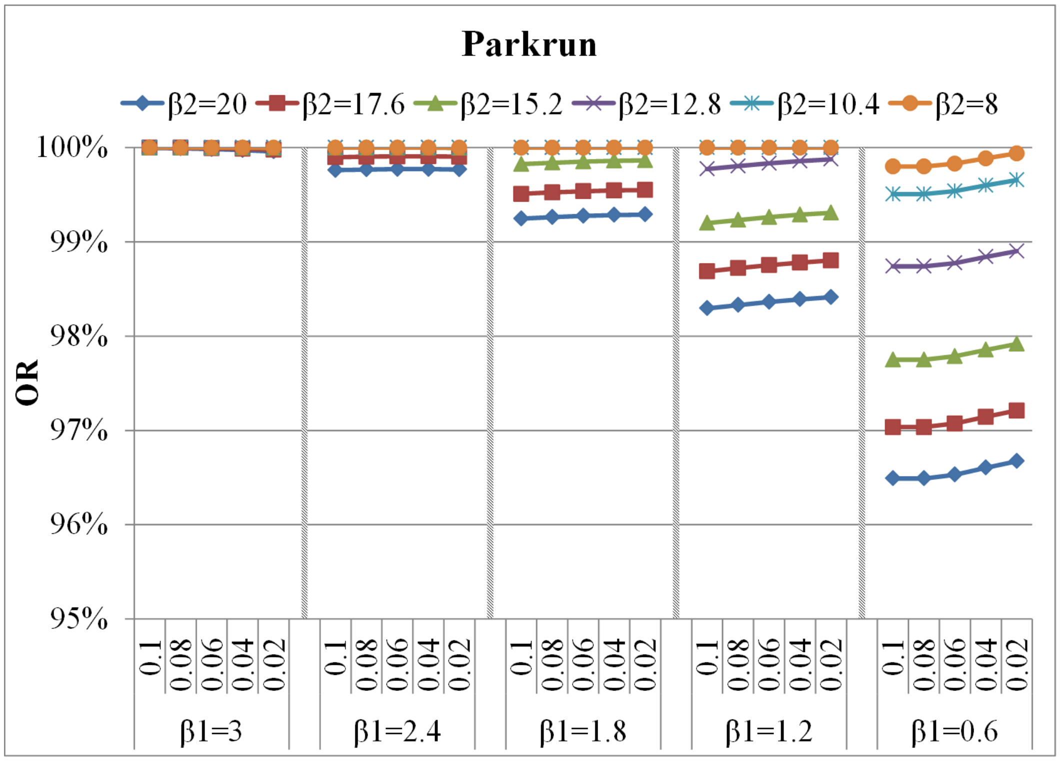
Generally, decreasing β_1 with a fixed value of β_2 will result in OR loss. Figure 6.6 shows the average OR loss when β_1 changes from 3 to 0.6. The values reported in this figure are averaged over the β_0 and β_2 values for a specific β_1.

It is mentioned in Section 4.5 that if the lower bound solution produced by the first iteration is feasible, then $OR = 100\%$. The total OR value will be reduced in each iteration if the lower bound solution is recognized as an infeasible one thanks to the trade-off that exists between RDC and CC values. Therefore RDC values should be sacrificed in order to reduce the complexity and satisfy the violated constraint. The OR decrement is dependent on the extent to which the lower bound solution is infeasible.

Similarly, decreasing β_2 has a positive effect on OR when β_1 is fixed. Allocating less budget to layer 2 enhances the chance of satisfying the constraints of the other layers int the initial solution obtained in the first step. Table 6.2 shows the changes in the average OR values when β_2 is reduced from 20 to 8 in the four video sequences. The values reported in this Table are averaged over the β_0 and β_1 for a specific β_2.

It should be noted that β_0 variation has negligible effect on OR due to the relatively small size of the base layer compared to other layers, but it is still clear in Figures 6.7-6.10 that the changes are measurable.

In summary, the OR achieved by our heuristic method is at least 84% in all tested cases with an average optimality ratio of at least 97%.



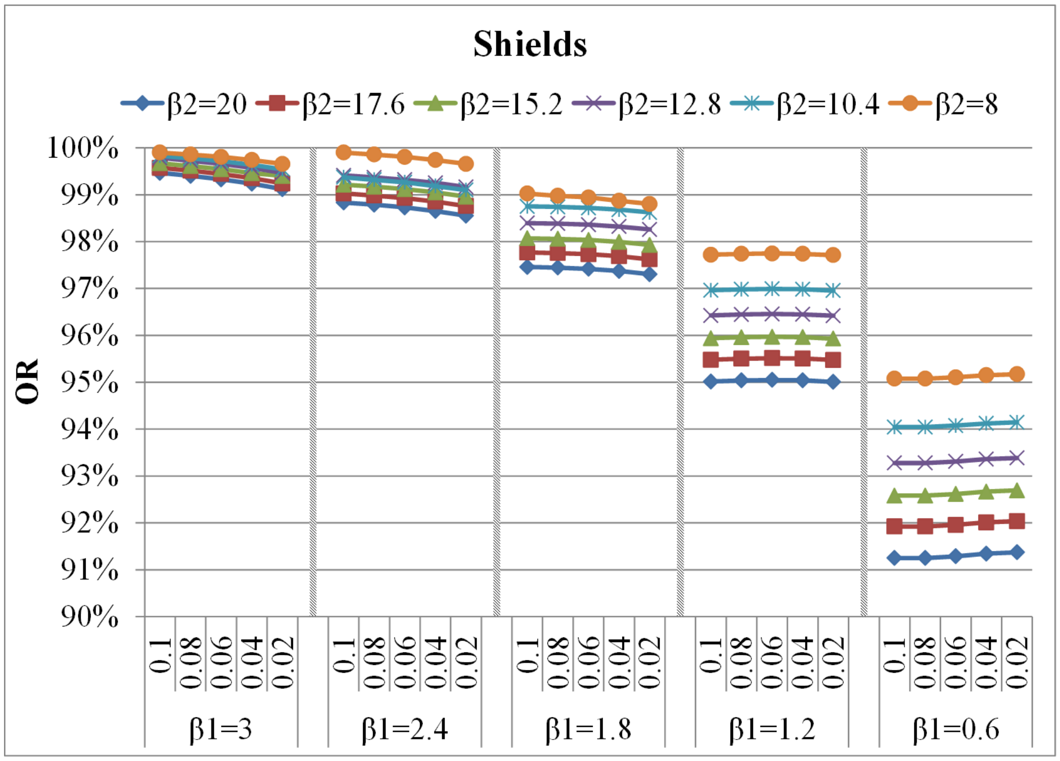
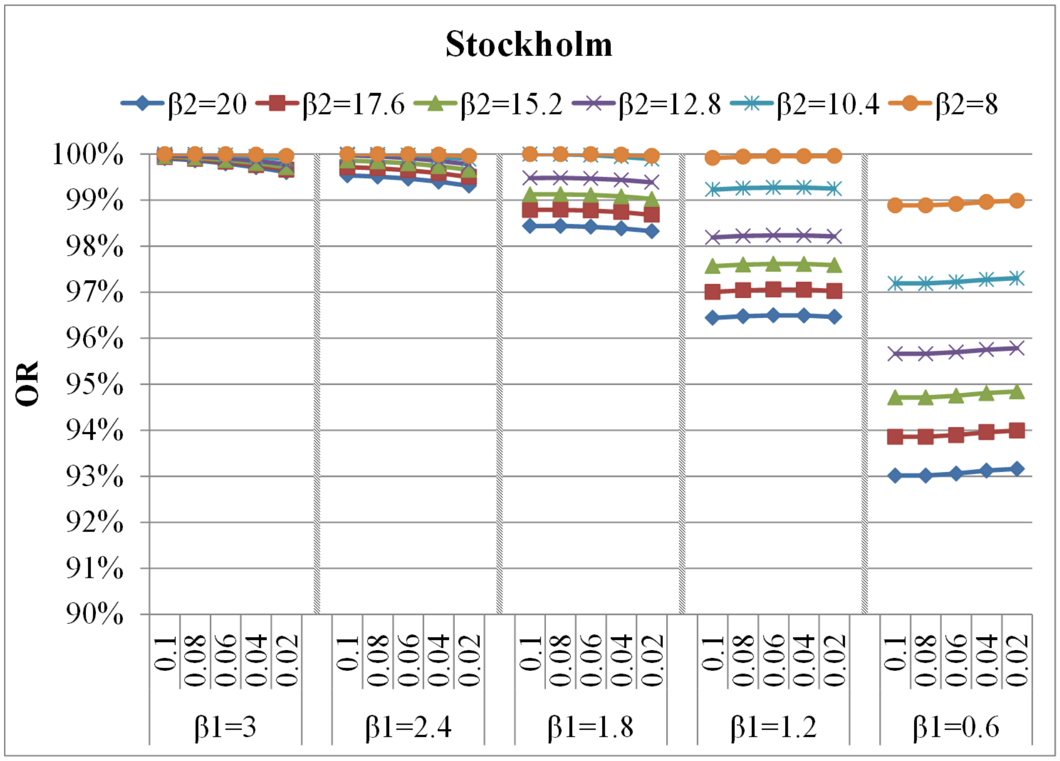


Figure 6.7: Optimality Ratio for different β_0, β_1, and β_2 for GOP size 5.



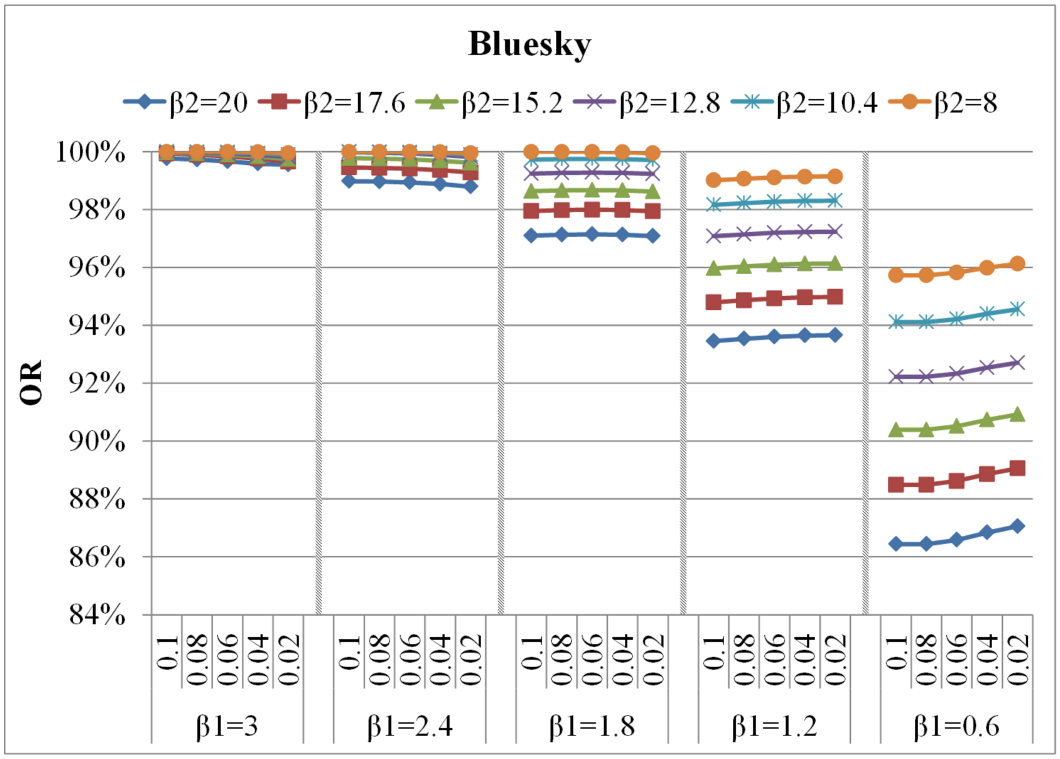
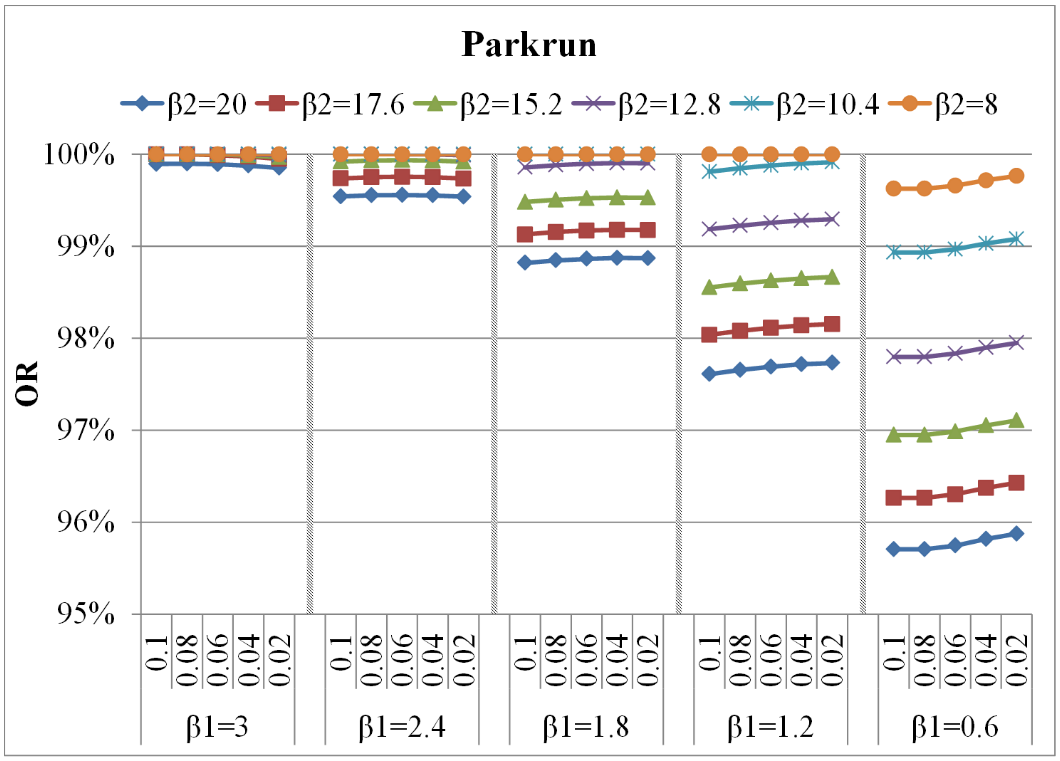


Figure 6.8: Optimality Ratio for different β_0, β_1, and β_2 for GOP size 5.



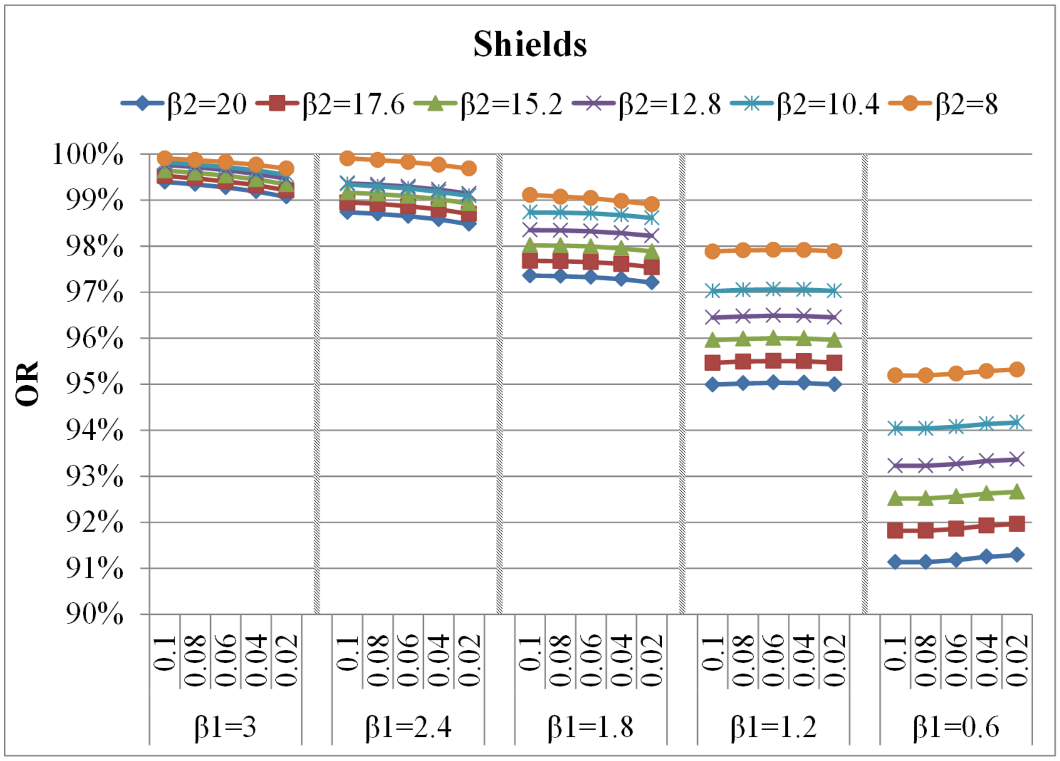
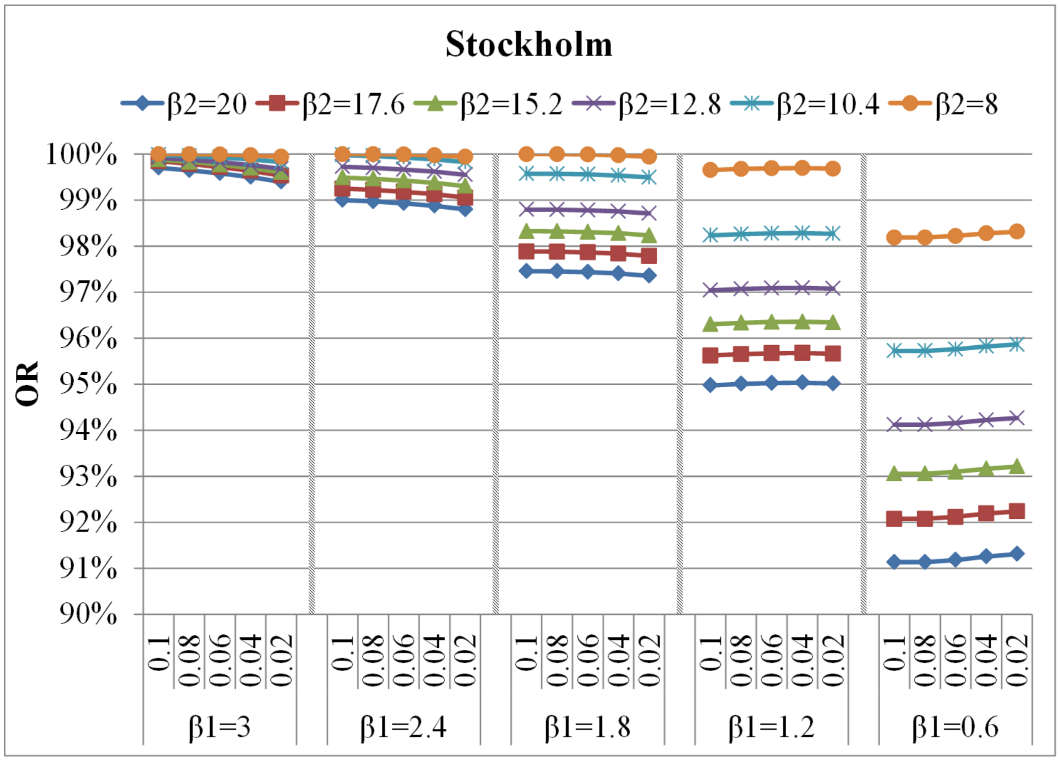


Figure 6.9: Optimality Ratio for different β_0, β_1, and β_2 for GOP size 9.



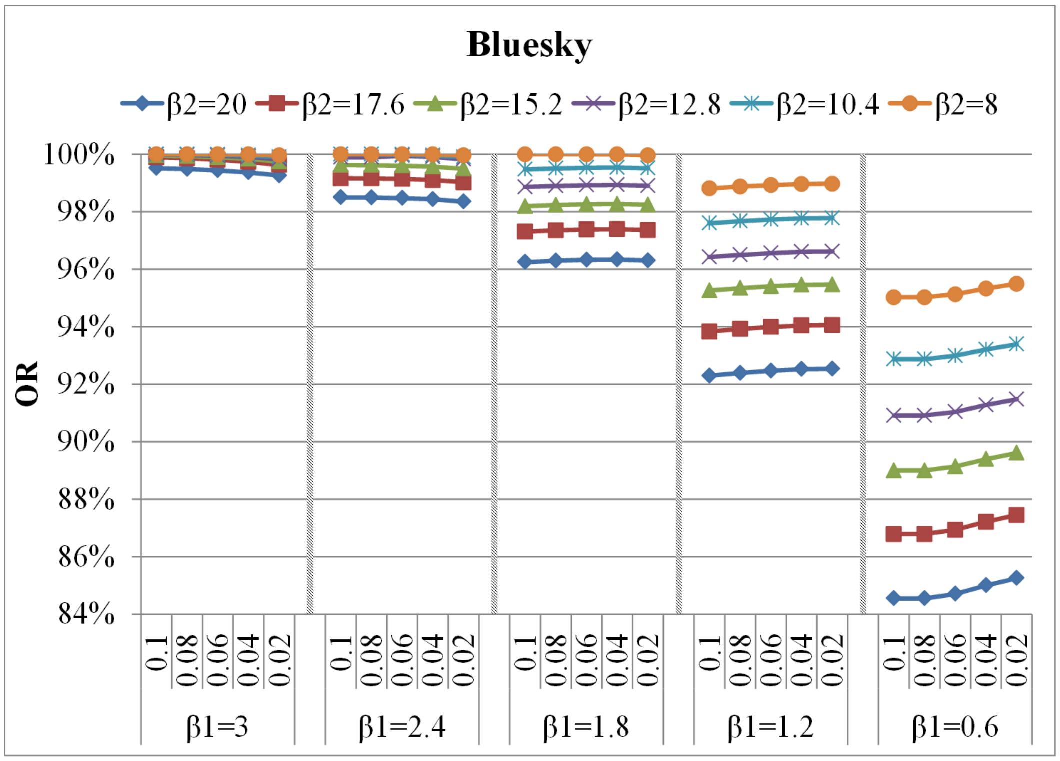


Figure 6.10: Optimality Ratio for different β_0, β_1, and β_2 for GOP size 9.

Sequence	GOP_5		GOP_9	
	$\beta_2 = 20$	$\beta_2 = 8$	$\beta_2 = 20$	$\beta_2 = 8$
Parkrun	98.8%	100%	98.3%	99.9%
Shields	96.4%	98.3%	96.3%	98.4%
Stockholm	97.4%	99.8%	96.4%	99.6%
Bluesky	95.2%	99%	94.3%	98.8%

Table 6.2: Average OR values over β_0 and β_1 with fixed β_2.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

Mobile multimedia systems are becoming increasingly power-hungry while battery technology is not keeping pace. This increases the importance of power-aware video codecs. The computational complexity of video codecs, which consists of CPU operations and memory accesses, is one of the main factors affecting power consumption. The trade-off between the computational complexity of the decoding process and the rate-distortion of the output stream is one the main features that can be tailored according to user preferences.

In this thesis, we formulated the rate-distortion optimization problems and presented efficient methods for encoder to monitor the resource requirement of the decoder for both single and multiple receiver scenarios, and we used experiments with the H.264/AVC video codec and the SVC extension of H.264 to evaluate our methods.

Our formulation of the rate-distortion optimization problem for the single receiver scenario is a multiple-choice knapsack problem. Our hybrid method is a combination of dynamic programming, scaling techniques, and greedy heuristics which attempts to find an optimal balance between the execution time and the optimality ratio for a consecutive series of video frames. Our experiments with H.264/AVC show that our method achieves up to 97% of the optimal video quality while at the same time guaranteeing that the computational complexity needed to decode the video does not exceed a specific threshold defined by a user.

The generalization of our formulation to the multiple receiver scenario is a multi-dimensional multiple-choice knapsack problem in which the number of dimensions (complexity constraints) corresponds to the number of layers. We assume that each layer has its own computational complexity threshold and rate-distortion values which are reflected in the constraints and the objective function. Our method is a combination of dynamic programming and greedy heuristics which first determines a lower bound solution and then converges towards an optimal solution by removing constraints recursively. Our experiments with the SVC extension of H.264 show that our method achieves an average of more than 97% of the optimal video quality.

7.2 Future Work

Our future work will mainly focus on expanding the current work to a heterogeneous environment with a large number of users in which each user can subscribe to a group with a complexity demand. To achieve such a system, we will create m group of users with n_i subscribers and β_i demand ($0 \leq i \leq m$). The goal is to optimize the total rate-distortion of all $m \cdot n_i$ users while satisfying the complexity demand of each single user. It is not possible to allocate a group for each costumer because of the large impact of the number of layers on the running time mentioned in Section 4.5. The number of groups (layers) and the complexity threshold (β_i) should be chosen to minimize the objective function as much as possible while satisfying the complexity constraints and solving the problem in a feasible time frame.

Another future aspect of this thesis is to achieve real-time performance. In live-stream broadcasting and video conference application, time plays a crucial factor for both transmitter and receiver. If the timing overhead of the software or hardware exceeds a particular threshold, it will cause undesirable effects on the video streaming experience such as lagging and jitter. Therefore, designing a real-time decoder complexity-aware encoding system becomes a vital contribution in such applications. It should be noted that the network packet loss or congestion is not part of our concern. Rather, we focus on reliable real-time performance of the video codec. One method that can potentially tackle this is to adopt parallel processing of a group of frames or macroblocks. Parallel encoding has the advantage of efficient CPU utilization, however it can increase the processing time by creating extra locks on the shared resources and thread switching overhead. Therefore, proper consideration must be taken to optimize this trade-off in such systems so that they can process a group of frames or macroblocks simultaneously for real-time performance. In addition, a hardware implementation of the proposed algorithm can be explored to deliver higher efficiency by removing unnecessary software code lines and customizing to algorithm's structure.

Moreover, as another avenue of future work, the battery usage in different decoder complexity aware methods can be measured precisely to study the exact amount of energy saving achieved by our proposed methods. One possible approach is to translate the time complexity to power consumption per second in the decoding device when clock frequency and circuit voltage are controlled. The following two bit-streams can be decoded and compared to observe how much power saving can be achieved.

1. Bit-stream encoded with optimum visual quality and bit-rate.
2. Bit-stream encoded with best visual quality and bit-rate while considering decoder complexity constraints with our method.

Another possible way to measure power consumption is to use a device that can report instantaneous power usage. This method can be adopted to roughly measure the power needed for decoding and rendering. Nevertheless, it suffers from the noise created by the power consumption of the operating system and other processes running in the background.

Bibliography

[1] `http://www.vidizmo.com/solutions/healthcare/health-academy/`. [Online].

[2] `http://www.streamingwell.tv/`. [Online].

[3] Global internet phenomena report. `https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf`, May 2014.

[4] Y. Abu-Lebdeh and I. Davidson. *Nanotechnology for Lithium-Ion Batteries*. Springer, 2012.

[5] A. Alinejad, R. Istepanian, and N. Philip. Medical quality of service analysis of ultrasound video streaming over LTE networks. In *XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013*, pages 1915–1918. Springer, 2014.

[6] J. Bean. Multiple choice knapsack functions. Technical report, University of Michigan, 1987.

[7] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz. Energy consumption modeling of H.264/AVC video decoding for GPP and DSP. In *Euromicro Conf. on Digital System Design (DSD)*, pages 890–897. IEEE, 2013.

[8] K. Berger and F. Galea. An efficient parallelization strategy for dynamic programming on gpu. In *2013 IEEE 27th Int. Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, pages 1797–1806, May 2013.

[9] B. Bross, W. Han, G. Sullivan, J. Ohm, and T. Wiegand. High efficiency video coding (HEVC) text spec. draft 10 (for FDIS & consent). In *JCT-VC Doc. JCTVC-L1003, 12th Meeting*, volume 1, Jan. 2013.

[10] T. Burd and R. Brodersen. Processor design for portable systems. In *Technologies for wireless computing*, pages 119–137. Springer, 1996.

[11] T. Burd, T. Pering, A. Stratakos, and R. Brodersen. A dynamic voltage scaled microprocessor system. *IEEE Journal of Solid-State Circuits*, 35(11):1571–1580, Nov 2000.

[12] G. Cernigliaro, F. Jaureguizar, J. Cabrera, and N. Garcia. Low complexity mode decision and motion estimation for H.264/AVC based depth maps encoding in free viewpoint video. *IEEE Trans. on Circuits and Systems for Video Technology*, 23(5):769–783, May 2013.

[13] J. Choi and Y. Ho. Deblocking filter algorithm with low complexity for h.264 video coding. In *Proceedings of the 9th Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, PCM '08, pages 138–147, Berlin, Heidelberg, 2008. Springer-Verlag.

[14] T. da Fonseca and R. de Queiroz. Energy-constrained real-time H.264/AVC video coding. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1739–1743, 2013.

[15] A. Greenberg and J. Zanetis. The impact of broadcast and streaming video in education. `http://www.cisco.com/web/strategy/docs/education/ciscovideowp.pdf`, March 2012.

[16] D. Grois and O. Hadar. Complexity-aware adaptive pre-processing scheme for region-of-interest spatial scalable video coding. *IEEE Trans. on Circuits and Systems for Video Technology*, PP(99):1–1, 2014.

[17] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu. Power-rate-distortion analysis for wireless video communication under energy constraints. *IEEE Trans. on Circuits and Systems for Video Technology*, 15(5):645–658, May 2005.

[18] D. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

[19] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/AVC baseline profile decoder complexity analysis. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7):704–716, 2003.

[20] M. Jamali Langroodi, J. Peters, and S. Shirmohammadi. Complexity aware encoding of the motion compensation process of the H.264/AVC video coding standard. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, pages 103–108, New York, NY, USA, 2013. ACM.

[21] M. Jamali Langroodi, J. Peters, and S. Shirmohammadi. Decoder-complexity-aware encoding of motion compensation for multiple heterogeneous receivers. *ACM Trans. on Multimedia Computing, Communications and Applications (TOMM)*, page revised version in review, Feb 2015.

[22] Joint Video Team. Advanced video coding for generic audiovisual services of ISO/IEC MPEG & ITU-T VCEG, , ITU-T rec. H.264 and ISO/IEC 14496-10 advanced video coding, edition 5.0 (incl. SVC extension). Technical report, MPEG/ITU-T, 2010.

[23] H. Jung and K. Ryoo. An intra prediction hardware architecture with low computational complexity for hevc decoder. In *Future Information Communication Technology and Applications*, pages 549–557. Springer, 2013.

[24] J. Kleinberg and E. Tardos. *Algorithm design.* Pearson Education India, 2006.

[25] S. Lee and C. C. Kuo. Complexity modeling of spatial and temporal compensations in H.264/AVC decoding. *IEEE Trans. on Circuits and Systems for Video Technology*, 20(5):706–720, May 2010.

[26] S. Lee and C. C. Kuo. H.264/avc entropy decoder complexity analysis and its applications. *Journal of Visual Communication and Image Representation*, 22(1):61–72, Jan. 2011.

[27] Y. Lee, J. Kim, and C. Kyung. Energy-aware video encoding for image quality improvement in battery-operated surveillance camera. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems,*, 20(2):310–318, 2012.

[28] C. Li, D. Wu, and H. Xiong. Delay-power-rate-distortion model for H.264 video coding. In *IEEE China Summit Int. Conf. on Signal and Information Processing (ChinaSIP)*, pages 505–509, 2013.

[29] Z. Ma, H. Hu, and Y. Wang. On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding. *IEEE Trans. on Multimedia*, 13(6):1240–1255, Dec 2011.

[30] K. Nibbelink, S. Rajopadhye, and R. McConnell. 0/1 knapsack on hardware: A complete solution. In *2007 IEEE Int. Conf. on Application-specific Systems, Architectures, and Processors*, pages 160–167, July 2007.

[31] T. Ogunfunmi, O. Ndili, and P. Arnaudov. On low power fractional motion estimation algorithms for H.264. In *Workshop on Signal Processing Systems (SiPS)*, pages 103–108. IEEE, 2012.

[32] M. Orlandic and K. Svarstad. A high-throughput and low-complexity H.264/AVC intra 16x16 prediction architecture for HD video sequences. In *2013 21st Telecommunications Forum (TELFOR)*, pages 529–532, Nov 2013.

[33] M. Orlandic and K. Svarstad. A low complexity H.264/AVC 4x4 intra prediction architecture with macroblock/block reordering. In *2013 Int. Conf. on Reconfigurable Computing and FPGAs (ReConFig)*, pages 1–6, Dec 2013.

[34] I. Richardson. *H.264 and MPEG-4 video compression: video coding for next-generation multimedia.* 2004.

[35] A. Rodriguez, A. Gonzalez, and M. Malumbres. Hierarchical parallelization of an H.264/AVC video encoder. In *International Symposium on Parallel Computing in Electrical Engineering*, pages 363–368, 2006.

[36] M. Ronchetti, C. Zhu, Y. Li, and X. Niu. Perspectives of the application of video streaming to education. *Streaming Media Architectures, Techniques, and Applications: Recent Advances*, pages 411–428, 2011.

[37] S. Samii, M. Selkala, E. Larsson, K. Chakrabarty, and Z. Peng. Cycle-accurate test power modeling and its application to SoC test architecture design and scheduling. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(5):973–977, May 2008.

[38] S. Sankaraiah, H. Lam, J. Abdullah, and C. Eswaran. GOP level parallelism on H.264 video encoder for multicore architecture. In *Int. Conf. on Circuits, System and Simulation (IPCSIT)*, volume 7, pages 127–132, 2011.

[39] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 17(9):1103–1120, Sept 2007.

[40] M. Semsarzadeh, M. Jamali Langroodi, M. Hashemi, and S. Shirmohammadi. Complexity modeling of the motion compensation process of the H.264/AVC video coding standard. In *Int. Conf. on Multimedia and Expo (ICME)*, pages 925–930. IEEE, 2012.

[41] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao. Real-time video coding under power constraint based on H.264 codec. In *in Proc. Visual Communications and Image Processing (VCIP)*, volume 6508, pages 1–12. International Society for Optics and Photonics, 2007.

[42] Team Joint Video. H.264/AVC reference software version JM 16.2. `http://iphome.hhi.de/suehring/tml/download/old_jm`, 2011.

[43] M. Tehrani Moayyed and A. Eftekhari Moghadam. Power-aware/real-time H.264 video decoding with control theoretic approach. In *13th Iranian Conference on Fuzzy Systems (IFSC)*, pages 1–6. IEEE, 2013.

[44] T. J. Video. H.264/SVC reference software (JSVM 9.19) and manual. *CVS sever at garcon. ient. rwth-aachen. de*, 2011.

[45] R. Wang, K. Mattick, and E. Dunne. Medical students perceptions of video-linked lectures and video-streaming. *Research in Learning Technology*, 18(1), 2010.

[46] Y. Wang, W. Zhang, Z. Zhang, and P. An. A low complexity deblocking filtering for multiview video coding. *IEEE Trans. on Consumer Electronics*, 59(3):666–671, Aug 2013.

[47] D. West. How mobile devices are transforming healthcare. *Issues in technology innovation*, 18:1–14, 2012.

[48] D. West. The evolution of video streaming and digital content delivery. 2014.

[49] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien. Joint Draft 10 of SVC amendment. Doc. JVT-W201, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, San Jose, CA, USA, Apr. 2007.

[50] C. Xiang, S. Wu, W. Wang, and F. Zhao. A research on charge and discharge strategy of hybrid batteries based on the electrochemical characteristics. In *Proceedings of 2013 35th International Telecommunications Energy Conference 'Smart Power and Efficiency' (INTELEC),*, pages 1–5. VDE, Oct 2013.

[51] J. Yang and B. An. Low-complexity rate-distortion optimization for robust H.264 video coding. In *2011 IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 1–4, July 2011.

[52] M. Yoshio, R. Brodd, and A. Kozawa. *Lithium-Ion Batteries.* Springer, 2009.

[53] H. Zhong, S. Shen, Y. Fan, and X. Zeng. A low complexity macroblock layer rate control scheme base on weighted-window for h.264 encoder. In *Proceedings of the 18th International Conference on Advances in Multimedia Modeling*, MMM'12, pages 563–573, Berlin, Heidelberg, 2012. Springer-Verlag.

[54] H. Zrida, A. Jemai, A. Ammari, and M. Abid. High level H.264/AVC video encoder parallelization for multiprocessor implementation. In *Design, Automation and Test in Europe Conference and Exhibition*, pages 940–945. IEEE, 2009.