7-1-2016

Joint Control of Quality, Complexity, and Rate for HEVC Intra Mode

CONG ZONG

CONG ZONG

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

MARIOS PATTICHIS , Chairperson

RAMIRO JORDAN

VINCE CALHOUN

Joint Control of Quality, Complexity, and Rate for HEVC Intra Mode

by

CONG ZONG

M.E., Computer Science, Tianjin University, 2014

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2016

Dedication

To my parents, Gangjun and Lifen,
their huge support help me
in the way to seek truth and brilliance.

Acknowledgments

This material is based upon work supported by the National Science Foundation under NSF AWD CNS-1422031. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

I would like to thank my advisor, Prof Marios Pattichis, for his long-time support and help during my master degree. His enthusiasm for knowledge and research truly encourage and inspire me.

I would like to thank my lab mate Yuebing Jiang. Without his brilliant idea and his original work on DRASTIC, I would not get this beautiful result.

Joint Control of Quality, Complexity, and Rate for HEVC Intra Mode

by

CONG ZONG

M.E., Computer Science, Tianjin University, 2014

M.S., Computer Engineering, University of New Mexico, 2016

Abstract

The thesis develops robust algorithms that are used to provide joint control of reconstructed video quality, computational complexity, and compression rate for intra-mode video encoding in HEVC. The approach uses a configuration parameter that controls the partitioning of the coding tree unit (CTU) so as to provide for finer control of the encoding process. By jointly sampling the quantization parameter and the configuration mode, the approach generates a finely-sampled, Pareto-optimal, rate-quality-performance surface.

A robust, spatially-adaptive control algorithm is proposed for solving the minimum bitrate, maximum quality, and minimum computational complexity optimization problems. The approach is demonstrated on 17 videos from four different classes. For all videos, the approach provides for substantial savings in computational complexity and bitrate, and slight improvements in image quality. Furthermore, the thesis demonstrates dynamic switching between the low, medium and high profiles within the same video.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

High-efficiency video coding (HEVC) has provided substantial improvements to video compression through the introduction of new coding tools. Examples of new HEVC technologies include recursive coding/transform units complex intra prediction modes, and asymmetric inter prediction unit division. Overall, HEVC aims at a 50% bit rate reduction at equivalent video quality levels [1]. Unfortunately, bitrate performance improvements come at substantial increase in computational complexity.

Recently, there have been many efforts to reduce computational complexity. For reducing inter encoding complexity, we have the introduction of several configuration modes as discussed in [2]. For reducing intra encoding complexity, we have the introduction of rough mode sets (RMS,[3]), gradient based intra prediction [4], and coding unit(CU) depth control[5]. To introduce our approach, let $\mathtt{T}$ denote encoding time per frame, $\mathtt{R}$ denote the number of bits per sample, and $\mathtt{Q}$ denote a measure of video quality (e.g., PSNR of average SSIM). Furthermore, let $\mathcal{C}$ denote the set of all possible video encoding configurations. We want to design methods that can solve $\min_{c \in \mathcal{C}} (\mathtt{T}, \mathtt{B}, -\mathtt{Q})$, where the negative sign in front of $\mathtt{Q}$ is needed in order to express our need to maximize quality (and hence minimize $-\mathtt{Q}$).

In order to jointly control $\mathtt{T}, \mathtt{R}$ and $\mathtt{Q}$, we provide bounds on each one of them. For improving performance and guarantee computations within specific time limits, let $\mathtt{T}_{\mathtt{max}}$ denote an upper bound on the encoding time. Similarly, for communicating within a specific bandwidth, let $\mathtt{B}_{\mathtt{max}}$ denote an upper bound on the available bits per pixel. Then, to guarantee a minimum level of quality, let $\mathtt{Q}_{\mathtt{min}}$ denote a lower bound on the encoded video quality. Thus, in general, we are only interested in encoding configurations that jointly satisfy: $(\mathtt{T} \leq \mathtt{T}_{\mathtt{max}}) \& (\mathtt{B} \leq \mathtt{B}_{\mathtt{max}}) \& (\mathtt{Q} \geq \mathtt{Q}_{\mathtt{min}})$. Clearly, for very low values of $\mathtt{T}_{\mathtt{max}}, \mathtt{B}_{\mathtt{max}}$ and high values of $\mathtt{Q}_{\mathtt{min}}$, it may not be possible to find an encoding configuration that satisfies the constraints. On the other hand, for very high values of $\mathtt{T}_{\mathtt{max}}, \mathtt{B}_{\mathtt{max}}$ and low values of $\mathtt{Q}_{\mathtt{min}}$, we can get a large number of encoding configurations that can satisfy the constraints. Thus, it makes sense to optimize for one of the objectives while placing bounds on the other two. We thus consider three optimization modes:

- The *maximum performance mode* provides the best computational performance by minimizing encoding time. An acceptable, optimal encoding configuration is obtained by solving:

$$\min_{c \in \mathcal{C}} \mathtt{T} \quad \text{subject to:} \quad (\mathtt{Q} \geq \mathtt{Q}_{\mathtt{min}}) \ \& \ (\mathtt{R} \leq \mathtt{R}_{\mathtt{max}}). \tag{1.1}$$

- The *minimum rate mode* reduces bitrate requirements without sacrificing quality or slowing down encoding time to an unacceptable level. The optimal configuration requires the solution of:

$$\min_{c \in \mathcal{C}} \mathtt{R} \quad \text{subject to:} \quad (\mathtt{Q} \geq \mathtt{Q}_{\mathtt{min}}) \ \& \ (\mathtt{T} \leq \mathtt{T}_{\mathtt{max}}). \tag{1.2}$$

- The *maximum quality mode*: provides the best possible quality without exceeding bitrate or computational requirements. The optimal encoding is selected by solving:

$$\max_{c \in \mathcal{C}} \mathtt{Q} \quad \text{subject to:} \quad (\mathtt{T} \leq \mathtt{T}_{\mathtt{max}}) \ \& \ (\mathtt{R} \leq \mathtt{R}_{\mathtt{max}}). \tag{1.3}$$

The modes given by equations (1.1)-(1.3) can be used to describe a large number of different, practical, scenarios. For example, for video streaming applications, we can simply set $\mathtt{T_{max}}$ to $\mathtt{T_{max}} = 1/\mathtt{fps}$ where $\mathtt{fps}$ denotes the number of frames per second at which the video is generated. We can also adapt to a time-varying communications channel by setting $\mathtt{R_{max}}$ to the time-varying, available bandwidth.

The methods and solutions proposed in this manuscript represent a substantial extension over prior research centered on dynamically reconfigurable hardware. A Dynamically Reconfigurable Architecture System (DRASTIC) for motion JPEG was described in [6] and earlier versions in conference papers [7], [8]. In [6], a dynamically reconfigurable DCT architecture was used to control dynamic power, bitrate, and quality. Hardware architectures that employ multi-objective optimization have been presented for the Discrete Periodic Radon Transform (DPRT) [9] (see earlier work in [10], [11]), the parallel pixel processors [12], and 2D filterbanks [13] (also see [14]). Dynamically reconfigurable filterbanks for video processing were presented in [15].

Earlier research also focused on the development of hardware cores for HEVC and H.264. Examples include a hardware architecture for intra-prediction for HEVC [16] and H.264 deblocking filters [17]. An attempt to optimize the quantization table to maximize perceptual quality has been developed in [18]. An earlier conference paper related to the current manuscript has been presented in [19], [20].

The thesis extends prior research by developing algorithms for jointly controlling the rate-quality-performance surface for intra-prediction applications. The approaches introduces a hierarchical parameterization of the partitioning of the coding unit so as to provide a dense sampling of the rate-quality-performance control surface. The thesis develops a model of the rate-quality-performance surface that is used to develop algorithms that can find appropriate encodings for the minimum bitrate, maximum quality, and maximum performance modes. The approach leads to substantial improvements over the standard use of the HEVC intra-encoding modes.

1.1 Thesis Statement

The thesis will develop a new methodology for fine, joint control of rate-quality-performance for HEVC intra-encoding applications. The new optimization approach is expected to yield significant improvements over the standard HEVC encoding approaches.

1.2 Contributions

The primary contribution of the thesis include:

- *Hierarchical coding unit (CU) partitioning for fine, joint control of rate-quality-performance:* Intra-encoding control is achieved by controlling the minimum size of the coding unit (CU). The minimum size encoding parameter ensures hierarchical partitioning. The minimum size ranges from 0 to 13, where 0 does not allow any partitioning (minimum size = 64×64), 1 supports top-level partitioning, and 13 supports the finest possible partitioning (minimum size = 4×4). The set of possible partitions is hierarchical in the sense that for minimum sizes i, j with $i < j$, j represents a finer partition of i. As a result, an increase in the minimum code size will always result in better coding performance since we have more choices. Thus, increasing the minimum code size increases quality, increase computational complexity, and bitrate. Similarly, decreasing the minimum code size will decrease quality, computational complexity, and bitrate.

- *Static and dynamic control of rate-quality-performance:* The thesis develops a model of how the rate-quality-performance surface depends on the minimum coding size and `QP` and uses the model to implement the minimum bitrate, maximum quality, and maximum performance modes. The approach also allows dynamic switching between modes.
- *System implementation validation on 17 standard video sequences:* The system is implemented using HM-11.0 and validated on 17 standard video sequences. For the same performance targets, the proposed approach achieves significant improvements over the original HEV st7C encoder for the same performance targets.

1.3 Thesis Overview

The remainder of the thesis is organized into 8 chapters. The first chapter provides an introduction, the thesis statement, a list of contributions, and a summary of the rest of the thesis.

Prior research in rate control for HEVC is given in chapter 2. In chapter 2 , we discuss three rate control algorithms implemented in the HEVC HM model: the unified RQ model [21, 22], the R-lambda model [23], the SATD model [24]. Chapter 3 summarizes the intra-encoding model and associated parameters. The intra-encoding is controlled by setting the `QP` and the minimum coding unit (CU) size. Here, note that increasing the CU size will simultaneously reduce quality, encoding time, and bitrate.

Chapter 5.3 provides details the joint control of rate, quality, and performance. The basic approach involves modeling the rate-quality-performance surface, model update, and fine control by adjusting `QP` and the minimum coding unit size.

Chapter 5 provides a summary of the basic rate-quality-performance results. The results are presented on 17 video sequences using all of the optimization modes. For the results, we consider low, medium, and high constraints on bitrate, quality, and performance.

Chapter 6 provides results based on dynamic adaptation between different modes. Dynamic adaptation is initialized using a model derived from the video database. The initial model is adapted to the current input video and used for switching between modes.

Chapter 7 discusses the contribution of the current approach compared against different approaches.

Chapter 8 provides concluding remarks and provides possible, future work.

Chapter 2

Rate Control in HEVC

In this chapter, we provide a summary of the rate-control methods that were implemented in the HEVC reference software. In section 2.1, we discuss the unified RQ model [21, 22]. In section 2.2, we summarize the R-lambda model [23]. Then, in section 2.3, we present sum of absolute tranformed distance (SATD) based rate control [24].

2.1 Rate control based on the Unified RQ model

The unified RQ (URQ) model is used in HM-6.1 [21, 22]. The proposed rate control algorithm works at the GOP level, the frame level, and the largest coding unit (LCU) level. The basic idea is to select the quantization parameter to meet specific bitrate requirements.

In what follows, let i be used to refer to the i-th GOP. Also, let j be used to refer to the j-th video frame.

Let $\mathtt{B_i(j)}$ denote the number of budget bits that are available to cover the needs of the j-th and the remaining frames in the GOP. Similarly, let $\mathtt{V_i(j)}$ denote the number of bits stored in the virtual buffer, $\mathtt{b_i(j-1)}$ denote the number of actual number bits required for the $(j-1)$-th frame, and $\mathtt{R_{AvgPic}}$ denote an estimate of the average number of bits per frame. At the GOP level, we have:

$$\mathtt{B_i(j)} = \begin{cases} \mathtt{R_{AvgPic}} \cdot \mathtt{N_{GOP}} - \mathtt{V_i(j)}, & \mathtt{j} = 0, \\ \mathtt{B_i(j-1)} - \mathtt{b_i(j-1)}, & \mathtt{j} = 1, 2, \dots, \mathtt{N_{GOP}}. \end{cases} \tag{2.1}$$

Next, let $\mathtt{bpp_i(j)}$ denote the average number of bits per pixel, $\mathtt{T_i(j)}$ denote the total number of bits, $\mathtt{N_{pixels,i}(j)}$ denote the total number of bits. Then, at the frame level, we clearly have:

$$\mathtt{bpp_i(j)} = \frac{\mathtt{T_i(j)}}{\mathtt{N_{pixels,i}(j)}}. \tag{2.2}$$

Let $\mathtt{MAD_{pred,i}(j)}$ denote the mean absolute deviation (MAD) for the j-th frame. To predict $\mathtt{MAD_{pred,i}(j)}$, let $\mathtt{MAD_{actual,i}(j-1-M)}$ denote the actual MAD for the $(j-1-M)$-th frame. We can estimate $\mathtt{MAD_{pred,i}(j)}$ using:

$$\mathtt{MAD_{pred,i}(j)} = \mathtt{a_1} \cdot \mathtt{MAD_{actual,i}(j-1-M)} + \mathtt{a_2} \tag{2.3}$$

where $M = 1$ when using a hierarchical reference structure, and a_1, a_2 are adaptively estimated.

Rate control at the j-th frame level is achieved by selecting appropriate values for $\mathtt{QP_i(j)}$. Using the target bitrate $\mathtt{bpp_i(j)}$, estimated MAD $\mathtt{MAD_{pred,i}(j)}$, the $\mathtt{QP_i(j)}$ can be selected by solving a second-order quadratic equation for $1/\mathtt{QP_i(j)}$ given by:

$$\mathtt{bpp_i(j)} = \alpha \cdot \frac{\mathtt{MAD_{pred,i}(j)}}{\mathtt{QP_i(j)}} + \beta \cdot \frac{\mathtt{MAD_{pred,i}(j)}}{\mathtt{QP_i^2(j)}} \tag{2.4}$$

where α, β are adaptively estimated.

Similarly, let $\mathtt{bpp_i(j,m)}$ denote the number of bits per pixel for the m-th CTU. Then, as done at the frame level, rate control at the largest coding unit (LCU) can be achieved by selecting $\mathtt{QP_i(j,m)}$ so as to satisfy:

$$\mathtt{bpp_i(j,m)} = \alpha \cdot \frac{\mathtt{MAD_{pred,i}(j)}}{\mathtt{QP_i(j,m)}} + \beta \cdot \frac{\mathtt{MAD_{pred,i}(j)}}{\mathtt{QP_i^2(j,m)}}. \tag{2.5}$$

2.2 Rate control based on the R-lambda Model

Rate control based on the R-lambda model was implmented in HM-9.0 [23]. Here, the `QP` is determined by the target rate allocated top from the top GOP level down to LCU level.

Let $\mathtt{R_{PicAvg}}$ denote the target bitrate for each video frame. We have that:

$$\mathtt{R_{PicAvg}} = \frac{\mathtt{R_{tar}}}{\mathtt{f}} \tag{2.6}$$

where $\mathtt{f}$ denotes the number of frames per second, and $\mathtt{R_{tar}}$ denotes the target number of bits per second.

Let $\mathtt{SW}$ denote the size of a smoothing window. The goal is to achieve $\mathtt{R_{PicAvg}}$ by distributing any deviation from the target to the remaining frames. This is accomplished by defining a new target for the average frame in the GOP using:

$$\mathtt{T_{AvgPic}} = \mathtt{R_{PicAvg}} + \frac{\mathtt{R_{PicAvg}} \cdot \mathtt{N_{coded}} - \mathtt{R_{coded}}}{\mathtt{SW}} \tag{2.7}$$

where $\mathtt{T_{AvgPic}}$ is the revised target, $\mathtt{R_{coded}}$ denotes the total number of bits used for $\mathtt{SW}$ video frames, and $\mathtt{N_{coded}}$ denotes the number of encoded video frames. Over the entire GOP, the target number of bits becomes:

$$\mathtt{T_{GOP}} = \mathtt{T_{AvgPic}} \cdot \mathtt{N_{GOP}} \tag{2.8}$$

where $\mathtt{N_{GOP}}$ denotes the number of video frames that make up the GOP.

Once the GOP target has been determined, we can allocate bits at the individual frame level. Let $\mathtt{Coded_{GOP}}$ denote the number of bits already allocated. Furthermore, let $\mathtt{w_i}$ denote the weight to be used for allocating bits for the *i*-th video frame. The number of bits to be allocated to the current video frame are then given by:

$$\mathtt{T_{CurrPic}} = \frac{(\mathtt{T_{GOP}} - \mathtt{Coded_{GOP}}) \cdot \mathtt{w_{CurrPic}}}{\sum_{\mathtt{NotCodedPic}} \mathtt{w_i}}. \tag{2.9}$$

Using the number of bits allocated to the entire picture, we can determine the number of bits to be allocated to each CTU. First, for each CTU, determine the MAD using:

$$\mathtt{MAD}_{\mathtt{CTU}} = \frac{1}{\mathtt{N}_{\mathtt{pixels}}} \sum_{\mathtt{i}} |\mathtt{pred}_{\mathtt{i}} - \mathtt{org}_{\mathtt{i}}| \tag{2.10}$$

where $\mathtt{pred}_{\mathtt{i}}$, $\mathtt{org}_{\mathtt{i}}$ determine the predicted (encoded) pixel and original pixel values. An allocation weight for each CTU is calculated using:

$$\mathtt{w}_{\mathtt{CTU}} = \mathtt{MAD}^2_{\mathtt{CTU}}. \tag{2.11}$$

Using the weights, the target number of bits for each CTU is then given by:

$$\mathtt{T}_{\mathtt{CurrCTU}} = \frac{(\mathtt{T}_{\mathtt{CurrPic}} - \mathtt{Bit}_{\mathtt{header}} - \mathtt{Coded}_{\mathtt{Pic}}) \cdot \mathtt{w}_{\mathtt{CurrCTU}}}{\sum_{\mathtt{NotCodedCTU}} \mathtt{w}_{\mathtt{i}}} \tag{2.12}$$

where $\mathtt{Bit}_{\mathtt{header}}$ denotes the number of bits allocated to the header, and $\mathtt{Coded}_{\mathtt{Pic}}$ denotes the number of bits that have already been used in the encoding.

To determine the $\mathtt{QP}$, we use the R-lambda model. The R-lambda model can be used to perform RD optimization at both the CTU and picture levels. The model assumes a rate-distortion (RD) relationship given by:

$$\mathtt{D}(\mathtt{R}) = \mathtt{C} \cdot \mathtt{R}^{-\mathtt{K}} \tag{2.13}$$

where $\mathtt{C}, \mathtt{K}$ denote constants, $\mathtt{R}$ denotes the rate, and $\mathtt{D}$ denotes the distortion. Starting from (2.13), we have:

$$\begin{aligned} \lambda &= -\frac{\partial D}{\partial R} \\ &= \mathtt{C} \cdot \mathtt{K} \cdot \mathtt{R}^{-\mathtt{K}-1} \\ &= \alpha_0 \cdot \mathtt{R}_0^{\beta} \\ &= \alpha \cdot \mathtt{bpp}^{\beta}. \end{aligned} \tag{2.14}$$

The $\mathtt{QP}$ is then estimated using:

$$\mathtt{QP} = 4.2005 \ln \lambda + 13.7122. \tag{2.15}$$

After encoding one CTU or one picture, α and β of (2.14) are updated using:

$$\alpha_{\texttt{new}} = \alpha_{\texttt{old}} + \delta_\alpha(\ln \lambda_{\texttt{real}} - \ln \lambda_{\texttt{comp}}) \cdot \alpha_{\texttt{old}} \tag{2.16}$$

$$\beta_{\texttt{new}} = \beta_{\texttt{old}} + \delta_\beta(\ln \lambda_{\texttt{real}} - \ln \lambda_{\texttt{comp}}) \cdot \ln \texttt{bpp}_{\texttt{real}} \tag{2.17}$$

where:

$$\lambda_{\texttt{comp}} = \alpha_{\texttt{old}} \cdot \texttt{bpp}_{\texttt{real}}^{\beta_{\texttt{old}}}, \tag{2.18}$$

and $\delta_\alpha = \texttt{0.1}$ and $\delta_\beta = \texttt{0.05}$.

2.3 Intra Frame Rate Control Based on SATD

SATD based intra frame rate control is implemented in HM-10.0 [24]. The approach is also based on an R-lambda model.

To introduce the approach, we begin with defining a relationship between the rate and the complexity of each CTU. Within each CTU, let $h_{i,j}$ denote the Hadamard transform coefficients computed over 8×8 image blocks. Then, SATD captures CTU complexity as given by:

$$\texttt{C} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |h_{i,j}|. \tag{2.19}$$

Using $\texttt{C}$, for a target rate $\texttt{R}_{\texttt{target}}$, we have:

$$\lambda = \alpha \cdot \left(\frac{\texttt{C}}{\texttt{R}_{\texttt{target}}} \right)^{\beta} \tag{2.20}$$

which leads to an expression of $\texttt{QP}$ given by:

$$\texttt{QP} = 4.2005 \cdot \ \ln \lambda + 13.7122. \tag{2.21}$$

The λ parameter is kept constant for the entire frame. To update λ, we update α, β in (2.20) using:

$$\alpha = \alpha \cdot e^{\Delta\lambda} \tag{2.22}$$

$$\beta = \beta + \frac{\Delta\lambda}{\ln(C/\mathtt{R}_{\mathtt{real}})} \tag{2.23}$$

where:

$$\Delta\lambda = \delta \cdot \beta \cdot (\ln \mathtt{R}_{\mathtt{real}} - \ln \mathtt{R}_{\mathtt{target}}) \tag{2.24}$$

and δ is a scaling parameter (set to 0.25 for our simulations). Then, the updated α, β are used to update λ using (2.20) and a new value for $\mathtt{QP}$ using (2.21).

For bit allocation, we need to determine $\mathtt{R}^{\mathtt{CTU}}_{\mathtt{target}}(i)$ which represents the target number of bits for the i-th CTU within each video frame. Let $\tilde{\mathtt{R}}_{\mathtt{left}}$ denote the remaining number of bits for the frame. Also, let $w(i)$ denote the weight allocated to the i-the CTU. We have:

$$\mathtt{R}^{\mathtt{CTU}}_{\mathtt{target}}(\mathtt{i}) = \mathtt{w}(\mathtt{i}) \cdot \tilde{\mathtt{R}}_{\mathtt{left}} \tag{2.25}$$

where the weights are given by:

$$\mathtt{w}(\mathtt{i}) = \frac{\mathtt{C}^{\mathtt{CTU}}(\mathtt{i})}{\sum_{j=i}^{M-1} \mathtt{C}^{\mathtt{CTU}}(\mathtt{j})} \tag{2.26}$$

and M denotes the total number of CTUs in the coded frame.

Chapter 3

Hierarchical coding

In this Chapter, we introduce a parametrization for controlling the partitioning of the coding tree unit (CTU). In terms of rate, quality, and performance, we demonstrate that the parametrization leads to a Pareto optimal surface. We also introduce a linear model for describing the relationship between the objectives and the parameters.

3.1 HM Intra Encoding

3.1.1 Basic ideas

We begin with basic defintions based on [1]. Color images are divided into coding tree units (CTUs) of equal squared sizes. Each CTU consits of $L \times L$ luma coding tree blocks (CTBs) and $(L/2) \times (L/2)$ chroma CTBs.

Blocks can be subdivided into smaller blocks. CTBs can be further subdivided into coding blocks (CBs). The CBs can be subdivided into prediction blocks (PBs). For intra-mode encoding, as described here, the PBs are set to be equal to the CBs, except for the smallest CBs. The smallest CBs can be further subdivided into smaller

PBs so as to allow for a finer subdivision. The combination of luma and chroma PBs form the prediction units (PUs).

Subdivision always supports quadtree decompositions. For quadtree decompositions, each square is split into 4 smaller squares. The approach is recursive. Each square can be further subdivided into its 4 constituent squares. In addition to quadtree decompositions, the subdivision of CBs into PBs also supports rectangular tiles as depicted in Fig. 3 of [1].

For residual encoding, the CBs can be further subdivided into transform blocks (TBs). This partitioning only supports quadtree decompositions. Thus, the resulting quadtree decomposition uses CBs and their constituent TBs.

3.1.2 Intra prediction using Rough Mode Sets

The HM11.0 reference software implementation [25] makes use of a rough mode set (RMS) for Luma prediction modes. The idea is to simplify the search for an optimal partitioning of the coding units through a simplified rate-distortion model. RMS includes 8 modes for 4×4 and 8×8 Coding Units and 3 modes for other CUs [26, 3]. Initially, the sum of absolute Hardmarf transform coefficients are used as a measure of distortion and the initial mode as a measure of rate. The best prediction mode is determined based on the RMS where the distortion is measured as the sum of square errors in the reconstruction and the rate based on the number of bits used when the largest transform unit (TU) is deployed. Due to the reduction in the number of chroma pixels, RMS is not used for chroma predition. Instead, for chroma prediction, the best croma prediction mode is selected from 5 possible modes based on their RD performance. Here, the RD performance is once again measured in terms of the number of bits for the largest TU and the sum of squared errors in the reconstruction.

An exhaustive subdivision process is used for determining the optimal partitioning of the coding blocks into the transform blocks. The final partition is based on the best RD performance.

The reconstructed image quality can be further enhanced using a deblocking filter (DBF) and sample adaptive offests (SAO). DBF and SAO are not used in our approach since they will slow-down the decoding process.

3.2 Scalable Partitioning of the Coding Tree Unit

In this section, we introduce a scalable partitioning of the coding tree unit. The basic approach can be used to generate a Pareto-optimal surface of the rate, quality, and performance parameters.

Consider a full, quadtree decomposition of the coding tree unit. Each block is subdivided into four constituent subblocks. We then list all of the blocks in a breadth first fashion and assign a processing id to each one as listed in Fig. 3.1.

In order to model the partitioning, we introduce a configuration parameter: `Config`. Initially, for `Config` $= 0$, no splitting is allowed. For the first full split, we assign `Config` $= 5$ (see [20] for other cases). In terms of splitting, we have the following additional splittings:

`Config` $= 6$: Split first 32×32 block into 16×16 blocks.

`Config` $= 7$: Split first two 32×32 blocks into 16×16 blocks.

`Config` $= 8$: Split three 32×32 blocks into 16×16 blocks.

`Config` $= 9$: Split all 32×32 blocks into 16×16 blocks.

Similarly, `Config` $= 10, 11, 12, 13$ refers to splitting the first, first two, first three, and all four 16×16 blocks. We present the case of `Config` $= 6$ in Fig. 3.1. We have the splitting of the original 64×64 block into four 32×32 blocks and the further splitting of the first 32×32 block into its four 16×16 constituent blocks.

We thus have that larger values of the `Config` parameters will lead us to consider finer partitioning of the coding tree unit. Furthermore, it is important to note that partitions associated with higher configuration values also include partitions associated with lower configuration values. Since we are following a breadth-first-search ordering, higher configurations simply partition additional blocks that were not previously split. As a result, we expect that finer partitioning will always yield improved coding performance.

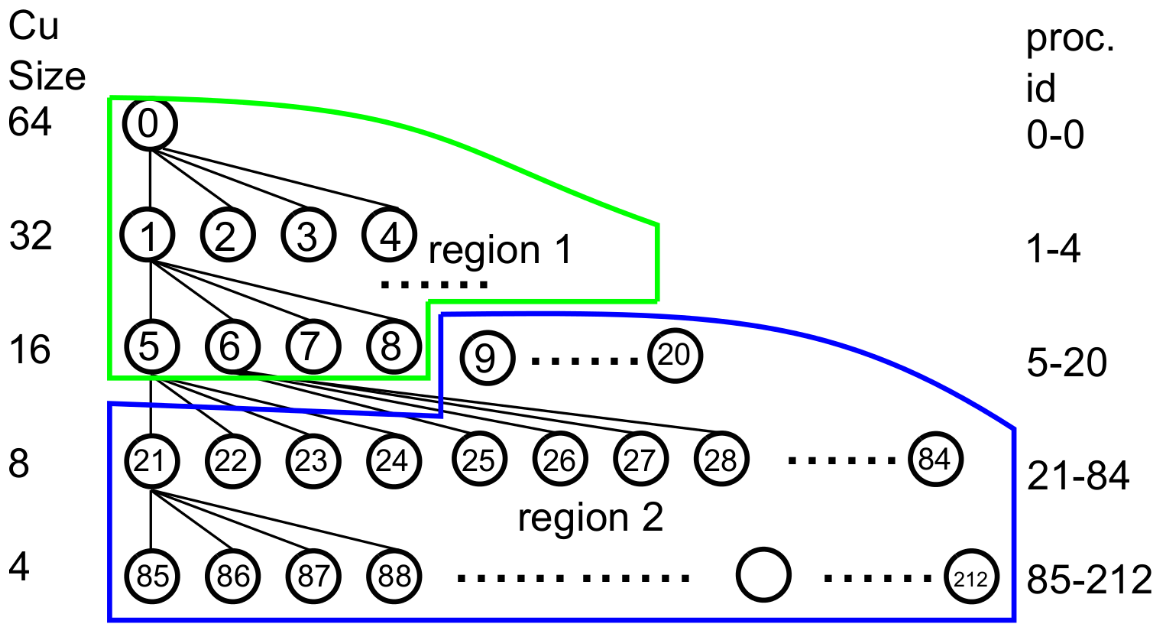


Figure 3.1: Scalable Coding Tree Unit (CTU) partitioning following a breadth-first-search splitting pattern. Each block is recursively partitioned into four sub-blocks using a quadtree decomposition. The case of `Config` = 6 is shown. The labeled partitioned block ids are also shown.

3.2.1 Pareto front using scalable partitioning

We present a multi-objective optimization example based on the proposed CTU partitioning. In Figs. 3.2, 3.3 and 3.4, we demonstrate the performance of our approach in terms of time (seconds per sample), rate (bits per sample), and quality (PSNR) for the RaceHorsesC video (832×480). The space was generated by using $\texttt{QP} \in [6, 51)$ with step=3 and $\texttt{Config} = 0, 1, 2, \ldots, 13$ For each case, we only consider the median values from the first 6 frames of the video.

The approach generated 210 Pareto-optimal configurations. In other words, it is not possible to improve on any one of the objectives without sacrificing on the remaining objective(s). As expected, higher configurations lead to better RD performance at increased complexity. Similarly, increasing `QP` leads to lower rates, higher distortions, and reduced computational complexity. Overall, the Pareto surface is smooth, without any inflection points.

3.3 A Simple Linear Model

We consider a simple linear model for describing the relationship between the objectives and the parameters. We consider:

$$\begin{aligned}
\texttt{Q} &= a_1 \cdot \texttt{QP} + b_1 \cdot \texttt{Config} + c_1 \\
\texttt{T} &= a_2 \cdot \texttt{QP} + b_2 \cdot \texttt{Config} + c_2 \\
\texttt{R} &= a_3 \cdot \texttt{QP} + b_3 \cdot \texttt{Config} + c_3
\end{aligned} \tag{3.1}$$

where `Q` is measured in terms of the mean squared error (MSE), `T` denotes the time (in ns, 10^{-9} second) required for processing a single pixel, and `R` denotes the number of bits per sample. This simple model will be adaptively updated for each video. A detailed control algorithm will also be carefully developed.

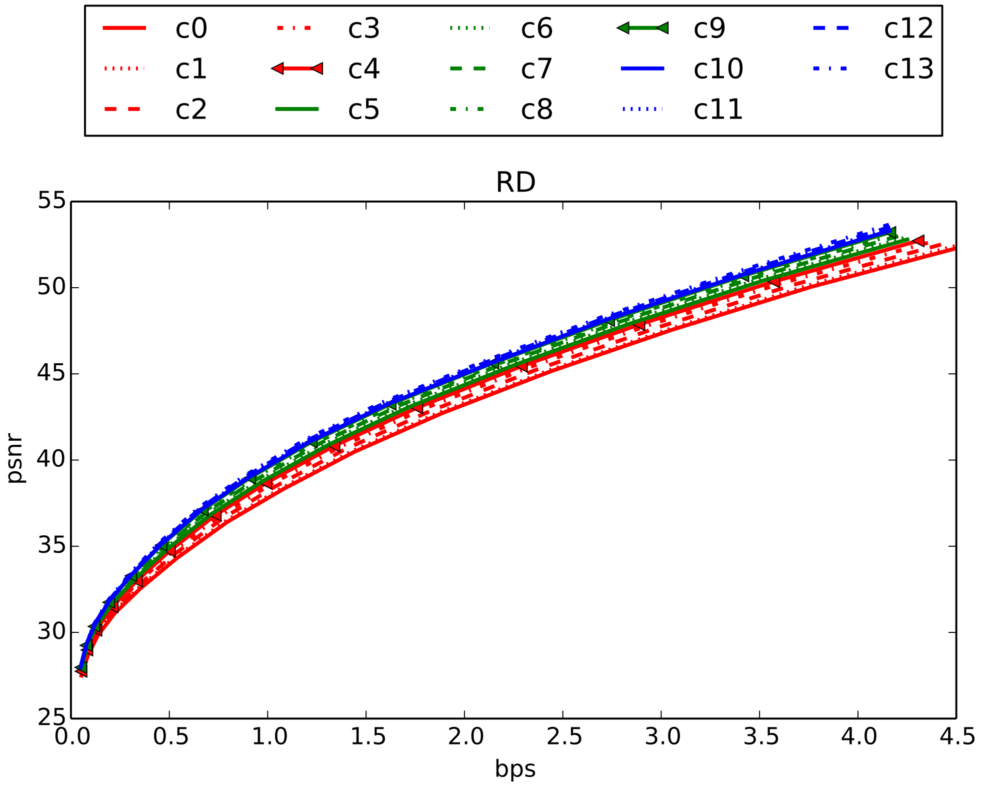


Figure 3.2: Rate-distortion projection of multi-objective space for RaceHorses video.

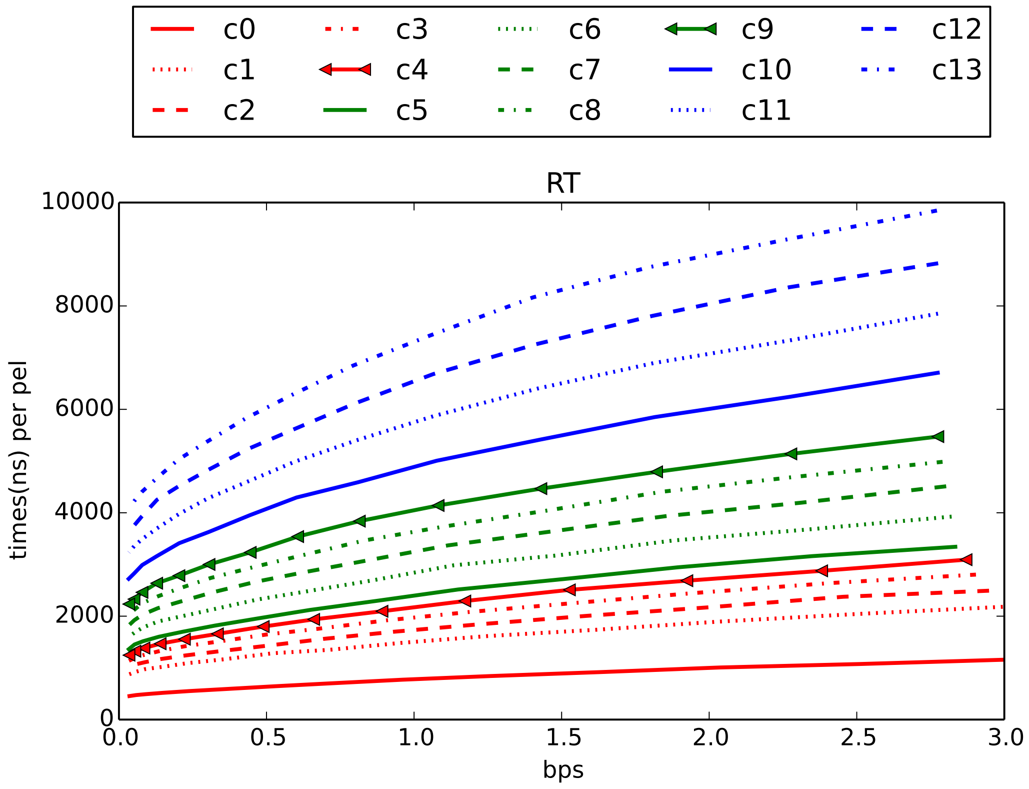


Figure 3.3: Rate-complexity projection of multi-objective space for RaceHorses video.

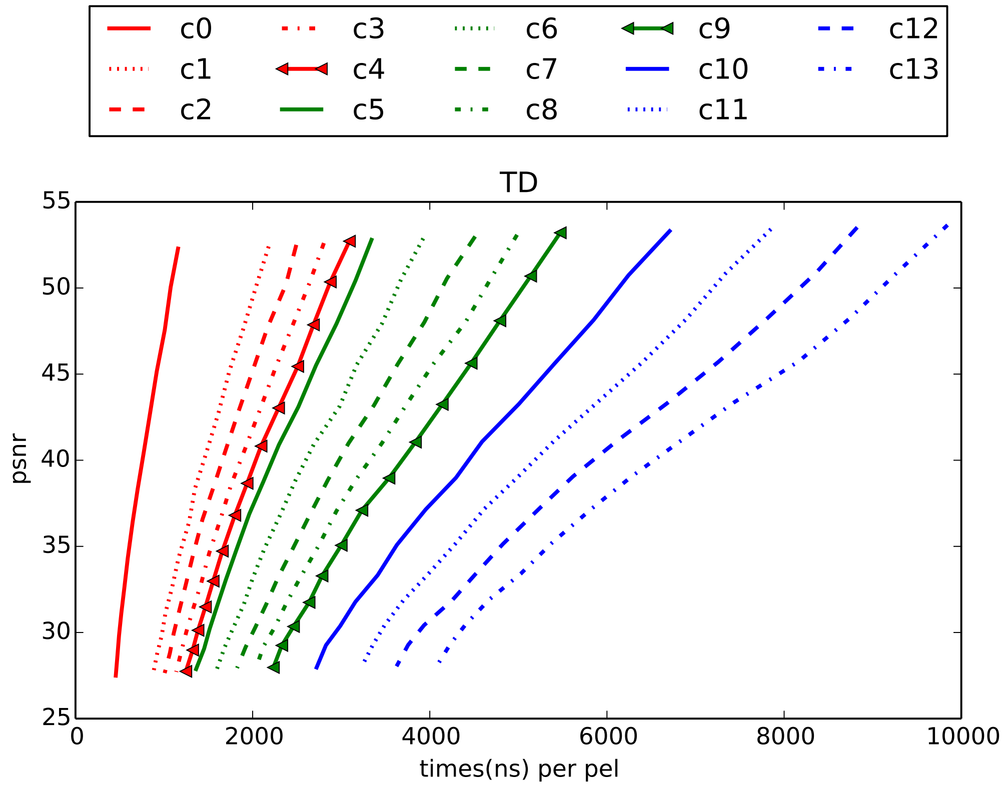


Figure 3.4: Complexity-distortion projection for multi-objective space for RaceHorses video.

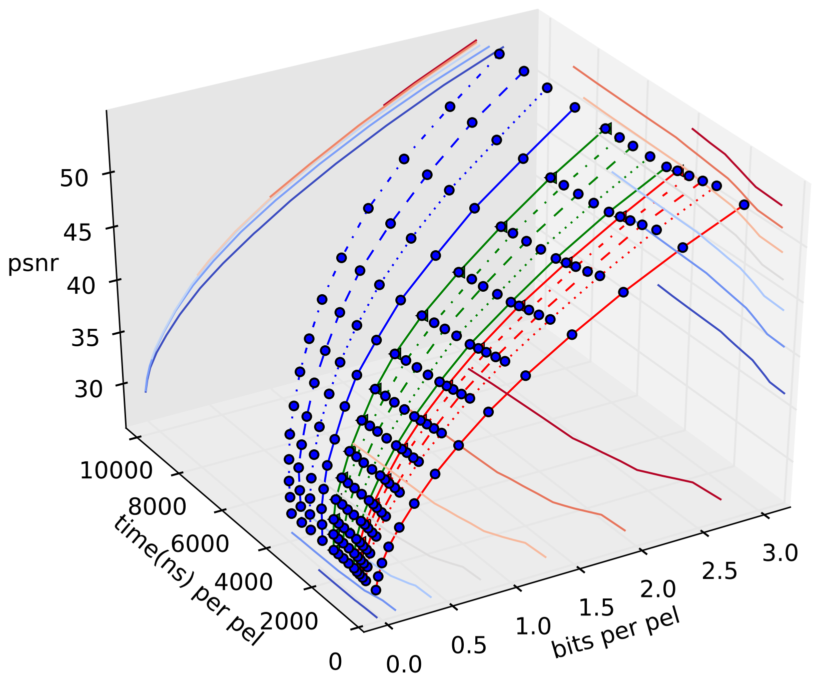


Figure 3.5: Multi-objective performance space for RaceHorses video. The space is generated by varying `QP` and `Config` and estimating complexity, rate, and performance.

Chapter 4

Rate-quality-performance control

4.1 Algorithm overview

The basic system is shown in Fig. 4.6. The proposed approach allocates time, quality, and rate to each CTU by controlling `QP` and `Config`. A feedback loop is used to provide measurements of time, quality, and rate to the control algorithm. The main control algorithm is presented in Fig. 4.1. and Fig. 4.3. The basic idea is to encode each CTU independendly while staying within the budget allocated to the entire frame.

The basic components of the approach are covered in detail in the remaining sections. Budget allocation is described in section 4.2. Spatially adaptive model updating is described in section 4.3. Robust estimation of the encoding parameters is covered in section 4.4.

4.2 Budget allocation

Budget allocation refers to not only to bit allocation, but also quality and complexity allocation. We use $\texttt{R}_{\texttt{target}}$, $\texttt{Q}_{\texttt{target}}$ and $\texttt{T}_{\texttt{target}}$ for the target rate, quality and complexity. We use bits per sample (all is referred as pixel in video encoding) for the rate, Peak Signal-to-Noise Ratio (PSNR), Mean of Square Error (MSE), and Sum of Square Error (SSE) for image quality, and nano-seconds per sample for complexity measurements. In our model, performance budget allocation is based on the pre-computed mean absolute deviation (MAD) computed by the HEVC reference standard

4.2.1 Rate Budget Allocation

Bit allocation requires that we assign the encoding bits for each CTU. Our bit allocation strategy is not simple average bit allocation for all CTUs. Instead, bit allocation is based on pre-computed MAD that also take into account uncontrolled, internal factors of the HEVC that are associated with live video streaming.

We estimate the required number of bits per pixel $\texttt{bpp}_{\texttt{target}}$ using:

$$\texttt{bpp}_{\texttt{target}} = \frac{\texttt{R}_{\texttt{target}}/\texttt{f} - \texttt{HeaderBits}}{\texttt{N}_{\texttt{pixels}}} \tag{4.1}$$

where $\texttt{R}_{\texttt{target}}$ denotes the target number of bits per second for each video frame, $\texttt{f}$ denotes the number of frames per second, $\texttt{N}_{\texttt{pixels}}$ denotes the number of pixels in each frame, and $\texttt{HeaderBits} = 25$ are used for storing the header for HEVC intra-frame encoding. Each frame gets $\texttt{R}_{\texttt{target}}$ bits using:

$$\texttt{R}_{\texttt{target}} = \texttt{N}_{\texttt{pixels}} \cdot \texttt{bpp}_{\texttt{target}}. \tag{4.2}$$

Using $\mathtt{R_{coded}}$, the total number of bits already used in the current frame, we estimate the number of bits remaining for the rest of the image using:

$$\mathtt{R_{left}} = \mathtt{R_{target}} - \mathtt{R_{coded}} \tag{4.3}$$

where $\mathtt{R_{left}}$ denotes the number of bits allocated in the budget that are still available.

Let $\mathtt{R_{adj}}$ refer to the budget correction that we need to make based on mean absolute deviation (MAD). In other words, $\mathtt{R_{adj}}$ is used as given by

$$\mathtt{R_{allocated}} = \mathtt{R_{left}} - \mathtt{R_{adj}} \tag{4.4}$$

to modify the number of bits that have been allocated for the entire frame. We adjust the budget using

$$\mathtt{R_{adj}} = \mathtt{R_{coded}} - \left(1 - \frac{\mathtt{D_{left}}}{\mathtt{D_{total}}}\right) \cdot \mathtt{R_{target}} \tag{4.5}$$

where $\mathtt{D_{left}}$ refers to the pre-computed MAD sum for the remaining CTUs, and $\mathtt{D_{total}}$ refers to the total MAD allocated for the current frame. In (4.5), our goal is to weight bit allocation to be proportional to the remaining MAD that needs to be accounted for. After encoding each CTU using (4.5), $\mathtt{D_{left}}$ gets reduced. $\mathtt{D_{left}}$ should converge to zero. Thus, effectively, the use of (4.5) is meant to ensure that the remaining CTUs get a number of bits that is proportional to their contribution towards the reduction of $\mathtt{D_{total}}$ to zero. After updating $\mathtt{R_{allocated}}$ by substituting (4.5) into (4.4), we allocate the number of bits for the current, $\mathtt{i}$-th CTU using

$$\mathtt{R_{target,i}} = \left(\frac{\mathtt{D_i}}{\mathtt{D_{remaining}}}\right) \cdot \mathtt{R_{allocated}} \tag{4.6}$$

where $\mathtt{D_i}$ refers to the MAD reduction associated with the $\mathtt{i}$-th CTU, $\mathtt{D_{remaining}}$ refers to the MAD still left to do for the entire frame.

4.2.2 Complexity Budget Allocation

Similar to bit allocation, the complexity budget for each CTU is based on the pre-computed MAD. The encoding time per pixel $\mathtt{time\_per\_pixel_{target}}$ is computed using

$$\texttt{time\_per\_pixel}_{\texttt{target}} = \frac{\texttt{Time}_{\texttt{target}}}{\texttt{N}_{\texttt{pixels}}} \tag{4.7}$$

where $\texttt{Time}_{\texttt{target}}$ denotes the number of seconds allocated per frame. The total amount of time allocated to the entire frame $\texttt{T}_{\texttt{target}}$ is given by

$$\texttt{T}_{\texttt{target}} = \texttt{N}_{\texttt{pixels}} \cdot \texttt{time\_per\_pixel}_{\texttt{target}}. \tag{4.8}$$

The amount of time left for encoding the remaining CTUs $\texttt{T}_{\texttt{left}}$ is given by

$$\texttt{T}_{\texttt{left}} = \texttt{T}_{\texttt{target}} - \texttt{T}_{\texttt{coded}} \tag{4.9}$$

where $\texttt{T}_{\texttt{coded}}$ refers to the total amount of bits already used. The allocated time for each CTU is adjusted using $\texttt{T}_{\texttt{adj}}$ given by

$$\texttt{T}_{\texttt{adj}} = \texttt{T}_{\texttt{coded}} - \left(1 - \frac{\texttt{D}_{\texttt{left}}}{\texttt{D}_{\texttt{total}}}\right) \cdot \texttt{T}_{\texttt{target}} \tag{4.10}$$

based on remaining MAD to cover, as done for the rate. The allocated time for entire CTU is similarly update using

$$\texttt{T}_{\texttt{allocated}} = \texttt{T}_{\texttt{left}} - \texttt{T}_{\texttt{adj}}. \tag{4.11}$$

Finally, the amount of allocated for the CTU is given by its share of the remaining MAD:

$$\texttt{T}_{\texttt{target,i}} = \left(\frac{\texttt{D}_{\texttt{i}}}{\texttt{D}_{\texttt{remaining}}}\right) \cdot \texttt{T}_{\texttt{allocated}}. \tag{4.12}$$

4.2.3 Quality Budget Allocation

Image quality is measured using the PSNR. At the CTU level, it is more efficient to work with the sum of squared error (SSE). Thus, there is a need to convert back and forth between PSNR and SSE budget requirements. As for rate and complexity, allocation is based on the MAD.

PSNR requirements are converted into SSE requirements using

$$\mathtt{Q_{target}} = \mathtt{SSE_{target}} = \frac{2^{(2\cdot \mathtt{bitDepth})} \cdot \mathtt{N_{pixels}}}{10^{\mathtt{PSNR}/10}} \tag{4.13}$$

where $\mathtt{SSE_{target}}$ refers to the allocated SSE for the entire frame, and `bitDepth` refers to the number of bits used to represent each pixel. After encoding a CTU, the remaining SSE budget is similarly given by:

$$\mathtt{Q_{left}} = \mathtt{Q_{target}} - \mathtt{Q_{coded}}. \tag{4.14}$$

Adjustments are similarly made using

$$\mathtt{Q_{adj}} = \mathtt{Q_{coded}} - \left(1 - \frac{\mathtt{D_{left}}}{\mathtt{D_{total}}}\right) \cdot \mathtt{Q_{target}} \tag{4.15}$$

and

$$\mathtt{Q_{allocated}} = \mathtt{Q_{left}} - \mathtt{Q_{adj}}. \tag{4.16}$$

Also, the CTU SSE is given by

$$\mathtt{SSE_{target,i}} = \left(\frac{\mathtt{D_i}}{\mathtt{D_{remaining}}}\right) \cdot \mathtt{SSE_{allocated}}. \tag{4.17}$$

4.2.4 Budget Reallocation

Significant content variation can lead to mis-prediction of the required budgets for each frame. In such cases, no action is taken if the variations stay withing the budgets. However, when mis-prediction results in budget deficits, we need to re-allocate the remaining budget to avoid significant artifacts in the reconstructed video. Thus, after the budget is used up, the remaining budget needs to be adjusted to minimize the budget violation.

Budget violations are reduced by reducing the estimates of the remaining budget using:

$$\mathtt{B_{adj}} = \alpha \cdot (\mathtt{D_{i,left}}/\mathtt{D_i}) \cdot \mathtt{B_{target}} \tag{4.18}$$

$$\mathtt{T}_{\mathtt{adj}} = \alpha \cdot (\mathtt{D}_{\mathtt{i,left}}/\mathtt{D}_{\mathtt{i}}) \cdot \mathtt{T}_{\mathtt{target}} \tag{4.19}$$

$$\mathtt{SSE}_{\mathtt{adj}} = \alpha \cdot (\mathtt{D}_{\mathtt{i,left}}/\mathtt{D}_{\mathtt{i}}) \cdot \mathtt{SSE}_{\mathtt{target}} \tag{4.20}$$

where α was set to 0.15 after experimenting with different videos. Clearly, $\alpha = 0$ would lead to significant artifacts while $\alpha = 1$ would not attempt to minimize budget violations and would thus allow significant changes in video content to violate the constraints.

4.3 Spatially Adaptive Model Update

The rate-quality-complexity model is spatially adapted to the input video content. A linear model is built based on the encoding of three neighboring CTUs as depicted in Fig. 4.3.

Let $i = 1, 2, 3$ denote the neighboring CTUs. Furthermore, let each CTU be encoded using the pair of ($\mathtt{QP}_{\mathtt{i}}$, $\mathtt{Config}_{\mathtt{i}}$) to results in ($\mathtt{SSE}_{\mathtt{i}}, \mathtt{T}_{\mathtt{i}}, \mathtt{R}_{\mathtt{i}}$). To estimate the linear model, define the parameter matrix $\mathtt{A}$ using:

$$\mathtt{A} = \begin{bmatrix} \mathtt{a1} & \mathtt{b1} & \mathtt{c1} \\ \mathtt{a2} & \mathtt{b2} & \mathtt{c2} \\ \mathtt{a3} & \mathtt{b3} & \mathtt{c3} \end{bmatrix}. \tag{4.21}$$

Then the basic linear model is described by:

$$\begin{bmatrix} \mathtt{SSE}_{\mathtt{i}} \\ \mathtt{T}_{\mathtt{i}} \\ \mathtt{R}_{\mathtt{i}} \end{bmatrix} = \begin{bmatrix} \mathtt{a1} & \mathtt{b1} & \mathtt{c1} \\ \mathtt{a2} & \mathtt{b2} & \mathtt{c2} \\ \mathtt{a3} & \mathtt{b3} & \mathtt{c3} \end{bmatrix} \begin{bmatrix} \mathtt{QP}_{\mathtt{i}} \\ \mathtt{Config}_{\mathtt{i}} \\ \mathtt{1} \end{bmatrix}. \tag{4.22}$$

Suppose that the 3 CTU encodings use 3 different pairs of ($\mathtt{QP}_{\mathtt{i}}$, $\mathtt{Config}_{\mathtt{i}}$). In this case, we expect that the 3 rows of [$\mathtt{QP}_{\mathtt{i}}$ $\mathtt{Config}_{\mathtt{i}}$ 1] should also be linearly independent since the ranges of QP and Config are quite different. Thus, when working with three different CTU encodings, we can estimate the parameters using:

$$\begin{bmatrix} \mathtt{a1} \\ \mathtt{b1} \\ \mathtt{c1} \end{bmatrix} = \begin{bmatrix} \mathtt{QP}_1 & \mathtt{Config}_1 & 1 \\ \mathtt{QP}_2 & \mathtt{Config}_2 & 1 \\ \mathtt{QP}_3 & \mathtt{Config}_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathtt{SSE}_1 \\ \mathtt{SSE}_2 \\ \mathtt{SSE}_3 \end{bmatrix}, \tag{4.23}$$

$$\begin{bmatrix} \mathtt{a2} \\ \mathtt{b2} \\ \mathtt{c2} \end{bmatrix} = \begin{bmatrix} \mathtt{QP}_1 & \mathtt{Config}_1 & 1 \\ \mathtt{QP}_2 & \mathtt{Config}_2 & 1 \\ \mathtt{QP}_3 & \mathtt{Config}_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathtt{T}_1 \\ \mathtt{T}_2 \\ \mathtt{T}_3 \end{bmatrix}, \quad \text{and} \tag{4.24}$$

$$\begin{bmatrix} \mathtt{a3} \\ \mathtt{b3} \\ \mathtt{c3} \end{bmatrix} = \begin{bmatrix} \mathtt{QP}_1 & \mathtt{Config}_1 & 1 \\ \mathtt{QP}_2 & \mathtt{Config}_2 & 1 \\ \mathtt{QP}_3 & \mathtt{Config}_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathtt{R}_1 \\ \mathtt{R}_2 \\ \mathtt{R}_3 \end{bmatrix}. \tag{4.25}$$

For robust model update, we also consider the case when the neighboring CTUs do not use 3 independent encodings. In this case, we select $[\mathtt{a_i}\ \mathtt{b_i}\ \mathtt{c_i}]$ associated with the best predictions. To implement this approach, for the i-th CTU, we compute the prediction errors using:

$$\begin{aligned} \mathtt{SSE}_{\mathtt{error,\,i}} &= |\mathtt{SSE_i} - a_1 \cdot \mathtt{QP_i} - b_1 \cdot \mathtt{Config_i} - c_1|, \\ \mathtt{R}_{\mathtt{error,\,i}} &= |\mathtt{R_i} - a_2 \cdot \mathtt{QP_i} - b_2 \cdot \mathtt{Config_i} - c_2|, \quad \text{and} \\ \mathtt{T}_{\mathtt{error,\,i}} &= |\mathtt{T_i} - a_3 \cdot \mathtt{QP_i} - b_3 \cdot \mathtt{Config_i} - c_3|. \end{aligned}$$

Then, we build the model by using the coefficients associated with the minimum prediction errors. For example, for $\mathtt{A}_{1,\mathtt{i}} = [\mathtt{a}_{1,\mathtt{i}}\ \mathtt{b}_{1,\mathtt{i}}\ \mathtt{c}_{1,\mathtt{i}}]$, we solve:

$$\min_{\mathtt{i}} \mathtt{SSE}_{\mathtt{error,i}} \tag{4.26}$$

and use the $\mathtt{A}_{1,\mathtt{j}}$ associated with j-th CTU model that minimizes (4.26). The idea is also demonstrated in Fig. 4.3.

Another problem occurs in coming up with an initial model for the first row and first column in each frame. For this case, we create virtual CTUs above the first row and to the left of the first column as shown in Fig. 4.4. Then, for each virtual CTU we simply compute the Pareto front based on the average of the current encodings. For the first frame, we simply use an initial model trained on other videos. After a few frames, we simply compute the Pareto front from the current video. Here, we note that the Pareto front is obtained through an exhaustive evaluation of all possible `Config` and `QP` values. However, the cost of estimating the Pareto front is restricted to CTUs over a few frames and offline computations using other videos.

4.4 Robust Estimation of CTU Encoding Parameters

We use the updated linear models to estimate values for `QP` and `Config` that can satisfy the constraints and minimize bitrate, maximize quality, or minimize complexity. We also provide a robust approach for minimizing constraint violations.

We use the minimum bitrate mode to demonstrate the basic concepts. All other models are similar and will not be repeated here. As explained in the section 4.2, the constraints are used to determine target values for `Q`, `T`, `R` as needed. For the minimum bitrate mode, we want to match the constraints on quality $\mathtt{Q_{target}}$ and time $\mathtt{T_{target}}$. We use the linear model to determine the encoding parameters:

$$\begin{bmatrix} \mathtt{Q_{target}} \\ \mathtt{T_{target}} \end{bmatrix} = \begin{bmatrix} \mathtt{a}_1 & b_1 & c_1 \\ \mathtt{a}_2 & b_2 & c_2 \end{bmatrix} \begin{bmatrix} \mathtt{QP} \\ \mathtt{Config} \\ 1 \end{bmatrix}. \tag{4.27}$$

Using (4.27), we estimate initial values of the encoding parameters using:

$$\begin{bmatrix} \mathtt{QP_{est}} \\ \mathtt{Config_{est}} \end{bmatrix} = \begin{bmatrix} \mathtt{a}_1 & b_1 \\ \mathtt{a}_2 & b_2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathtt{Q_{target}} - \mathtt{c1} \\ \mathtt{T_{target}} - \mathtt{c2} \end{bmatrix} \tag{4.28}$$

We then round $\texttt{QP}_{\texttt{est}}$ and $\texttt{Config}_{\texttt{est}}$ to the nearest integer values. We then use the model given by

$$\begin{aligned}
\texttt{Q} &= a_1 \cdot \texttt{QP} + b_1 \cdot \texttt{Config} + c1 \\
\texttt{T} &= a_2 \cdot \texttt{QP} + b_2 \cdot \texttt{Config} + c2 \\
\texttt{R} &= a_3 \cdot \texttt{QP} + b_3 \cdot \texttt{Config} + c3
\end{aligned} \tag{4.29}$$

to perform a local search with $\texttt{QP} \in [\texttt{QP}_{\texttt{est}} - 2, \texttt{QP}_{\texttt{est}} + 2]$ and $\texttt{Config} \in [\texttt{Config}_{\texttt{est}} - 2, \texttt{Config}_{\texttt{est}} + 2]$ for the minimum bitrate that also satisfies the constraints. Alternatively, if no parameters can satisfy the constraints, we compute the normalized constraint violations using:

$$\texttt{norm}(\texttt{X}) = \frac{\texttt{X} - \texttt{X}_{\texttt{min}}}{\texttt{X}_{\texttt{mean}}}. \tag{4.30}$$

We then select the $(\texttt{QP}, \texttt{Config})$ pair that minimizes the total normalized constraint violation as given in Table 4.1. for the minimum bitrate mode.

Similarly, for the maximum quality mode, we first use the target budget values for bitrate and performance to determine initial estimates and select optimal encoding parameters based on local search or minimum constraint violation. Then, for the minimum complexity mode, we use the target bitrate and quality for the initial search.

4.5 Constraint Update for Valid Encoding

While the linear model is simple and robust, it can fail to produce valid values for $\texttt{QP}$ and $\texttt{Config}$. This failure occurs because the linear model does not impose any restrictions on the constraints. Thus, the constraints end up being significantly above or below the rate-performance-quality surface. When the constraints are significantly off, we automatically modify them to bring them close to the control surface.

Table 4.1: Constraint violation objectives. When no configuration will satisfy the constraints, we select ($\texttt{QP}, \texttt{Config}$) that minimizes the normalized constraints.

Mode	**Objective (minimum)**
Minimum Rate	$\texttt{norm}(\texttt{abs}(\texttt{MSE}_{\texttt{est}} - \texttt{MSE}_{\texttt{target}}))$ $+\texttt{norm}(\texttt{abs}(\texttt{Time}_{\texttt{est}} - \texttt{Time}_{\texttt{target}}))$
Minimum Complexity	$\texttt{norm}(\texttt{abs}(\texttt{MSE}_{\texttt{est}} - \texttt{MSE}_{\texttt{target}}))$ $+\texttt{norm}(\texttt{abs}(\texttt{BPS}_{\texttt{est}} - \texttt{BPS}_{\texttt{target}}))$
Maximum Quality	$\texttt{norm}(\texttt{abs}(\texttt{Time}_{\texttt{est}} - \texttt{Time}_{\texttt{target}}))$ $+\texttt{norm}(\texttt{abs}(\texttt{BPS}_{\texttt{est}} - \texttt{BPS}_{\texttt{target}}))$

For valid encodings, we require that $\texttt{QP} \in [0, 51]$ and $\texttt{Config} \in [0, 13]$. When either parameter falls out of range, we modify the constraints to produce valid encodings.

In general, rate, constraint, and computation complexity are non-linearly related. Our linear model is excellent for local approximations to the non-linear relationship. However, globally, the linear model will not work.

We present examples of the non-linear relationships between distortion and rate in Fig. 4.5. We fit the curves that describe the relationships between any pair of constraints using:

$$\begin{aligned}
\texttt{T} \quad &= a1 \cdot \texttt{SSE}^{b1}, \quad a1 > 0,\ b1 < 0. \\
\texttt{SSE} \quad &= a2 \cdot \texttt{R}^{b2}, \qquad a2 > 0,\ b2 < 0. \\
\texttt{T} \quad &= a3 \cdot \texttt{R}^{b3}, \qquad a3 > 0,\ b3 > 0.
\end{aligned} \tag{4.31}$$

Next, we provide detailed explanations of how to modify the constraints for the minimum rate algorithm. As for the linear model, we use the neighboring CTU encodings to adaptively estimate the relationships between the constraints (see Fig.4.6). The main algorithm for estimating $\texttt{T} = \texttt{a} \cdot \texttt{SSE}^{\texttt{b}}$ is given in Fig. 4.7. Based on the relationship, we move either the quality or the complexity constraint to lie on the curve as given in Fig. 4.8. Similarly, for the minimum complexity mode, we estimate $\mathtt{SSE} = \mathtt{a} \cdot \mathtt{R}^{\mathtt{b}}$ as given in Fig. 4.9 and update the constraints as given in Fig. 4.10. The model update and algorithm for the maximum quality (minimum distortion) model is given in Figs. 4.11 and 4.12.

We also account for the case of failing to estimate the model. For example, if the left and top CTUs are encoded in the same way, we simply use the configuration from the last CTU. Similarly, if the constraint update is excessive, we also use the configuration from the last CTU.

The updated constraints are used for estimating new, valid values for `QP` and `Config`. We prevent large changes by requring that the `QP` to remain within ± 4 of the average of the neighboring CTUs. Furthermore, the final encoding parameters are forced to stay within the valid ranges.

```
1: Estimate budgets for T, Q, R for all CTUs.
2: Estimate QP and Config using initial model.

   ▷ Encode frame by iterating through the CTUs.
3: for each CTU in current frame do

       ▷ Robust allocation T, Q, R within available budgets.
4:     Allocate T, Q, R based on available budgets.
5:     Update remaining budgets for T, Q, R.
6:     if any remaining budget < 0 then
           ▷ Adjust budget to minimize the violation.
7:         ReAllocate CTU budgets using a fraction
                   of the remaining total frame budget.
8:     end if

       ▷ Robust model update
9:     Update model using three neighboring CTUs.
10:    if model update failed then
11:        Update model with neighboring CTU model
               that gave best prediction.
12:    end if

       ▷ Robust parameter estimation and optimization.
13:    Estimate QP and Config based on the model.
14:    Solve optimization problem using local search.
15:    if either QP or Config is out of range then
           ▷ Update constraints and fix encodings
16:        Update constraints and estimate new
               estimates of QP and Config.
17:        Constrain QP to be within ±4 of
               neighboring CTUs.
18:        Enforce QP and Config within valid ranges.
19:    end if

       ▷ Encode CTU and store encoding parameters.
20:    Encode CTU using QP and Config.
21:    Compute T, Q, R for current CTU.
22:    Save QP, Config, T, Q, R and CTU location for
               model updates.
23: end for
```

Figure 4.1: Common framework for mode implementation

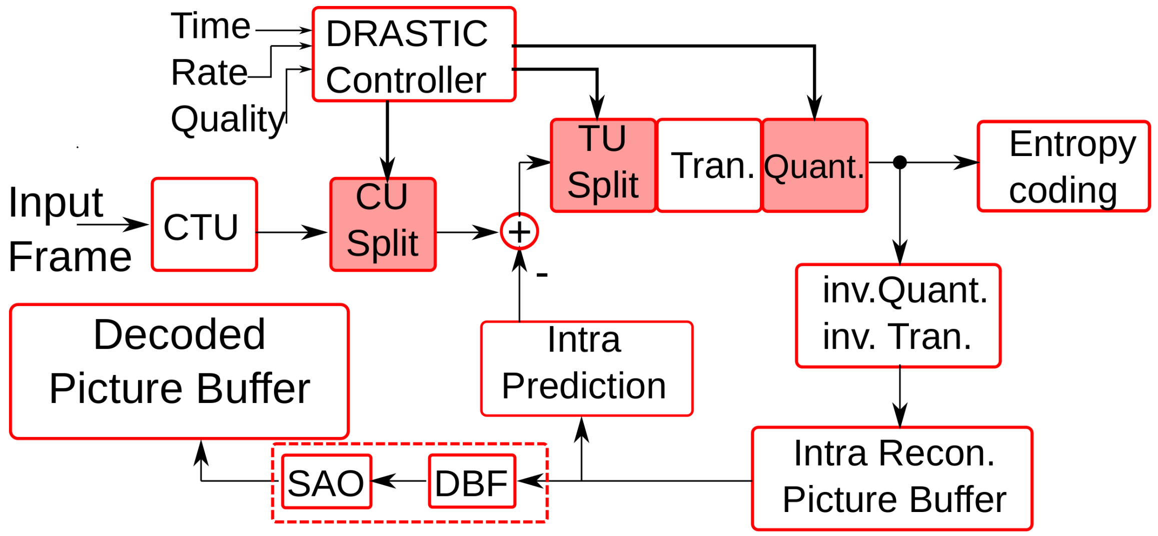


Figure 4.2: Diagram for DRASTIC HEVC Intra Encoding System with Time, Rate and Quality input. Time, Rate, and Quality are already computed in the standard HEVC implementation.

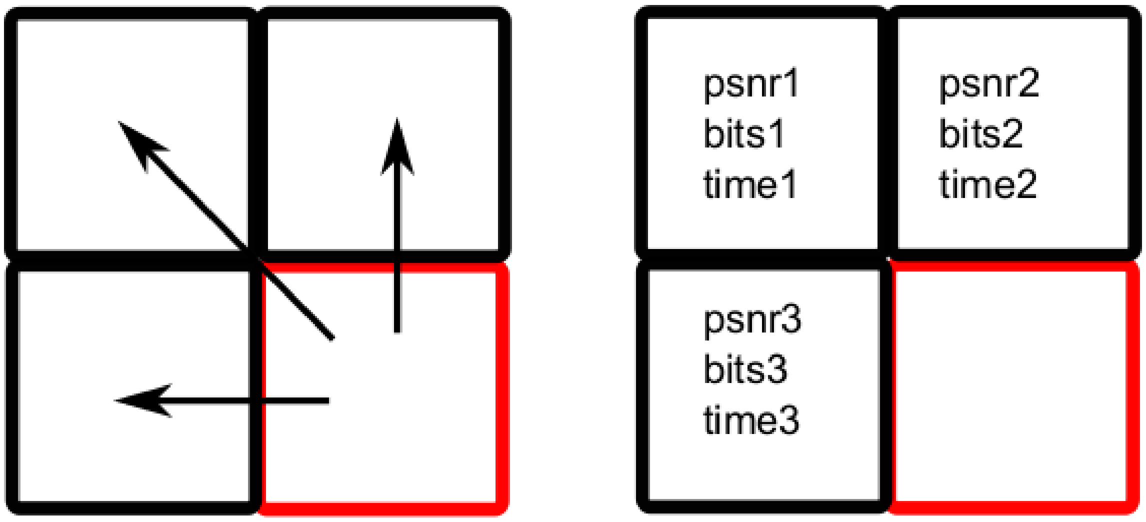


Figure 4.3: Model update using neighboring CTUs. For the CTU indexed as $(\mathtt{CTU_y}, \mathtt{CTU_x})$, the 3 neighbor CTUs are indexed as $(\mathtt{CTU_y}, \mathtt{CTU_x} - \mathtt{1})$, $(\mathtt{CTU_y} - \mathtt{1}, \mathtt{CTU_x} - \mathtt{1})$ and $(\mathtt{CTU_y} - \mathtt{1}, \mathtt{CTU_x})$. When the neighboring CTUs share encodings, the model is constructed using the best predictions (see text). Thus, it is possible for a model to select model parameters associated with $\mathtt{SSE_1}$, $\mathtt{R_2}$, $\mathtt{T_3}$ because each of them gave minimum errors.

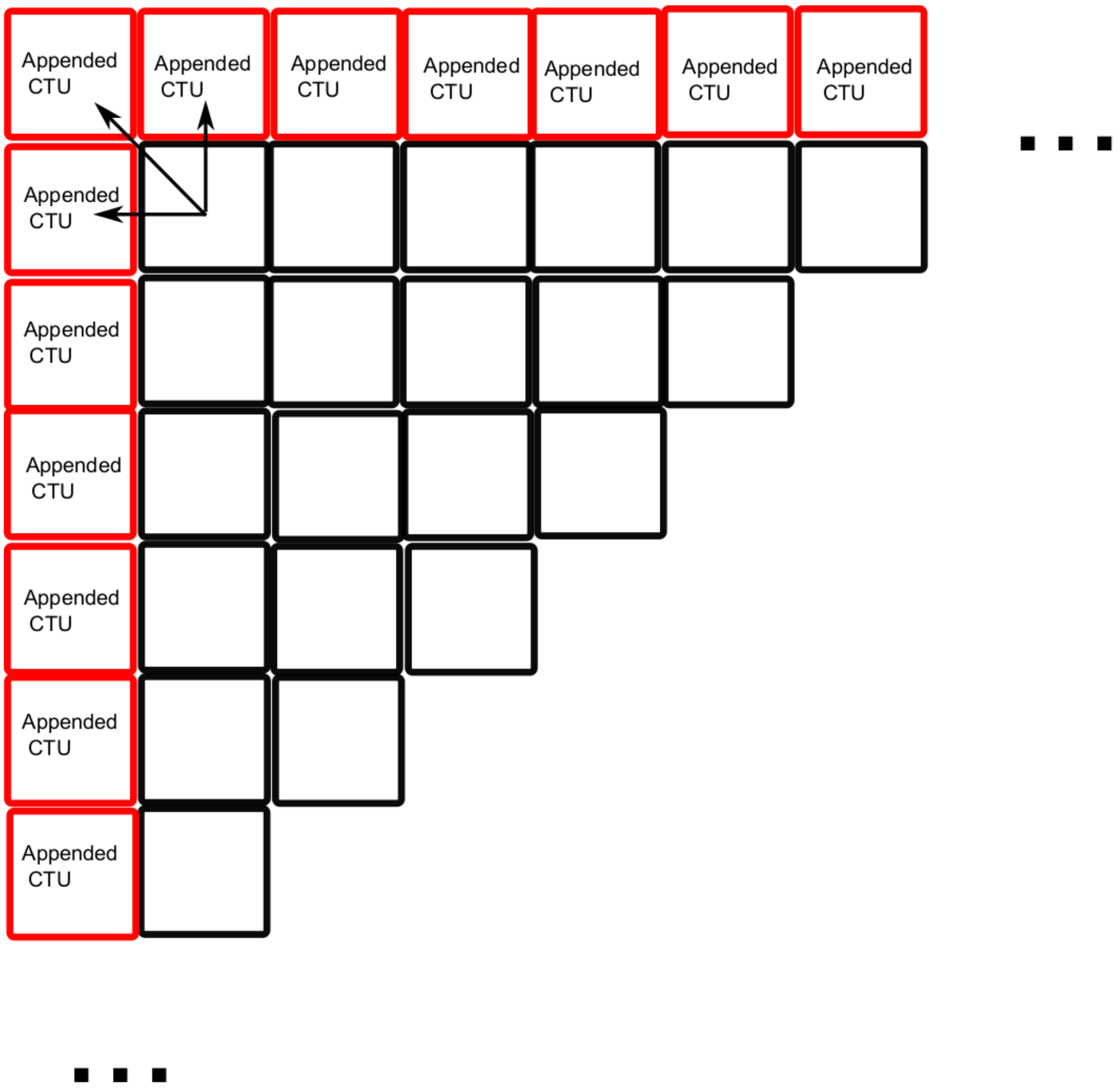


Figure 4.4: Model updates for the first row and column. The virtual CTU encodings assume the Pareto front that is initialized from other videos and then updated based on the encodings of the first few frames of the current video.

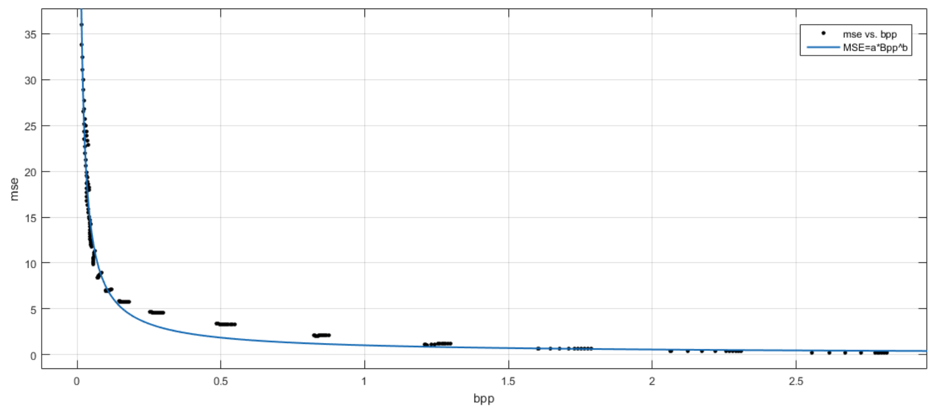


Figure 4.5: Non-linear relationship between distortion(in MSE) and rate(in bpp)

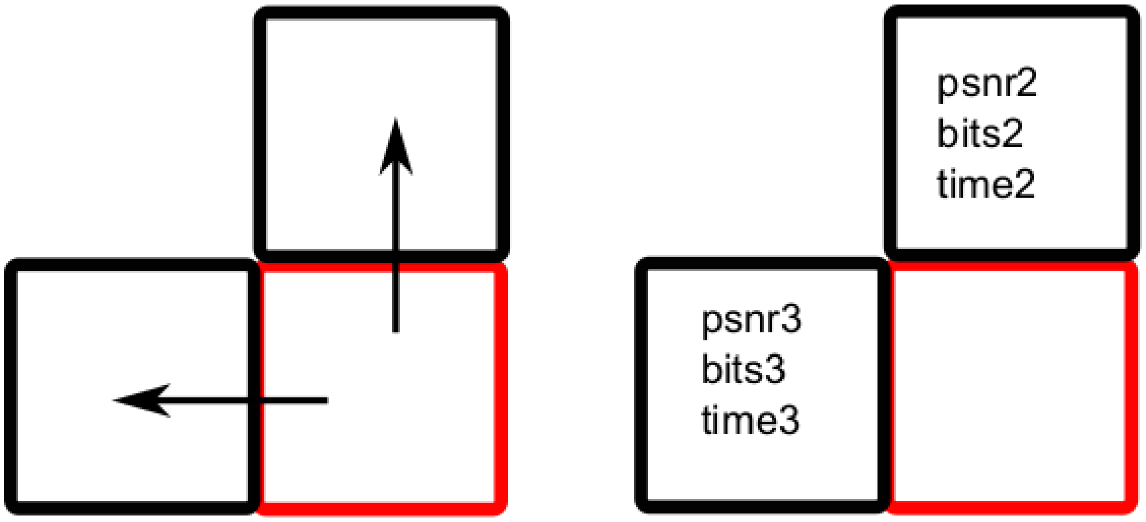


Figure 4.6: Performance constraint model update using neighbor CTUs. for CTU indexed as $(\mathtt{CTU_y}, \mathtt{CTU_x})$, we will refer to 2 neighbor CTUs indexed as $(\mathtt{CTU_y}, \mathtt{CTU_x} - \mathtt{1})$ and $(\mathtt{CTU_y} - \mathtt{1}, \mathtt{CTU_x})$

```
   ▷ Use CTU SSE and times T to estimate $a, b$.
1: if $(\mathtt{SSE_{top}} \mathrel{!=} \mathtt{SSE_{left}})$ and $(\mathtt{T_{top}} \mathrel{!=} \mathtt{T_{left}})$ then
2:     $b = \log(\mathtt{T_{top}}/\mathtt{T_{left}})/\log(\mathtt{SSE_{top}}/\mathtt{SSE_{left}})$
3:     $a = \mathtt{T_{top}}/\mathtt{SSE_{top}}^{b}$
4: end if
```

Figure 4.7: Time-quality relationship model update for minimum bitrate mode.

```
    ▷ Estimate ratios associated with current CTU.
1:  $\mathtt{T_{used}} \leftarrow \mathtt{T}/\mathtt{T_{target,i}}$
2:  $\mathtt{SSE_{used}} \leftarrow \mathtt{SSE}/\mathtt{SSE_{target,i}}$
3:  if $(\mathtt{SSE_{used}} > 1)$ and $(\mathtt{T_{used}} > 1)$ then
        ▷ Above the target.
4:      if $(\mathtt{SSE_{used}} \leq \mathtt{T_{used}})$ then
            ▷ Reduce time to meet the curve.
5:          $\mathtt{T_{target,i}} = a \cdot \mathtt{Q_{target}}^{b}$
6:      else
            ▷ Reduce SSE to meet the curve.
7:          $\mathtt{Q_{target}} = (\mathtt{T_{target}}/a)^{1/b}$
8:      end if
9:  else
        ▷ Below the target.
10:     if $(\mathtt{SSE_{used}} \geq \mathtt{T_{used}})$ then
            ▷ Increase time to meet the curve.
11:         $\mathtt{T_{target}} = (\mathtt{Q_{target}}/a)^{1/b}$
12:     else
            ▷ Increase SSE to meet the curve.
13:         $\mathtt{Q_{target}} = a \cdot \mathtt{T_{target}}^{b}$
14:     end if
15: end if
```

Figure 4.8: Constraint updates for minimum bitrate mode.

```
   ▷ Use CTU SSE and bitrates R to estimate $a, b$.
1: if $(\mathtt{SSE_{top}} \mathrel{!=} \mathtt{SSE_{left}})$ and $(\mathtt{R_{top}} \mathrel{!=} \mathtt{R_{left}})$ then
2:     $b = \log(\mathtt{SSE_{top}}/\mathtt{SSE_{left}})/\log(\mathtt{R_{top}}/\mathtt{R_{left}})$
3:     $a = \mathtt{SSE_{top}}/\mathtt{R_{top}}^{b}$
4: end if
```

Figure 4.9: Quality-rate relationship model update for minimum computational complexity mode.

```
1: $\mathtt{R_{used}} \leftarrow \mathtt{R}/\mathtt{R_{target,i}}$
2: $\mathtt{SSE_{used}} \leftarrow \mathtt{SSE}/\mathtt{SSE_{target,i}}$
3: if $(\mathtt{R_{used}} > 1)$ and $(\mathtt{SSE_{used}} > \mathtt{1})$ then
4:     if $(\mathtt{SSE_{used}} \leq \mathtt{R_{used}})$ then
5:         $\mathtt{R_{target}} = (\mathtt{Q_{target}}/a)^{1/b}$
6:     else
7:         $\mathtt{Q_{target}} = a \cdot \mathtt{R_{target}}^{b}$
8:     end if
9: else
10:     if $(\mathtt{SSE_{used}} \geq \mathtt{R_{used}})$ then
11:         $\mathtt{R_{target}} = (\mathtt{Q_{target}}/a)^{1/b}$
12:     else
13:         $\mathtt{Q_{target}} = a \cdot \mathtt{R_{target}}^{b}$
14:     end if
15: end if
```

Figure 4.10: Constraint update for minimum computational complexity mode.

```
1:  ▷ Use CTU encoding times T and rates R
2:  ▷ to estimate $a, b$ for the model.
3: if $(\mathtt{T_{top}} \mathrel{!=} \mathtt{T_{left}}))$ and $(\mathtt{R_{top}} \mathrel{!=} \mathtt{R_{left}})$ then
4:     $b = \log(\mathtt{T_{top}}/\mathtt{T_{left}})/\log(\mathtt{R_{top}}/\mathtt{R_{left}})$
5:     $a = \mathtt{T_{top}}/\mathtt{R_{top}}^{b}$
6: end if
```

Figure 4.11: Time-rate relationship model update for maximum quality (minimum distortion mode).

```
1: $\mathtt{T_{used}} \leftarrow \mathtt{T}/\mathtt{T_{target,i}}$
2: $\mathtt{R_{used}} \leftarrow \mathtt{R}/\mathtt{R_{target,i}}$
3: if ($\mathtt{R_{used}} > \mathtt{1}$) and ($\mathtt{T_{used}} > 1$) then
4:     if ($\mathtt{T_{used}} \leq \mathtt{R_{used}}$) then
5:         $\mathtt{T_{target}} = a \cdot \mathtt{R_{target}}^{b}$
6:     else
7:         $\mathtt{R_{target}} = (\mathtt{T_{target}}/a)^{1/b}$
8:     end if
9: else
10:    if ($\mathtt{T_{used}} \geq \mathtt{R_{used}}$) then
11:        $\mathtt{R_{target}} = (\mathtt{T_{target}}/a)^{1/b}$
12:    else
13:        $\mathtt{T_{target}} = a \cdot \mathtt{R_{target}}^{b}$
14:    end if
15: end if
```

Figure 4.12: Constraint update for minimum distortion mode (maximum quality).

Chapter 5

Results for Basic Control of Rate-quality-performance

We describe the database of the test videos in section 5.1. We then describe the basic settings for the HEVC encoder in section 5.2. The constraints and the low, medium, and high profiles used for HEVC mode optimization are given in section 5.3. The results are given in section 5.4.

5.1 Test video sequences

We present results on 17 standard JCT-VC videos from four different classes. The videos, associated class, resolution, and frame rates are given in Table 5.1. The classes are differentiated based on the size of each frame. Within each class, we have different fame rates (FPS) and number of frames.

Table 5.1: Test Video Sequences.

Name	**Class**	**Frame Count**	**Resolution**	**FPS**
Traffic	A	150	2560x1600	30
PeopleOnStreet	A	150	2560x1600	30
NebutaFestival10bit	A	300	2560x1600	60
SteamLocomotiveTrain10bit	A	300	2560x1600	60
Kimono	B	240	1920x1080	24
ParkScene	B	240	1920x1080	24
Cactus	B	500	1920x1080	50
BQTerrace	B	600	1920x1080	60
BasketballDrive	B	500	1920x1080	50
RaceHorsesC	C	300	832x480	30
BQMall	C	600	832x480	60
PartyScene	C	500	832x480	50
BasketballDrill	C	500	832x480	50
RaceHorses	D	300	416x240	30
BQSquare	D	600	416x240	60
BlowingBubbles	D	500	416x240	50
BasketballPass	D	500	416x240	50

5.2 HM Encoder Setting

We compute baseline encodings using the HEVC HM-11.0 encoder. For the baseline encoding, we use the basic HEVC intra-mode encoding as detailed in Table 5.2. The encoder configuration is stored in the file titled `encoder_intra_main.cfg`.

5.3 Video Encoding Profiles and Constraints

For comparison purposes, we used the standard HEVC encoder HM-11.0 to compress the first 30 frames of each video sequence using different profiles. For each video, we generate HEVC encodings using: (i) low-profile for `QP` $= 37$, (ii) medium-profile for `QP` $= 32$, and (iii) high-profile for `QP` $= 27$. We set the constraints associated with the low, medium, and high profiles using the average rates, quality, and computational-

Table 5.2: The functions and techniques blocked in our experiments

Option	**Setting**
Sample adaptive offset filter (SAO)	Off
In-loop deblocking filter (LoopFilterDisable)	Off
Rate-distortion-optimized quantization (RDOQ)	Off
PCM (PCMEnabledFlag)	Off
Rate-distortion-optimized quantization for transform-skipped TUs (RDOQTS)	Off
Transform-skipping mode decision for 4x4 TUs (TransformSkip)	Off
Reduced testing of the transform-skipping mode decision for chroma TUs(TransformSkipFast)	Off

complexity for each case. For each video frame, we consider a constraint to be satisfied if it is not violated by more than 5%. Encoding times are based on computations using an Intel(R) Xeon(R) Processor E5-2630 v3 (8 cores, 2.4GHz, Turbo, HT, 20M, 85W) using Ubuntu 14.05 that was used to run HM11.0 standard software.

5.4 Results

The results are given in Tables 5.3, 5.4, and 5.6. In what follows, we present different image examples that demonstrate the advantages of using joint control of rate-performance-quality.

We present comparative encodings of the 10th frame of the RacehorseC (832x480) video in Fig. 5.1. Compressions with different optimization modes using `QP` = `32` and `Config` = `13` are shown in Fig. 5.2.

An example of significant bitrate savings is shown in Fig. 5.3. In this example, based on RaceHorseC, the constraints on time and quality were successfully met. Yet, using the high profile, the bitrate was reduced by 20.2% without sacrificing image quality.

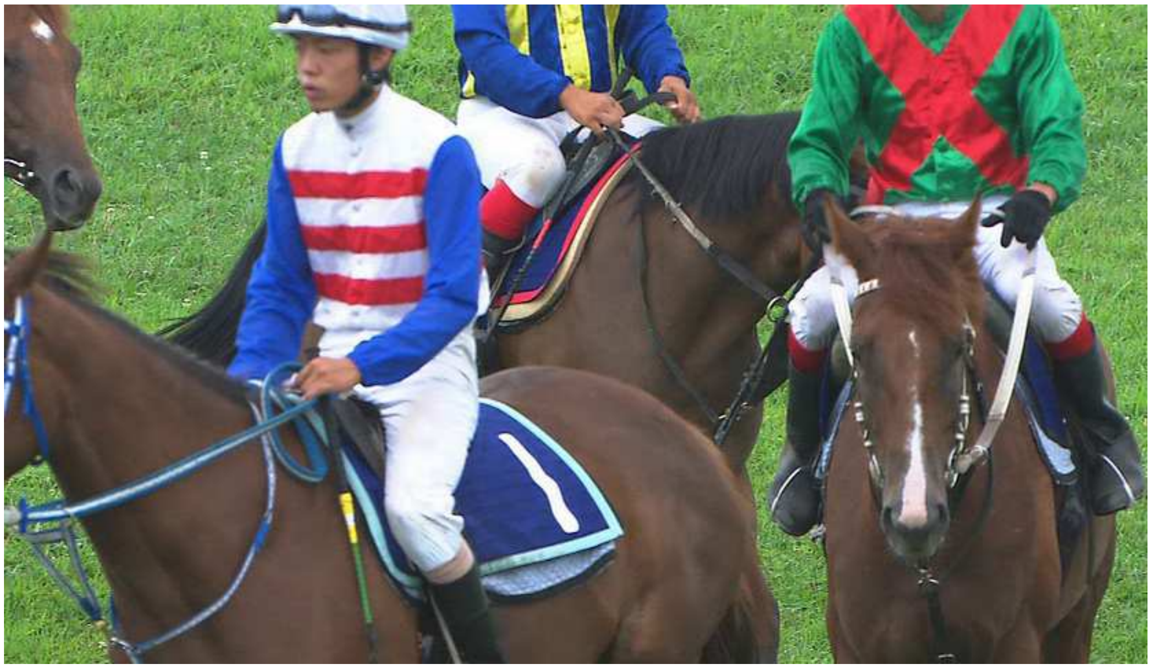


Figure 5.1: Original 10th frame in test sequence PartyScene (832x480).

An example of significant bitrate savings is demonstrated in Fig. 5.4. For the RaceHorseC video, using the high profile, bitrate and quality constraints are met at a substantial 37.1% reduction in bitrate.

An increase in image quality is demonstrated in Fig. 5.5. In this example, time and bitrate constraints are met with a PSNR gain of 0.383 using the high profile.

We provide an example of image quality enhanced in Figs. 5.6, 5.7a, and 5.7b. In the example, for baseline encoding, it is not possible to see the hair under the chin of the horse. On the other hand, the horse hair is easy to see in minimum distortion (maximum image quality) encoding. Here, for the minimum distortion mode, quality was enhanced without requiring extra computational complexity or a higher bitrate.

We also present statistical summaries for each mode and profile in Table 5.6. For the minimum bitrate mode, we have average bitrate savings in the range of

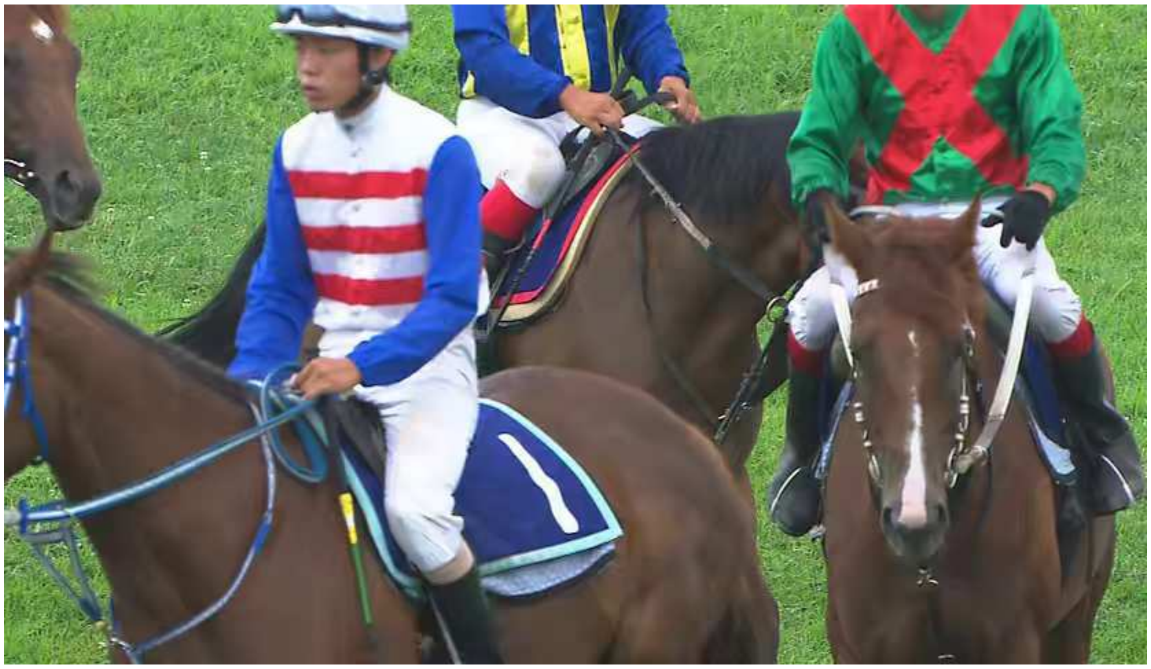


Figure 5.2: Baseline encoding using `QP` = `32` and `Config` = `13` for 10th frame of test sequence PartyScene (832x480).

15 to 20 percent. Here, it is important to note that we have the highest average bitrate savings of 20 percent for the high-quality profile. We also have substantial percentage savings using the minimum complexity mode. The savings range from 38 to 46 percentage that can nearly double the encoding frame-rate (at 50 percent). Average savings for the maximum image quality mode range from 0.38 to 0.94 dB. The low-quality mode get the maximum improvement at nearly 1dB.

Table 5.3: Results for Minimum Rate Mode for the low, medium, high profiles. The savings over standard HEVC encodings are given under improvements (Imp). For each profile, we list (i) the savings over standard HEVC encodings (Sav), and (ii) the percentage of video frames that satisfied the constraints.

	Low		**Med**		**High**	
Name	**Constr**	**Sav**	**Constr**	**Sav**	**Constr**	**Sav**
Traffic	100%	16.3%	100%	22.8%	100%	21.5%
PeopleOnStreet	100%	12.3%	100%	14.4%	96%	18.7%
NebutaFestival10bit	100%	18.1%	100%	19.5%	100%	19.8%
SteamLocomotiveTrain10bit	100%	8.3%	100%	2.8%	100%	-6.4%
Kimono	100%	2.8%	100%	12.4%	100%	19.2%
ParkScene	100%	23.7%	100%	27.7%	93.3%	24.3%
Cactus	100%	18.0%	100%	-1.9%	83.3	-9.8%
BQTerrace	73%	22.9%	73.3%	17.6%	83.3%	18.6%
BasketballDrive	100%	12.0%	100%	29.5%	100%	41.2%
RaceHorsesC	100%	21.7%	100%	20.2%	96.7%	18.9%
BQMall	100%	11.7%	100%	18.1%	86.7%	27.1%
PartyScene	96.7%	23.4%	100%	16.8%	93.3%	14.8%
BasketballDrill	100%	17.0%	100%	25.3%	90%	29.2%
RaceHorses	100%	12.1%	100%	19.0%	100%	26.7%
BQSquare	100%	6.7%	100%	12.5%	100%	13.0%
BlowingBubbles	100%	18.3%	100%	21.4%	96.7%	32.7%
BasketballPass	100%	13.9%	100%	18.8%	96.7%	34.0%

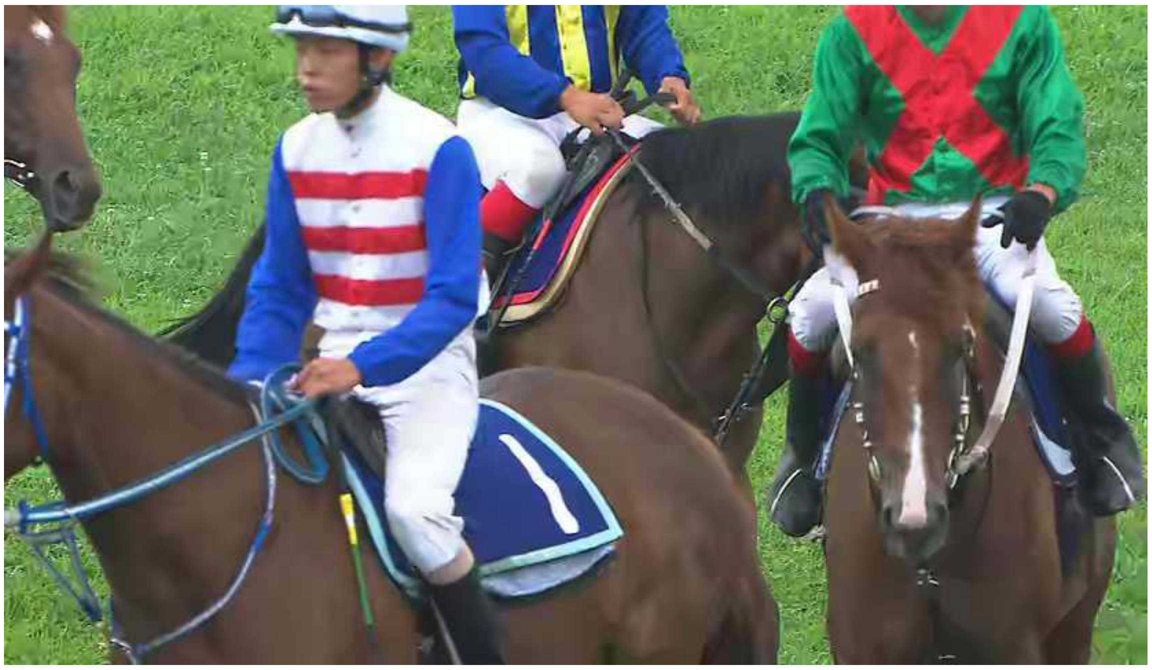


Figure 5.3: Minimum bitrate encoding for the high profile for 10th frame of test sequence PartyScene (832x480).

Table 5.4: Results for the Minimum Complexity Mode.

Name	**Low**		**Medium**		**High**	
	Constr	**Sav**	**Constr**	**Sav**	**Constr**	**Sav**
Traffic	100%	51.0%	100%	45.0%	100%	48.1%
PeopleOnStreet	100%	44.2%	100%	43.6%	100%	46.0%
NebutaFestival10bit	100%	56.7%	100%	52.8%	100%	48.7%
SteamLocomotiveTrain10bit	100%	58.0%	100%	58.0%	100%	54.3%
Kimono	100%	58.8%	100%	54.9%	100%	47.6%
ParkScene	100%	52.6%	100%	46.5%	100%	51.5%
Cactus	100%	52.2%	100%	48.7%	100%	56.5%
BQTerrace	100%	43.0%	100%,	41.8%	100%	47.4%
BasketballDrive	93.3%	49.9%	100%	54.4%	100%	56.2%
RaceHorsesC	93.3%	52.4%	100%	37.1%	100%	23.0%
BQMall	100%	45.8%	100%	32.6%	100%	27.2%
PartyScene	100%	43.3%	100%	42.4%	100%	42.5%
BasketballDrill	93.3%	40.7%	100%	39.2%	100%	41.4%
RaceHorses	93.3%	36.0%	100%	26.5%	100%	26.4%
BQSquare	83.3%	26.6%	100%	20.1%	90%	15.0%
BlowingBubbles	93.3%	42.3%	100%	22.5%	100%	14.1%
BasketballPass	80%	37.4%	100%	26.1%	93.3%	15.0%

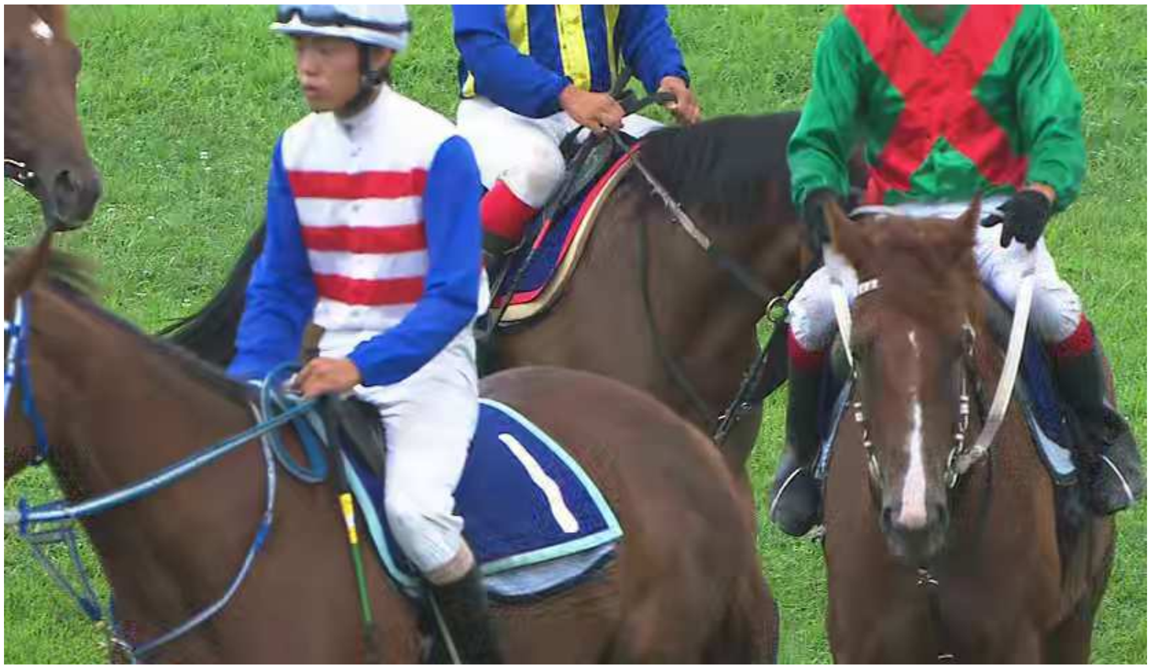


Figure 5.4: Minimum computational complexity mode for PartyScene (832x480) using the high profile mode.

Table 5.5: Results for the Minimum Distortion (maximum quality) Mode.

Name	**Low**		**Medium**		**High**	
	Constr	**Sav**	**Constr**	**Sav**	**Constr**	**Sav**
Traffic	93.3%	0.715	96.6%	0.295	90%	0.276
PeopleOnStreet	16.6%	1.129	26.6%	0.774	36.6%	0.630
NebutaFestival10bit	63.3%	1.263	86.6%	0.852	100%	0.846
SteamLocomotiveTrain10bit	33.3%	1.118	60%	0.632	86.6%	0.542
Kimono	46.6%	0.698	46.6%	0.016	80%	-0.025
ParkScene	30%	0.636	43.3%	0.361	63.3%	0.310
Cactus	66.6%	0.562	70%	0.190	70%	0.312
BQTerrace	16.6%	0.760	56.6%	0.069	46.6%	0.291
BasketballDrive	10%	0.571	6.6%	0.168	30%	-0.083
RaceHorsesC	86.6%	1.164	96.6%	0.671	96.6%	0.525
BQMall	50%	1.072	50%	0.319	60%	-0.132
PartyScene	80%	1.014	80%	0.556	90%	0.684
BasketballDrill	90%	0.782	73.3%	0.412	83.3%	0.353
RaceHorses	96.6%	1.042	90%	0.628	90%	0.503
BQSquare	100%	1.415	100%	0.904	93.3%	1.105
BlowingBubbles	63.3%	0.958	63.3%	0.474	56.6%	0.281
BasketballPass	90%	0.998	93.3%	0.269	93.3%	0.118

Table 5.6: Overall savings for all modes and profiles.

Percentage savings for minimum bitrate mode

Profile	***Mean***	***Std Dev***	***25th Pctle***	***Median***	***75th Pctle***
Low	15.2	6.0	11.85	16.3	20.0
Medium	17.5	8.0	13.45	18.8	22.1
High	20.2	12.9	16.70	19.8	28.2

Percentage savings for minimum computational complexity mode

Profile	***Mean***	***Std Dev***	***25th Pctle***	***Median***	***75th Pctle***
Low	46.5	8.6	41.5	45.8	52.5
Medium	40.7	11.8	29.6	42.4	50.8
High	38.9	15.2	24.7	46.0	50.1

PSNR (dB) savings for maximum image quality mode savings

Profile	***Mean***	***Std Dev***	***25th Pctle***	***Median***	***75th Pctle***
Low	0.9351	0.2527	0.7065	0.998	1.1235
Medium	0.4464	0.2694	0.2295	0.412	0.6515
High	0.3844	0.3267	0.1970	0.312	0.586

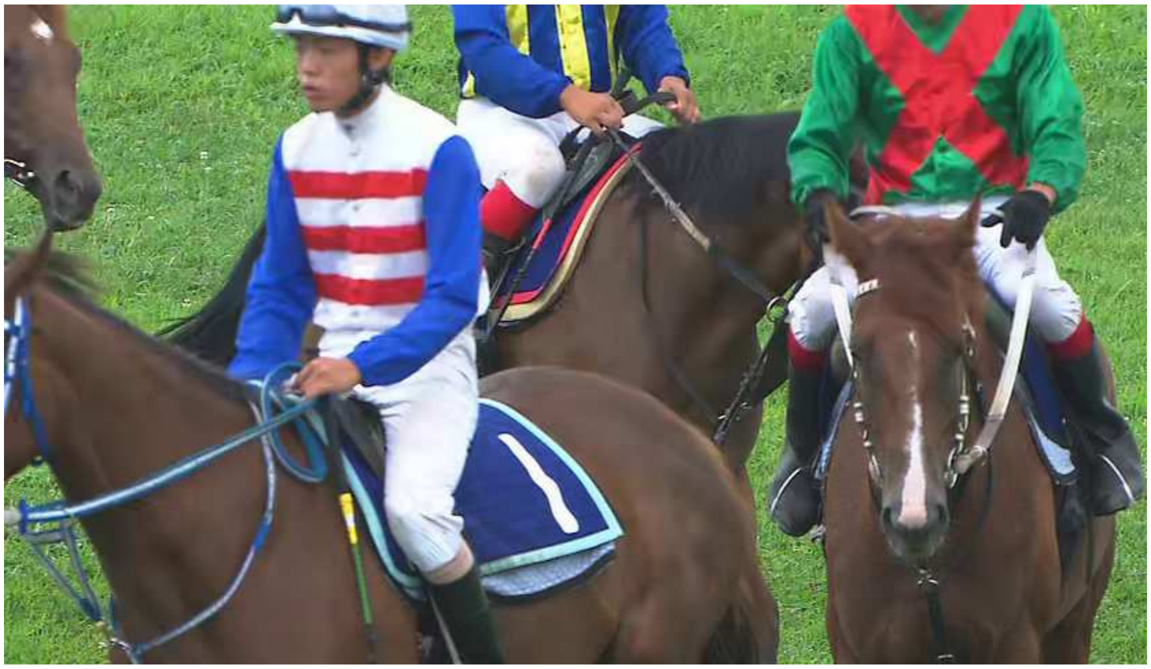


Figure 5.5: Minimum distortion (maximum quality) mode for the high profile for 10th frame of test sequence PartyScene (832x480).

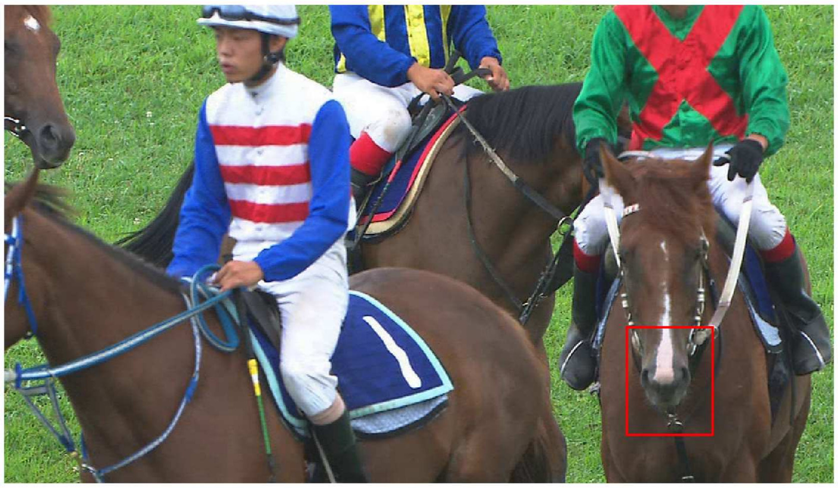
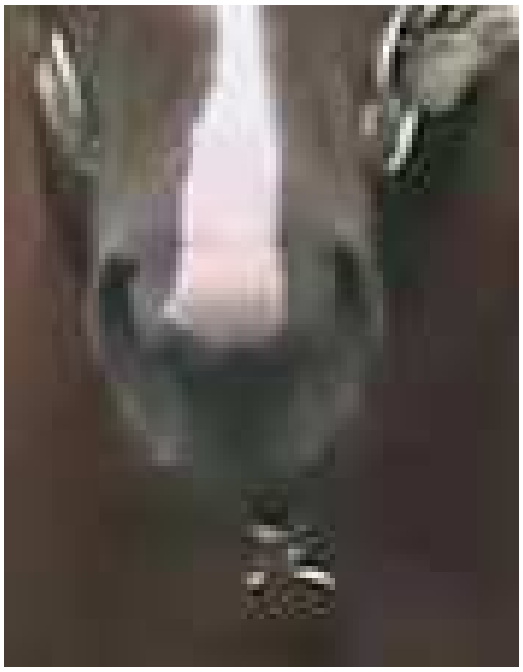


Figure 5.6: Uncompressed original 10th frame of RaceHorseC (832x480).


(a) Compressed 10th frame of RaceHorseC (832x480) using baseline encoding with HM-11.0 for high profile.

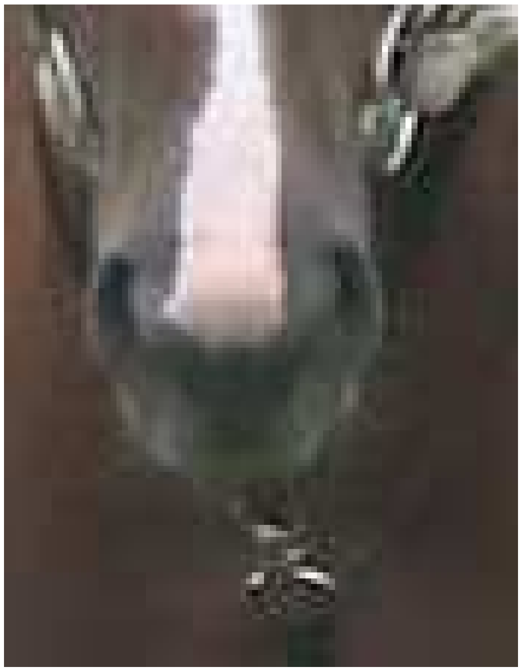

(b) Compressed 10th frame of RaceHorseC (832x480) for the minimum distortion mode for the high profile.

Figure 5.7: The detail enhancement comparison

Chapter 6

Results for Dynamic Control of Rate-quality-performance

The proposed joint-control method can also be used to dynamically switch between profiles and different optimization modes. In this chapter, we will demonstrate switching between profiles to satisfy different time-varying constraints. Switching is performed over several frames to avoid visual artifacts due to rapid encoding switching.

6.1 Profile Switching

We consider the RacehorseC (832x480) as an example test video sequence. The goal of our example is to demonstrate the ability to switch from a low profile to a medium and then a high profile.

For our examples, we define the low, medium, and high profiles by fixing `QP` to 27, 32 and 37 respectively. Furthermore, for comparing to the proposed approach, for controlling both the bit rate and PSNR, we use the full range depth configuration (`Config` = `13`). We require frames 0-30 to use the low profile, frames 31-60 to use the medium profile, and frames 61-90 to use the high profile.

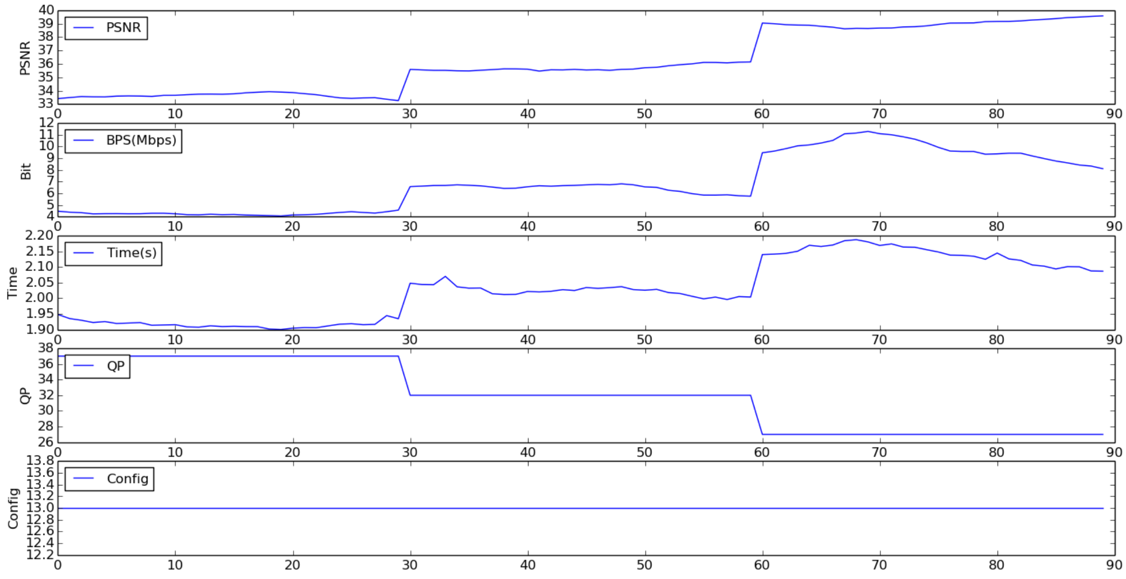


Figure 6.1: Profile switching by varying `QP` without the use of the joint-control algorithm. There is no way to enforce constraint satisfaction. The example is demonstrated on the PartyScene video sequence (832x480).

Profile switching for the minimum bitrate mode is shown in Fig. 6.2. To generate the example, the constraints are listed in Table 6.1. The example demonstrates dynamic adaptation of the PSNR to approximate the constraint. Furthermore, the PSNR constraint is met by avoiding any violations that are more than 5% less than the constraint value. Adaptation is achieved by adjusting the `QP` and `Config` parameters together. For the plots, for each video frame, we present the average `QP` and `Config` values for all CTUs.

Dynamic profile switching for the minimum complexity mode is shown in Fig.

Table 6.1: Constraints for switching profiles for the minimum bitrate mode.

Profile	**Performance Constr. in nano-seconds/frame**	**Quality Constr. PSNR**
High Profile	2,139,956,683	37.019
Medium Profile	2,024,068,684	33.695
Low Profile	1,916,881,835	31.628

6.3. The corresponding constraints are given in Table 6.2. The example clearly shows that the PSNR constraints have been exceeded while never exceeding the bitrate requirements. The complexity gets reduced as compared to the standard switching example of Fig. 6.2.

Profile switching for the minimum distortion (maximum image quality) mode is shown in Fig. 6.4. The associated constraints are given in Table 6.3. The example

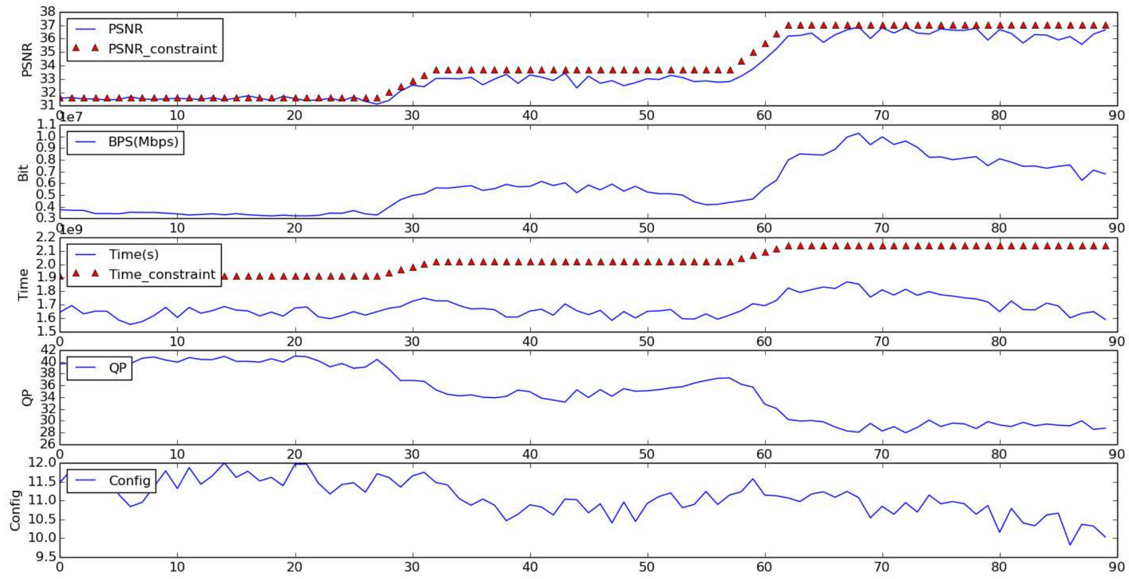


Figure 6.2: Profile switching for the minimum rate mode using the PartyScene video sequence (832x480).

Table 6.2: Constraints for switching profiles using the minimum computational complexity mode.

Profile	**Bitrate Constr. in Kbps**	**Quality Constr. in PSNR**
High Profile	10261.4	37.02
Medium Profile	6760.9	33.67
Low Profile	4478.8	31.63

clearly demonstrates that all of the constraints have been met. Furthermore, there is a nice, steady increase in the PSNR.

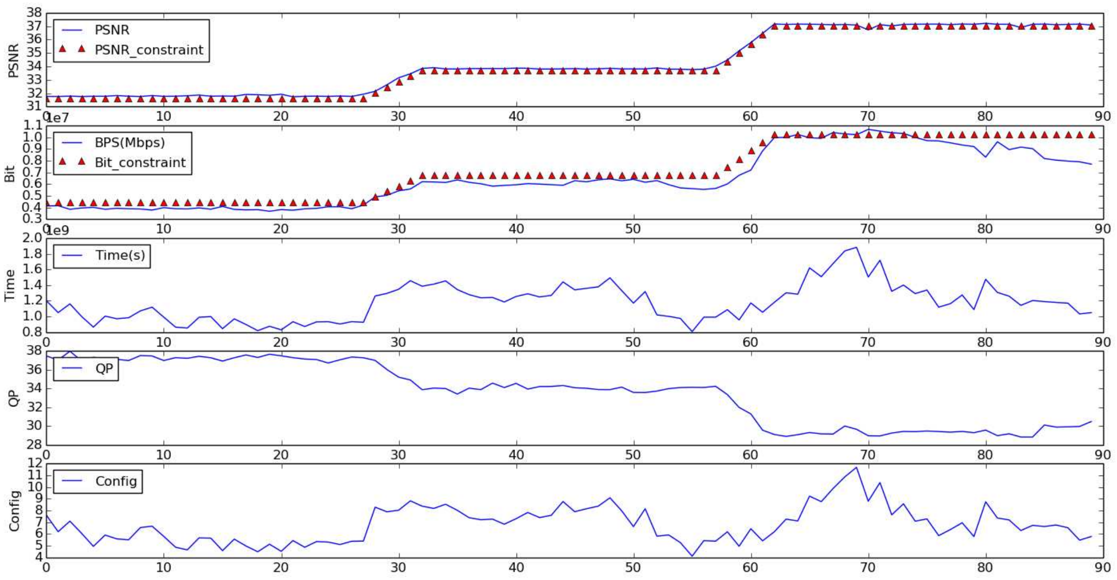


Figure 6.3: Profile switching using the minimum complexity mode for the PartyScene video sequence (832x480).

Table 6.3: Constraints for switching profiles for the minimum distortion (maximum quality) mode.

Profile	**Bitrate Constr. in Kbps**	**Quality Constr. PSNR**
High Profile	10261.4	37.02
Medium Profile	6760.9	33.70
Low Profile	4478.8	31.63

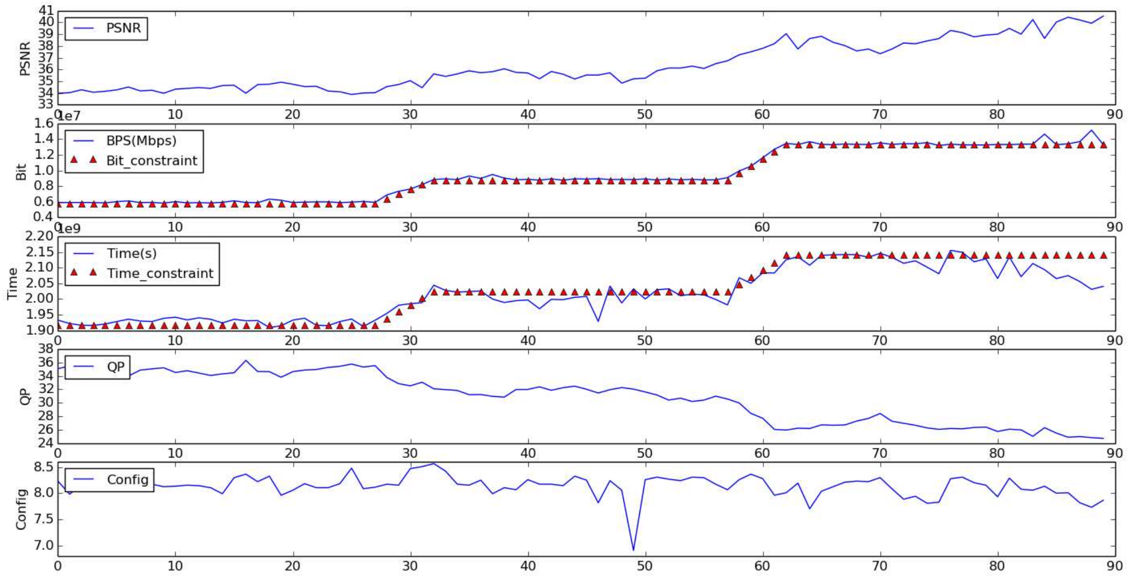


Figure 6.4: Profile switching using the minimum distortion mode (maximum quality mode) for the PartyScene test sequence (832x480).

Chapter 7

Discussion

The proposed approach provides new algorithms that allow for the joint control of rate, complexity, and quality. Prior methods have been restricted to rate control. We provide comparisons of improvements achievable through the use of the best rate control algorithms to show that the savings are similar to our minimum distortion (maximum image quality) mode. Nevertheless, we note that, unlike rate-control methods, the minimum distortion mode also allows to constrain computational complexity.

Rate control algorithms are used to either constrain the bitrate used for each frame or to maintain a constant bitrate while delivering higher quality. Instead, the use of multi-objective optimization allows us to use control surfaces that can be used to solve many different constraint optimization problems, not just problems involving the encoding rate. In fact, the approach is easily generalized to handle any number of objectives.

Next, we summarize results from each approach. Results from the unified R-Q algorithm for the random access mode are shown in Table 7.1. For the R-Lambda algorithm, results are shown in Table 7.2. For the SATD algorithm, results are shown in Table 7.3. Then, results for the minimum distortion mode are shown in Table 7.4.

From the tables, we can see that the rate control algorithms perform somewhat better than the minimum distortion model. However, the comparisons are misleading because the rate-control algorithms do not address complexity issues. The rate-control algorithms do not control computational complexity and thus there is no guarantee that they can meet a complexity bound.

Table 7.1: RQ model performance with RA, LB and LP for fluctuating ranges of ΔkbpsΔ and ΔPSNR

Mode	**Random Access**		**Low Delay B**		**Low Delay P**	
	$\Delta kbps$	$\Delta PSNR$	$\Delta kbps$	$\Delta PSNR$	$\Delta kbps$	$\Delta PSNR$
Avg	0.72%	0.42db	0.64%	0.61db	0.66%	0.39dB

Table 7.2: R Lambda model performance with RA, LB and LP for fluctuating ranges of Δkbps and ΔPSNR

Name	**Random Access**		**Low Delay B**		**Low Delay P**	
	$\Delta kbps$	$\Delta PSNR$	$\Delta kbps$	$\Delta PSNR$	$\Delta kbps$	$\Delta PSNR$
Avg	0.22%	1.08 dB	0.06%	0.29 dB	0.13%	0.29 dB

Table 7.3: SATD model performance with RA and LB for fluctuation ranges of kbps and PSNR

	α and β update	
	Before frame encoding	**After frame encoding**
	$\Delta kbps$	$\Delta kbps$
Avg	3.52%	0.74%

Table 7.4: Savings based on the minimum distortion mode (maximum quality) based on 5% bound violation.

Name	**Low**		**Medium**		**High**	
	$\Delta kbps$	$\Delta PSNR$	$\Delta kbps$	$\Delta PSNR$	$\Delta kbps$	$\Delta PSNR$
Avg	5%	0.935db	5%	0.446db	5%	0.384db

Chapter 8

Conclusion

8.1 Conclusion

The thesis presented a robust methodology for the joint control of rate-performance-quality for intra-mode HEVC encoding. The approach relies on the use of multi-objective optimization that can be easily extended to support more objectives. The joint-control algorithm has been used to solve the minimum bitrate, minimum complexity, and minimum distortion (maximum quality) optimization problems subject to constraints. As a result, the algorithm achieved significant reductions in bitrate, complexity, and distortion while satisfying realistic constraints.

8.2 Future Work

This section provides a summary of different directions to be considered in future research. There are three major recommendations.

In future research, the recommendation is to extend the concepts to different layers of HEVC and other encoding standards. For example, we can consider extensions from the GOP down to the TU levels. By working with more video encoding layers, the expectation is that we can provide significant improvements over the methods proposed in the thesis.

There is also room for improving budget allocation, model estimation, and constraint updates. The idea is to ensure that the approach works on many different scenarios based on more flexible conditions. The development of more flexible methods should also lead to performance improvements.

The approach should also be extended to adapt as a function of video content. The basic idea is to adjust the constraints and switch modes based on real-time video content. For this approach, the goal would be to support switching at the Group of Pictures (GOP) level. It would be very beneficial to be able reconsider the optimization problems from the GOP down to the Coding Tree Unit(CTU) level. An example would include encoding regions of interest through time to support live streaming.

References

[1] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 22, no. 12, pp. 1649–1668, 2012.

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

[3] L. Zhao, L. Zhang, S. Ma, and D. Zhao, "Fast mode decision algorithm for intra prediction in hevc," in *2011 IEEE Visual Communications and Image Processing (VCIP),*, nov. 2011, pp. 1 –4.

[4] W. Jiang, H. Ma, and Y. Chen, "Gradient based fast mode decision algorithm for intra prediction in hevc," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet),*, april 2012, pp. 1836 –1840.

[5] G. Correa, P. Assuncao, L. Agostini, and L. da Silva Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics,*, vol. 57, no. 4, pp. 1866–1874, November.

[6] Y. Jiang and M. Pattichis, "A dynamically reconfigurable architecture system for time-varying image constraints (drastic) for motion jpeg," *Journal of Real-Time Image Processing*, pp. 1–17, 2014.

[7] ——, "Dynamically reconfigurable dct architectures based on bitrate, power, and image quality considerations," in *19th IEEE International Conference on Image Processing (ICIP)*, Sept 2012, pp. 2465–2468.

[8] ——, "A dynamically reconfigurable dct architecture for maximum image quality subject to dynamic power and bitrate constraints," in *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, April 2012, pp. 189–192.

[9] C. Carranza, D. Llamocca, and M. Pattichis, “Fast and scalable computation of the forward and inverse discrete periodic radon transform,” *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 119–133, Jan 2016.

[10] ——, “The fast discrete periodic radon transform for prime sized images: Algorithm, architecture, and vlsi/fpga implementation,” in *IEEE Southwest Symposium on Image Analysis and Interpretation*, april 2014, pp. 169–172.

[11] ——, “A scalable architecture for implementing the fast discrete periodic radon transform for prime sized images,” in *IEEE International Conference on Image Processing*, Oct 2014, pp. 1208–1212.

[12] D. Llamocca and M. Pattichis, “A dynamically reconfigurable pixel processor system based on power/energy-performance-accuracy optimization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 3, pp. 488–502, March 2013.

[13] ——, “A self-reconfigurable platform for the implementation of 2d filterbanks with real and complex-valued inputs, outputs, and filter coefficients,” *VLSI Design*, vol. 2014, p. 5, 2014.

[14] D. Llamocca, C. Carranza, and M. Pattichis, “Separable fir filtering in fpga and gpu implementations: Energy, performance, and accuracy considerations,” in *21st Conference on Field Programmable Logic and Applications (FPL)*, Sept 2011, pp. 363–368.

[15] D. Llamocca and M. Pattichis, “Dynamic energy, performance, and accuracy optimization and management using automatically generated constraints for separable 2d fir filtering for digital video processing,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 4, pp. 31:1–31:30, dec 2014.

[16] Y. Jiang, D. Llamocca, M. Pattichis, and G. Esakki, “A unified and pipelined hardware architecture for implementing intra prediction in hevc,” in *IEEE Southwest Symposium on Image Analysis and Interpretation*, April 2014, pp. 29–32.

[17] Y. Jiang and M. Pattichis, “A dynamically reconfigurable deblocking filter for h.264/avc codec,” in *Asilomar Conference on Signals, Systems and Computers*, Nov 2012, pp. 2189–2193.

[18] ——, “Jpeg image compression using quantization table optimization based on perceptual image quality assessment,” in *Proc. 45th Asilomar Conference on Signals, Systems and Computers*, Nov 2011, pp. 225–229.

[19] Y. Jiang, G. Esakki, and M. Pattichis, "Dynamically reconfigurable architecture system for time-varying image constraints (drastic) for hevc intra encoding," in *Asilomar Conference on Signals, Systems and Computers*, Nov 2013, pp. 1112–116.

[20] Y. Jiang, C. Zong, and M. Pattichis, "Scalable hevc intra frame complexity control subject to quality and bitrate constraints," in *3rd IEEE Global Conference on Signal and Information Processing*, 2015, in press.

[21] J. N. J. Y. D. S. K. I. B. S. H. Choi, "Jctvc-h0213,rate control based on unified rq model for hevc," in *ISO/IEC JTC1/SC29 WG11*, San Jose, USA, Feburary 2012.

[22] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," April 2001.

[23] H. L. L. L. J. Z. B. Li, "Jctvc-k0103, rate control by r-lambda model for hevc," in *ISO/IEC JTC1/SC29 WG11*, Shanghai, China, October 2012.

[24] X. W. M. Karczewicz, "Jctvc-m0257, intra frame rate control based on satd," in *ISO/IEC JTC1/SC29 WG11*, Incheon,Korea, April 2013.

[25] I.-K. Kim, K. McCann, K. Sugimoto, B. Bross, and W.-J. Han, "High efficiency video coding (hevc) test model 11 (hm11) encoder description," 2013.

[26] J. C. Yinji Piao, Junghye Min, "Encoder improvement of unified intra prediction, jctvc-c207," 2010.