Modeling of Power, Rate and Perceptual Quality of Scalable Video and Its Applications

D I S S E R T A T I O N

for the Degree of

Doctor of Philosophy (Electrical Engineering)

Zhan Ma

January 2011

Modeling of Power, Rate and Perceptual Quality of Scalable Video and Its Applications

D I S S E R T A T I O N

Submitted in Partial Fulfillment
of the REQUIREMENTS for the

Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

POLYTECHNIC INSTITUTE OF NEW YORK UNIVERSITY

by

Zhan Ma

January 2011

Approved:

_____
Department Head

_____
Date

Copy No. ____

Approved by the Guidance Committee:

Major: Electrical and Computer Engineering

Yao Wang
Professor of
Electrical and Computer Engineering

Hai (Helen) Li
Assistant Professor of
Electrical and Computer Engineering

Kyeong Yang
Director, Video Technologies
Dialogic Inc.

Minor: Computer Science

Edward K. Wong
Associate Professor of
Computer Science and Engineering

VITA

Zhan Ma was born in China on September 20, 1982. He received the B.S. and M.S. degrees in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2006, respectively. While pursuing the M.S. degree, he joined the national digital audio and video standardization (AVS) workgroup to participate in standardizing the video coding standard in China. Since August 2006, he has been a Ph.D. student at Electrical Engineering Department in Polytechnic Institute of New York University, Brooklyn, NY 11201 under the supervision of Professor Yao Wang. From May 2008 to April 2009, May 2009 to August 2009, May 2010 to August 2010, he interned at the Corporate Research, Thomson Inc. (now Technicolor), NJ, Texas Instruments Inc., TX, and Sharp Labs of America, WA, respectively. During his thesis study, he published 12 technical papers and had 11 patents pending. He has conducted the research in the fields of energy efficient video processing, scalable video coding and power consumption, rate and perceptual quality modeling of the scalable video. He received the 2006 Special Contribution Award from the national digital audio and video standardization workgroup, China for his contribution in standardizing the AVS Part 7, and 2010 Patent Incentive Award from Sharp Labs of America.

To my wife Yi Chen, our parents and my grandma
for their love and support.

ACKNOWLEDGEMENT

First, I am indebted to Professor Yao Wang for her invaluable support, inspiration and guidance throughout the course of this dissertation research. Her extensive knowledge, enlightened direction and continuous help encourage me for my entire Ph.D study and future career. I would like to thank other members of my thesis committee. Professor Hai (Helen) Li, Professor Edward Wong and Dr. Kyeong Yang have closely supervised my dissertation research. Their patient help and constructive advice make this thesis better.

I would also like to thank Hao Hu and Meng Xu, who worked with me closely on many projects through my Ph.D period. I am also very grateful to my colleagues at Thomson Corporation Research, Texas Instruments, Inc., Sharp Labs of America: Dr. Daniel Luo, Dr. Yunfei Zheng, Dr. Minhua Zhou, Dr. Do-Kyoung Kwon, Dr. Andrew Segall for their help and insightful technical discussion. Leveraging this opportunity, I acknowledge the members of the Video Lab: including Dr. Beibei Wang, Dr. Zhengye Liu, Dr. Ozgu Alay, Cagdas Bilen, Yen-Fu Ou, Xuan Zhao and Xiaozhong Xu, for their collaboration and discussion. I also want to thank other members and friends in our department: Professor Yong Liu, Professor X.-K. Chen, Dr. Yanming Shen, Dr. Pei Liu and Dr. Chao Liang. Because of all of them, my life at Poly is much more enjoyable.

In particular, I want to thank my dear wife, Yi Chen, for her continuous understanding, caring and love since our high school, and our parents and my grandma for their continual support and encouragement.

This work is supported by National Science Foundation under Grant No. 0430145, and the New York State Center for Advanced Technology in Telecommunications (CATT) at Polytechnic Institute of New York University, Brooklyn, New York, USA.

AN ABSTRACT

Modeling of Power, Rate and Perceptual Quality of Scalable Video and Its Applications

by

Zhan Ma

Advisor: Yao Wang

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy (Electrical Engineering)

January 2011

Wireless video streaming and playback on mobile handhelds has become a very popular application because of the advances of the high speed wireless access network and semiconductor technologies. How to deliver video content to different battery powered mobile users (receivers) over heterogeneous networks is a fundamental problem to be solved. To tackle this problem, a common approach is to use scalable video which can be adapted according to sustainable receiving bandwidth and the available decoding power at the receiver. A key challenge in this approach is how to determine the appropriate spatial, temporal, and amplitude resolution (STAR) at which to decode the video, to maximize the perceptual quality, given the bandwidth and power constraints. The solution of this problem requires accurate models that relate the quality, rate, and decoding complexity with the STAR. Such models enable us to solve the aforementioned problem with an analytically tractable solution.

In this thesis, we focus on the effect of temporal and amplitude resolutions (i.e. frame rate and quantization stepsize) on the quality, rate and complexity. Three analytical models, i.e., perceptual quality, rate and decoding complexity (power consumption) models are

developed to explore the impact of temporal and amplitude resolution. We have applied an "impact separation" methodology to differentiate the effect by temporal resolution (i.e., frame rate) and amplitude resolution (i.e., quantization stepsize) respectively, and proposed single variable functions to express the separable effects of frame rate and quantization. The overall models are then the product of a function of frame rate, and a function of the quantization stepsize, where the model parameters are content dependent. We have evaluated our proposed models using the actual measurement data, such as raw MOS (mean opinion score) from subjective tests, bit rate and complexity points by real video encoding and decoding. Results show that our models can accurately estimate the measurement data with very small root mean square error (RMSE) and high Pearson correlation (PC).

We further investigated how how to estimate the model parameters using content features. Toward this, we have implemented a lightweight, simple pre-processor using macroblock based integer motion estimation. We have selected a set of features which can be easily obtained from this simple pre-processor, such as the frame difference, displaced frame difference, motion vector magnitude, motion direction activity, video contrast, etc. Results show that, with a proper feature combinations, we can estimate the parameters very well. Using our proposed models, we developed efficient algorithm to solve the power-rate constrained scalable video adaptation problem analytically, i.e., providing the best decoded video quality given the network bandwidth and local battery power constraints.

Our solution for video adaptation based on the bandwidth and power constraints assumes that the receiver adapts its processor operating frequency based on the needed number of operations. This requires accurate prediction of the decoding complexity at the frame or GOP (group of picture) level. Towards this goal we further developed a frame decoding complexity prediction algorithm along with video decoding. According to our simulation results using various videos with different contents, resolutions and bit rates, our model can predict the frame decoding complexity very well (i.e., average relative error less than 3%). The frame decoding complexity can be also extended to the GOP level model, which not only improves the complexity prediction accuracy but also reduces the overhead. Also, we devise our frame decoding complexity prediction algorithm on mobile platform to guide

the processor voltage and frequency adaptation. In devices using dynamic voltage and frequency scaling (DVFS), our complexity prediction based solution can achieve up to 50% power saving for popular ARM processor, and more than 70% saving for Intel Pentium mobile architecture.

List of Publications

Publications

1. Zhan Ma, Hao Hu and Yao Wang, "On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding," *submitted to IEEE Trans. on Multimedia*, Nov. 2010.

2. Zhan Ma, Meng Xu and Yao Wang, "Power-rate optimized scalable video adaptation," *prepared for IEEE Trans. on Circuits and Systems for Video Technology*, Dec. 2010.

3. Zhan Ma, Hao Hu and Yao Wang, "DVFS-enabled energy efficient video decoding," IEEE ComSoc MMTC e-letter, pp.20-24, July 2010.

4. Yen-Fu Ou, Zhan Ma, Tao Liu and Yao Wang, "Perceptual Quality Assessment of Video Considering both Frame Rate and Quantization Artifacts," *accepted by IEEE Trans. on Circuits and Systems for Video Technology*, Sept. 2010.

5. Yuanyi Xue, Yen-Fu Ou, Zhan Ma and Yao Wang, "Perceptual Video Quality Assessment On A Mobile Platform Considering Both Spatial Resolution And Quantization Artifacts," *Proc. of PacketVideo*, Hong Kong, December 2010.

6. Zhan Ma, Jiancong Luo, Peng Yin, Cristina Gomila and Yao Wang, "Smoothed Reference Inter-layer Texture Prediction for Bit Depth Scalable Video Coding," *Proc. of VCIP*, San Jose, Jan. 2010.

7. Yen-Fu Ou, Zhan Ma, and Yao Wang, "Modeling the Impact of Frame Rate and Quantization Stepsizes and Their Temporal Variations on Perceptual Video Quality: A Review of Recent Works," *Proc. of IEEE CISS*, Princeton, NJ, March 2009.

8. Yao Wang, Zhan Ma, and Yen-Fu Ou, "Modeling Rate and Perceptual Quality of Scalable Video as Functions of Quantization and Frame Rate and Its Application in Scalable Video Adaptation," *Proc. of PacketVideo*, Seattle, WA, May 2009, (*invited paper*).

9. Zhan Ma, Zhongbo Zhang and Yao Wang, "Complexity Modeling of H.264 Entropy Decoding," *Proc. of IEEE ICIP*, Oct., 2008

10. Zhan Ma and Yao Wang, "Complexity Modeling of SVC Decoding," *Proc. of IEEE ICASSP*, April 2008.

11. Yen-Fu Ou, Zhan Ma and Yao Wang, "A Novel Quality Metric for Compressed Video Considering both Frame Rate and Quantization Artifacts," *Proc. of VPQM*, Arizona, 2009.

12. Yen-Fu Ou, Tao Liu, Zhi Zhao, Zhan Ma and Yao Wang, "Modeling the Impact of Frame Rate on Perceptual Quality of Video," *Proc. of IEEE ICIP*, Oct., 2008.

JCTVC Proposal

1. Zhan Ma and Andrew Segall, "System for graceful power degradation," Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC JTC1 MPEG, Doc. JCTVC-B114, Geneva CH, July 2010

Pending Patents and Disclosures

1. Zhan Ma, Andrew Segall, "System for graceful power degradation of video processing," July 2010.

2. Zhan Ma, Andrew Segall, "System for low resolution power reduction with deblocking," July 2010.

3. Zhan Ma, Andrew Segall, "System for low resolution power reduction with high resolution deblocking," July 2010.

4. Zhan Ma, Andrew Segall, "System for low resolution power reduction with deblocking flag," June 2010.

5. Zhan Ma, Andrew Segall, "System for low resolution power reduction with compressed image," June 2010.

6. Zhan Ma, Andrew Segall, "System for low resolution power reduction with low resolution intra prediction," August 2010.

7. Zhan Ma, Andrew Segall, "System for low resolution power reduction with low resolution motion-compensation," August 2010.

8. Zhan Ma, Andrew Segall, "System for frame buffer compression with edge directed interpolation," August 2010.

9. Zhan Ma, Do-Kyoung Kwon, "Methods and apparatus for classification based perceptual video coding and its application on rate control," August 2009.

10. Zhan Ma, Jiancong Luo, Peng Yin, "Methods and apparatus for motion compensation with smoothed reference frame in bit depth scalability," February 2009.

11. Jiancong Luo, Zhan Ma, Peng Yin, "Methods and apparatus for smoothed tone mapping of bit depth scalability" February, 2009.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Challenge, Motivation and Our Approach

It is indeed the multimedia era now. Due to the advanced wireless access network and semiconductor technologies, wireless multimedia services, particularly, wireless video services are becoming ubiquitous in our daily life. We can easily use our SmartPhone to watch the streaming video via WiFi or 3G network, we can also record video clips (or take pictures) using the popular mobile devices and upload them to the social sharing network community (like Youtube, Facebook, Twitter, etc). Furthermore, we can even use our SmartPhone to do live video conferencing, such as the FaceTime introduced by the Apple iPhone 4. Among them, wireless video streaming and playback (WVSP) is one of the most popular applications.

As shown in Figure 1.1, there are two fundamental problems for a successful wireless video streaming and playback application. One is how to deliver the same content to different users over different access networks without introducing much operation overhead. The other is how to deal with the limited battery power supply for current mobile handhelds without losing much video playback quality. In principle, these two problem can be rephrased as – *how to deliver the same video with the best quality under the network bandwidth and battery power constraints at mobile device?*, i.e.,

$$\max\ Q \quad \text{subject to} \quad R \leq R_0 \quad P \leq P_0. \tag{1.1}$$

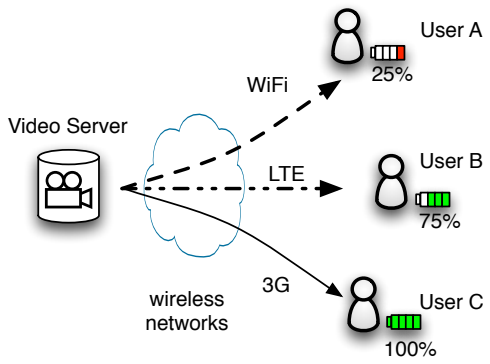


Figure 1.1: Illustrative example of ubiquitous wireless video streaming to different mobile users with different remaining battery power over diverse wireless/mobile access networks.

where Q is the decoded video quality, R_0 and P_0 are the sustain network bandwidth and battery power constraints for a typical mobile target. Specially, there are three aspects of the problem (1.1),

- How to deliver the same video to different users via heterogeneous access networks (e.g., WiFi, 3G, wired, etc)?
- How to ensure that the decoded video has the "best" quality for each receiver?
- How to prolong the battery life of the mobile handhelds without sacrificing the video playback quality much?

An efficient and effective solution for above concerns promises the success of the application.

We propose to use the scalable video to solve the diversities introduced by the underlying access networks and subscribed mobile receivers. In practice, a single scalable video stream can be easily truncated into substreams with different reconstruction quality levels (e.g., in terms of temporal resolution, amplitude enhancement [1] and spatial augment) to meet the underlying network and end-user differences. Compared with the video transcoding approach which usually requires the powerful server to do computational intensive

[1] We use the "amplitude scalability" to note the conventional quality or SNR scalability in scalable video to avoid ambiguity with perceptual quality definition.

video transcoding, scalable video just needs a lightweight adaptor to do the bitstream adaptation. Compared with simulcast, scalable video can reduce the total network bandwidth requirement dramatically, especially when supporting many subscribers with different access network bandwidth. To tackle the other two concerns, we have developed three analytical models to address the perceptual quality, rate and power consumption for scalable video with the focus on the temporal and amplitude scalability [2]. To further reduce the power consumption in video decoding, we have devised an efficient frame or GOP (group of pictures) decoding complexity prediction algorithm to guide the voltage and clock rate adjustment of the underlying processor so as to save the video decoding energy.

To explore the impact of temporal and amplitude scalability on the perceptual quality, rate and decoding power consumption, we propose to use an "impact separation" methodology. Specially, we normalize the raw data points for any given combination of temporal and amplitude resolution, by the data points at maximum temporal and amplitude resolution respectively, and then investigate how to use appropriate analytical functions to model the separated normalized effects. As a result, each model is expressed as the product of a function of temporal resolution (i.e., frame rate) and a function of amplitude level (i.e., quantization stepsize), with two or three model parameters in total. For practical use, we also explore the model parameter prediction using content features. Results indicate that by choosing proper feature combinations, we can estimate the model parameters very well. With the content predicted parameters, we also show that model predicted value, such as mean opinion score (MOS) for quality model, bit rates for rate model, complexity (i.e., in terms of cycles per second) for complexity model, can accurately predict the actual corresponding data points collected by experiments. We first give a brief introduction of the scalable video coding in the following section. More details on model development will be unfolded in subsequent chapters.

[2]The spatial scalability impact will be deferred as our future research.

1.2 Scalable Video Coding

Scalable video coding (SVC) refers to the scalable extension of the H.264/AVC [1], which is the latest video coding standard providing the temporal, amplitude and spatial scalability. SVC has been actively researched for at least two decades. Several international standards, such as MPEG-2 video [2], H.263 [3], and MPEG-4 visual [4], also define their scalable profiles. However, none of them has been accepted and used in the market. Many reasons have caused the failure of the scalable video coding in the history. Two of the most important reasons are the poor coding efficiency [compared with the single layer coding] and the dramatic complexity demands for enabling the scalable tools. Because of these defects, no commercial product is available in the market to support those scalable tools provided by previous video coding standards.

Unlike its predecessors, SVC [3] inherits the well-designed structure of the H.264/AVC and introduces additional new tools (such as inter-layer intra prediction, inter-layer mode and/or motion prediction, as well as the residual prediction) to remove the inter-layer redundancy so as to provide the scalability efficiently. An optimized SVC encoder [5, 6] can provide the scalable videos with just +10% bit rate overhead in comparison with the H.264/AVC single layer coding. Because of the same network interface design shared by both SVC and H.264/AVC (i.e., network abstraction layer (NAL) structure), and its own layered structure, SVC is friendly to the error-prone network transmission that promises the dominant role of the SVC in the networked video area. Moreover, the complexity issue is well studied during the SVC standardization phase. Tools are chosen to provide the decent coding efficiency without introducing much decoding complexity. One important feature of the SVC is the "*single loop motion compensation*" or "*single loop decoding*", which constraints the decoding loop as to always close at the target layer without extra reference layer pixel domain reconstruction. Thus, compared with conventional single layer H.264/AVC decoding, layered SVC decoding doesn't bring too much computational overhead [6]. Chapter 4 will present the detailed analysis and discussion regarding the SVC

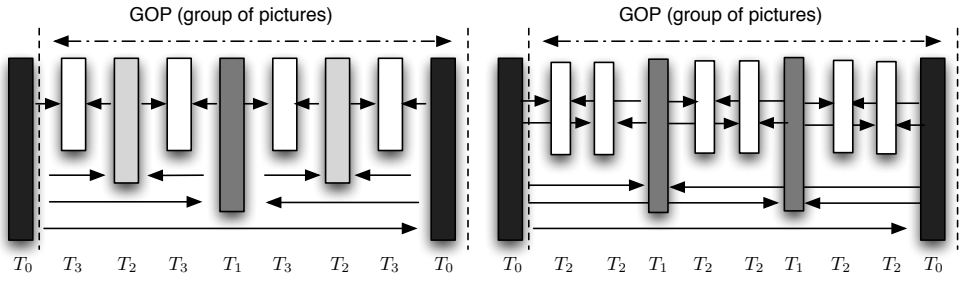
[3] SVC is used to indicate the scalable extension of the H.264/AVC unless we point out explicitly.

decoding complexity.

Table 1.1: SVC scalability and related tools

scalability	tool
temporal	hierarchical B picture (either dyadic or non-dyadic)
spatial, amplitude	inter-layer intra prediction inter-layer mode/motion prediction inter-layer residual prediction

Three basic types of the scalability are supported by the SVC, i.e. temporal, spatial and amplitude scalability. As known, additional tools are developed to enable these scalability as listed in Table 1.1. Hierarchical B pictures is used to provide the temporal scalability. Compared with the conventional B pictures, hierarchical B only requires high-level syntax signaling changes and also improves the coding performance. Dyadic or non-dyadic structure can be enabled to present different temporal resolutions to satisfy different application scenarios as shown in Figure 1.2. In our work, we will focus on the dyadic hierarchical prediction structure.



(a) dyadic (b) non-dyadic

Figure 1.2: Hierarchical B prediction structure enabled in SVC, T_k indicates the temporal resolution at level k.

Layered structure is employed in SVC to enable the amplitude and spatial scalability. To reduce the inter-layer redundancy, inter-layer intra prediction, inter-layer mode/motion prediction and inter-layer residual prediction are used adaptively. Because of the resolution change for spatial scalability, additional upsampling and inter-layer deblocking operations

are required. This thesis work considers the joint amplitude and temporal scalable coding, while leaves the spatial scalability and overall combined scalability as future study. Amplitude scalability, also noted as SNR or "quality" scalability in the literature, provides the signal amplitude refinements from "poor" to "excellent" signal reconstruction. Simply, we can control the quantization parameter (QP) at each layer to refine the video amplitude fidelity. For example, we usually choose higher QP at base layer, and use smaller QP at enhancement layer to improve the signal fidelity. There are two types of amplitude scalability supported in SVC. One is the coarse grain scalability (CGS) that is the special case of spatial scalability with identical video resolution for each layer. The motion compensation (MCP) has to be conducted within the same layer in CGS [6]. The other one is the medium grain scalability (MGS), which allows higher enhancement layer reconstruction signal as reference to improve the coding efficiency. However, once the packets at enhancement layer are lost, it will introduce the decoder drift. Thus, SVC introduces the *key picture* to delimit the decoder drift between two key frames [6]. Also, transform coefficients splitting and packetization can be combined with MGS as well to provide more rate truncation points. In our research, we have constrained the MGS with MCP at current layer to avoid the decoder drift. With decoder drift impact, it is hard to evaluate the perceptual quality of the decoded video. This constrained MGS is similar as the CGS, but with more amplitude layers (e.g., up to 5 in our work). CGS can only support up to 3 layers because it uses the same signaling syntax as the spatial scalability. More details about SVC technology and related topics can be found in [5–9].

1.3 Dissertation Outline

This thesis is organized as follows. In Chapter 2, we first introduce the perceptual quality metric of scalable video considering both frame rate and quantization artifacts. We propose to separate the impact of frame rate and quantization, and use two single parameter exponential functions to characterize the perceptual quality in terms of frame rate and quantization respectively.

In Chapter 3, we apply the same idea to separate the impact of the frame rate and quantization on the bit rate, each of which can be well captured by a single parameter power function. Together with the maximum bit rate encoded at maximum frame rate and finest quantization, there are three content dependent parameters in our proposed rate model.

In Chapter 4, we first derive the complexity model for joint temporal and amplitude scalable video decoding as the product of a single parameter power function of the quantization stepsize (where the parameter is temporal resolution adaptive and can be also modeled as the power function of the frame rate), and a linear function of the temporal resolution. Overall, there are also three parameters for the complexity model as the rate model. By incorporating the power consumption model for ARM processor, we extend our complexity model to the power consumption model for scalable video decoding on ARM platform. In addition to power consumption or complexity modeling considering the temporal and amplitude resolution variation, we also propose the on-line complexity prediction along with the video decoding. The estimated frame decoding complexity can be used to dynamically adapt the voltage and clock rate of underlying processor so as to save more power.

In Chapter 5, we explore the model parameter prediction using content features. We have implemented a simple, lightweight pre-processor with a macroblock (i.e., 16x16) based integer motion estimation engine ahead of real encoding to collect the necessary information for feature extraction, such as *residual signal*, *motion fields* and *original video signal*. In addition to present the parameter prediction performance, we also provide the model accuracy with estimated parameters by comparing the prediction and original raw data.

In Chapter 6, we present two popular applications using our developed models. At first, we apply our models to guide the resource constrained scalable video adaptation, under network bandwidth and mobile battery capacity constraints. Using our models, we can solve the problem analytically without taking too much computing resource. The second application is using our proposed complexity prediction algorithm to adapt the voltage and frequency of the underlying processor so as to save video playback power. Our complexity

prediction based DVFS (dynamic voltage and frequency scaling) scheme has been evaluated on mobile architectures, including mobile laptop and SmartPhone platforms.

Chapter 7 concludes the thesis and discuss some future research directions.

To help the understanding, we first summarize the notations and abbreviations frequently used throughout this thesis in Table 1.2.

Table 1.2: Summary of notations and abbreviations used in this thesis

name	notation
q	quantization stepsize with ideal range (0, +∞) in our system, we use $q_{\min}$ = 16 and $q_{\max}$=104.
t	frame rate within (0, 30] Hz, in our simulation, we set $t_{\max} = 30$ Hz and $t_{\min} = 1.875$ Hz.
$Q(q,t)$	perceptual quality encoded using q at frame rate t $Q_{\max}$ as the maximum quality at $t_{\max}$ and $q_{\min}$
$R(q,t)$	video bit rate encoded using q at frame rate t $R_{\max}$ as the maximum bit rate at $t_{\max}$ and $q_{\min}$
$C(q,t)$	video decoding complexity at (q,t)-th layer $C_{\max}$ as the maximum complexity at $t_{\max}$ and $q_{\min}$
$P(q,t)$	SVC decoding power consumption model at (q,t)-th layer
$\mu_{\mathbf{X}}$ $\sigma_{\mathbf{X}}$	mean of X standard deviation of X
CIF QCIF WVGA 720p	common intermediate format with resolution at 352x288 quarter CIF with resolution at 176x144 wide video graphics array with resolution at 832x480 high definition video with resolution at 1280x720
MOS PSNR	mean opinion score, which is a numerical indication of the perceived quality of compressed video or auido peak signal to noise ratio
DVFS	dynamic voltage and frequency scaling, which is a technique widely adopted in modern processors to conserve the energy

Chapter 2

Perceptual Quality Metric of Scalable Video

In this chapter, we explore the impact of frame rate and quantization on perceptual quality of a video. We propose to use the product of a spatial quality factor that assesses the quality of decoded frames without considering the frame rate effect and a temporal correction factor for quality, which reduces the quality assigned by the first factor according to the actual frame rate. We find that the temporal correction factor for quality follows closely an inverted falling exponential function, whereas the quantization effect on the coded frames can be captured accurately by an exponential function of the quantization stepsize q. The proposed model is analytically simple, with each function requiring only a single content-dependent parameter. The proposed overall metric has been validated using our subjective test scores as well as those reported by others in the literature [10]. For all data sets examined, our model yields high Pearson correlation (higher than 0.95) with measured MOS. In the following Chapter 5, we further investigate how to predict parameters of our proposed model using content features derived from the original videos. Using content predicted parameters, our model still fits with measured MOS with high correlation (over 0.97 in average).

2.1 Motivation and Related Works

Development of objective quality metrics that can automatically and accurately measure perceptual video quality is becoming more and more important as video applications

become pervasive. Prior work in video quality assessment is mainly concerned with applications where the frame rate of the video is fixed. The objective quality metric compares each pair of corresponding frames in deriving a similarity score or distortion between two videos with the same frame rate. In many emerging applications targeting for heterogeneous users with different display devices and/or different communication links, the same video content may be accessed with varying frame rate, frame size or quantization (assuming the video is coded into a scalable stream with spatial/temporal/amplitude scalability). In applications permitting only very low bit rate video, one often has to determine whether to code an original high frame-rate video at the same frame rate but with significant quantization, or to code it at a lower frame rate with less quantization. In all proceeding scenarios as well as many others, it is important being able to objectively quantify the perceptual quality of a video that has been subjected to both quantization and frame rate reduction.

There have been several works studying the impact of frame rate artifacts on perceptual video quality. In a recent review of frame rate effect on human perception of video [11], it is found that frame rate around 15 Hz seems to be a threshold of humans' satisfaction level, but the exact acceptable frame rate varies depending on video content, underlying application, and the viewers. In addition, the authors of [12] proposed that the preferred frame rate decreases as video bandwidth decreases, and two switching bandwidths corresponding to the preferred frame rates were derived. The work in [13] investigated the preferred frame rate for different types of video. In [14], a particular high-motion type of coded video sequences (sports game) was explored. It was found that high spatial quality is more preferable than high frame rate for small screens. However, no specific quality metric, which can predict the perceived video quality, were derived in these works [11–14].

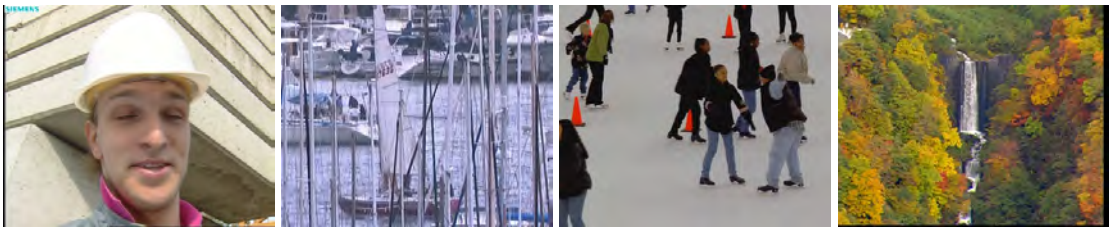
The works in [15–17] proposed quality metrics that consider the effect of frame rate. The work in [15] used logarithmic function of the frame rate to model the negative impact of frame rate dropping on perceptual video quality in the absence of compression artifacts. The model was shown to correlate well with subjective ratings for both CIF and QCIF videos. However, this model requires two content-dependent parameters, which may limit its applicability in practice. The metric proposed in [16] explores the impact of regular

and irregular frame drop. The quality of each video scene is determined by weighting and normalizing a logarithm function of temporal fluctuation and the frame dropping severity. Finally, the overall quality of the entire video is the average of the quality indices over all video scene segments. The work in [17] also considers the impact of both regular and irregular frame drops and examines the jerkiness and jitter effects caused by different levels of strength, duration and distribution of the temporal impairment. However, [16] did not provide a single equation, which can predict the perceptual quality of regular frame drops, and even though [17] did, the proposed quality model has four parameters, and the authors did not consider how to derive these parameters from the underlying video.

Besides the study of frame rate impact on perceptual quality, Feghali *et al.* proposed a video quality metric [18] considering both frame rate and quantization effects. Their metric uses a weighted sum of two terms, one is the PSNR of the interpolated sequences from the original low frame-rate video, another is the frame-rate reduction. The weight depends on the motion of the sequences. The work in [19] extended that of [18] by employing a different motion feature in the weight. The work in [20] proposed a quality metric considering block-fidelity, content richness fidelity, spatial-textural, color, and temporal masking. They combined all these components into a quality index to predict the perceptual quality. This model involves sophisticated processing to extract content components from video sequences. Hence, it may not be applicable for practical application.

Our proposed model uses the product of a spatial quality factor (SQF) and a temporal correction factor for quality (TCFQ). The first term assesses the quality of video due to the quantization, and the TCFQ reduces the quality assigned by the first metric according to the actual frame rate. Our model has only two content-dependent parameters, and correlates very well with subjective ratings obtained in our subjective tests as well as subjective scores reported in other papers [10], with significantly higher correlation than the metrics proposed in other works as well as reduced complexity.

(a) akiyo (b) city (c) crew (d) football



(e) foreman (f) harbour (g) ice (h) waterfall

Figure 2.1: CIF (352x288) resolution test video sequences.

2.2 Subjective Quality Assessment

2.2.1 Test Sequence Pool

Seven video sequences, “akiyo”, “city”, “crew”, “football”, “foreman”, “ice”, “waterfall”, all in CIF (352×288) resolution at original frame rate 30 fps, are chosen from JVT (Joint Video Team) test sequence pool [21], which are shown in Figure 2.1. All these sequences are coded using the JSVM [22], which is the reference software for the scalable extension of H.264/AVC (SVC) developed by JVT. Each sequence is encoded with 5 amplitude and 5 temporal layers using constrained MGS. We extract bitstreams with four temporal layers corresponding to the frame rates of 30, 15, 7.5, 3.75 Hz, and each temporal layer in turn has four amplitude layers created with QP equal to 28, 36, 40, and 44[1], respectively. A processed video sequence (PVS) is created by decoding a scalable bitstream up to a certain temporal and amplitude layer.

The subjective rating tests for the seven sequences were done in two separate experi-

[1]Different from JSVM default configuration utilizing different QPs for different temporal layers, the same QP is chosen among all temporal layers at a certain amplitude layer.

ments. In the first experiment, 64 PVSs from four sequences (“akiyo”, “city”, “crew”, and “football”) were rated, varying among four frame rates (30, 15, 7.5 and 3.75 Hz) and four QP levels (28, 36, 40, and 44). In the second experiment, 60 PVSs from five sequences (“akiyo”, “football”, “foreman”, “ice” and “waterfall”) are rated. In this case, we still test among four frame rates but only among 3 QP levels (28, 36 and 40). This is because the results from the first session show that it is very hard for the viewers to tell the difference between QP=40 and 44. We included the two common sequences (“akiyo” and “football”) in both experiments, so that we can determine an appropriate mapping between the subjective ratings from two experiments, following the algorithm described in [23].

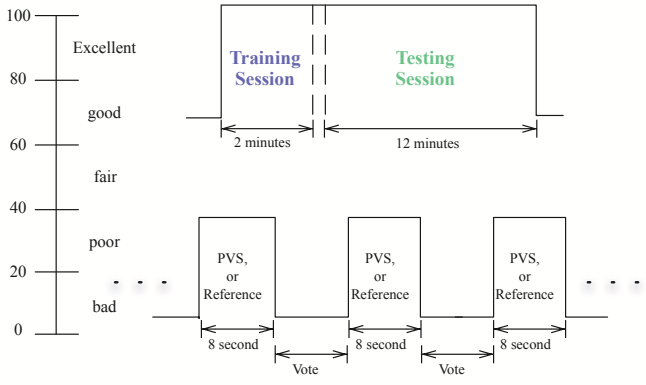


Figure 2.2: Illustration of subjective quality rating test.

2.2.2 Test Configuration

The subjective quality assessment, illustrated in Figure 2.2, is carried out by using a protocol similar to ACR (Absolute Category Rating) described in [24]. In the test, a subject is shown one PVS at a time, and is asked to provide an overall rating at the end of the clip. The rating scale ranges from 0 (worst) to 100 (best) with text annotations shown next to the rating numbers as shown in Figure 2.2. Most of the viewers for both of the subjective test are engineering students from Polytechnic Institute of New York University, with age 23 to 35. Other details regarding each experiment are given below.

1. The first experiment:

   In order to shorten the duration of the test, the experiment is divided into two subgroups. Each of them contains 38 processed video sequences and lasts about 14 minutes. Each subgroup test consists of two sessions, a training session and a test session. The training session (about 2 minutes) is used for the subject to accustom him/herself to the rating procedure and ask questions if any. The training clips including PVSs from "Soccer" and "Waterfall" are chosen to expose viewers to the types and quality range of the testing clips. The PVSs in the test session (about 12 minutes) are ordered randomly so that each subject sees the video clips in a different order. Thirty one non-expert viewers who had normal or corrected-to-normal vision acuity participated in one or two subgroup tests. There are on average 20 ratings for each PVS.

2. The second experiment:

   Each subgroup contains 24 PVSs. The training clips (6 PVSs) are picked from the entire PVS pool except the sequences included in the testing session and the selections of testing points are uniformly distributed among the entire range. The sequences in the test session are also ordered randomly. Thirty three non-expert viewers who had normal or corrected-to-normal vision acuity participated in one or two subgroup tests. There are on average 16 ratings for each PVS.

2.2.3 Data Post-Processing

Given the rating range from 0 to 100, different viewers' scores tend to fall in quite different subranges. The raw score data should be normalized before analysis. We first find the minimum and maximum scores given by each viewer for a specific source sequence, we then find the median of the minimum (resp. maximum) scores by all viewers for this source sequence. All viewers' scores for the same source video are normalized by the resulting median of minimum and median of maximum. We then average normalized viewer ratings for the same processed video sequence to determine its mean opinion score (MOS). More

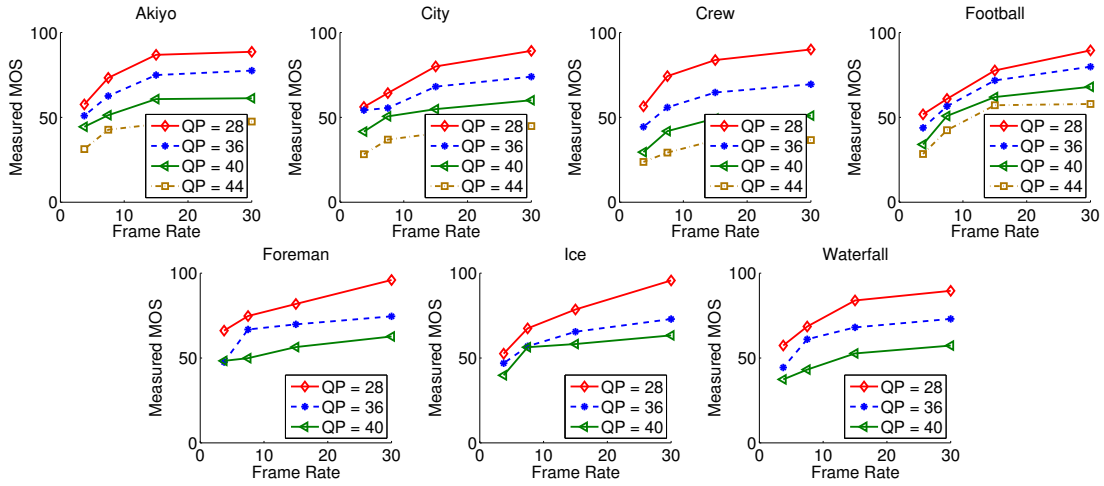


Figure 2.3: Measured MOS against frame rate at different QP. The average 95% confident interval of all the sequences is 20.89.

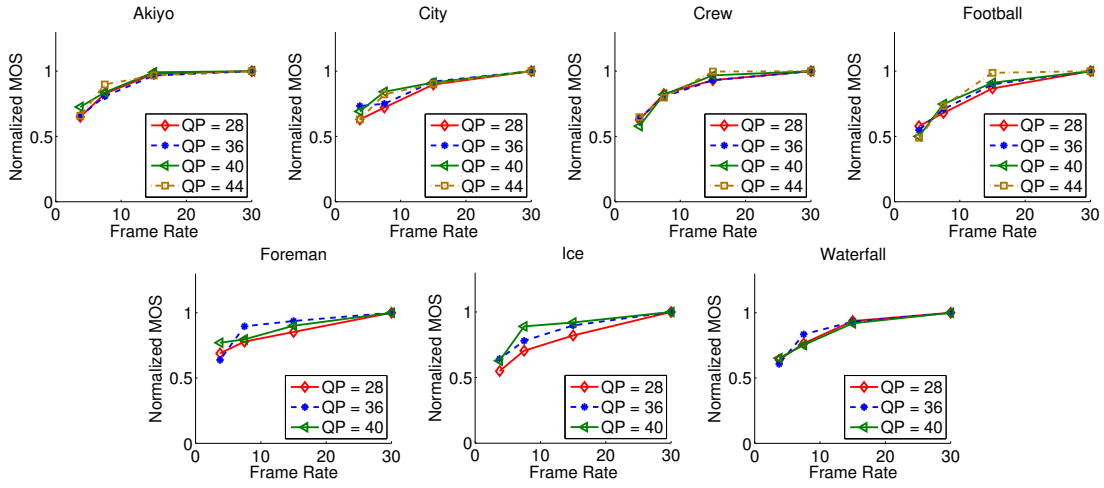


Figure 2.4: Normalized MOS against frame rate at different QP.

details regarding how to perform the data post-processing can be found in [10].

After screening there are on average 15 and 14 user ratings for each PVS in the first and second experiments, respectively. Figure 2.3 presents the subjective test results. We see that no matter what QP level is, MOS reduces consistently as the frame rate decreases. In order to examine whether the reduction trend of the MOS against the frame rate is independent of the quantization parameter, we plot in Figure 2.4, the normalized MOS, which is the ratio of the MOS with the MOS at the highest frame rate (30 Hz in our case), at the same QP. We see that these normalized curves corresponding to different QPs almost overlap with each other, indicating that the reduction of the MOS with frame rate is quite independent of the QP.

2.3 Proposed Quality Metric

Our quality model is extended from our earlier work [10]. Instead of using the PSNR in our model [10], we try to use the quantization stepsize to relate the picture spatial quality directly. As described earlier, results in Figures 2.3 and 2.4 suggest that the impact of frame rate and that of quantization is separable. We focus on examining the impact of frame rate on the quality, under the same quantization stepsize; while trying to use prior models to characterize the impact of quantization stepsize q on the quality, when the video is coded at a fixed frame rate. The proposed model is written generally as

$$Q(q,t) = Q_{\max} Q_q(q; t_{\max}) Q_t(t; q), \tag{2.1}$$

where $Q_{\max} = Q(q_{\min}, t_{\max})$,

$$Q_q(q; t_{\max}) = \frac{Q(q, t_{\max})}{Q(q_{\min}, t_{\max})}$$

is the normalized quality versus quantization stepsize (NQQ) under the maximum frame rate $t_{\max}$;

$$Q_t(t; q) = \frac{Q(q,t)}{Q(q, t_{\max})}$$

is the normalized quality vs. temporal resolution (NQT) under the same quantization stepsize q. Note that $Q_{\max}Q_q(q;t_{\max})$ models the impact of quantization on the quality when the video is coded at the highest frame rate $t_{\max}$; while $Q_t(t;q)$ describes how the quality reduces when the frame rate reduces, under the same q. In other words, $Q_t(t;q)$ corrects the predicted quality by $Q_{\max}Q_q(q;t_{\max})$ based on the actual frame rate, and for this reason is also called Temporal Correction Factor for Quality (TCFQ).

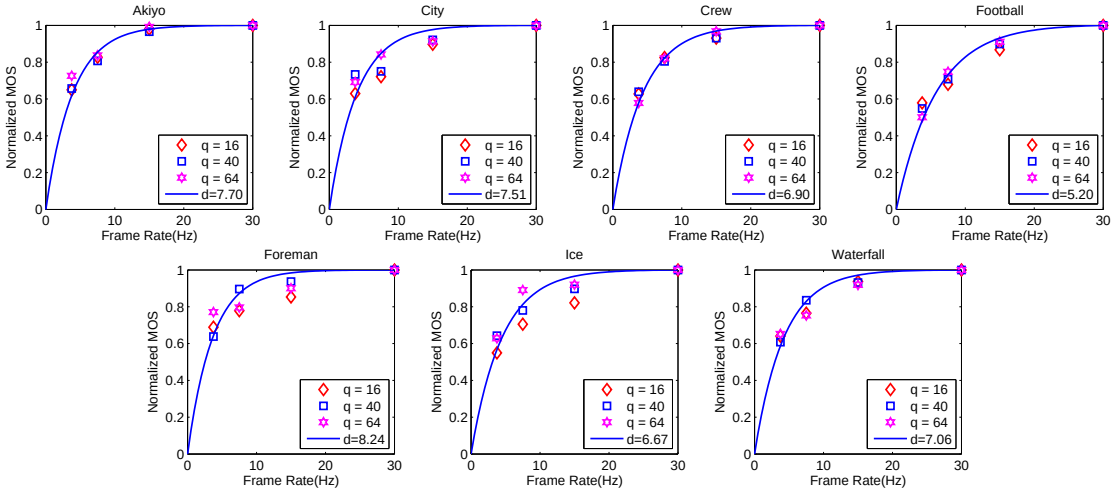


Figure 2.5: Normalized quality vs. temporal resolution (NQT), for different quantization stepsize q. Points are measured data, curves are predicted quality using Eq. (2.2).

To see how the normalized quality ratings $Q_q(q;t)$ and $Q_t(t;q)$ vary with q and t respectively, we plot the measured data from our subjective tests in Figures 2.5 and 2.6. Unlike the rate data presented in Chapter 3, where the effects of quantization stepsize q and frame rate t are quite separable, there are noticeable interactions between t and q in their impact on the perceptual quality. This interaction in fact is well known, but not well understood. However, as seen in Figure 2.5, the effect of q on the NQT curves $Q_t(t;q)$ is inconsistent and relatively small. Also these variations may be in part due to viewer inconsistency during the subject tests. To reduce the model complexity, we choose to model the $Q_t(t;q)$ curves by a function of t only, denoted by $Q_t(t)$. For the model for $Q_q(q;t_{\max})$, we use only the measured NQQ data at the frame rate $t_{\max}$.

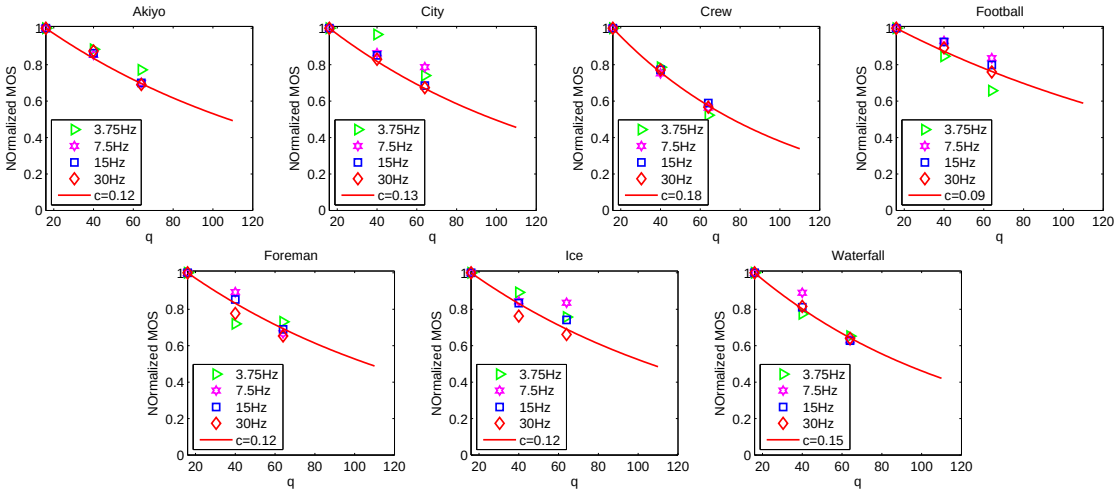


Figure 2.6: Normalized quality versus the quantization stepsize (NQQ) for different frame rates t. Points are measured data and curves are predicted quality for $t = 30$ Hz, using Eq. (2.3).

2.3.1 Model for Temporal Correction Factor For Quality (TCFQ) $Q_t(t)$

In a prior work [25], we have investigated the impact of the frame rate on the perceptual quality of uncompressed video, and found that the normalized quality can be modeled very accurately by an inverted exponential falling function. Here we adopt the same function:

$$Q_t(t) = \frac{1 - e^{-d\frac{t}{t_{\max}}}}{1 - e^{-d}}. \tag{2.2}$$

As can be seen in Figure 2.5, this function can predict the normalized MOS very well. We can see that the fitting is quite accurate for all sequences. Note that the parameter d characterizes how fast the quality drops as the frame rate reduces, with a smaller d indicating a faster drop rate. The d values for different sequences are provided in Figure 2.5. As expected, sequences with higher motion have faster drop rates (smaller d). To demonstrate the influence of the video content on the parameter, Figure 2.5 shows the TCFQ curves for different videos. We can clearly see that d is larger for slower motion sequences.

The model in (2.2) is chosen by comparing several one-parameter functions, including the exponential falling function in Eq. (2.2), the power function $(\frac{t}{t_{\max}})^d$, and the logarithmic

function $\frac{\log(1+d\frac{t}{t_{\max}})}{\log(1+d)}$. By evaluating several data sets [10], it is shown that the inverted exponential function in Eq. (2.2) is the best.

2.3.2 Model for Normalized Quality vs. Quantization $Q_q(q)$

To model the variation of the perceptual quality with quantization when the video is coded at a fixed frame rate $t_{\max}$, in our earlier work [10,26], we assume that under the same quantization parameter q, the PSNR of decoded frames at frame rate $t_{\max}$ would be similar to the PSNR of decoded frames at a reduced frame rate t. So we use PSNR computed at frame rate t to estimate the quality of the video coded at $t_{\max}$. Based on the prior work in [27], we use a sigmoidal function to relate the PSNR with the perceptual quality, with two parameters. In the current work, based on measured NQQ points $Q_q(q; t_{\max})$ shown in Figure 2.6, we propose to use an exponential function to capture the quality variation with q at the highest frame rate $t_{\max}$, i.e.,

$$Q_q(q) = e^c e^{-c\frac{q}{q_{\min}}}, \tag{2.3}$$

with c as the model parameter. Compared with the original two parameter sigmoid function proposed in [10, 26], the single parameter exponential function is simpler and easier to analyze. Comparing the measured and predicted quality ratings shown in Figure 2.6, we see that the model captures the quantization-induced quality variation very well at the highest frame rate.

2.3.3 The Overall Quality Metric

Combining Eqs. (2.1), (2.2) and (2.3), the overall video quality model can be expressed as

$$Q(q,t) = Q_{\max}\frac{e^{-c\frac{q}{q_{\min}}}}{e^{-c}}\frac{1-e^{-d\frac{t}{t_{\max}}}}{1-e^{-d}}. \tag{2.4}$$

Note that $Q_{\max}$ is the MOS given for the video at $q_{\min}$ and $t_{\max}$. Generally, this value can be estimated by some preliminary subjective tests. In our subjective tests, the ratings

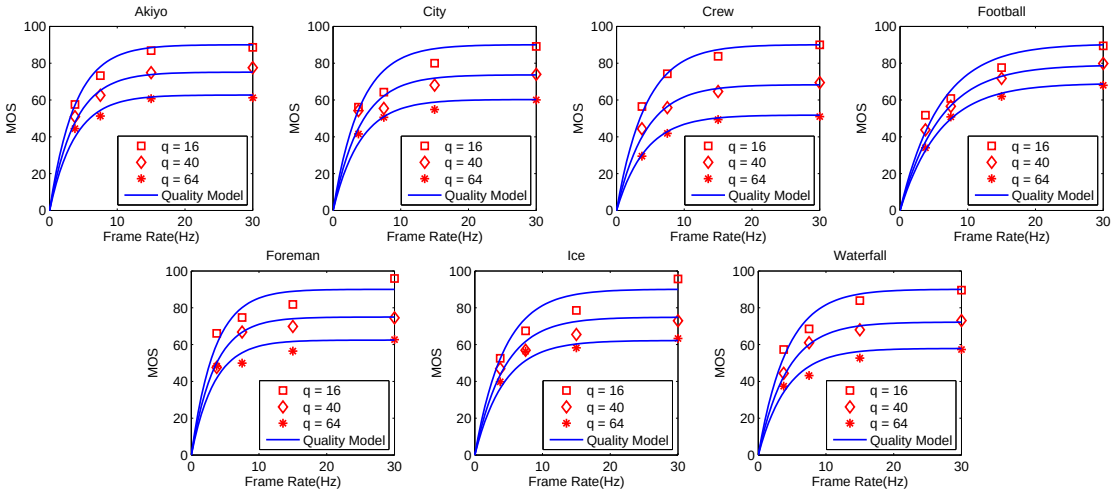


Figure 2.7: Quality vs. quantization stepsize and frame rate. Points are measured MOS data; curves are predicted quality using Eq. (2.4)

are given in the range of 0 to 100. But the viewers seldom give a rating of 100, even for very high quality video, as is commonly observed in subjective tests. What is surprising and fortunate is that the MOS values for the videos coded at $q_{\min}$ and $t_{\max}$ are very close to each other for all seven test sequences, about 90. Therefore, we set $Q_{\max}$ to 90 in our model. Note that on the more common MOS scale of 1 to 5, 90 out of 0 to 100 would correspond to a MOS of $0.9 \times 4 + 1 = 4.6$.

Table 2.1: Parameters for the quality model and model accuracy

	akiyo	city	crew	football	foreman	ice	waterfall	ave.
c	0.12	0.13	0.18	0.09	0.12	0.12	0.15	
d	7.70	7.51	6.90	5.20	8.24	6.67	7.06	
$\frac{\text{RMSE}}{Q_{\max}}$	3.06%	6.41%	2.50%	4.54%	5.49%	5.38%	3.65%	4.40%
PC	0.99	0.95	0.99	0.98	0.94	0.96	0.98	0.97

Figure 2.7 compares the measured and predicted quality ratings by the model in (2.4). The parameters c, d are obtained by least square error fitting. Table 2.1 summarizes the parameters and the model accuracy in terms of RMSE and Pearson correlation (PC) values for the seven sequences. Overall, the proposed model, with only two content-dependent parameters, predicts the MOS very well, for sequences "akiyo" and "crew", with a very

high Pearson correlation (> 0.99), defined as

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}, \tag{2.5}$$

where x_i and y_i are the measured and predicted rates, and n is the total number of available samples. The model is less accurate for "foreman" and "city", but still has a quite high PC. We would like to point out that the measured MOS data for these two sequences do not follow a consistent trend at some quantization levels, which may be due to the limited number of viewers participating the subjective tests.

2.4 Discussion and Summary

As shown in previous plots, the model parameters c and d are video content dependent. The model will be more useful if the model parameters can be predicted from some content features derived from the original or compressed video signals. The details regarding the quality model parameters prediction will be discussed in Chapter 5 together with the parameters for proposed rate and complexity models. In addition to verify the accuracy of our proposed quality metric using these seven sequences, we also apply our model to other data sets proposed in the literature. All comparison results show that our model outperforms other related works with high PC. More details can be found in our published work [10].

In summary, this chapter presents our perceptual quality metric considering both frame rate and quantization artifacts. Based on the experimental data, we have found that the impact of the frame rate and quantization is separable, therefore, we propose two single-parameter exponential functions to capture the quality variation in terms of the variation of frame rate and quantization stepsize respectively. Results show that our proposed model can estimate the subjective MOS very well.

Chapter 3

Rate Model of Scalable Video

This chapter presents the rate modeling for scalable video with focus on the joint temporal and amplitude scalability. Like what we have done for the perceptual quality model, we separate the impact of the quantization and frame rate, and propose two single-parameter power functions to model the rate in terms of the quantization and frame rate respectively. As shown, our proposed model can predict the actual video rate very accurate (with average PC > 0.99, average RMSE $< 1.5\%$) for videos with different content activities, resolutions, and etc. We have also found that the rate model parameters are highly content dependent. The model will be more useful if the parameters can be predicted accurately and easily. More details on rate model parameter prediction using content features can be found in Chapter 5.

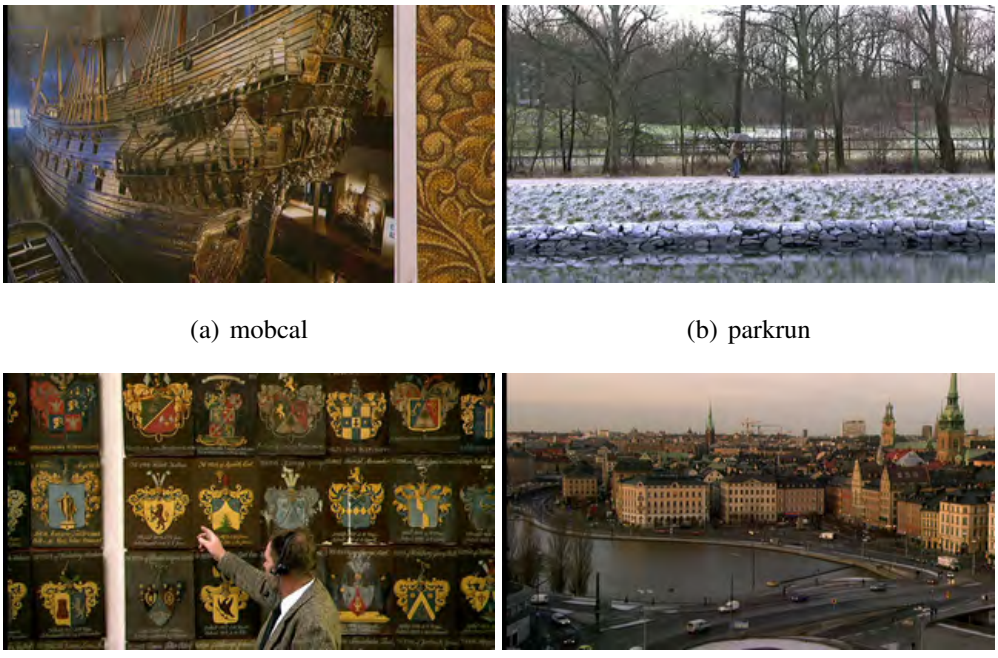
3.1 Introduction

Our proposed rate model, i.e., $R(q,t)$, which relates the video bit rate R with the quantization stepsize q and frame rate t. To the best of our knowledge, no prior work has considered joint impact of frame rate and quantization on the bit rate. However, several prior works have considered rate modeling in non-scalable video, and have proposed models that relate the average bit rate versus quantization stepsize q. Ding and Liu reported the

(a) BasketballDrill (b) BQMall

(c) PartyScene (d) RaceHorses

Figure 3.1: WVGA (832x480) resolution test video sequences.



(a) mobcal (b) parkrun

(c) shields (d) stockholm

Figure 3.2: High definition 720p (1280x720) resolution test video sequences.

following model [28],

$$R = \frac{\theta}{q^{\gamma}}, \tag{3.1}$$

where θ and γ are model parameters, with $0 \leq \gamma \leq 2$. Chiang and Zhang [29] suggested the following model

$$R = \frac{A_1}{q} + \frac{A_2}{q^2}, \tag{3.2}$$

This so-called quadratic rate model has been used for rate-control in MPEG-4 reference encoder [30]. We note that by choosing A_1 and A_2 appropriately, the model in (3.2) can realize the inverse power model of (3.1) with any $\gamma \in (1, 2)$. Only the quadratic term was included in the model by Ribas-Cobera and Lei [31], i.e.,

$$R = \frac{A}{q^2}. \tag{3.3}$$

More recently, He [32] proposed the ρ-model,

$$R(\mathsf{QP}) = \theta\left(1 - \rho(\mathsf{QP})\right), \tag{3.4}$$

with ρ denoting the percentage of zero quantized transform coefficients with a given quantization parameter. This model has been shown to have high accuracy for rate prediction. A problem with the ρ-model is that it does not provide explicit relation between QP and ρ. Therefore, it does not lend itself to theoretical understanding of the impact of QP on the rate.

3.2 Proposed Rate Model

In our work on rate modeling, we focus on the impact of frame rate t on the bit rate R, under the same quantization stepsize q; while using prior models to characterize the impact of q on the rate, when the video is coded at a fixed frame rate. Towards this goal, we recognize that $R(q, t)$ can be written as

$$R(q, t) = R_{\max} R_q(q; t_{\max}) R_t(t; q), \tag{3.5}$$

where $R_{\max} = R(q_{\min}, t_{\max})$ is the maximum bit rate obtained with a chosen minimal quantization stepsize $q_{\min}$ and a chosen maximum frame rate $t_{\max}$;

$$R_q(q; t_{\max}) = \frac{R(q, t_{\max})}{R(q_{\min}, t_{\max})}$$

is the normalized rate vs. quantization stepsize (NRQ) under the maximum frame rate $t_{\max}$, and

$$R_t(t; q) = \frac{R(q, t)}{R(q, t_{\max})}$$

is the normalized rate vs. temporal resolution (NRT) under the same quantization stepsize q. Note that the NRQ function $R_q(q; t_{\max})$ describes how does the rate decreases when the quantization stepsize q increases beyond $q_{\min}$, under the frame rate $t_{\max}$; while the NRT function $R_t(t; q)$ characterizes how does the rate reduces when the frame rate decreases from $t_{\max}$, under the same quantization stepsize q. We also call $R_t(t; q)$ the temporal correction factor for rate (TCFR), as it describes how to correct the rate estimate by $R_{\max} R_q(q; t_{\max})$ based on the actual temporal resolution. As will be shown later by experimental data, the impact of q and t on the bit rate is actually separable, so that $R_t(t; q)$ can be represented by a function of t only, denoted by $R_t(t)$, and $R_q(q; t)$ by a function of q only, denoted by $R_q(q)$.

To see how quantization and frame rate respectively influence the bit rate, we encoded several test videos using the SVC reference software JSVM [22] and measured the actual bit rates corresponding to different q and t. Specifically, eight CIF (i.e., 352x288) resolution video sequences, “akiyo”, “city”, “crew”, “football”, “foreman”, “harbour”, “ice”, “waterfall”, four WVGA (i.e., 832x480) videos, “BasketballDrill”, “BQMall”, “PartyScene”, “RaceHorses” and four high definition (HD) 720p (i.e., 1280x720) sequences, “mobcal”, “parkrun”, “shields”, “stockholm” are encoded into 5 temporal layers using dyadic hierarchical prediction structure, with frame rates 1.875, 3.75, 7.5, 15, and 30 Hz, respectively, and each temporal layer contains 5 amplitude layers obtained with quantization parameter (QP) of 44, 40, 36, 32, 28.[1] Those test videos are shown in Figures 2.1, 3.1, 3.2, respectively. We believe that our rate model can be applied widely by evaluating this large test

[1] Different from the JSVM default configuration utilizing different QPs for different temporal layers, the same QP is applied to all temporal layers at each amplitude layer.

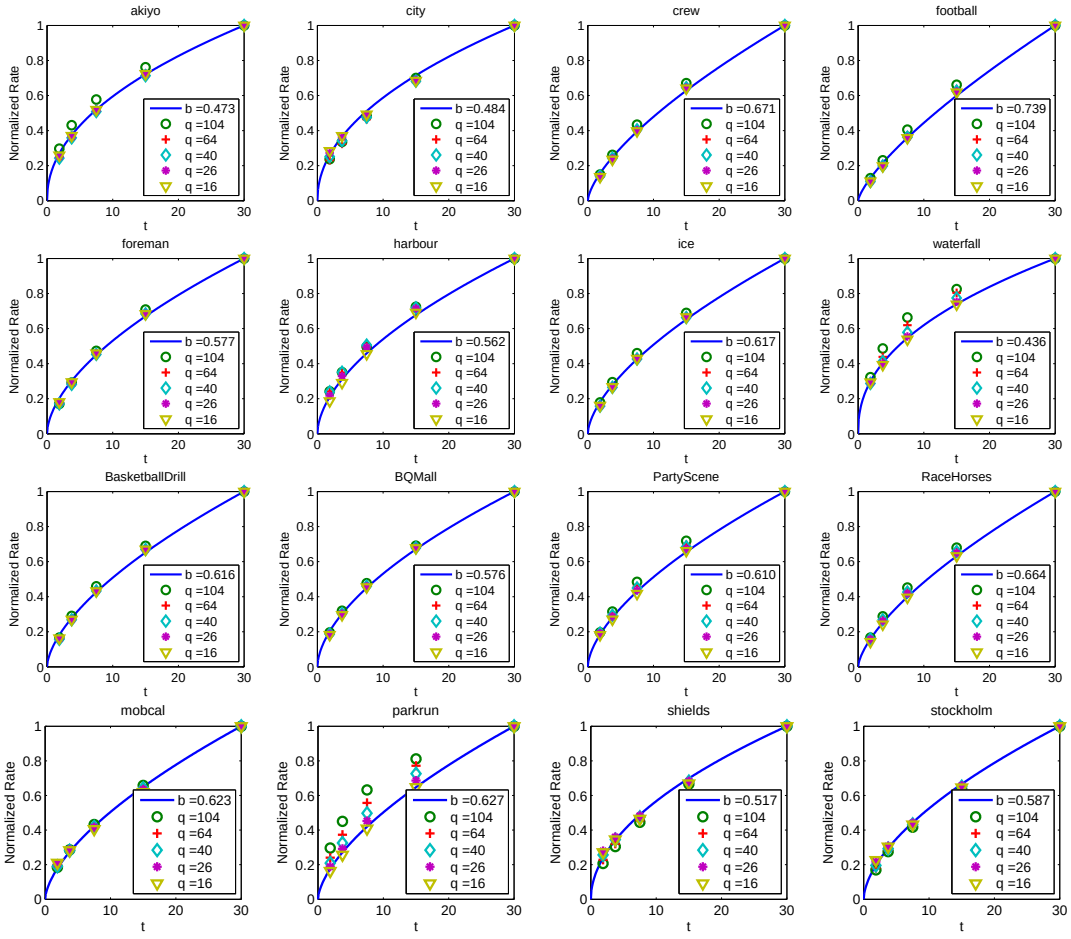


Figure 3.3: Normalized rate vs. temporal resolution (NRT) using different quantization stepsize (q). Points are measured rates, curves are predicted rates by the model of Eq. (3.6).

video pool with different content feature, resolution, etc. Using the H.264/AVC mapping between q and QP, $q = 2^{(\mathsf{QP}-4)/6}$, the corresponding quantization stepsizes are 104, 64, 40, 26, 16, respectively.

The bit rates of all layers are collected and normalized by the rate at the highest frame rate, i.e., $t_{\max} = 30$ Hz, to find NRT points $R_t(t;q) = R(q,t)/R(q,t_{\max})$, for all t and q considered, which are plotted in Figure 3.3. As shown in Figure 3.3, the NRT curves obtained with different quantization stepsizes overlap with each other, and can be captured by a single curve quite well. Similarly, the NRQ curves $R_q(q;t) = R(q,t)/R(q_{\min},t)$ for different frame rates t are also almost invariant with the frame rate t, as shown in Figure 3.8. These observations suggest that the effects of q and t on the bit rate are separable, i.e., the quantization-induced rate variation is independent of the frame rate and vice verse. Therefore, the overall rate modeling problem is divided into two parts, one is to devise an appropriate functional form for $R_t(t)$, so that it can model the measured NRT points for all q in Figure 3.3 accurately, the other is to derive an appropriate functional form for $R_q(q)$ that can accurately model the measured NRQ points in Figure 3.8 for $t = t_{\max}$. Note that in fact, the $R_q(q)$ model fits the NRQ points obtained at all different t. Generally for given $q_{\min}$ and $t_{\max}$, $R_{\max}$ depends on the video content. $R_{\max}$ prediction using the content features will be explored in the following Chapter 5. The derivation of the models $R_q(q)$ and $R_t(t)$ are explained in details as follows.

3.2.1 Model for Temporal Correction Factor for Rate (TCFR) $R_t(t)$

As explained earlier, $R_t(t)$ is used to describe the reduction of the normalized bit rate as the frame rate reduces. Therefore, the desired property for the $R_t(t)$ function is that it should be 1 at $t = t_{\max}$ and monotonically reduces to 0 at $t = 0$. Based on the measurement data in Figure 3.3, we choose a power function, i.e.,

$$R_t(t) = \left(\frac{t}{t_{\max}}\right)^b. \tag{3.6}$$

Figure 3.3 shows the model curve using this function along with the measured data. The parameter b is obtained by minimizing the squared error between the modeled rates

and measured rates. It can be seen that the model fits the measured data points very well. We also provide the theoretical analysis to prove the power function expression is the right choice to relate the bit rate in terms of the temporal resolution. More details regarding the proof of the power function expression for $R_t(t)$ will be unfolded as follows.

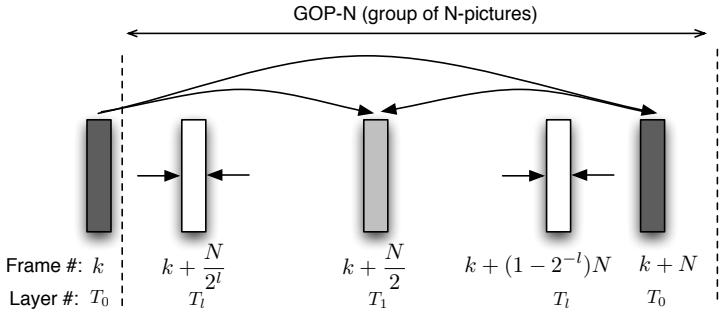


Figure 3.4: Illustration of the dyadic hierarchical prediction structure used to provide the temporal scalability in SVC.

Theoretical analysis of $R_t(t)$ using power function approximation

As shown in Figure 3.4, a N-picture GOP is exemplified. Because of the dyadic hierarchical prediction structure applied, temporal resolution or frame rate will be doubled when another temporal refinement is added. Let l be the temporal layer number, and assume the same quantizer used at different temporal layers, therefore, according to the rate-distortion theory [33, 34], the distortion at l-th temporal layer can be expressed as

$$\sigma_{D,l}^2 = \sigma_{p,l}^2 2^{-2\lambda \overline{R_l}} \tag{3.7}$$

where $\overline{R_l}$ is the bit rate per pixel for image at temporal level l, and $\sigma_{p,l}^2$ is the variance of temporal predictive error signal. Let $\sigma_{D,l}^2 = \sigma_{D,0}^2$ because of the same quantizer applied at different temporal layers, therefore, bits per pixel at l-the layer can be deducted as follows:

$$\sigma_{p,l}^2/\sigma_{p,0}^2 = 2^{-2\lambda(\overline{R_0} - \overline{R_l})}, \tag{3.8}$$

$$\log_2\left(\sigma_{p,l}^2/\sigma_{p,0}^2\right) = -2\lambda\left(\overline{R_0} - \overline{R_l}\right), \tag{3.9}$$

$$\overline{R_l} = \overline{R_0} + \frac{1}{2\lambda}\log_2\left(\sigma_{p,l}^2/\sigma_{p,0}^2\right). \tag{3.10}$$

For simplicity, we use the σ_l^2 to replace $\sigma_{p,l}^2$ to indicate the predictive error signal variance. Given the l-th temporal layer, the bits per pixel $\overline{R_l}$ is

$$\overline{R_l} = \overline{R_0} + \frac{1}{2\lambda}\log_2\left(\sigma_l^2/\sigma_0^2\right) = \overline{R_0} + G(\sigma_l;\lambda), \tag{3.11}$$

where we assume $G(\sigma_l;\lambda) = \frac{1}{2\lambda}\log_2\left(\sigma_l^2/\sigma_0^2\right)$ as the bit rate refinement due to the temporal predictive coding.

As argued above, we have derived the $\overline{R_l}$ (bits per pixel) at l-th temporal layer as a function of $\overline{R_0}$ (bits per pixel) at base layer (i.e., $l = 0$) and predictive error signal refinements (i.e., σ_l^2/σ_0^2). However, in practice, "bits per second [bps]" is more widely used to represent the bit rate. Let R_l denote the bit rate (bits per second) up to l-th layer, given the picture resolution at M and lowest frame rate at t_0, we can obtain the bit rate (bits per second) for successive temporal layers (up to):

$$\text{TL\# 0:} \qquad R_0 = \overline{R_0}Mt_0, \tag{3.12}$$

$$\begin{aligned}\text{TL\# 1:} \qquad R_1 &= \overline{R_1}M(t_1 - t_0) + R_0 \\ &= \overline{R_0}Mt_1 + G(\sigma_1;\lambda)M(t_1 - t_0) \\ &= Mt_0\left(2\overline{R_0} + G(\sigma_1;\lambda)\right),\end{aligned} \tag{3.13}$$

$$\begin{aligned}\text{TL\# 2:} \qquad R_2 &= \overline{R_2}M(t_2 - t_1) + R_1 \\ &= \overline{R_0}Mt_2 + M(t_2 - t_1)G(\sigma_2;\lambda) + M(t_1 - t_0)G(\sigma_1;l) \\ &= Mt_0\left(4\overline{R_0} + 2G(\sigma_2;\lambda) + G(\sigma_1;\lambda)\right),\end{aligned} \tag{3.14}$$

$$\begin{aligned}\text{TL\# } l\text{:} \qquad R_l &= \overline{R_0}Mt_l + M\sum_{i=1}^{l}(t_i - t_{i-1})G(\sigma_i;\lambda) \\ &= Mt_0\left(2^l\overline{R_0} + \sum_{i=1}^{l}2^{i-1}G(\sigma_i;\lambda)\right).\end{aligned} \tag{3.15}$$

Because of the dyadic hierarchical prediction structure applied, we can know that $t_l = 2^{l-1}t_0$ with $l \geq 1$. To derive the $\overline{R_0}$, we can let $\sigma_{D,0}^2 = \sigma_q^2$, where σ_q^2 represents the distortion due to the quantization in video coding, therefore,

$$\sigma_q^2 = \epsilon^2\sigma_0^2 2^{-2\lambda\overline{R_0}} \tag{3.16}$$

$$\overline{R_0} = \frac{1}{2\lambda}\log_2\left(\epsilon^2\sigma_0^2/\sigma_q^2\right) \tag{3.17}$$

where ϵ and λ are the signal parameters. By combing Eq. (3.17) and Eq. (3.15), we can see that the bit rate at l-layer, i.e., R_l, is a function of σ_l^2 and related signal parameters, such as λ and ϵ, etc. Thus, we have recognized that we have to derive the σ_l^2 explicitly so as to express the R_l quantitatively.

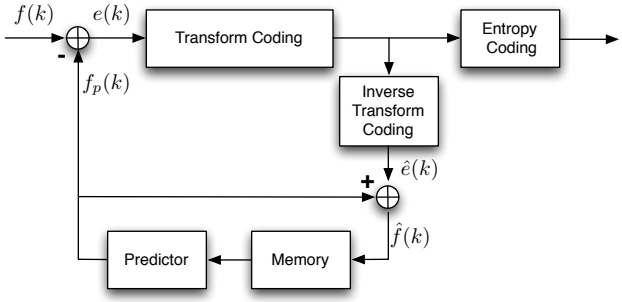


Figure 3.5: Illustration of the hybrid predictive and transform coding structure, $f(k)$ is noted as the original k-th frame video signal, $f_p(k)$ means the prediction signal of $f(k)$ and $\hat{f}(k)$ is the reconstructed k-th frame which will be buffered into memory as reference for future frame coding.

A hybrid predictive and transform coding structure is illustrated in Figure 3.5. Usually, the predictive coding can be either spatial intra prediction or temporal inter prediction. Our focus here is the temporal inter prediction which in practice employes the motion estimation to find the best match from reference pictures. As shown in Figure 3.4, $(k+N)$-th frame uses the backward k-th frame as its reference, and $(k+N/2)$-th frame uses the backward k-th frame and forward $(k+N)$ frame to construct the predictor bidirectionally, i.e.,

$$f_p(k+N) = \hat{f}(k) \tag{3.18}$$

$$f_p(k+N/2) = \alpha\hat{f}(k) + (1-\alpha)\hat{f}(k+N) \tag{3.19}$$

where α is the weighting factor to show the percentage of the signal from backward reference. Further prediction signal can be derived similarly. With the focus on the temporal predictive signal, we simply use the original signal to replace the reconstructed signal as reference, i.e., $\hat{f}(k) \approx f(k)$, and $\hat{f}(k+N) \approx f(k+N)$, etc. Therefore, we can obtain the

variance of the prediction error signal at each layer step by step as follows:

$$\sigma_0^2 = \mathrm{E}\left[(f(k+N) - f(k))^2\right] = 2\sigma^2\left(1-\rho^N\right), \tag{3.20}$$

$$\begin{aligned}\sigma_1^2 &= \mathrm{E}\left[(f(k+N/2) - \alpha f(k) - (1-\alpha)f(k+N))^2\right] \\ &= 2\sigma^2\left((\alpha^2+(1-\alpha)^2)(1-\rho^{N/2}) + \alpha(1-\alpha)(1-\rho^{N/2})^2\right),\end{aligned} \tag{3.21}$$

$$\sigma_l^2 = 2\sigma^2\left((\alpha^2+(1-\alpha)^2)(1-\rho^{N/2^l}) + \alpha(1-\alpha)(1-\rho^{N/2^l})^2\right). \tag{3.22}$$

where $\rho = \mathrm{E}\left[f(k+1)f(k)\right]$ as the correlation between two successive frames [35]. By assuming the $\alpha = 0.5$, we can further reach at

$$\sigma_l^2 = 0.5\sigma^2\left(3 - 4\rho^{N/2^l} + (\rho^{N/2^l})^2\right). \tag{3.23}$$

By combining Eq. (3.17), Eq. (3.15) and Eq. (3.23),

$$\begin{aligned}R_l &= Mt_0\left(2^l\overline{R_0} + \frac{1}{2\lambda}\sum_{i=1}^{l} 2^{i-1}\log_2(\sigma_i^2/\sigma_0^2)\right) \\ &= Mt_0\left(\overline{R_0} + \sum_{i=1}^{l} 2^{i-1}\log_2\left(\epsilon^2\sigma_0^2/\sigma_q^2\right) + \frac{1}{2\lambda}\sum_{i=1}^{l} 2^{i-1}\log_2(\sigma_i^2/\sigma_0^2)\right) \\ &= Mt_0\left(\overline{R_0} + \frac{1}{2\lambda}\sum_{i=1}^{l} 2^{i-1}\log_2\left(\frac{\epsilon^2\sigma_0^2}{\sigma_q^2}\frac{\sigma_i^2}{\sigma_0^2}\right)\right) \\ &= Mt_0\left(\overline{R_0} + \frac{1}{2\lambda}\sum_{i=1}^{l} 2^{i-1}\log_2\left(\frac{\epsilon^2\sigma_i^2}{\sigma_q^2}\right)\right) \\ &= \frac{Mt_0}{2\lambda}\left(\log_2\left(\frac{2\epsilon^2\sigma^2(1-\rho^N)}{\sigma_q^2}\right)\right. \\ &\qquad \left. + \sum_{i=1}^{l} 2^{i-1}\log_2\left(\frac{\epsilon^2\sigma^2(3-4\rho^{N/2^i}+(\rho^{N/2^i})^2)}{2\sigma_q^2}\right)\right)\end{aligned} \tag{3.24}$$

As known for the temporal scalable video, each layer l corresponds to an individual frame rate t. For the dyadic prediction structure, frame rate t at l-th layer can be expressed as

$$t = \frac{2^l}{N}t_{\max}, \tag{3.25}$$

where N is the GOP length and $t_{\max}$ is the maximum video frame rate (i.e., $t_{\max} = 30$ Hz in our work). Therefore, bit rate at l-th layer, i.e., R_l is equivalent to the bit rate at frame rate t, $R_t(t)$, i.e.,

$$R_t(t) = \frac{Mt_0}{2\lambda}\left(\overline{R_0} + \sum_{i=1}^{\log_2\left(\frac{t}{t_{\max}}N\right)} \frac{t_i N}{2t_{\max}} \log_2\left(\frac{\epsilon^2\sigma^2\left(3 - 4\rho^{\frac{t_{\max}}{t_i}} + (\rho^{\frac{t_{\max}}{t_i}})^2\right)}{2\sigma_q^2}\right)\right), \tag{3.26}$$

where $t_i = (2^i/N)t_{\max}$ is the actual video frame rate at i-th temporal layer. Assuming $N = 16$, $\lambda = 1$, $\epsilon = 1$, $\sigma_q = 1$, $\sigma = 10$ and $\rho = 0.95$, we can plot the normalized bit rate for each layer and approximate it using a power function accurately which is shown in Figure 3.6.

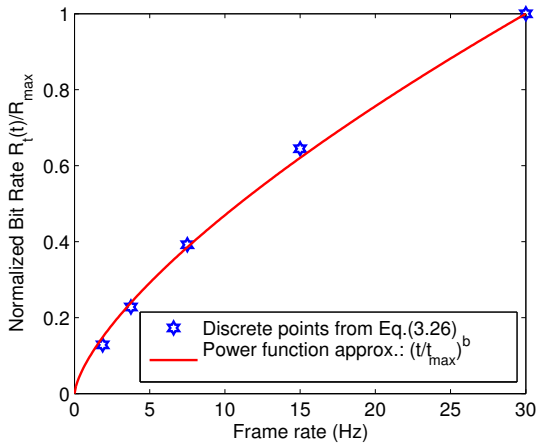


Figure 3.6: Normalized rate vs. frame rate t (NRT) and its power function approximation, where $\frac{R_t(t)}{R_{\max}} = \left(\frac{t}{t_{\max}}\right)^b$, t is the frame rate corresponding to each layer level l, $b = 0.6881$ derived by least-square-error fitting.

We also conduct the practical video coding to obtain the different rate points at different temporal layers. For different video sequences, we select the different ρ and λ (we simply let $\alpha = 0.5$) to derive the analytical rate points given the base layer bit rate (R_0). Here, we set the R_0 using the experimental rate point at base temporal level. To avoid the impact introduced by fractional interpolation, we set the integer motion compensation to encode the videos. According to our experiments, we do notice that the fractional motion compensation affects the value of ρ, λ and α, but rate points are all well matched by

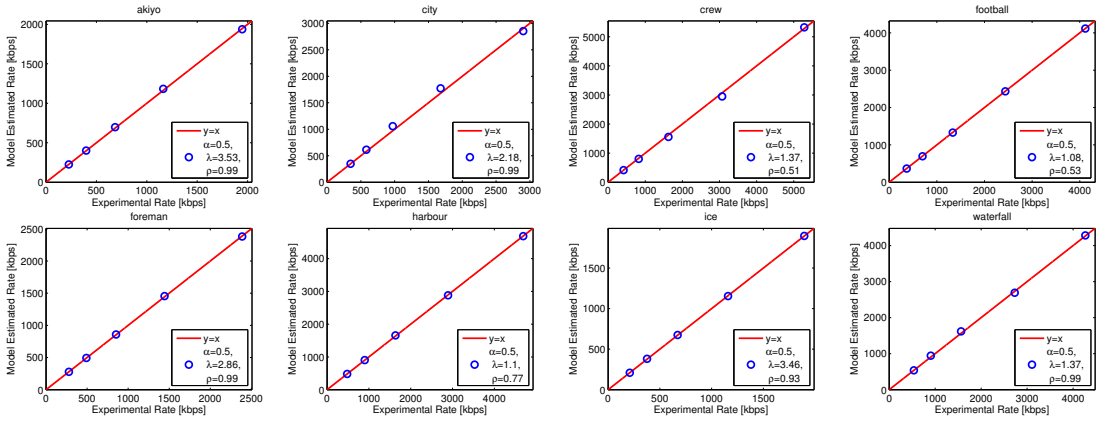


Figure 3.7: Comparison between experiment rates and analytical rates prediction.

power function approximation. As shown in Figure 3.7, by choosing proper parameters for different video content, the actual experimental rate points extracted at different temporal level can be well predicted using the analytical rate model (3.26). Therefore, video bit rate at different temporal resolution can be well captured by a single parameter power function (3.6).

3.2.2 Model for Normalized Rate vs. Quantization $R_q(q)$

Analogous to the $R_t(t)$ function, $R_q(q)$ is used to describe the reduction of the normalized bit rate as the quantization stepsize increases at a fixed frame rate. The desired property for the $R_q(q)$ function is that it should be 1 at $q = q_{\min}$ and monotonically reduces to 0 as q goes to infinity. Based on the measurement data in Figure 3.8, we choose an inverse power function, i.e.,

$$R_q(q) = \left(\frac{q}{q_{\min}}\right)^{-a}. \tag{3.27}$$

Figure 3.8 shows the model curve using this function along with the measured data. It can be seen that the model fits the measured data points very well. The parameter a characterizes how fast the bit rate reduces when q increases. Interestingly all four test sequences have very similar a values. We also tried some other functional forms, including falling exponential. We found that the inverse power function yields the least fitting error. It is

noted that the model in (3.27) is consistent with the model proposed by Ding and Liu [28], i.e., Eq. (3.1), for non-scalable video, where they have found that the parameter a is in the range of 0-2.

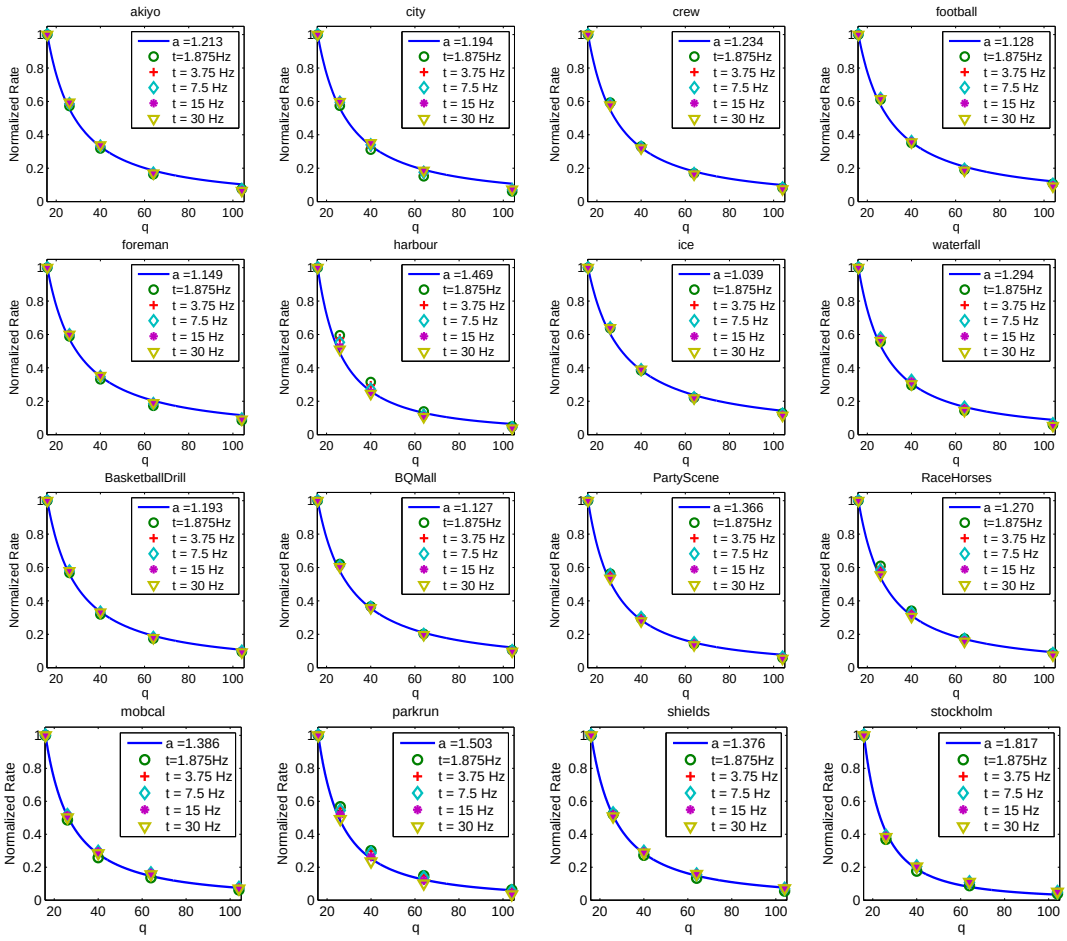


Figure 3.8: Normalized rate vs. quantization stepsize (NRQ) using different frame rates t. Points are measured rates, curves are predicted rates by the model of Eq. (3.27).

3.2.3 The Overall Rate Model

Combining Eqs. (3.5), (3.6), and (3.27), we propose the following rate model

$$R(q,t) = R_{\max}\left(\frac{q}{q_{\min}}\right)^{-a}\left(\frac{t}{t_{\max}}\right)^{b}, \tag{3.28}$$

where $q_{\min}$ and $t_{\max}$ should be chosen based on the underlying application, and $R_{\max}$ is the actual rate when coding a video at $q_{\min}$ and $t_{\max}$. a and b are the model parameters. The actual rate data of all test sequences with different combinations of q and t, and the corresponding estimated rates via the proposed model (3.28) are illustrated in Figure 3.9, we note that the model predictions fit very well with the experimental rate points. The model parameters, a and b, are obtained by minimizing the root mean squared errors (RMSE) between the measured and predicted rates corresponding to all q and t. Table 3.1 lists the parameter values. Also listed are the fitting error in terms of relative RMSE/$R_{\max}$, and the Pearson correlation (PC) bewteen measured and predicted rates. We see that the model is very accurate for all sequences, with very small relative RMSE and very high PC.

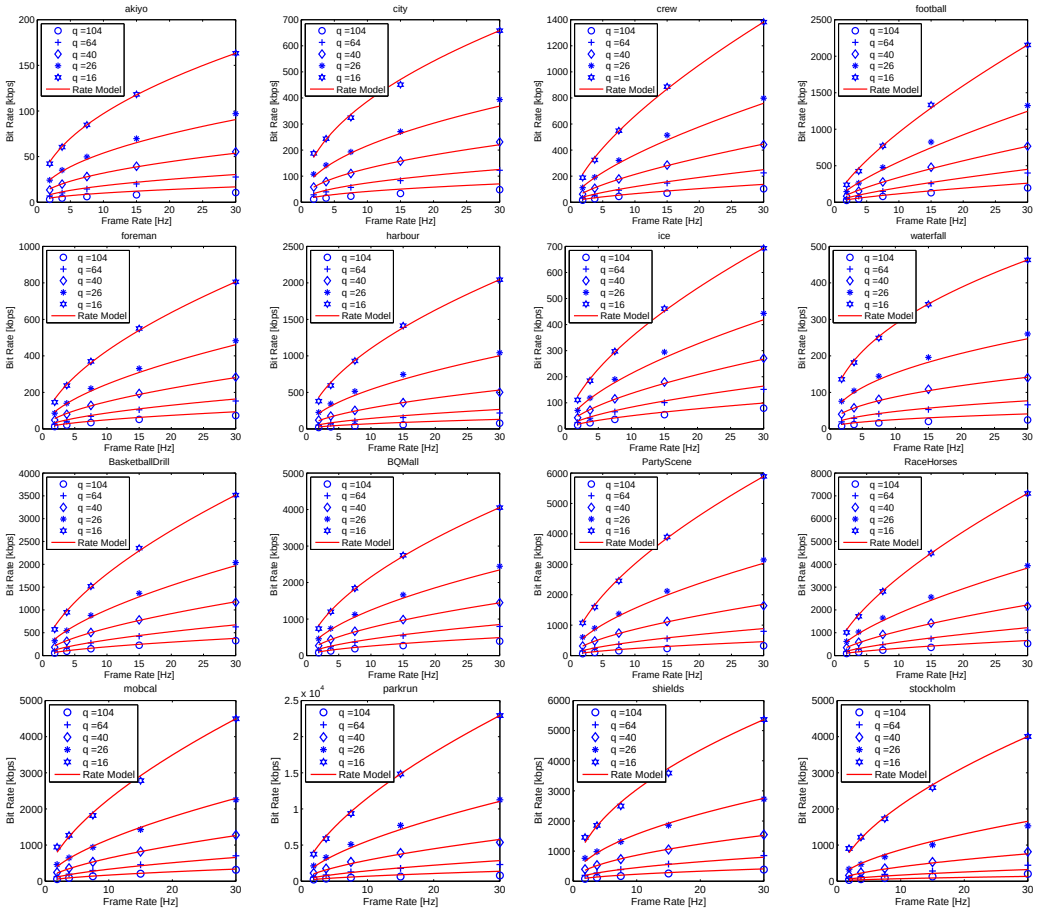


Figure 3.9: Experimental rate points and predicted rates using the rate model (3.28).

Table 3.1: Parameters for the rate model and model accuracy

seq.	a	b	$R_{\max}$ [kbps]	RMSE/$R_{\max}$	PC
akiyo	1.213	0.473	163	1.547%	0.9984
city	1.194	0.484	658	1.6738%	0.9977
crew	1.234	0.671	1382	1.2453%	0.9989
football	1.128	0.739	2154	1.5371%	0.9983
foreman	1.149	0.577	805	1.3055%	0.999
harbour	1.469	0.562	2044	1.5327%	0.9981
ice	1.039	0.617	693	1.4428%	0.9986
waterfall	1.294	0.436	463	1.4677%	0.9984
BasketballDrill	1.193	0.616	3517	0.9904%	0.9993
BQMall	1.127	0.576	4054	1.1346%	0.9991
PartyScene	1.366	0.61	5889	0.995%	0.9992
RaceHorses	1.27	0.664	7104	0.9247%	0.9992
mobcal	1.386	0.623	4495	1.1008%	0.999
parkrun	1.503	0.627	22920	1.2282%	0.9986
shields	1.376	0.517	5368	1.1799%	0.9988
stockholm	1.817	0.587	4006	1.4762%	0.9984
ave.				1.30%	0.9987

Note that parameter a characterizes how fast the bit rate reduces when q increases. A larger a indicates a faster drop rate. Parameter b indicates how fast the rate drops when the frame rate decreases, with a larger b indicating a faster drop. As expected, the "Football" sequence, which has higher motion, has a larger b and "Akiyo", has the least b.

In scalable video adaptation where a full-resolution scalable stream is already generated, one can easily derive the model parameters from the rates corresponding to several different (t, q) combinations using least squares fitting. In applications requiring estimation of model parameters from the original video sequence (e.g. for encoder optimization), it will be important to characterize the relation between $a, b, R_{\max}$ and some content features. Study of the relation between the model parameters and video content will be discussed in Chapter 5.

3.3 Discussion and Summary

In this chapter, we propose to separate the impact of frame rate t and quantization q on the bit rate, and apply two single-parameter power functions to relate the bit rate in terms of frame rate and quantization stepsize respectively. We have found that using power function for bit rate versus quantization stepsize is consistent with other works in the literature. We also provide the theoretical analysis to show the power function is the right choice to approximate the bit rate in terms of the frame rate. Additionally, we have conducted simulations using a large range of test videos with different content activities (such as motion, texture, etc) and resolutions. From the presented results, we see that our proposed model can predict the actual encoding bit rates very well. Overall, the average PC is above 0.99, and the average RMSE is less than 1.5% for all test sequences. As shown, the parameters in rate model, i.e., a, b and $R_{\max}$ are highly content dependent, which will be elaborated in Chapter 5.

Chapter 4

Power Consumption Modeling and Prediction

This chapter presents the power consumption modeling work for scalable video decoding with the focus on joint temporal and amplitude scalability. Specifically, we first develop the complexity model in terms of the amplitude and temporal resolutions, i.e., quantization stepsize and frame rate. Then, the power consumption model is devised by mapping the instant complexity requirements (i.e., cycles per second) to the processor power consumption (i..e, watt). Like what we have done for the perceptual quality and rate models, we also apply the same "impact separation" methodology to model the normalized complexity in terms of frame rate and quantization stepsize respectively. In addition to the scalable video decoding complexity modeling, we also analyze the complexity prediction from frame to frame or GOP to GOP so as to guide the dynamic voltage and frequency scaling (DVFS) of the underlying processor. To accurately capture the frame decoding complexity, we decompose the video decoder into 6 decoding modules (DM), each of which has an unique complexity unit (CU) to address basic operations required for such DM. The average cycles required by a certain CU are either constants or can be predicted easily by a simple linear predictor. Currently, we propose to embed the number of CUs as the side information to guide the decoder to do accurate complexity estimation.

4.1 Motivation

Limited battery power supply is a crucial problem for the popular mobile video applications. Video decoding usually requires more energy compared with other services, such as audio playback, texting, etc, which is due to its complicated computational operations and dramatic data transfer within buffers. The shortage of battery power is more serious for high-definition (HD) video decoding using the advanced video coding standard, such as H.264/AVC and its scalable extensions [1, 7]. Generally, there are two sources of energy dissipation during video decoding [36]. One is the memory transfer. The other is CPU cycles. Both are power consuming. We will focus on the computational complexity induced power consumption and defer the off-chip memory complexity related power dissipation investigation for our future study.

It is essential and necessary to know the power consumption of the scalable video decoding so as to propose power efficient schemes. Moreover, power consumption can be expressed as a function of the required video decoding complexity (in terms of cycles per second) [37], i.e.,

$$P = \Phi(C), \tag{4.1}$$

where P and C describe the power consumption and computational complexity for video decoding respectively, and $\Phi()$ abstracts the relationship between power consumption and complexity which should be fixed for a typical hardware architecture [37]. Therefore, it is necessary to have an accurate complexity model for scalable video decoding which can be used to derive the power consumption model through (4.1).

4.2 Scalable Video Decoding

We choose the SVC to provide the scalable video with the focus on the joint temporal and amplitude scalability. The temporal scalability is provided by the hierarchical B-pictures [38]. Our work employes the dyadic hierarchical prediction structure, where the temporal frame rate t is doubled when the temporal resolution is refined to the next higher

level. An 8-picture GOP is exemplified in Figure 1.2(a) and the possible frame rates are 30, 15, 7.5, 3.75Hz respectively assuming the maximum frame rate is 30 Hz. Medium grain scalability (MGS) is chosen to provide the amplitude scalability [6, 7]. To avoid the decoder drift, we have constrained the motion estimation and compensation at current layer. To well exploit the inter-layer correlation, we have enabled all inter-layer predictions, such as inter-layer intra prediction, inter-layer motion/mode prediction as well as the inter-layer residual prediction to create the amplitude scalable video.

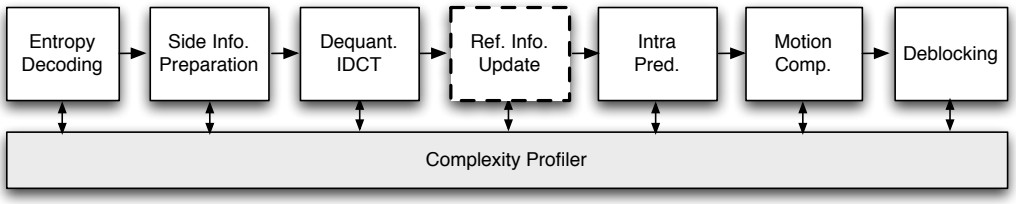


Figure 4.1: Scalable video decoding with single loop motion compensation, "reference information update" module is used to update the inter-layer prediction data structure.

To derive an accurate SVC decoding complexity model, we have to understand the steps for the scalable video decoding in details. Like what we have done in [39], we ignore the computational cycles required by high-level syntax parsing, such as parameter sets and slice header, and focus on the macroblock-level syntax parsing and decoding. According to our experiments presented in [39], the cycles for high-level syntax parsing are much less than 0.01% in comparison to the overall SVC decoding, which means that macroblock level parsing and decoding dominate the computing resource (and so as the energy).

As known, hierarchical B-picture is used to provide the temporal scalability, which only has the different high-level signaling compared with conventional B-picture in H.264/AVC. Therefore, the temporal enhancement decoding is similar as normal B-picture decoding. For the amplitude scalable video, in order to reduce the decoder complexity, *single loop motion-compensation* is required along with the SVC standardization [7]. Even with layered structure to provide the amplitude scalability, SVC decoder has the comparable decoder complexity to the H.264/AVC single layer decoder because of the single loop decoding design [5].

Table 4.1: SVC decoder modules

module	functionality
entropy decoding	MB overhead parsing and quantized transform coefficients (QTC) decoding
side info. preparation	side information initialization, such as mode, motion updates for surrounding blocks
dequant. and IDCT	MB dequantization and inverse transform
ref. info. update	inter-layer prediction update, such as inter-layer transform coefficient refinements, inter-layer motion, mode, residual updates, etc
intra prediction	MB intra prediction
motion compensation	reference signal fetch, interpolation and block reconstruction
deblocking	boundary strength calculation, edge filtering and filter buffer update

Instead of using the inefficient SVC reference software [22], we have developed our high modularized macroblock (MB) based SVC decoder targeting for the mobile handhelds. There are seven modules defined in our SVC decoder – entropy decoding, side information preparation, dequantization and inverse transform, reference information update, intra prediction, motion compensation and deblocking as illustrated in Figure 4.1. The bitstream is first fed into *entropy decoding* to obtain interpretable symbols for the following steps, such as side information (e.g., macroblock type, intra prediction modes, reference index, motion vector difference, etc) and quantized transform coefficients; the decoder then uses the parsed information to initialize necessary decoding data structures, which is so-called *side information preparation*. The block types, reference pictures, prediction modes, motion vectors, will be computed and filled in corresponding data structures for further usage. By this step, we let other decoding modules focus on their particular jobs, and this job isolation can make data preparation (for prediction purpose) and decoding more independent without interference. The *dequantization and inverse transform* are then called to convert quantized transform coefficients into block residuals. Before reaching the target layer, the lower layer information, such as residuals, transform coefficients, intra reconstructed pixels, will be updated and added up in *reference information update* module.

Because of the single loop decoding design, the decoding loop closes at the target layer only. At the target layer, decoded residuals are summed with predicted samples, from either *intra prediction* or *motion compensation* to form reconstructed signal. Finally, the *deblocking filter* is applied to remove blocky artifacts introduced by block based hybrid transform coding structure.

Usually, there is another module called MB writeout which is used to write out the reconstructed blocks. In our decoder implementation, we use the direct memory access (DMA) to do the MB write-out as well as the reference block fetch for motion compensation. Nowadays, DMA is a common feature for modern microprocessor, which runs for efficient data exchange between memory buffers and does not require CPU cycles when doing data transferring. Therefore, we don't include this module into our computational complexity modeling [1]. Table 4.1 summaries the decoding modules and their functionality. Each module can be well optimized using platform dependent instructions, such as MMX, SSE at x386 [40] and NEON at ARM-cortex architectures [41]. Because of the "single loop" design, compared with single layer H.264/AVC decoding, amplitude scalable video decoding just requires extra inter-layer references updates, such as transform coefficients, mode, motion and residual refinements, etc. For spatial scalability, additional upsampling and inter-layer deblocking operations are required, which is out of current work scope and deferred as our future research.

4.3 Complexity Modeling for Scalable Video Decoding

To model the complexity for SVC bitstream decoding, we implement the complexity profiler to collect the frame decoding complexity as shown in Figure 4.1. The complexity profiler can be supported by various chips, such as Intel Pentium mobile (Intel PM) [42] and ARM Cortex A8 (ARM) [43]. A specific instruction set [2] is called to record the processor

[1] On the other hand, we have emulated the MB writeout and reference block fetch simply using the memory operation functions provided by the standard C library, which means that the memory operation is part of CPU computing as well. Results show that such memory data exchange is quite constant, which doesn't affect our research results.

[2] Different platforms will use different instruction sets to write/read the processor state in specific registers.

state just before and after desired module, and the difference is the consumed computing cycles. The number of computational cycles spent in complexity profiling is less than 0.001% of the cycle number desired by the regular decoding module according to our measurement data. Hence it is negligible. The details about how to implement the complexity profiler for the Intel and ARM platforms can be found in [44]. For the SVC decoding complexity modeling, we measure the decoding complexity on the ARM CPU [42]. For the following complexity prediction for H.264/AVC decoding, we measure the complexity for both Intel PM and ARM CPUs [43]. Like bit rate points for each temporal and amplitude layer combination, we can obtain the complexity requirements (i.e., in terms of mega cycles per second) for each layer as well.

The same "impact separation" methodology used for quality and rate models is applied here as well, i.e., we focus on the impact of frame rate t on the complexity C, under the same quantization stepsize q; while characterizing the impact of q on the complexity, when the video is coded at a fixed frame rate. Towards this goal, the complexity model $C(q,t)$ can be written as

$$C(q,t) = C_{\max} C_t(t; q_{\min}) C_q(q; t), \tag{4.2}$$

where $C_{\max}$ is the maximum complexity demanded by decoding bitstreams coded at maximum frame rate and minimum quantization stepsize, i.e.,

$$C_{\max} = C(q_{\min}, t_{\max}), \tag{4.3}$$

$$C_t(t; q_{\min}) = \frac{C(t, q_{\min})}{C(t_{\max}, q_{\min})}$$

is the normalized complexity versus temporal resolution (NCT) under the minimum quantization stepsize $q_{\min}$ and

$$C_q(q; t) = \frac{C(t, q)}{C(t, q_{\min})}$$

is the normalized complexity versus quantization (NCQ) at any given any frame rate t. Note that NCQ function $C_q(q; t)$ tells how does complexity decreases as the quantization stepsize

increases beyond $q_{\min}$, given any frame rate t; while NCT function $C_t(t; q_{\min})$ characterizes how does complexity reduces as the frame rate decreases from $t_{\max}$, under the minimum quantization stepsize $q_{\min}$. As will shown later by experimental data, we have found that NCQ function $C_q(q; t)$ can be modeled by a function of q with the model parameter as a function of the frame rate t, denoted by $C_q(q; s(t))$ with $s(t)$ as the frame rate dependent parameter, and NCT function $C_t(t; q)$ is independent of the q, denoted by $C_t(t)$.

To see how complexity relates the temporal and amplitude resolution, we decode the video bitstreams created using the same test video pool for rate model, including videos with different content activities (such as texture, motion, contrast, etc) and different resolution (i.e., CIF, WVGA and 720p). The test bitstreams are generated with five temporal (i.e., corresponding to the frame rates at 1.875, 3.75, 7.5, 15, 30 Hz) and five amplitude layers (i.e., corresponding to the QPs at 44, 40, 36, 32, 28), therefore, we can have 25 extracted bitstreams corresponding to the different combinations of the temporal and amplitude resolutions. Each extracted substream is decoded to collect the complexity. The cycles of all layers are normalized by the cycle number at the highest frame rate to find NCT points, which are plotted in Figure 4.2. As shown, the NCT points overlap for different quantization stepsize and can be captured by a single curve quite well. Similarly, the NCQ curves for different frame rate are plotted in Figuer 4.3. Unlike NCT curves, NCQ curves are frame rate dependent. However, we have found that the NCQ curves can be also predicted accurately by a function of quantization stepsize with a frame rate dependent parameter. Therefore, the overall complexity modeling work is divided into two parts, one is to develop an appropriate functional form for $C_t(t)$, so that it can model the measured NCT points accurately, the other is to devise a proper function for $C_q(q; s(t))$ to model the measured NCQ points accurately for every frame rate t. The derivation of respective $C_t(t)$ and $C_q(q; s(t))$ are explained as follows.

4.3.1 Model for Normalized Complexity vs. Temporal Resolution $C_t(t)$

As suggested earlier, $C_t(t)$ is used to describe the reduction of the normalized complexity as the frame rate reduces. Therefore, the desired property for the $C_t(t)$ function

is that it should be 1 at $t = t_{\max}$ and monotonically reduces to 0 at $t = 0$. Based on the measurement data in Figure 4.2, we choose a linear function to model the NCT curves, i.e.,

$$C_t(t) = \frac{t}{t_{\max}}, \tag{4.4}$$

where $t_{\max}$ is a known variable (e.g., $t_{\max} = 30$ Hz in our work). We do notice that a power function, i.e., $(t/t_{\max})^{\varphi}$ with φ around 1 (e.g., $\varphi = 1.02$ for "stockholm" video), can provide better approximation. However, the reduction of the prediction error is limited. Thus, we choose the simple linear function to model the NCT curves.

Figure 4.2 shows the model curve using (4.4) along with the measured data. It can be seen that the model fits the measured data points very well.

4.3.2 Model for Normalized Complexity vs. Quantization $C_q(q; s(t))$

The same to the $C_t(t)$ function, $C_q(q; s(t))$ is applied to describe the reduction of the normalized complexity as the quantization stepsize increases (i.e., corresponding to decoding less amplitude layers) at any given frame rate t. The desired property for the $C_q(q; s(t))$ function is that it should be 1 at $q = q_{\min}$ and monotonically reduces to 0 as q goes to infinity. Based on the measured data in Figure 4.3, we choose an inverse power function, i.e.,

$$C_q(q; s(t)) = \left(\frac{q}{q_{\min}}\right)^{-s(t)}, \quad s(t) = g_1 \left(\frac{t}{t_{\max}}\right)^{-g_2}, \tag{4.5}$$

where $s(t)$ is the frame rate dependent parameter which can be captured by another two-parameter power function as well. Here, g_1 and g_2 are content dependent parameters.

Figure 4.3 shows the model curve with frame rate sensitive parameter (4.5) along with the measured data. In addition, $s(t)$ can be well fitted by choosing proper g_1 and g_2 for different sequences, as shown in Figure 4.4.

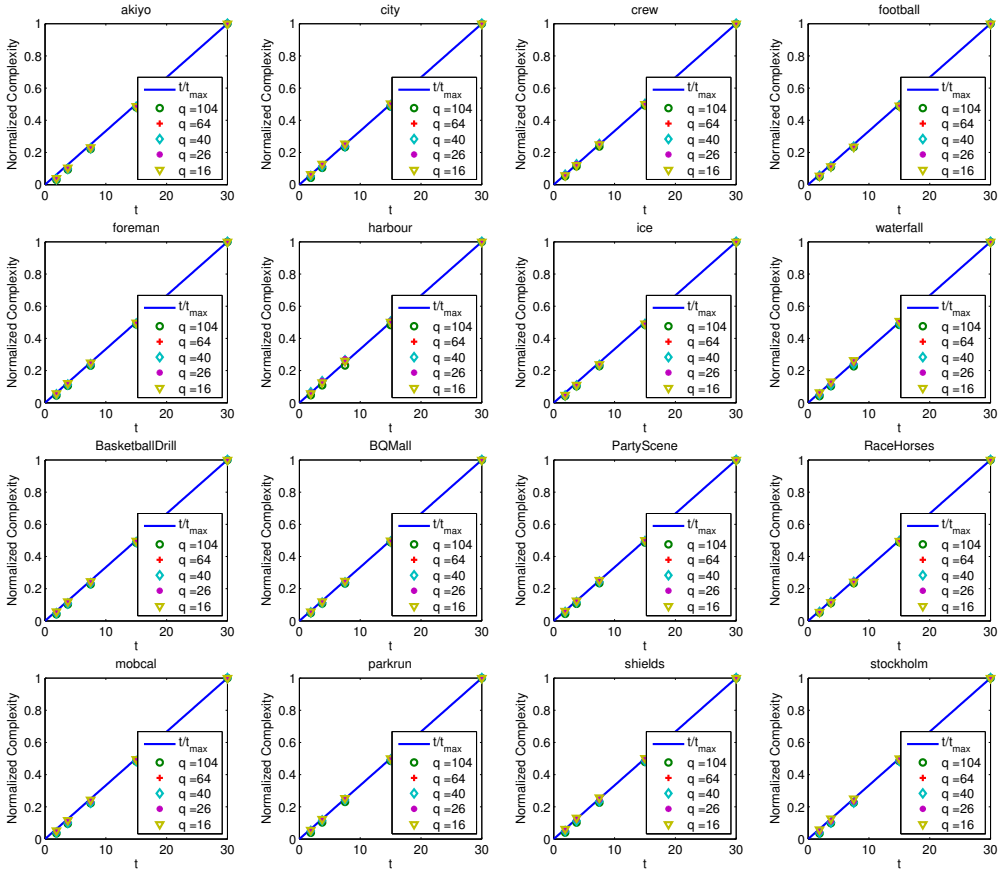


Figure 4.2: Normalized complexity versus temporal resolution (NCT) using different quantization stepsize q. Points are measured complexity, curves are predicted complexity by the model of (4.4).

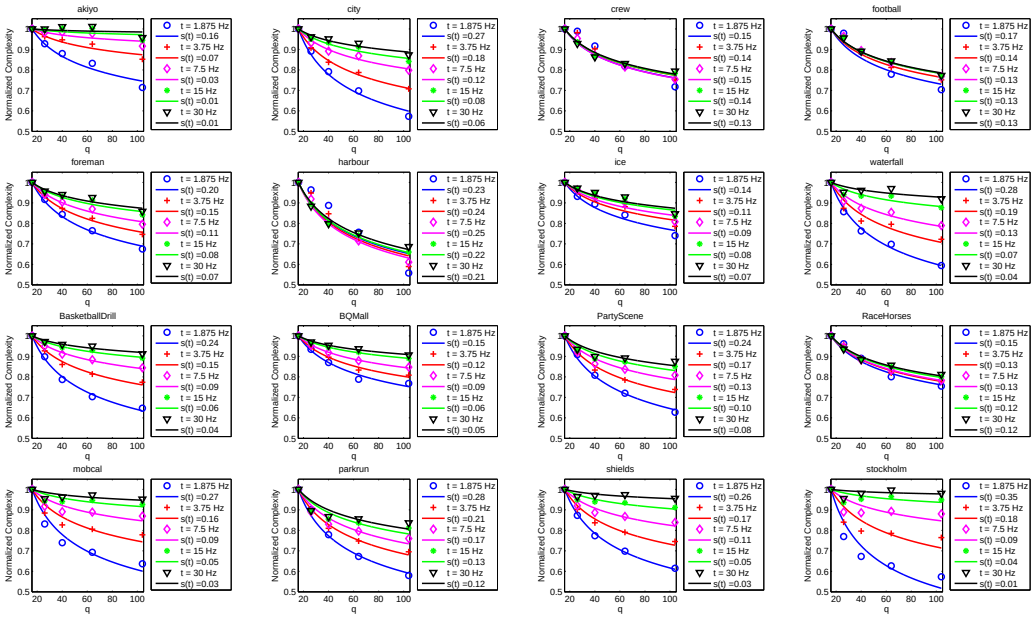


Figure 4.3: Normalized complexity versus quantization (NCQ) using different frame rate q. Points are measured complexity, curves are predicted complexity by the model of (4.5).

4.3.3 The Overall Complexity Model

Combining Eq. (4.3), Eq. (4.4) and Eq. (4.5), we can obtain the overall complexity model as

$$C(q,t) = C_{\max}\left(\frac{q}{q_{\min}}\right)^{-s(t)}\left(\frac{t}{t_{\max}}\right), \tag{4.6}$$

where

$$s(t) = g_1\left(\frac{t}{t_{\max}}\right)^{-g_2},$$

with g_1 and g_2 as content dependent parameters for overall complexity model. The actual complexity data of all test sequences with different combinations of q and t, and the corresponding estimated cycles via the proposed model (4.6) are illustrated in Figure 4.5, we note that the model predictions fit very well with the experimental complexity points. The model parameters, g_1 and g_2 are obtained by minimizing the mean squared errors (MSE) between the measured and predicted complexity. Table 4.2 lists the parameter values, fitting errors in terms of the relative RMSE, and the Pearson correlation (PC) between measured

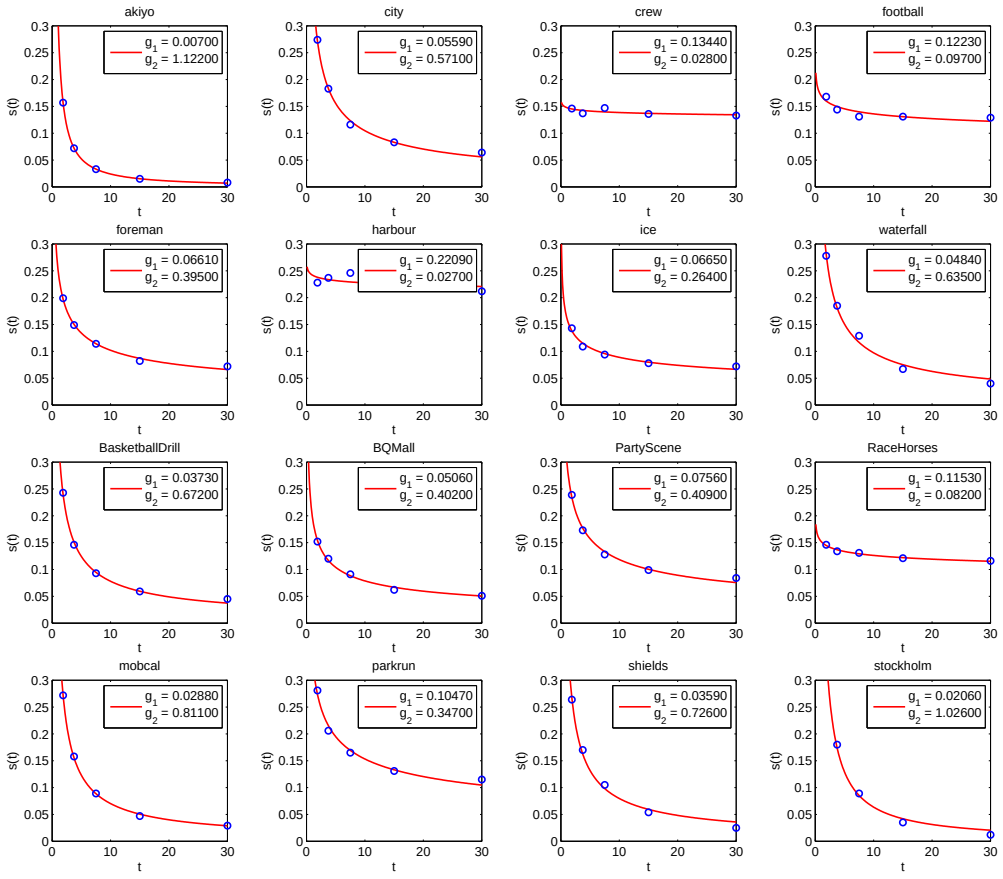


Figure 4.4: Power function approximation for $s(t)$ where g_1 and g_2 are the content dependent parameters.

and predicted complexity. We see that the model is very accurate for all sequences, with very small relative RMSE and very high PC.

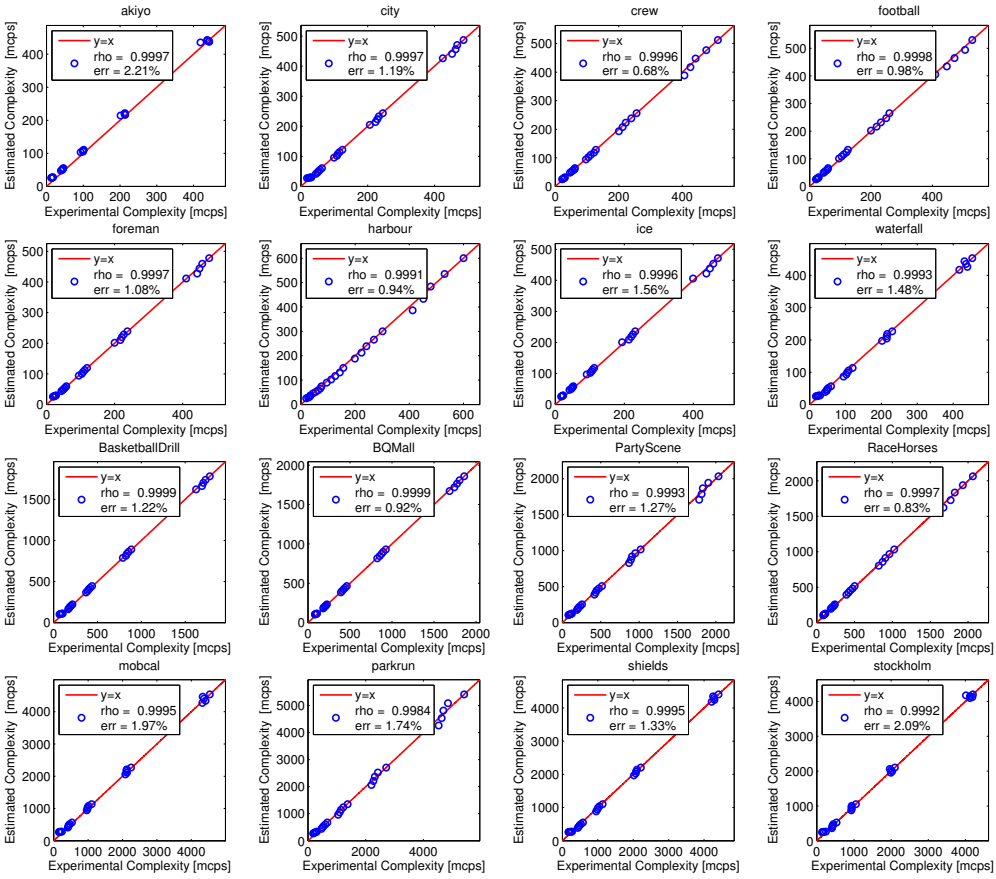


Figure 4.5: Comparison between measured complexity and predicted complexity using the overall complexity model (4.6), prediction error is presented using both Pearson correlation (PC) coefficients and RMSE normalized by the $C_{\max}$.

4.3.4 Power Consumption Model for ARM Processor

As discussed above, we can use the Eq. (4.1) to relate the processor power consumption in terms of the complexity workload requirements. Currently, popular chips, such as Intel Pentium mobile [42] and ARM [43], which are widely deployed on mobile devices, can support dynamic voltage/frequency scaling (DVFS) according to the processor's instantaneous workload and temperature, etc, or by user defined manner, so as to save energy [45].

Table 4.2: Parameters for complexity model and model accuracy

seq.	g_1	g_2	$C_{\max}$ [mcps]	RMSE/$C_{\max}$	PC
akiyo	0.007	1.122	443	2.21%	0.9997
city	0.056	0.571	487	1.19%	0.9997
crew	0.134	0.027	512	0.68%	0.9993
football	0.122	0.097	530	0.98%	0.9997
foreman	0.066	0.396	478	1.08%	0.9997
harbour	0.221	0.027	600	0.94%	0.9984
ice	0.067	0.263	471	1.56%	0.9996
waterfall	0.048	0.636	453	1.48%	0.9994
BQMall	0.051	0.402	1858	0.92%	0.9999
RaceHorses	0.115	0.083	2064	0.83%	0.9995
BasketballDrill	0.037	0.671	1781	1.21%	0.9999
PartyScene	0.076	0.409	2034	1.27%	0.9992
mobcal	0.029	0.812	4535	1.97%	0.9995
parkrun	0.105	0.347	5413	1.74%	0.9975
shields	0.036	0.724	4407	1.33%	0.9996
stockholm	0.021	1.024	4201	2.09%	0.9993
ave.				2.15%	0.9994

Table 4.3: Supported Dynamic Voltages and Clock Rates of ARM Processor on TI OMAP35x EVM

OPP	5	4	3	2	1
Voltage (volt)	1.35	1.27	1.20	1.00	0.95
Max. Freq. (MHz)	600	550	500	250	125

Typically, for a DVFS-capable processor, there are four kinds of power consumption, i.e.,

$$P_{\texttt{tot}} = P_{dyn} + P_{stc} + P_{dp} + P_{on}, \tag{4.7}$$

where

$$P_{dyn} = K_{eff} V_{dd}^2 f \tag{4.8}$$

is the dynamic power with K_{eff} as the effective circuit capacitance, V_{dd} is the supportable voltage, and f is the clock frequency. P_{stc} is the static power due to the leakage sources, such as subthreshold leakage (I_{sub}), the reverse bias junction current (I_j) and gate leakage current (I_g), etc and it can be written as

$$\begin{aligned} P_{stc} &= L_g(V_{dd} I_{sub} + |V_{bs}| I_j + V_{dd} I_g), \\ I_{sub} &= K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}}, I_g = K_6 e^{K_7 V_{dd}}, \end{aligned} \tag{4.9}$$

where L_g, V_{bs}, I_j, K_3, K_4, K_5, K_6 and K_7 are constants [46,47]. The leakage power cannot be neglected when the circuit feature sizes become smaller (below 90 nanometers) [46]. In particular, for many processors deployed in popular mobile handhelds, which are fabricated using 70nm or even smaller technology node, the static power cannot be ignored. P_{on} is a constant, which always exists once the processor is turned on. P_{dp} is the short circuit power, i.e., $P_{dp} = V_{dd} I_{dp}$ where I_{dp} is the average direct path current. Typically, V_{dd} is related to f by

$$V_{dd} = \omega f^{\phi} + \theta, \tag{4.10}$$

where parameters ω, ϕ and θ are approximated by fitting the supportable pairs of f_k and V_k by the underlying platform. Hence we can approximate the total power as a convex function of the voltage [3], noted as

$$P_{\texttt{tot}}(V_{dd}) = p_1 V_{dd}^{\gamma_1} + p_2 V_{dd} e^{\gamma_2 V_{dd}} + p_3, \tag{4.11}$$

where $1 < \gamma_1 < 2, \gamma_2 > 0$, and $p_i, i = 1, 2, 3$ are constants for a specified processor. For simplicity, we can use $P(V_{dd})$ instead of the $P_{\texttt{tot}}(V_{dd})$.

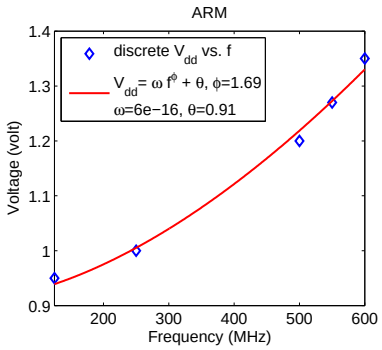


Figure 4.6: Relation between voltage and frequency for ARM processors on TI OMAP35x.

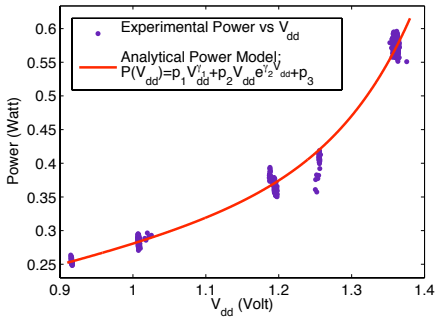


Figure 4.7: Power consumption model for ARM processor, parameters are obtained via least square error fitting.

Table 4.4: Parameters for ARM power consumption model

ω	ϕ	θ	p_1	p_2	p_3	γ_1	γ_2
6×10^{-16}	1.69	0.91	0.145	1.12×10^{-5}	0.12	1.44	7.05

Specifically, we have derived the analytical power consumption for the ARM processor deployed on our TI OMAP35x EVM [48] [39]. The ARM processor on the TI OMAP35x board only supports discrete set of voltage and frequency levels, as shown in Table 4.3 [48]. Each pair of voltage and frequency is associated with a CPU operating point (OPP) state. To derive the analytical power consumption model for ARM processor, we fit the voltages and clock rates in Table 4.3 and have found that the voltage and frequency are related by (4.10) with $\phi = 1.69$, $\omega = 6 \times 10^{-16}$ and $\theta = 0.91$ as depicted in Figure 4.6. To evaluated the relation between power and the voltage, we collect the instant power and its corresponding voltage (by choosing a certain OPP), plot them as scatter points in Figure 4.7, and find its best fit using the power model in (4.11). Because of the unique mapping between supply voltage and its clock rate for the ARM processor as shown in Table 4.3 and Figure 4.6, we can derive the power consumption model for ARM as a function of the clock rate f, i.e., by combing Eqs. (4.11) and (4.10), we can reach at

$$P(f) = p_1 \left(\omega f^{\phi} + \theta\right)^{\gamma_1} + p_2 \left(\omega f^{\phi} + \theta\right) e^{\gamma_2 \left(\omega f^{\phi} + \theta\right)} + p_3, \tag{4.12}$$

[3]Here, we merge the P_{dyn} and P_{dp} together and estimate them via the first item in Eq. (4.11).

where the parameters are listed in Table 4.4. Furthermore, we have found that a simplified power function can well approximate the power consumption with respect to the clock rate for the ARM processor, as shown in Figure 4.8. Therefore, Eq. (4.12) can be rewritten as

$$P(f) = \kappa_1 f^{\varphi} + \kappa_2, \tag{4.13}$$

where $\kappa_1 = 3.06 \times 10^{-10}$, $\varphi = 3.19$, $\kappa_2 = 0.26$ and f is the clock rate in mega hertz (MHz). As known, the parameters are fixed for a typical platform.

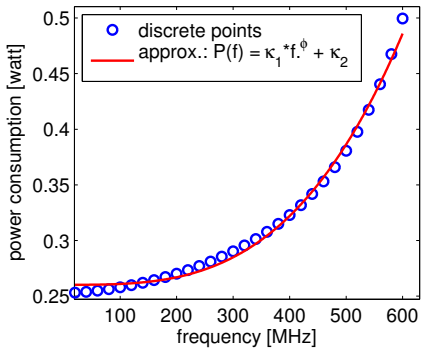


Figure 4.8: Simple power function approximation for power consumption model in terms of the clock rate f [MHz], where $\kappa_1 = 3.06 \times 10^{-10}$, $\varphi = 3.19$, $\kappa_2 = 0.26$

In previous Section 4.3.3, we have developed the complexity model in terms of the frame rate and quantization stepsize, which indicates how many *cycles per second* are required to conduct the decoding for a certain combination of the temporal and amplitude resolution. In other words, *cycles per second* means the exact clock rate required in Hz. Hence, we can map the complexity to the processor frequency directly, $f = C(q,t)$. Combing Eqs. (4.1), (4.6) and (4.13), we can obtain the power consumption model for scalable video decoding on ARM processor as

$$P(q,t) = \Phi(C(q,t)) = \kappa_1 C(q,t)^{\varphi} + \kappa_2. \tag{4.14}$$

4.4 Complexity Prediction for H.264/AVC Decoding

In previous sections, we have presented the complexity model for scalable video decoding with focus on the joint temporal and amplitude scalability, and applied a two-parameter function to model the impact of complexity with respect to the frame rate and quantization stepsize. Moreover, we extend the power consumption model for scalable video decoding on ARM processor as a function of the frame rate and quantization stepsize, i.e., $P(q,t)$, by combing the power consumption model of processor clock rate (4.13) and SVC decoding complexity model (4.6). As known, the proposed power consumption model of scalable video decoding can be applied to guide "smart" SVC adaption according to the receiver battery status. After deciding the exact layers to extract, another problem arises is whether we can further reduce the power consumption when conducting the video decoding so as to save more power. One decent solution to achieve this is adapting the the voltage and frequency of the underlying processor according to the instant video decoding complexity workload, and this dynamic voltage/frequency scaling (DVFS) function is supported by various chips and microprocessors, such as Intel Pentium mobile [42] and ARM cortex A8 [43]. In devices using dynamic voltage and frequency scaling (DVFS), being able to accurately predict the complexity of successive decoding intervals is critical to reduce the power consumption [37].

In this section, we will focus on the computational complexity modeling of the H.264 video decoding and defer the off-chip memory complexity investigation for our future study. Specifically, we extend our prior work [49] beyond the entropy decoding complexity and consider all modules involved in H.264/AVC video decoding, including entropy decoding, side information preparation, dequantization and inverse transform, intra prediction, motion compensation, and deblocking. First of all, we define these modules as *decoding modules* (DMs), and denote their complexity (in terms of clock cycles) over a chosen time interval by $C_{\texttt{DM}}$. The proposed model is applicable to any time interval, but the following discussion will assume an interval is one video frame. Furthermore, we abstract the basic, common operations needed by each DM as its *complexity unit* (CU), so that $C_{\texttt{DM}}$

is the product of the average complexity of one CU over one frame, $k_{\mathtt{CU}}$ and the number of CUs required by this DM over this frame, $N_{\mathtt{CU}}$. For example, the CU for the entropy decoding DM is the operation of decoding one bit, and the complexity of this DM, $C_{\mathtt{vld}}$, is the average complexity of decoding one bit, $k_{\mathtt{bit}}$, times the number of bits in a frame, $N_{\mathtt{CU}} = n_{\mathtt{bit}}$. That is $C_{\mathtt{vld}} = k_{\mathtt{bit}} n_{\mathtt{bit}}$. Among several possible ways to define the CU for a DM, we choose the definition that makes the defined CU either *fairly constant* or *accurately predictable by a simple linear predictor*. Note that the CU complexity may vary from frame to frame because the corresponding CU operations change due to the adaptive coding tools employed in the H.264/AVC. For example, in H.264/AVC, adaptive in-loop deblocking filter is used to remove block artifacts, which applies different filters according to the information of adjacent blocks, such as coding mode, quantization parameters, motion vector difference etc. Basically, a strong filter is employed for intra coded blocks, while a normal filter or no filter for inter coded blocks. The percentage of intra and inter coded block is highly content dependent, thus the average cycles required by the deblocking for one block, $k_{\mathtt{dblk}}$, would vary largely from frame to frame. Also, CU complexity varies among different hardware platforms. Therefore, we also explore how to predict the average complexity of a CU for a new frame, from the measured CU complexity in the previous frames. Meanwhile, we assume that the number of CUs, $N_{\mathtt{CU}}$, can be embedded into the bitstream to enable the decoder to conduct complexity prediction.

We measure the decoding complexity on both the Intel Pentium mobile CPU [42] (as an example of a general purpose architecture) and ARM Cortex A8 processor [43] (as an example of embedded architecture) to derive and validate our proposed complexity model. In the following Chapter 6, we make use of our complexity model to adapt voltage and clock rate on these platforms to evaluate the achievable energy saving on mobile handhelds, and further measure the actual power consumption on the TI OMAP35x EVM board [48], to validate our analytical results.

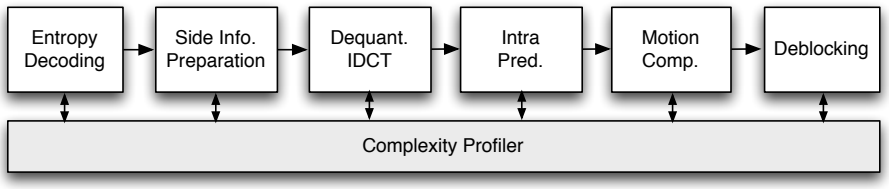


Figure 4.9: Illustration of H.264/AVC Decoder Decomposition.

4.4.1 H.264/AVC Decoder Abstraction

The H.264/AVC decoder can be decomposed into the following basic decoding modules (DMs): entropy decoding, side information preparation, dequantization and IDCT, intra prediction, motion compensation and deblocking as shown in Figure 4.9. Compared with the SVC decoder decomposition in Figure 4.1, there is an additional module called "reference information update" which is used to update the inter-layer reference information, such as mode, motion, intra reconstructed pixels and residuals, etc. For H.264/AVC single layer decoding, this additional module is not required. In addition to the collect the overall complexity for each layer earlier, we embed the complexity profiler in the H.264/AVC decoder to collect the cycles for each DM. As explained earlier, a H.264/AVC decoder can be decomposed into 6 DMs: entropy decoding (`vld`), side information preparation (`sip`), dequantization and inverse transform (`itrans`), intra prediction (`intra`), motion compensation (`mcp`) and deblocking (`dblk`). Each DM requires both memory transfer and computation by the CPU. For instance, the temporal reference block must be fetched into the processor to form the reconstructed signal of the current block. Because the mobile devices have limited on-chip memory space, we must store the temporal reference frame(s) into off-chip memory, and fetch the required block as needed. These on-chip and off-chip memory transfer operations can be done via the direct memory access (DMA) routine, which is a feature of modern computers and microprocessors. Using the DMA, the memory data exchange can be performed independently without demanding CPU cycles. In our work, the memory data transfer will be operated by the DMA without consuming the processor resource. For example, the MCP can be sliced into three major parts: refer-

ence block fetch, interpolation and block reconstruction (e.g., sum and clip). The reference block fetch is conducted by the DMA. Only interpolation and block reconstruction are conducted by the processor, and they contribute to the computational complexity. Furthermore, instead of analyzing the video decoding complexity at macroblock level, we will discuss the complexity of the H.264/AVC video decoding at frame level, and further GOP level.

Table 4.5: Essential DM and its CU in the H.264/AVC decoder

DM	Functionality	CU
`vld`	side info. parsing and quantized transform coefficients decoding	bit parsing
`itrans`	inverse transform module	MB dequant. & IDCT
`intra`	inverse intra prediction	MB intra pred.
`mcp`	motion-compensation	Half-pel interpolation
`dblk`	deblocking filter	α-point filtering
`sip`	side info. data structure init.	MB data structure init.

For each DM, we define a unique complexity unit (CU) to abstract required fundamental operations. For example, for the entropy decoding DM, the CU is the process involved in decoding one bit, whereas for the `itrans` DM, the CU is the process involved in dequantization and inverse transform for one macroblock (MB). Note that a CU includes all essential operations needed for a basic processing unit (a bit for `vld`, a MB for `itrans`) in a DM, instead of the basic arithmetic or logic Ops, such as `add`, `shift`, etc. Table 4.5 summarizes each DM and its corresponding CU.

Let $C_{\texttt{DM}}$ denote the required computation cycles to decode one frame by a particular DM, then the overall frame decoding complexity is the sum of the individual complexity required by each DM. As shown in Section 4.4.2, the complexity of each DM can be written as the product of $k_{\texttt{CU}}$ - the complexity of one CU, and $N_{\texttt{CU}}$ - the number of CUs required for decoding each frame. We further explain the CU identified for each DM and the corresponding $k_{\texttt{CU}}$ and $N_{\texttt{CU}}$ in Section 4.4.2.

We choose to focus on two different platforms analyzing the decoding complexity: the IBM ThinkPad T42 using the Intel PM processor and TI OMAP35x EVM board using the ARM processor. Table 4.6 provides the configuration of these two hardware platforms.

Table 4.6: Experiment Environment

item	General-purpose System	Embedded System
Platform	ThinkPad T42	TI OMAP35x EVM
Processor	Intel PM 1.6GHz	ARM Cortex A8 600MHz
OS	Ubuntu 8.04	Arago Embedded Linux
RAM	1G	256M

The former is representative of laptops using a low-power general purpose microprocessor, while the latter is typical of SmartPhones or other handheld devices. We developed our own H.264/AVC decoding software that can run on these two platforms efficiently. Targeting for low complexity mobile applications, we have not considered CABAC (Context-Adaptive Binary Arithmetic Coding), interlace coding, 8x8 transform, quantization matrix and error resilient tools (e.g., flexible macroblock order, arbitrary slice order, redundant slice, data partition, long term reference, etc). The baseline, main, and high profiles are supported but without the tools listed above, while supportable levels are constrained by the underlying hardware capability. For example, the decoder can decode bitstreams smoothly up to level 3 on our OMAP platform, and up to level 3.2 using Intel PM based ThinkPad T42 [4]. Our decoder operates at the macroblock level [50], given the limited on-chip memory on mobile processors, following the block diagram shown in Figure 4.9. In our implementation, we use DMA to write back reconstructed samples from on-chip buffer to off-chip memory and fetch a large chunk of data into on-chip memory for motion compensation, e.g., 3 macroblock lines, which means if the motion vector (MV) of current MB is within this range, there is no need to do on-the-fly reference block fetching. It is possible that the MV is out of this range (i.e., exceeding 3 macroblock lines) and need the interrupt of CPU to fetch such reference block. However, according to our simulations, such events happen with a very small probability (less than 1% in our experiments). Please note that our decoder implementation represents a typical implementation for embedded systems and hence the complexity model derived for our decoder is generally applicable.

In order to validate our complexity model, we have created test bitstreams using stan-

[4]The decoder will crash with insufficient memory when we try to decode bitstreams at higher levels for OMAP board, while running very slow for Intel PM platform.

Table 4.7: Supported Encoder Features

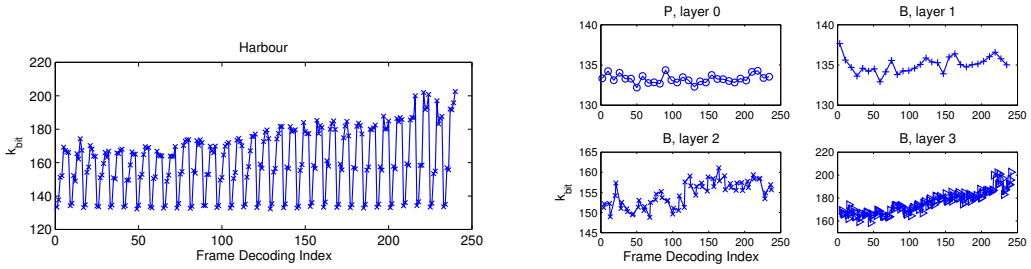
Item	Description
GOP length	8-frame
reference number	1
slice map	1 slice per frame
entropy coding	CAVLC, Exp-Golomb, Fix-length
transform	4×4
intra prediction	intra4x4, intra16x16
inter prediction	variable-block size from 16x16 to 4x4
interpolation	up to $1/4$-pixel
deblocking	enabled

dard test sequences, e.g., Harbor, Soccer, Ice, News, all at CIF (i.e., 352×288) resolution. These four video sequences have different content activities, in terms of texture, motion activities, etc. A large quantization parameter (QP) range, from 10 to 44 in increments of 2, is chosen to create the test bitstreams. Particularly, we enable the dyadic hierarchical-B [51] prediction structure in the encoder, thus the test bitstreams inherently support the temporal scalability [6]. The reference software of scalable extension of the H.264/AVC video coding standard (SVC) [5], i.e., JSVM [22], is used for generating the H.264/AVC compliant test bitstreams. Note that these bitstreams are with the temporal scalable functionality enabled. The adopted encoder setting is described in Table 4.7. These created bitstreams are decoded on both Intel PM and ARM featured platforms and the complexity per DM as well as the total complexity per frame are measured.

4.4.2 Frame-level H.264/AVC Decoding Complexity Modeling

In this section, we identify the CU for each DM and further consider how to predict the CU complexity from frame to frame.

[5] Compared with the H.264/AVC reference software, i.e., JM, JSVM outputs the same encoded bitstream using the same encoding configuration, but with slight difference in high level header signaling, which doesn't affect our work.



(a) (b)

Figure 4.10: Variation of $k_{\texttt{bit}}$ when decoding "Harbour" at QP=28 on the Intel PM platform. (a) In the frame decoding order over the entire sequence; (b) In the frame decoding order over different temporal layers.

Entropy Decoding

Intuitively, we model the entropy decoding complexity as the product of the bit decoding complexity and the number of bits involved, i.e.,

$$C_{\texttt{vld}} = k_{\texttt{bit}} \cdot n_{\texttt{bit}}, \tag{4.15}$$

where $k_{\texttt{bit}}$ is the average cycles required for decoding one bit, and $n_{\texttt{bit}}$ is the number of bits for a given frame. Note that $n_{\texttt{bit}}$ can be exactly obtained after de-packetizing the H.264/AVC NAL (network abstraction layer) unit [1]. The total bits in H.264/AVC bitstream are mainly spent for side information, and Quantized Transform Coefficients (QTC). Generally, the average cycles required by bit parsing for the side information and QTC are different [49]. Because the percentage of bits spent on each part varies with the video content and the bit rate, the average cycles required per bit parsing $k_{\texttt{bit}}$ cannot be approximated well by a constant. As exemplified in Figure 4.10(a) for "Harbour" at QP 28, $k_{\texttt{bit}}$ varies largely in decoding order. However, after decomposing frames into different temporal layers, we have found that $k_{\texttt{bit}}$ changes more slowly from frame to frame in the same temporal layer, as shown in Figure 4.10(b). Thus we can update $k_{\texttt{bit}}$ for the current frame using the actual bits and the consumed cycles by entropy decoding for the previous decoded frame in the same layer. Although only data for "Harbour" at QP 28 for Intel PM

platform are presented here, the data for all other sequences at different QPs are similar according to the measured complexity data.

The estimated and actual cycles of all test bitstreams are plotted in Figure 4.11 for both Intel and ARM platform. From this figure, it is noted that the actual complexity can be well estimated by Eq. (4.15) and the proposed method for predicting $k_{\texttt{bit}}$.

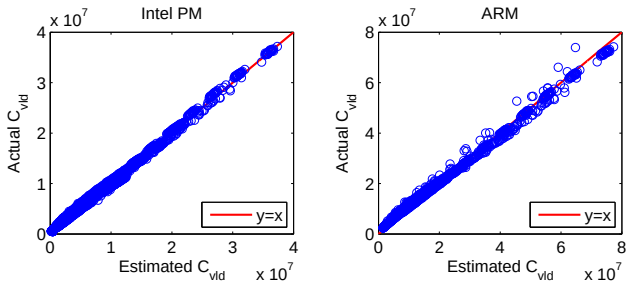


Figure 4.11: Illustration of entropy decoding complexity estimation using Eq. (4.15), $k_{\texttt{bit}}$ is predicted using complexity data from the same layer nearest decoded frame. The actual and estimated $C_{\texttt{vld}}$ of four test videos at all QPs are presented.

Side Information Preparation

After parsing the overhead information in the bitstream, macroblock type, intra prediction mode, sub-macroblock type, reference index, motion vectors, etc, are obtained and stored in proper data structure for future reference. We further include macroblock sum/clip, deblocking boundary strength calculation into SIP DM. Let $k_{\texttt{MBsip}}$ represent the average clock cycles for side information preparation per macroblock, and $n_{\texttt{MB}}$ the number of macroblock per frame. The total complexity for SIP can be written as

$$C_{\texttt{sip}} = k_{\texttt{MBsip}} \cdot n_{\texttt{MB}}, \tag{4.16}$$

Generally, $k_{\texttt{MBsip}}$ depends on the frame type. For example, in intra mode, we don't need to fill the motion vector and reference index structures. For uni-directional prediction in P-frame, we only need to fill the forward prediction related data structure, whereas for bi-directional prediction in B-frame, we need to fill both forward and backward related data

structures. We have found from measured complexity data that $k_{\texttt{MBsip}}$ is almost constant in the same temporal layer, but different among temporal layers. Thus, we have predicted $k_{\texttt{MBsip}}$ from prior decoded frame in the same layer as with entropy decoding.

Dequantization and IDCT

Only 4-by-4 integer transform and scalar de-quantization are considered in our current work [6]. We have unified the dequantization and IDCT as a single decoding module, i.e., "`itrans`". In H.264/AVC, the dequantization and IDCT can be skipped for zero macroblocks, and only operate at non-zero macroblocks. We have found that the computation complexity of macroblock dequantization and IDCT is fairly constant for all non-zero blocks. Therefore, given a frame, the complexity consumed by `itrans` can be written as,

$$C_{\texttt{itrans}} = k_{\texttt{MBitrans}} \cdot n_{\texttt{nzMB}} \tag{4.17}$$

where $n_{\texttt{nzMB}}$ is the number of non-zero macroblock per picture. $k_{\texttt{MBitrans}}$ describes the complexity of macroblock de-quantization and IDCT, and is a constant.

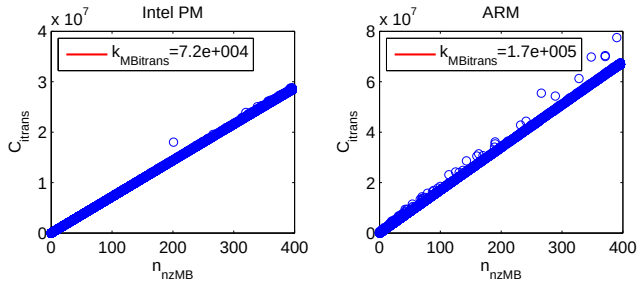


Figure 4.12: Complexity consumption (in cycles) dissipated in itrans DM against the non-zero MBs for all CIF resolution test videos. Parameters are obtained via Least-square-error fitting.

Figure 4.12 shows the measured complexity for the itrans DM for Intel and ARM platforms, respectively. It is shown that for a given implementation platform, $k_{\texttt{MBitrans}}$ is indeed a constant independent of the sequence content.

[6] In H.264/AVC, there is a second stage transform, i.e., Hadamard transform, applied on the luma DC (e.g., for intra16x16 mode), and chroma DC coefficients. For simplicity, we merge these hadamard transforms into 4-by-4 integer transform. Also, we defer the adaptive transform (with 8-by-8) for our future work.

Intra Prediction

In intra prediction module, adaptive 4×4 and 16×16 block-based predictions are used for luma component, and 8×8 block based prediction is used for chroma. There are 4 prediction modes of intra16x16, 9 prediction modes of intra4x4 for luma component, and 4 prediction modes of intraCr for chroma components. We have found from experimental data that there is no need to differentiate among different intra prediction types. Rather, we can just model the total complexity due to intra prediction by

$$C_{\texttt{intra}} = k_{\texttt{intraMB}} \cdot n_{\texttt{intraMB}}, \tag{4.18}$$

where $k_{\texttt{intraMB}}$ denotes the average complexity of performing intra-prediction for one intra-coded MB (averaged over all intra-prediction types), and $n_{\texttt{intraMB}}$ is the the number of intra macroblock per frame.

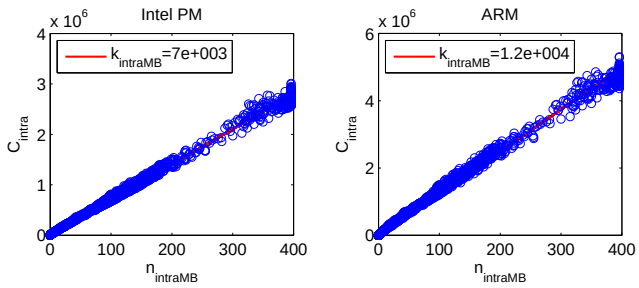


Figure 4.13: Intra prediction complexity $C_{\texttt{intra}}$ against the corresponding number of intra MB $n_{\texttt{intraMB}}$. Intra prediction complexity data from four different test videos at different QPs are presented, and can be well fitted by Eq. (4.18).

We collect and plot the number of intra macroblock and its corresponding intra prediction complexity for each frame from all test videos decoding data in Figure 4.13. It is shown that the model (4.18) works pretty well for different video content at different quantization levels (i.e., compressed via different QP), and parameter $k_{\texttt{intraMB}}$ is constant for a specific implementation on a target platform.

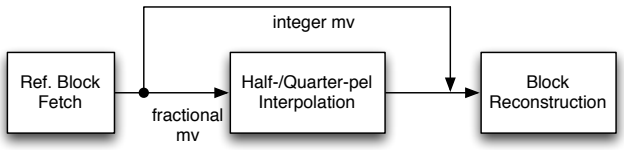


Figure 4.14: Modularized motion-compensation in H.264/AVC

Motion Compensation

The overall motion compensation module is divided into three parts: reference block fetching, interpolation and block reconstruction (sample sum and clip), as depicted in Figure 4.14. As mentioned above, the reference block fetching is conducted by DMA which does not consume CPU cycles. Only interpolation and block reconstruction are discussed in current section. Note that, on block reconstruction step, the compensated signal and residual block are added prior to being fed into deblocking filter, and its computational complexity can be treated as a constant because of the fixed *sum* and *clip* operations per macroblock. Thus, our major work in motion compensation module of video decoding is to model the complexity dissipated on the fractional accuracy pixel interpolation. Our experiment results show that the complexity of chroma interpolation can be approximated by a constant. The luma interpolation will further be addressed in details in subsequent sections. For simplicity, the term "interpolation" stands for the "luma interpolation" unless we point out exactly.

A naive way to model the complexity of MCP would be

$$C_{\texttt{mcp}} = k_1 \cdot n_{\texttt{mcpMB}}, \tag{4.19}$$

where k_1 is the unit complexity for macroblock interpolation, and $n_{\texttt{mcpMB}}$ is the number of macroblocks requiring interpolation. However, according to our extensive simulations, it is hard to predict k_1 accurately using any information from a given bitstream, or the complexity of the MCP DM in previous frames. This is because the MCP complexity for each MB depends on the motion vector for the MB. When both components of the MV are integers, no computation complexity is needed, whereas when one or both components are fractional numbers, more computations are required.

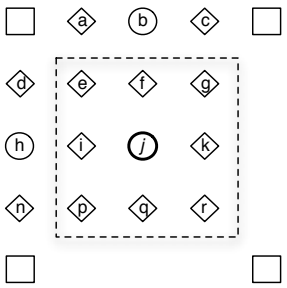


Figure 4.15: Fractional pixel interpolation in H.264/AVC with "□" ,"◯", "◇" standing for integer, half-, quarter-pel positions. The fractional points inside "dashed" box need half-accuracy interpolation twice.

Instead of investigating at block level, we analyze the MCP complexity at pixel level. In H.264/AVC, 6-tap Wiener filtering is applied to interpolate a half-pel pixel, while 6-tap Wiener plus 2-tap bilinear filtering are required for quarter-pel interpolation. Typically, the cycles required by 6-tap Wiener filtering and bilinear filtering are constants for a specified implementation, thus the complexity dissipated for interpolation is determined by the number of 6-tap Wiener and bilinear filtering operations. In Figure 4.15, we sketch the integer, half-pel, and quarter-pel positions according to the interpolation defined in the H.264/AVC standard [1]. The "□" positions are directly obtained via DMA from off-chip memory. The other 15 fractional positions need to be interpolated on-the-fly, and they consume different complexity because they require different interpolation filters. Due to the on-chip buffer limitation of embedded system architecture, which does not permit frame-based interpolation, whether to do interpolation is determined by the parsed motion vector pairs, i.e., ($\mathtt{mv}_\mathtt{x}$, $\mathtt{mv}_\mathtt{y}$) for a block.

Note that there are complexity differences in half-pel interpolation operations. For example, in Figure 4.15, pixel "b" and "h" can be created via 6-tap Wiener filter at one time. However, position "j" should be computed after creating "b" or "h". Thus, "b" and "h" require one 6-tap filtering, and "j" needs 6-tap filtering twice. Let us assume the unit complexity for constructing "b", "h" and "j" are k_b, k_h and k_j respectively. Assuming the unit complexity of 6-tap Wiener filtering is $k_{\mathtt{half}}$, then we can write $k_b = k_h = k_{\mathtt{half}},\ \ k_j =$

$2 \cdot k_{\texttt{half}}$.

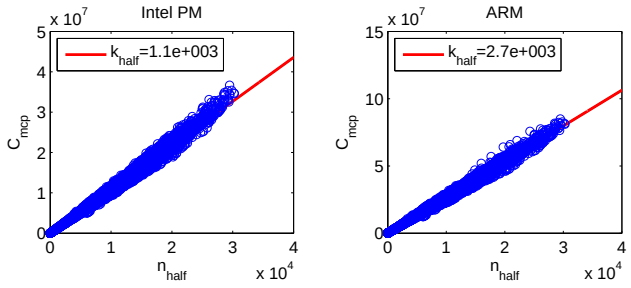


Figure 4.16: Interpolation complexity $C_{\texttt{mcp}}$ against the number of 6-tap Wiener interpolation filtering ($n_{\texttt{half}}$) required. All interpolation complexity of four different videos at different QPs are collected and presented together.

As explained in the standard, the quarter-pel pixels will be computed from adjacent half and/or integer pixels using bilinear filter [1]. Then, the 12 quarter-pel positions can be categorized into two classes: one needs a 6-tap plus a bilinear filter, such as “a”, “c”, “d”, “n”, and the other requires twice 6-tap filtering plus a bilinear operation, like “e”, “f”, “g”, “i”, “k”, “p”, “q”, “r”. Based on our measured complexity data, we have found that the half-pel interpolation using the 6-tap Wiener filter dominates the overall interpolation complexity, thus, the computational complexity of bilinear filtering can be neglected to simplify our further exploration. Therefore, we propose to approximate the MCP complexity by the product of the number of 6-tap Wiener filtering operations and the unit complexity required to perform one 6-tap Wiener filtering [7], i.e.,

$$C_{\texttt{mcp}} = k_{\texttt{half}} \cdot n_{\texttt{half}}, \tag{4.20}$$

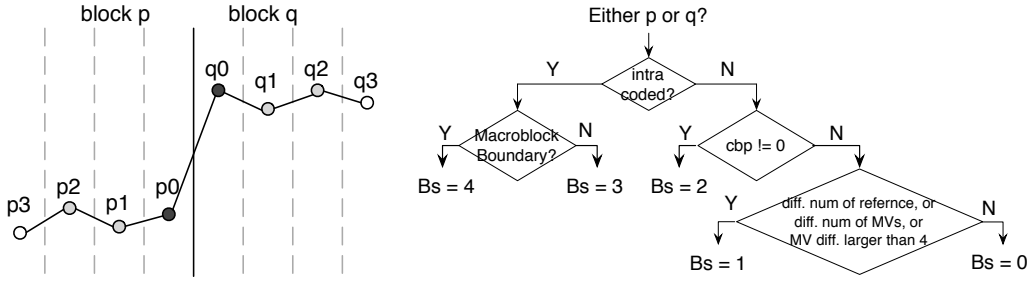
where $k_{\texttt{half}}$ is average complexity required to conduct one 6-tap Wiener filtering, and $n_{\texttt{half}}$ is the number of 6-tap filterings needed in decoding a frame. In the encoder, once we know the motion vector of each block, we can obtain the exact $n_{\texttt{half}}$. We can embed $n_{\texttt{half}}$ in the

[7] We found that there was slight difference between interpolation complexity for P and B pictures. Specifically, there was a constant offset for B picture interpolation (e.g., less than 2% compared with total frame decoding cycles in our simulation on Intel PM), however, compared with the total complexity consumed by whole frame decoding, this constant offset can be ignored.

bitstream header of each frame to enable the decoder to predict the complexity associated with motion compensation. Parameter $k_{\texttt{half}}$ is fairly constant for a fixed implementation.

The actual $n_{\texttt{half}}$ and cycles consumed by the MCP module have been collected by decoding all test bitstreams, and plotted in Figure 4.16. Note that the model (4.20) can quite accurately express the relationship between MCP (i.e., interpolation) complexity and the number of half-pel filtering operations.

Deblocking



(a) edge-crossing pixels of p and q (b) boundary strength calculation

Figure 4.17: 4×4 block edge illustration and boundary strength decision in H.264/AVC.

In the H.264/AVC video coding standard, an adaptive in-loop deblocking filter [1] is applied to all 4×4 block edges for both luma and chroma components, except picture boundaries [8]. There are several options defined in the H.264/AVC standard [1] to inform the decoder to apply the proper filter. However, in this paper, we only consider the two basic options, i.e., option 0 and 1 which indicate to enable and disable deblocking filter respectively [9]. Figure 4.17 depicts the block edge with related edge-crossing pixels distributed in left and right blocks, and the boundary strength decision tree defined in the H.264/AVC [1, 52]. Figure 4.17(b) shows that the boundary strength (e.g., `Bs` level) is determined by the block type, CBP (code block pattern), reference index difference, and

[8] Actually, the filter could be applied to 8×8 block edges if 8×8 transform is adopted, and the filtering operation will be disabled at some slice boundaries by enabling high-level filter syntax controlling. However, in our discussion, we just concern one slice per picture, and adopt only 4×4 as basic block size.

[9] The conditional filter crossing slice boundary, and separated filters for luma and chroma components are not considered in this paper.

motion vector difference from block p and q. According to our simulations, the complexity of calculating Bs can be treated as a constant, but with slight difference for I/P/B-picture. In our complexity modeling, the complexity of Bs calculation is merged with the SIP complexity $C_{\texttt{sip}}$ as discussed before in Sec. 4.4.2. Here, we only consider the edge filtering operations and its computational demands for in-loop deblocking.

The filtering strength is categorized into 3 classes according to the value of computed Bs, i.e.,

$$\texttt{Bs} = \begin{cases} 4 & \text{Strong Filtering} \\ 1,2,3 & \text{Normal Filtering} \\ 0 & \text{No Filtering.} \end{cases}$$

Typically, for intra-coded blocks, Bs takes value of 4 and 3, while Bs is 0, 1, 2, or 3 for inter-coded blocks (including skip mode). Therefore, the total complexity can be written as

$$C_{\texttt{dblk}} = k_{\texttt{intraMB,dblk}} \cdot n_{\texttt{intraMB}} + k_{\texttt{interMB,dblk}} \cdot n_{\texttt{interMB}}, \tag{4.21}$$

where $k_{\texttt{intraMB,dblk}}$ and $k_{\texttt{interMB,dblk}}$ are the average complexity required for filtering intra-coded and inter-coded macroblocks, respectively, and $n_{\texttt{intraMB}}$ and $n_{\texttt{interMB}}$ are the number of intra and inter coded macroblock, respectively. Since we have embedded $n_{\texttt{intraMB}}$ for intra prediction, $n_{\texttt{interMB}}$ is a known parameter given a frame resolution, i.e., $n_{\texttt{interMB}} = n_{\texttt{MB}} - n_{\texttt{intraMB}}$. For intra-coded block, because only $\texttt{Bs} = 4$ and 3 are employed, $k_{\texttt{intraMB,dblk}}$ is relatively constant; while for inter coded macroblock, $k_{\texttt{interMB,dblk}}$ varies with respect to the percentage of the zero Bs edges. In addition to the boundary strength control, edge-crossing pixel difference is also used to determine whether to do filtering. Thus, even with non-zero Bs, edge filtering can be also skipped for crossing pixels as defined in the standard [1]. Therefore, there is large variation in $k_{\texttt{interMB,dblk}}$. Also, because of the non-stationarity video content distribution, the deblocking complexity is hard to predict with a reasonable accuracy based on our experimental data, and Eq. (4.21) is not a good model for estimation.

As defined in [1], different Bs leads to different filtering operations on edge crossing pixels, e.g., p_i and q_j with $i \in \{0,1,2,3\}$. Typically, for $\texttt{Bs} = 0$, there is no filter applied.

For $\texttt{Bs} = 4$, the strongest filter will be employed, which uses all pixels, i.e., $p_i,\ q_i, i = \{0, 1, 2, 3\}$ to modify p_i and q_i with $i = 0, 1$ and 2, as depicted in Figure 4.17(a). For $\texttt{Bs} =$ 3, 2 or 1, six edge-crossing pixels, i.e., p_i and q_i with $i = 0, 1, 2$ are used to update the p_i and q_i with $i = 0, 1$. In addition to the Bs, we also need to calculate the difference of edge-crossing pixels for a certain pixel line, such as

$$\delta_\alpha = |p_0 - q_0|,$$
$$\delta_\beta = |p_0 - p_1|, \quad \text{or} \quad |q_0 - q_1|.$$

If δ_α and δ_β are less than predetermined Alpha and Beta thresholds defined in [1], filtering operations corresponding to Bs are applied; Otherwise, the deblocking is skipped even with non-zero Bs.

Table 4.8: Correlation coefficients for n_α and n_β

Seq.	Harbour	Ice	News	Soccer	ave.
Corr. coeff.	0.985	0.995	0.992	0.990	0.991

For simplicity, we define α-points (i.e., $\alpha_{\texttt{pts}}$) and β-points (i.e., $\beta_{\texttt{pts}}$) to categorize all edge-crossing pixels as depicted in Figure 4.17(a) which are required to do filtering, i.e.,

$$\alpha_{\texttt{pts}} = \{p_0, q_0\}$$
$$\beta_{\texttt{pts}} = \{p_1, q_1, p_2, q_2\}.$$

Thus, the deblocking complexity is the sum of cycles dissipated among α-points and β-points,

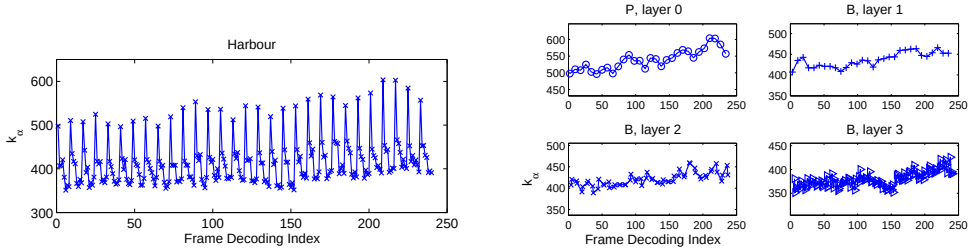
$$C_{\texttt{dblk}} = k'_\alpha n_\alpha + k_\beta n_\beta, \tag{4.22}$$

where k'_α and k_β are the average cycles required to do α-point and β-point filtering, and n_α and n_β are the numbers of respective α-point and β-point per frame. We have found from our experimental data the decision to filter β-point is highly correlated the decision to filter α-point, i.e., once α-point requires filtering operations, the corresponding β-points

will also be filtered with very high probability (i.e., > 0.99 on average as exemplified in Table 4.8). Thus, Eq. (4.22) can be reduced to

$$C_{\texttt{dblk}} = (k'_\alpha + k_\beta \cdot \kappa)\, n_\alpha \stackrel{\text{def}}{=} k_\alpha \cdot n_\alpha, \tag{4.23}$$

with k_α denoting the generalized average complexity for filtering α-points.



(a) (b)

Figure 4.18: Illustration of k_α in frame decoding order for Intel PM platform: (a) overall sequence decoding (b)frame decomposition for different layers.

Typically, k_α varies from frame to frame due to the content adaptive tools used in conducting deblocking filtering. We have found that k_α changes slowly from frame to frame in the same temporal layer as illustrated in Figure 4.18(b). As with complexity modeling for entropy decoding, instead of using a fixed k_α, we predict k_α of the current frame using previous frame deblocking complexity in the same layer and the embedded number of α-points. Figure 4.19 demonstrates that the proposed model in Eq. (4.23) and method for predicting k_α can accurately predict the deblocking complexity.

The Overall Frame-level Complexity Model

From the above discussion, we conclude that each DM complexity can be abstracted as the product of its CU complexity and the number of involved CU. The total complexity required by frame decoding can be expressed as

$$C_{\texttt{frame}} = \sum\nolimits_{\texttt{DM}} C_{\texttt{DM}} = \sum\nolimits_{\texttt{DM}} k_{\texttt{CU(DM)}} \cdot N_{\texttt{CU(DM)}}, \tag{4.24}$$

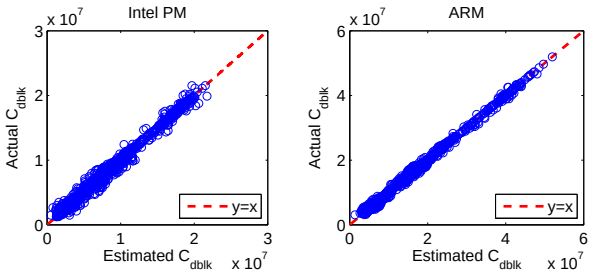


Figure 4.19: Actual deblocking complexity against estimated complexity for both Intel PM and ARM processors.

where $k_{\mathtt{CU(DM)}}$ indicates the complexity of the CU for a particular DM, and $N_{\mathtt{CU(DM)}}$ is the number of CUs involved in a DM. Table 4.9 lists the CU for each DM, and its corresponding abstraction. We assume that the number of CUs in each frame for different CUs, i.e., $N_{\mathtt{CU}}$, can be embedded into the video bitstream packets as the metadata to conduct decoding complexity estimation. For example, one can packetize the raw H.264/AVC bitstream into a popular container, e.g., FLV, and put $N_{\mathtt{CU}}$ information in the container header field. Note that $n_{\mathtt{bit}}$ and $n_{\mathtt{MB}}$ are not embedded since they can be obtained by de-packetizing the NAL unit of H.264/AVC bitstream, parsing the sequence and picture parameter sets before frame decoding. Therefore, only four numbers, $n_{\mathtt{nzMB}}$, $n_{\mathtt{intraMB}}$, $n_{\mathtt{half}}$ and n_{α}, need to be embedded using 8 bytes. Even for videos coded at the very low bit rate of 96kbps, this embedded overhead only counts for 1.5% of the video bit rate. For GOP-level complexity prediction (see Section 4.4.3), the overhead is even smaller. As for the CU complexity, i.e., $k_{\mathtt{CU}}$, as shown in the previous subsections, for a given implementation platform, it is a constant for some CU, whereas for some other CU (i.e., bit parsing, SIP and α-point filtering), it needs to be predicted from the measured complexity of the previous frame in the same temporal layer. In practice, we can set the initial $k_{\mathtt{CU}}$ to some default values for decoding the first frame. Alternatively, we can pre-decode one frame in each temporal layer (or one GOP for GOP model) to obtain the specific $k_{\mathtt{CU}}$ of each involved CU ahead of real video playback. Once we initialize all $k_{\mathtt{CU}}$ for a target platform, we update them automatically frame by frame according to the actual DM complexity and number of involved CU (i.e., $N_{\mathtt{CU}}$) of the previous decoded frame. Table 4.9 summarizes whether a $k_{\mathtt{CU}}$ is a constant or needs

prediction. The constant $k_{\texttt{CU}}$ is further listed in Table 4.10 for Intel and ARM processors.

Table 4.9: CU abstraction for each DM

DM	CU	$k_{\texttt{CU}}$	$N_{\texttt{CU}}$ (over a frame or GOP)
vld	Bit parsing	$k_{\texttt{bit}}$: predicted	$n_{\texttt{bit}}$: # of total bits
sip	side info. preparation	$k_{\texttt{MBsip}}$: predicted	$n_{\texttt{MB}}$: # of total MBs
dblk	α-point deblocking	k_{α}: predicted	n_{α} : # of α-points
itrans	MB dequant. & IDCT	$k_{\texttt{MBitrans}}$: constant	$n_{\texttt{nzMB}}$: # of non-zero MBs
intra	MB intra prediction	$k_{\texttt{intraMB}}$: constant	$n_{\texttt{intraMB}}$: # of intra MBs
mcp	Half-pixel interpolation	$k_{\texttt{half}}$: constant	$n_{\texttt{half}}$: # of half-pel interp.

Table 4.10: Constant $k_{\texttt{CU}}$ for Intel PM and ARM processors (in terms of CPU clock cycle)

	$k_{\texttt{MBitrans}}$	$k_{\texttt{intraMB}}$	$k_{\texttt{half}}$
Intel PM	7.2×10^4	7×10^3	1.1×10^3
ARM	1.7×10^5	1.2×10^4	2.7×10^3

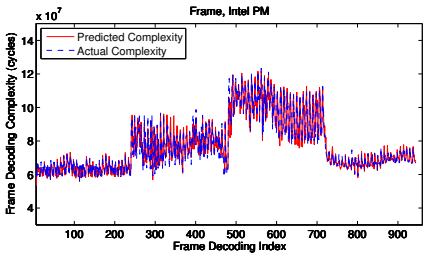
To verify the accuracy of this estimation strategy, we collect the actual and predicted frame decoding complexity for all four test videos with QP ranging from 10 to 44., and calculate their prediction error. Let $\delta(i)$ denote the relative prediction error for frame i. We calculate the mean and standard deviation (STD) of $\delta(i)$ over all frames and over all sequences coded using different QPs, as a measure of the prediction accuracy. As shown in the simulation results listed in Table 4.12, the prediction error is very small, with small mean and STD (i.e., both less than 3% on average). We also present the predicted and actual frame complexity in decoding order for the concatenated video consisting of "News", "Soccer", "Harbour" and "Ice" in Figure 4.20(a-b) at QP 24. Results for other QPs are similar according to our experiments. Based on these results, our proposed model can estimate the frame decoding complexity for the H.264/AVC video decoding very well.

Table 4.11: Rate control Configuration

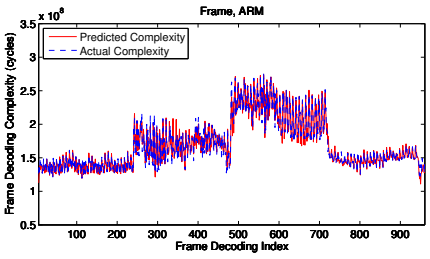
resolution	# frame	bit rate (kbps)
QCIF (176×144)	1100	250
CIF (352×288)	1100	500
4CIF (704×576)	1400	1000

Table 4.12: Normalized Prediction Error (mean μ and standard deviation σ) for Intel PM and ARM platform

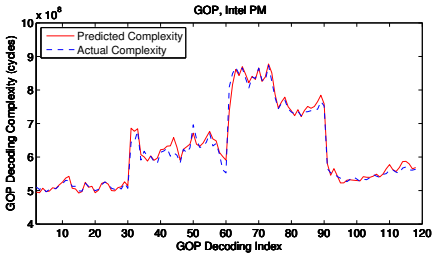
Seq.	Intel PM				ARM			
	Frame		GOP		Frame		GOP	
	μ	σ	μ	σ	μ	σ	μ	σ
Harbour	3.22%	3.20%	1.37%	1.21%	3.30%	3.33%	1.28%	1.19%
Ice	2.00%	1.68%	1.20%	0.88%	2.38%	2.36%	1.84%	1.65%
News	1.36%	1.17%	1.39%	1.11%	1.32%	1.11%	1.34%	1.11%
Soccer	3.32%	2.98%	2.92%	2.58%	3.62%	3.22%	2.97%	2.71%
ave.	2.48%	2.26%	1.72%	1.45%	2.66%	2.51%	1.86%	1.65%



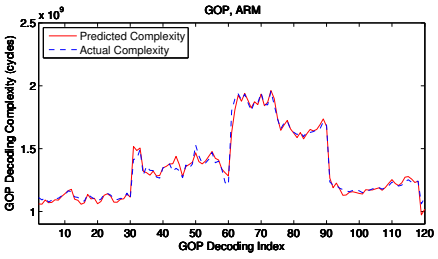
(a) $\mu = 2.37\%, \sigma = 2.36\%$



(b) $\mu = 2.40\%, \sigma = 2.49\%$

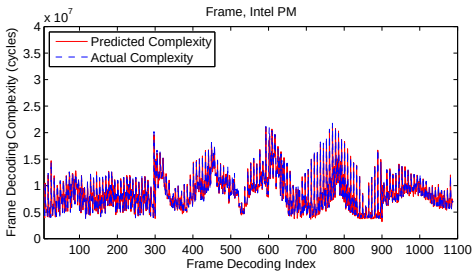


(c) $\mu = 1.85\%, \sigma = 1.80\%$

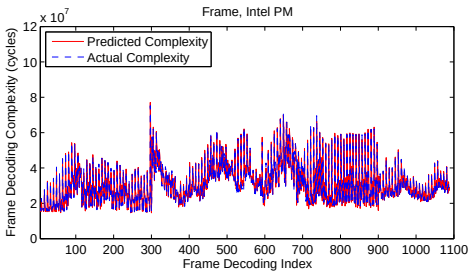


(d) $\mu = 1.66\%, \sigma = 1.89\%$

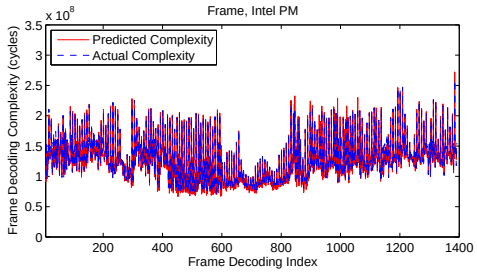
Figure 4.20: Illustration of predicted and actual profiled complexity (in terms of cycles) of concatenated sequences (in the order of "News", "Soccer", "Harbour" and "Ice") at QP 24 for frame and GOP-level respectively.



(a) QCIF@250bps, $\mu = 2.50\%, \sigma = 2.20\%$



(b) CIF@500kbps, $\mu = 3.00\%, \sigma = 2.90\%$



(c) 4CIF@1Mbps, $\mu = 3.10\%, \sigma = 2.30\%$

Figure 4.21: Illustration of predicted and actual profiled complexity (in terms of cycles) of different resolution concatenated sequences using rate control for frame-level complexity model.

Performance under Rate Control and Different Spatial Resolutions

The results reported so far are for decoding videos coded using constant QP, and at the CIF resolution. To verify the accuracy of the complexity model for videos coded under variable QP (due to rate control) and other spatial resolutions, we also created bitstreams using the JSVM [22] for three resolutions, QCIF, CIF and 4CIF. As before, we concatenate 4 different videos to form a test sequence under each resolution. For QCIF and CIF resolution, we use the videos in the order of "News", "Soccer","Harbour", "Ice" while the 4CIF resolution sequence is the concatenation of "Soccer", "Harbour","Ice", "Crew" and "City" [10]. Table 4.11 gives the sequence length and bit rate setting for QCIF, CIF and 4CIF respectively.

As shown in Figure 4.21, our complexity model can accurately predict the decoding complexity for different videos with various content activities, at different resolutions and bit rates. Because of the space limit, we choose to present the results on the Intel platform only. ARM based simulations have the similar high accuracy under rate control and different spatial resolution.

4.4.3 GOP-level H.264/AVC Decoding Complexity Model

As shown in the previous section, the proposed model can predict the decoding complexity for each video frame with a high accuracy, assuming that the number of CUs required for each DM of each frame, $N_{\mathtt{CU}}$, can be embedded in the bit stream, and that the decoding complexity for each DM can be measured for each decoded frame and used to predict the $k_{\mathtt{CU}}$ for the next frame in the same temporal layer. Here, we extend the complexity model from frame-level to GOP-level, and show that the same model still works well, where $\hat{N}_{\mathtt{CU}}$ now denotes the number of CUs required for each DM over each GOP, and decoding complexity for each DM over the entire GOP can be measured and used to predict the $k_{\mathtt{CU}}$ for the next GOP. Let $C_{\mathtt{gop}}$ describe the complexity dissipated for decoding

[10] We don't have "News" video at 4CIF resolution.

a GOP, it can be written as

$$C_{\mathtt{gop}} = \sum\nolimits_j C_{\mathtt{frame}}(j) = \sum\nolimits_j \sum\nolimits_{\mathtt{CU}} k_{\mathtt{CU}}(j) N_{\mathtt{CU}}(j)$$
$$= \sum\nolimits_{\mathtt{CU}} k_{\mathtt{CU}} \sum\nolimits_j N_{\mathtt{CU}}(j) = \sum\nolimits_{\mathtt{CU}} k_{\mathtt{CU}} \hat{N}_{\mathtt{CU}}, \tag{4.25}$$

where $\hat{N}_{\mathtt{CU}} = \sum_j N_{\mathtt{CU}}(j)$ is the number of a particular CU over the entire GOP, such as the number of the intra macroblock per GOP, GOP bits number, etc. In practice, even when $k_{\mathtt{CU}}(j)$ varies from frame to frame, we can use $k_{\mathtt{CU}}$ to denote the average CU complexity across a GOP and use (4.25) to estimate the total complexity of a GOP. Like what we have proposed in frame based model, $k_{\mathtt{CU}}$ will be updated GOP by GOP using the previous GOP complexity data. Similarly, we assume $\hat{N}_{\mathtt{CU}}$ can be embedded into the packetized stream at the GOP header.

To validate our above proposal, we plot the measured cycles consumed by GOP decoding and estimate complexity using our proposed model for the four test videos at QP 24 on both Intel PM and ARM platforms in Figure 4.20(c-d). These figures show that the GOP level prediction works very well. We also provide the mean and standard deviation for the GOP-level complexity prediction error in Table 4.12. Note that the GOP-level prediction improves the accuracy compared to the frame-level model according to the results listed in Table 4.12 and pictured in Figure 4.20. This is because the average CU complexity over a GOP varies more slowly than that over a frame, and hence the prediction of $k_{\mathtt{CU}}$ at the GOP level is more accurate.

Compared with frame based complexity prediction, GOP level complexity model only needs to store the metadata at GOP level instead of frame level, thus reduces the overhead. Also, for dynamic voltage/frequency scaling, we only need to adjust the voltage/frequency at the beginning of every GOP instead of every frame.

4.5 Discussion and Summary

In summary, the work presented in this Chapter consists of two parts: in the first half, we develop a complexity model for scalable vide decoding considering the tempo-

ral and amplitude variations, i.e., $C(q,t)$. Three content dependent parameters, i.e., $C_{\max}$, g_1 and g_2, will be further discussed in the subsequent chapter. We then extend the complexity model $C(q,t)$ to the power consumption model for the ARM processor. Since the ARM processor is widely used in SmartPhones, mobile Pads, etc, our proposed power consumption model for scalable video decoding can be applied practically. Together with the quality model and rate model in Section 2.3.3 and 3.2.3, we can conduct the power-rate constrained scalable video adaptation for a typical mobile video streaming scenario, where network condition and battery power capacity are both constrained in practice. Please refer to the following Chapter 6 for more details regarding employing our proposed models in real applications.

The second part deals with the complexity prediction along with the video decoding. The overall video decoder is decomposed into 6 decoding modules (DM), each of which is controlled by a unique *complexity unit* (CU). We have defined those CUs to ensure that its average complexity (i.e., per frame or per GOP) is either constant or can be easily predicted from previous decoded data. In the current study, we embed the number of CUs, i.e, $N_{\texttt{CU}}$, as the metadata in the header field of the container (such as FLV, MKV, etc) associated with the video bitstream. According to our simulation data, our proposed complexity prediction algorithm can accurately estimate the frame or GOP decoding complexity for various videos with different contents, resolutions and bitrates. Our proposed prediction algorithm can be used to dynamically adapt the voltage and frequency of the underlying processor (DVFS) so as to save the power. More details regarding the complexity prediction based DVFS will be presented in Chapter 6.

Chapter 5

Model Parameter Prediction

As shown in previous chapters, our model parameters are video content dependent. The models will be very useful if the model parameters can be accurately predicted from some content features. In this chapter, we investigate the video content features, such as frame difference (FD), displaced frame difference (DFD), motion activities, etc, and use linear weighted features to estimate the parameters for our proposed models. We first explore and describe various features parameter prediction. We then present the stepwise feature selection approach for selecting a subset of features that minimize the cross-validation error for all test sequences.

5.1 Video Content Feature

As shown in previous chapters, we have applied an "impact separation" methodology to differentiate the joint temporal and amplitude impact on perceptual quality, rate and power consumption (or complexity). Each model is well explained by the product of a function in terms of the frame rate, and a function in terms of the quantization stepsize. Overall, we have eight content dependent parameters as listed in Table 2.1, 3.1 and 4.2. As shown in the experimental results, the parameters are indeed content dependent. To well explore the correlation between content features and model parameters, we analyze all model parameters, and establish the general relationship between them and contents. For example, $R_{\max}$

is the maximum bit rate, which is mainly composed of the bits for encoding motion vectors and residuals, therefore, $R_{\max}$ should be a function of the motion vector and residual features. In H.264/AVC, the motion and residual information have the direct impact on the decoding complexity [39]. Whether to do interpolation or deblocking is highly depending on the motion vector (or motion vector difference), i.e., interpolation will be skipped if the block has the integer motion vector, and the deblocking filter will be disabled if the motion vector difference of neighboring blocks is smaller than a predefined threshold. Roughly speaking, $C_{\max}$ – maximum cycles required for decoding full-resolution bitstream, can be estimated as a function of motion and residual as well. Overall, we correlate the model parameters with motions, residual signal, video frame contrast, etc. Specifically, we have defined three input signals to derive the features, one is the *residual (error) signal*, such as frame difference (FD), displaced frame difference (DFD), etc; the second one is the *original signal*; and the third one is the *motion fields*. A set of content features is trained and built upon those three inputs, as detailed in Table 5.1. In general, this set of content features consists of two subsets. One includes the original features derived from raw input sources. The other contains the inter-normalized features using the prior subset. More details regarding how to derive the value of the individual feature will be presented in the subsequent section.

Table 5.1: List of content features in consideration

input source	feature	
original features		
residual	frame difference (FD)	$\mu_{\texttt{FD}}$, $\sigma_{\texttt{FD}}$
	displaced frame difference (DFD)	$\mu_{\texttt{DFD}}$, $\sigma_{\texttt{DFD}}$
motion	motion vector magnitude (MVM)	$\mu_{\texttt{MVM}}$, $\sigma_{\texttt{MVM}}$
	motion direction activity (MDA)	$\sigma_{\texttt{MDA}}$
original	video contrast	$\sigma_{\texttt{org}}$
inter-normalized features		
	normalized FD (NFD) by contrast	$\eta(\mu_{\texttt{FD}}, \sigma_{\texttt{org}}) = \mu_{\texttt{FD}}/\sigma_{\texttt{org}}$
	normalized DFD (NDFD) by contrast	$\eta(\mu_{\texttt{DFD}}, \sigma_{\texttt{org}}) = \mu_{\texttt{DFD}}/\sigma_{\texttt{org}}$
	normalized MVM by contrast	$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{org}}) = \mu_{\texttt{MVM}}/\sigma_{\texttt{org}}$
	normalized MVM by $\sigma_{\texttt{MVM}}$	$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MVM}}) = \mu_{\texttt{MVM}}/\sigma_{\texttt{MVM}}$
	normalized MVM by $\sigma_{\texttt{MDA}}$	$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MDA}}) = \mu_{\texttt{MVM}}/\sigma_{\texttt{MDA}}$

5.2 Feature Extraction Preprocessor

In [10], we extract features by enabling all possible encoding features in H.264/AVC, such as fractional accuracy interpolation, variable block size motion compensation, etc. These features are accurate but not practical in real applications. Alternatively, we propose to use a simple, lightweight pre-processor ahead of real encoding to obtain the features. In this pre-processor, we apply a simplified motion compensation engine to obtain the DFD and motion fields. To reduce the complexity, only macroblock based (i.e., 16x16) integer motion estimation is used in our preprocessor. Besides, we apply the pre-processor upon the original raw video signal in order to avoid the coding effects, such as quantization. For a N-picture video source, we would have frame difference, displaced frame difference, motion field signal with the length of N-1 pictures after applying the pre-processor frame-by-frame [1]. For any i-th FD picture, we can calculate its mean and standard deviation, i.e., $\mu_{\mathtt{FD}}(i)$ and $\sigma_{\mathtt{FD}}(i)$, then, the mean and standard deviation for whole sequence can be obtained via

$$\mu_{\mathtt{FD}} = \frac{1}{N-1}\sum_{i=1}^{N-1}\mu_{\mathtt{FD}}(i), \tag{5.1}$$

$$\sigma_{\mathtt{FD}} = \frac{1}{N-1}\sum_{i=1}^{N-1}\sigma_{\mathtt{FD}}(i). \tag{5.2}$$

The same method can be applied to obtain the mean and standard deviation of the DFD signal. Usually, we will have a pair of value, i.e., $(\mathtt{mv}_x, \mathtt{mv}_y)$ for any motion vector, then the motion vector magnitude and direction can be defined as

$$|\mathtt{mv}| = \sqrt{|\mathtt{mv}_x|^2 + |\mathtt{mv}_y|^2}, \tag{5.3}$$

$$\theta_{\mathtt{mv}} = \arctan(\mathtt{mv}_y/\mathtt{mv}_x). \tag{5.4}$$

Similarly, we can calculate the mean and standard deviation of the MVM and standard deviation of the MDA using the same method discussed above [2] Please note that, we exclude the zero motions, i.e., $(\mathtt{mv}_x, \mathtt{mv}_y) = (0, 0)$, to calculate the MDA feature.

[1] As known, the first frame is skipped without creating its FD, DFD, motion signals since there isn't reference frame for the first frame within a video sequence.

[2] The standard deviation of the MVM is the same as the motion activity intensity (MAI) defined in [10].

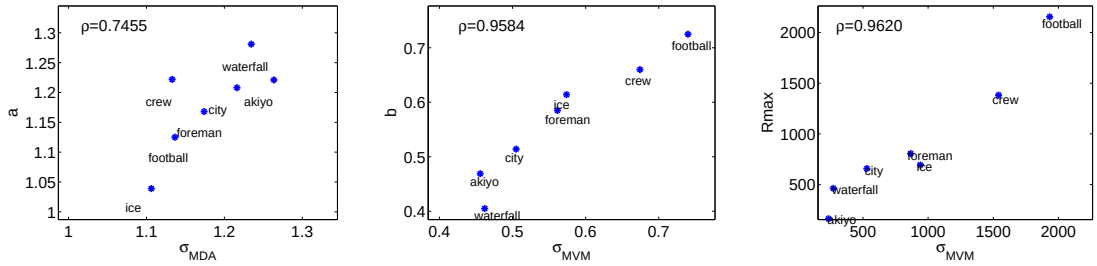
5.3 Model Parameter Prediction

We have defined a set of features for model parameter prediction as shown in earlier sections. In this section, we present the parameter prediction using the trained feature and show the prediction accuracy. According to our experimental data, we have found that a single feature can not give an accurate estimation, thus two or three features are combined together and their weighted sum is used to predict the model parameter, i.e., $\sum_k \omega_k F_k + \omega_0, k = 1, 2, \ldots, K$, where ω_k indicates the individual weighting coefficient for k-th feature F_k and K is the total number of features examined.

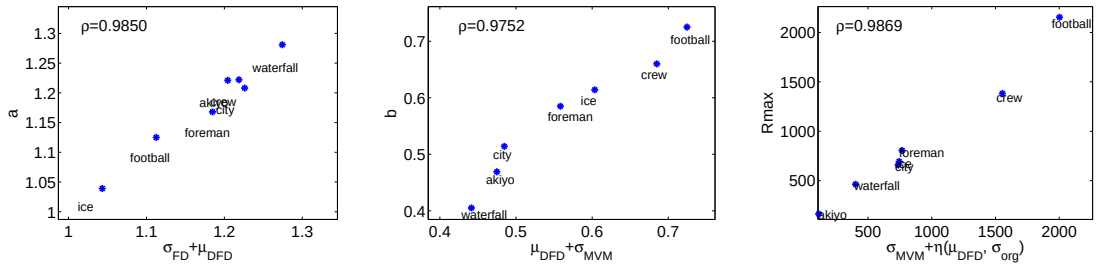
Moreover, we apply the cross-validation to choose the best features and their weighted coefficients. Specifically, we first choose a number of features, for example, two features are selected out of the feature set together instead of using the stepwise feedforward approach proposed in [10]. In order for the solution to be generalizable to other sequences outside our test sequences, we use the leave-one-out cross-validation error (CVE) criterion. Assume the total number of sequences is M, for a particular set of chosen features, we arbitrarily set one sequence as the test sequence and the remaining $M-1$ sequences as the training sequences. We determine the weights ω_k to minimize the mean square fitting error for the training sequences. We then evaluate the square fitting error for the test sequence. We repeat this process, each time using a different sequence as the test sequence. The average of the fitting errors for all the test sequences is the CVE associated with this feature set. The best features and weighting coefficients are chosen according to the minimum CVE. By using above procedure, we have examined all model parameters using one-feature, two-feature and three-feature combined prediction, as shown in Figures 5.1, 5.2, 5.3 for CIF resolution videos [3]. Table 5.2, 5.4 and 5.3 list the weighted coefficients, i.e., ω_k for CIF videos.

As shown, one-feature prediction doesn't work well for CIF resolution videos. Some parameters have been predicted with high accuracy by using two features, such as a, $R_{\max}$, $C_{\max}$; some parameters require three-feature prediction, such as c, d, g_1, g_2, etc. It is

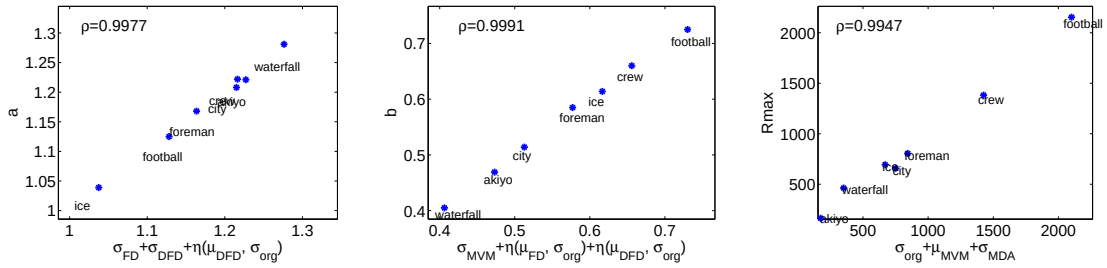
[3] Since we only have 7 test sequences for quality model, we have presented 7 CIF videos for all model related parameters.



(a) one feature

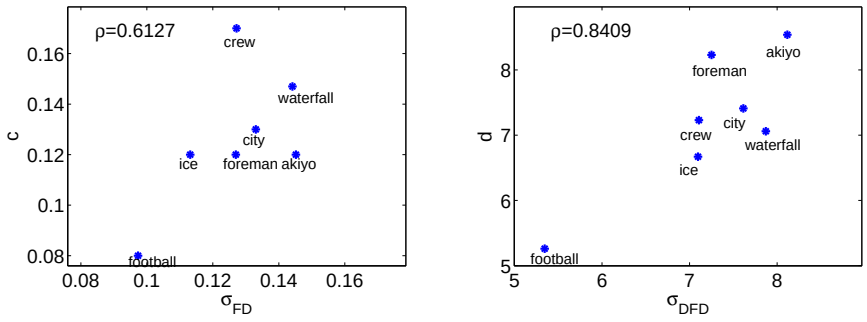


(b) two feature

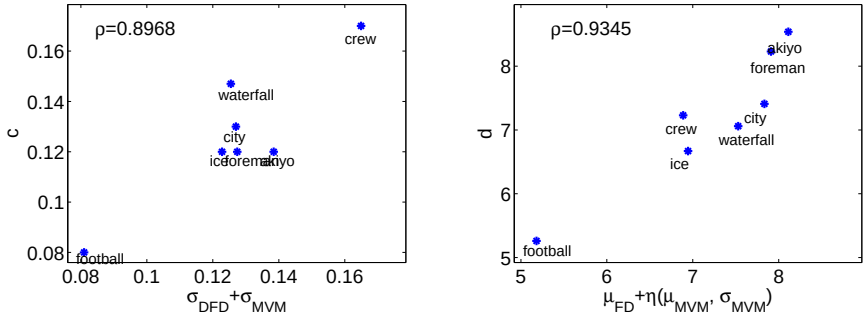


(c) three feature

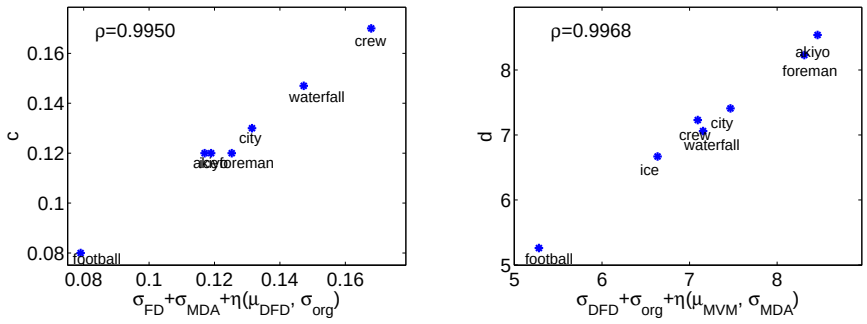
Figure 5.1: Illustration of rate model parameter prediction, i.e., $a, b, R_{\max}$ using content features for CIF resolution videos.



(a) one feature

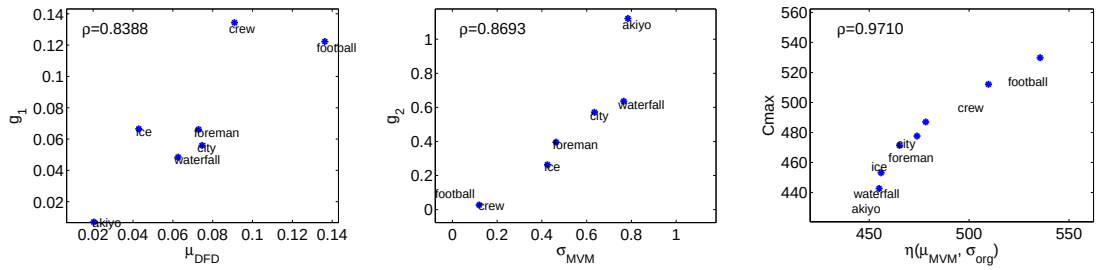


(b) two feature

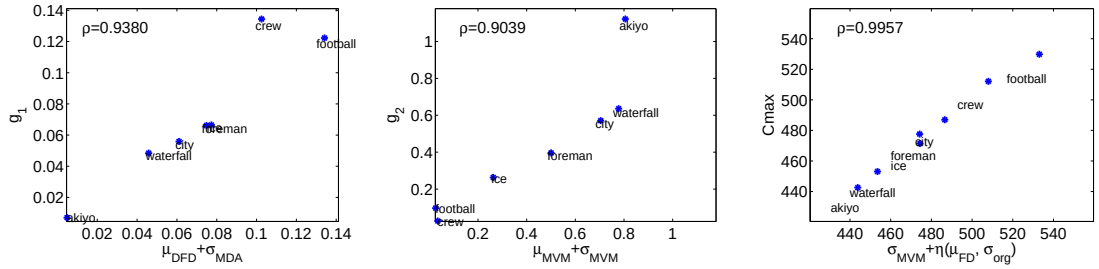


(c) three feature

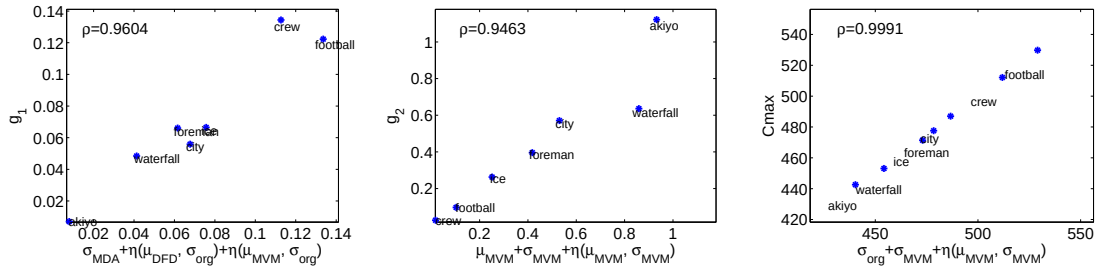
Figure 5.2: Illustration of quality model parameter prediction, i.e., c, d, using content features for CIF resolution videos.



(a) one feature



(b) two feature



(c) three feature

Figure 5.3: Illustration of complexity model parameter prediction, i.e., $g_1, g_2, C_{\max}$ using content features for CIF resolution videos.

noted that the PC will be improve when we choose more features, however, to reduce the complexity, we limit 3 as the maximum number of features applied for CIF videos. Let $\mathbf{Y}_{\texttt{CIF}} = [a, b, R_{\max}, c, d, g_1, g_2, C_{\max}]$ be the predicted parameter set and $\mathbf{X}$ be the feature set, according to our simulation data, we can obtain the feature set and corresponding weighting matrix for different feature combination cases. For instance, we can derive the feature set for CIF videos with one-feature, i.e.,

$$\mathbf{X}_{\texttt{CIF},1} = \left[\sigma_{\texttt{MDA}}, \sigma_{\texttt{MVM}}, \sigma_{\texttt{FD}}, \sigma_{\texttt{DFD}}, \mu_{\texttt{DFD}}, \eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{org}})\right], \tag{5.5}$$

where $\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{org}})$ is the $\mu_{\texttt{MVM}}$ normalized by the original video contrast $\sigma_{\texttt{org}}$. Thus, it requires seven original features, i.e., $\sigma_{\texttt{MDA}}, \sigma_{\texttt{MVM}}, \sigma_{\texttt{FD}}, \sigma_{\texttt{DFD}}, \mu_{\texttt{DFD}}, \mu_{\texttt{MVM}}, \sigma_{\texttt{org}}$, then, we derive the inter-normalized feature $\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{org}})$ to construct the feature set (5.5).

Additionally, results for WVGA and 720p videos are presented as well from Figure 5.4 to Figure 5.7. Since we only conduct the experiments on CIF videos to build the quality model, we only show the results regarding the prediction of rate and complexity model parameters. Similarly, we choose different combination of features to show the accuracy of the parameter prediction. It is noted that two-feature combined prediction is sufficient to produce accurate estimation. Table 5.5, 5.6, 5.7 and 5.8 list the weighted coefficients for best selected features. According to the simulation results, we have found that even for the same model parameter, we should use different feature combinations to estimate for different video resolutions. Moreover, a stable weighted coefficient matrix and feature set should be trained via a large amount of test videos with various content activities. Based on our experiments, our developed feature set and matrix can be applied to the similar video content as our test sequences. Since the chosen test sequences already cover a large range of content activities, we believe that our result can be applied widely (but of course not applicable to every video content).

5.4 Model Evaluation Using Predicted Parameters

In addition to presenting the above parameter prediction results (i.e., Figure 5.1 to 5.7), we also plug these predicted parameters into our proposed models, and verify the estimation

Table 5.2: Weighted coefficients matrix for model parameters: one feature, CIF video

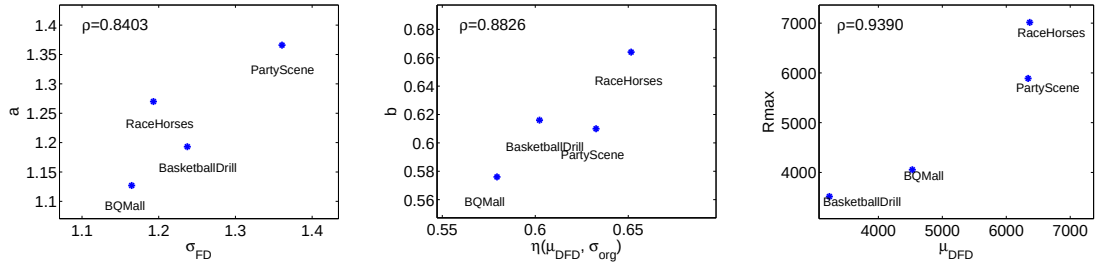
	$\sigma_{\texttt{MDA}}$	$\sigma_{\texttt{MVM}}$	$\sigma_{\texttt{FD}}$	$\sigma_{\texttt{DFD}}$	$\mu_{\texttt{DFD}}$	$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{org}})$	ω_0
a	-0.15						1.31
b		0.05					0.45
$R_{\max}$		322.24					182.63
c			-3.7×10^{-3}				0.15
d				-0.5			8.63
g_1					0.02		0.01
g_2		-0.16					0.81
$C_{\max}$						378.83	454.69

Table 5.3: Weighted coefficients matrix for model parameters: two feature, CIF video

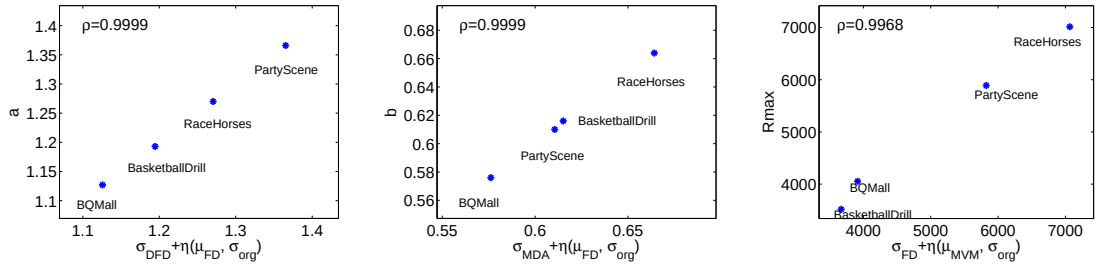
	a	b	$R_{\max}$	c	d	g_1	g_2	$C_{\max}$
ω_0	1.22	0.47	36.40	0.17	7.85	-0.02	0.85	441.01
$\sigma_{\texttt{FD}}$	-0.02							
$\sigma_{\texttt{DFD}}$				-0.03				
$\sigma_{\texttt{MVM}}$		0.07	245.43	0.022			-0.31	8.79
$\sigma_{\texttt{MDA}}$						0.05		
$\mu_{\texttt{FD}}$					-0.35			
$\mu_{\texttt{DFD}}$	0.03	-0.02				0.012		
$\mu_{\texttt{MVM}}$							0.11	
$\eta(\mu_{\texttt{FD}}, \sigma_{\texttt{org}})$								94.64
$\eta(\mu_{\texttt{DFD}}, \sigma_{\texttt{org}})$			2932.90					
$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MVM}})$					2.32			

Table 5.4: Weighted coefficients matrix for model parameters: three feature, CIF video

	a	b	$R_{\max}$	c	d	g_1	g_2	$C_{\max}$
ω_0	1.22	0.47	465.78	0.11	6.89	-0.01	1.05	462.89
$\sigma_{\texttt{FD}}$	-0.03			-0.01				
$\sigma_{\texttt{DFD}}$	0.05				-0.95			
$\sigma_{\texttt{MVM}}$		0.06					-0.38	11.48
$\sigma_{\texttt{MDA}}$			190.78	0.09		0.05		
$\sigma_{\texttt{org}}$			-7.27		0.05			-0.58
$\mu_{\texttt{MVM}}$			218.28				0.20	
$\eta(\mu_{\texttt{FD}}, \sigma_{\texttt{org}})$		0.43						
$\eta(\mu_{\texttt{DFD}}, \sigma_{\texttt{org}})$	0.39	-1.26		0.17		0.25		
$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MVM}})$							-0.29	
$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MDA}})$					0.43			
$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{org}})$						0.13		17.49



(a) one feature

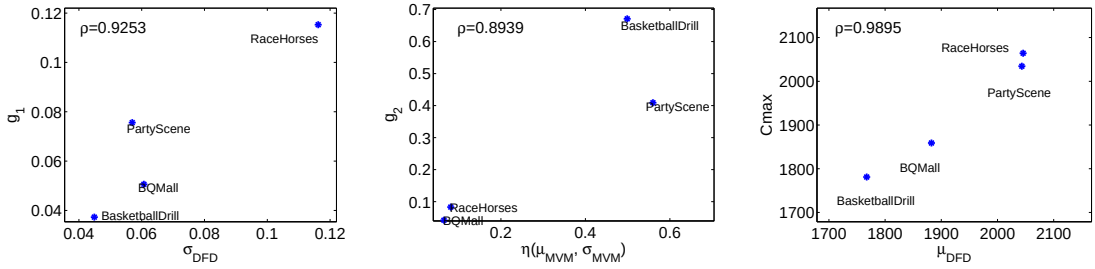


(b) two feature

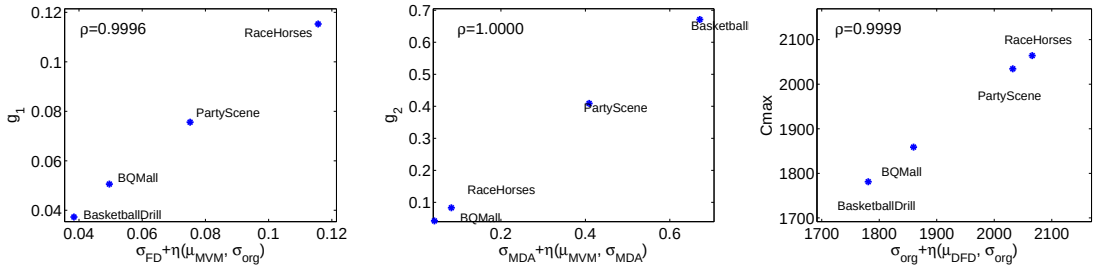
Figure 5.4: Illustration of rate model parameter prediction, i.e., $a, b, R_{\max}$ using content features for WVGA resolution videos.

Table 5.5: Weighted coefficients matrix for model parameters: one feature, WVGA video

	a	b	$R_{\max}$	g_1	g_2	$C_{\max}$
ω_0	1.48	0.46	-939.42	-0.18	0.72	1396.40
$\sigma_{\mathtt{FD}}$	-0.02					
$\sigma_{\mathtt{DFD}}$				0.06		
$\mu_{\mathtt{DFD}}$			1329.50			118.13
$\eta(\mu_{\mathtt{DFD}}, \sigma_{\mathtt{org}})$		1.36				
$\eta(\mu_{\mathtt{MVM}}, \sigma_{\mathtt{MVM}})$					-0.48	



(a) one feature

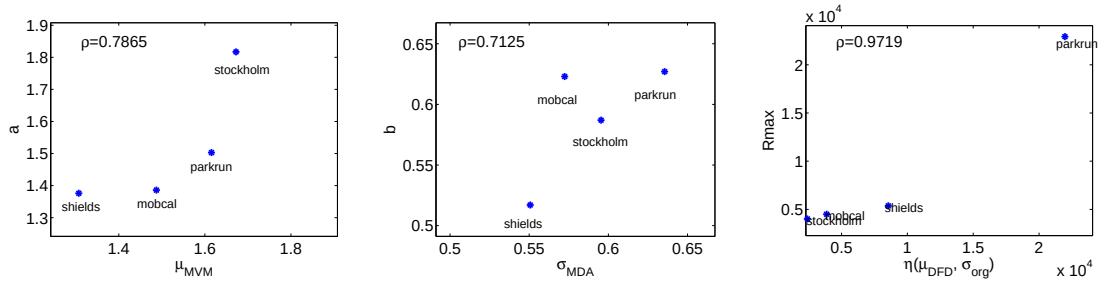


(b) two feature

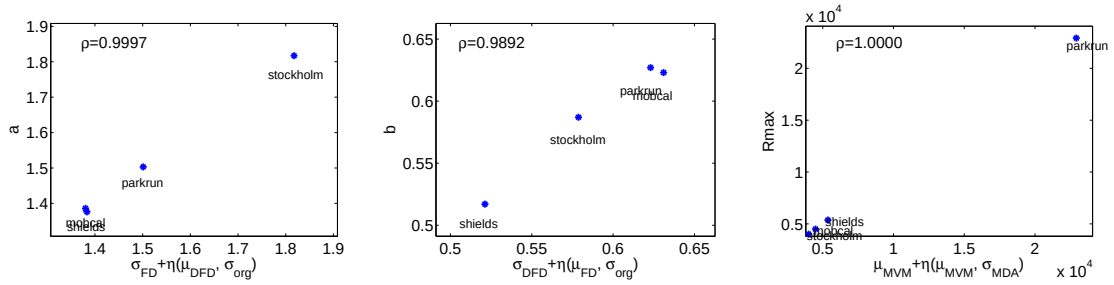
Figure 5.5: Illustration of complexity model parameter prediction, i.e., $g_1, g_2, C_{\max}$ using content features for WVGA resolution videos.

Table 5.6: Weighted coefficients matrix for model parameters: two feature, WVGA video

	a	b	$R_{\max}$	g_1	g_2	$C_{\max}$
ω_0	0.51	0.73	9419.70	0.14	-4.70	922.06
$\sigma_{\mathtt{FD}}$			-785.21	-0.02		
$\sigma_{\mathtt{DFD}}$	0.28					
$\sigma_{\mathtt{MDA}}$		-0.50			10.49	
$\sigma_{\mathtt{org}}$						9.80
$\eta(\mu_{\mathtt{FD}}, \sigma_{\mathtt{org}})$	-1.56	1.50				
$\eta(\mu_{\mathtt{DFD}}, \sigma_{\mathtt{org}})$						5447.80
$\eta(\mu_{\mathtt{MVM}}, \sigma_{\mathtt{MDA}})$					-1.66	
$\eta(\mu_{\mathtt{MVM}}, \sigma_{\mathtt{org}})$			53810.00	1.17		



(a) one feature

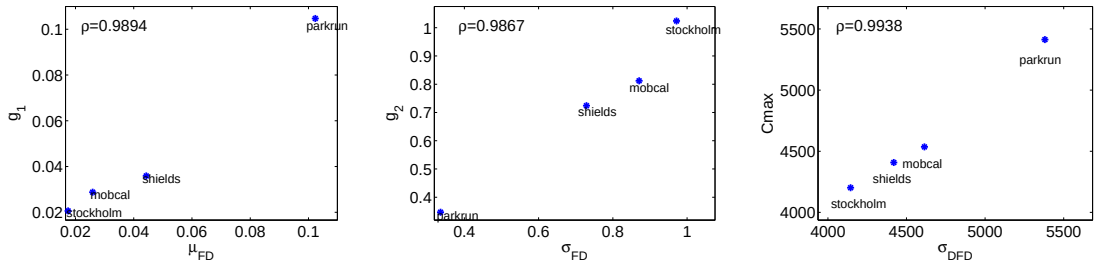


(b) two feature

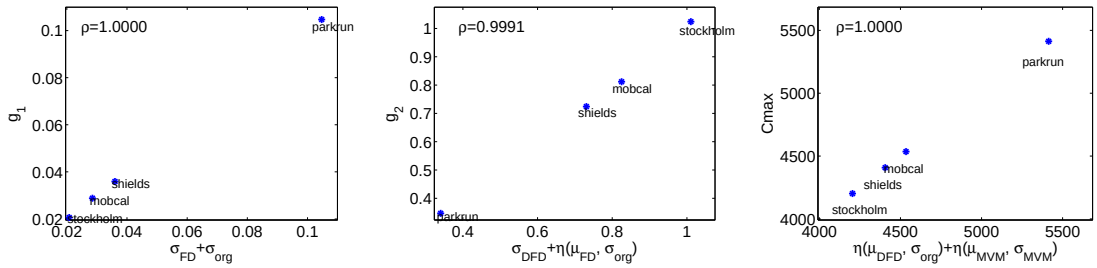
Figure 5.6: Illustration of rate model parameter prediction, i.e., $a, b, R_{\max}$ using content features for 720p resolution videos.

Table 5.7: Weighted coefficients matrix for model parameters: one feature, 720p video

	a	b	$R_{\max}$	g_1	g_2	$C_{\max}$
ω_0	2.46	0.66	-26220.00	-0.07	1.90	3532.80
$\sigma_{\texttt{FD}}$					-0.13	
$\sigma_{\texttt{DFD}}$						457.79
$\sigma_{\texttt{MDA}}$		-0.11				
$\mu_{\texttt{FD}}$				0.01		
$\mu_{\texttt{MVM}}$	-0.46					
$\eta(\mu_{\texttt{DFD}}, \sigma_{\texttt{org}})$			331450.00			



(a) one feature



(b) two feature

Figure 5.7: Illustration of complexity model parameter prediction, i.e., $g_1, g_2, C_{\max}$ using content features for 720p resolution videos.

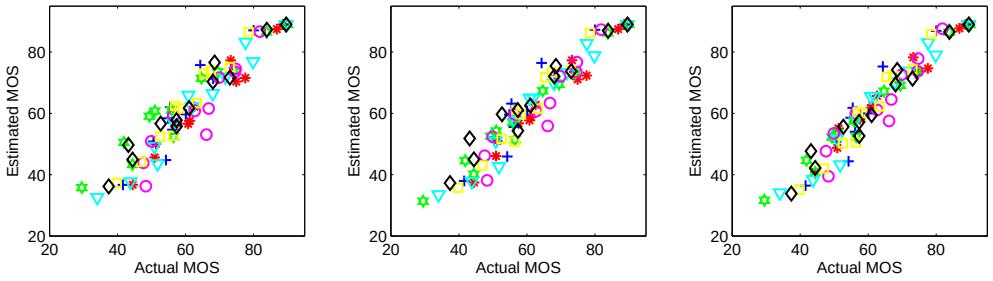
Table 5.8: Weighted coefficients matrix for model parameters: two feature, 720p video

	a	b	$R_{\max}$	g_1	g_2	$C_{\max}$
ω_0	2.27	0.64	-10615.00	-0.20	1.51	1471.30
$\sigma_{\texttt{FD}}$	-1.03			0.02		
$\sigma_{\texttt{DFD}}$		0.08			-0.13	
$\sigma_{\texttt{org}}$				1.7×10^{-3}		
$\mu_{\texttt{MVM}}$			3409.50			
$\eta(\mu_{\texttt{FD}}, \sigma_{\texttt{org}})$		-0.93			-1.80	
$\eta(\mu_{\texttt{DFD}}, \sigma_{\texttt{org}})$	78.29					39638.00
$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MDA}})$			2952.20			
$\eta(\mu_{\texttt{MVM}}, \sigma_{\texttt{MVM}})$						-602.10

accuracy with respect to the actual simulation data.

5.4.1 Predicted c and d for perceptual quality metric

Following the earlier discussion, parameters c and d can be predicted using different features (and their combinations). As shown in Figure 5.2, one feature does not work well, three-feature combination gives the excellent estimation and two-feature choice is in the middle. To see how the predicted c and d affect the predicted MOS provided by the quality model, we use the perceptual quality metric with predicted c and d to calculate the MOS scores and compare them with the original MOS data obtained by subjective rating. It is noted that such evaluation is more meaningful than presenting the parameter prediction only.



(a) one feature (b) two feature (c) three feature

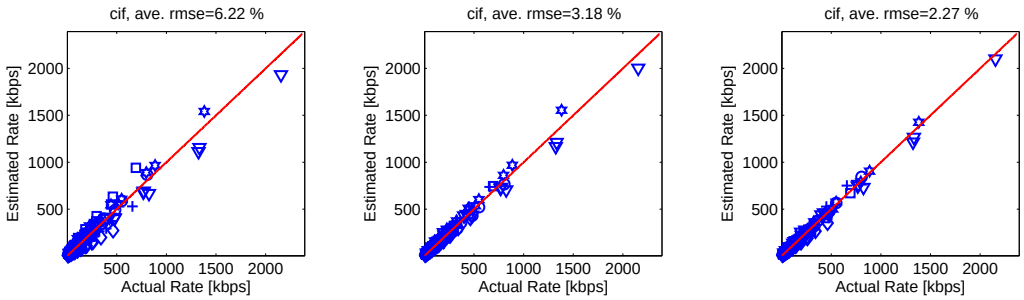
Figure 5.8: Quality model accuracy using predicted parameters c and d, scatter points are for all 7 CIF videos: (a) PC = 0.95, $e_\mu = 6.5\%$, $e_{\max} = 18\%$, (b) PC = 0.96, $e_\mu = 5.8\%$, $e_{\max} = 16\%$, (c) PC = 0.97, $e_\mu = 5.3\%$, $e_{\max} = 12\%$.

Figure 5.8 presents the prediction accuracy between predicted MOS using quality model and original collected MOS data. In addition to providing the Pearson correlation (PC) coefficient, we also define another prediction performance evaluation metric – *relative error*. For all tests, we will have a set of predicted MOS $\mathbf{Y}$ and a corresponding set of original MOS $\mathbf{X}$, then the *relative error* is

$$\mathbf{e} = \frac{|\mathbf{X} - \mathbf{Y}|}{\mathbf{X}}, \tag{5.6}$$

where we use its mean and maximum to quantify the accuracy of the prediction, i.e., e_μ and $e_{\max}$. Ideally, if we have both $e_\mu = 0$ and $e_{\max} = 0$, the prediction is perfect. However, in practice, we always have prediction error, where both $e_\mu > 0$ and $e_{\max} > 0$. Clearly, the smaller e_μ and $e_{\max}$, the better prediction. As shown in Figure 5.8, PC does not differ much when we choose to use one, two or three features. However, we have found that three-feature combined prediction indeed gives the best result of the relative error, i.e., points are more squeezed in subplot 5.8(c), and one feature prediction is the least accurate, i.e., points are more dispersed. Interestingly, we have found that two-feature combined prediction has the decent MOS prediction and already gives the good estimation, although the two-feature combined prediction does not work very well for some other parameters, such as shown in Figure 5.2 (i.e., parameter c only has 0.89 PC).

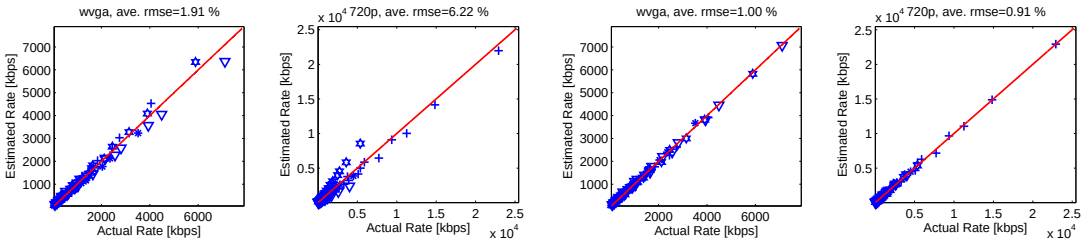
5.4.2 Predicted a, b and $R_{\max}$ for rate model



(a) one feature (b) two feature (c) three feature

Figure 5.9: Rate model accuracy using predicted parameters $a, b, R_{\max}$, scatter points are for all 7 CIF videos.

As shown in Figure 5.9, we present the prediction accuracy for CIF videos by comparing the actual rates and model estimated rates, where the model parameters are predicted using content features. According to our experimental results, rate prediction has very high PC (over 0.99) for all cases, i.e., one-feature, two-feature and three-feature. However, by calculating the average RMSE over all test sequences, we have found that three-feature



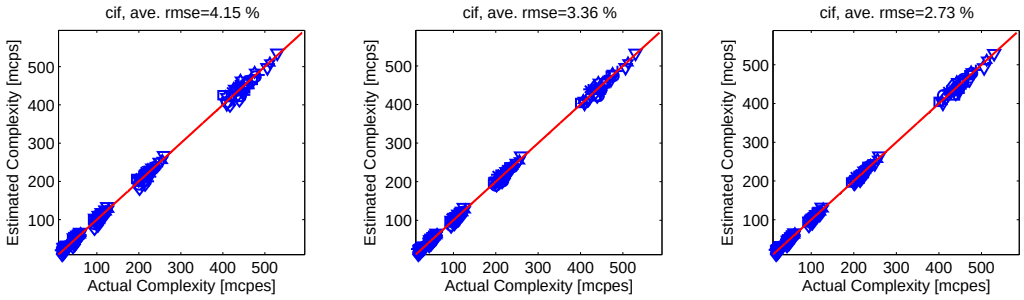
(a) one feature (b) two feature

Figure 5.10: Rate model accuracy using predicted parameters $a, b, R_{\max}$, scatter points are for 4 WVGA and 4 720p videos.

combined prediction gives the best result, where almost all scatter points are merged to the "perfect line" (i.e., $y = x$). On the other hand, we can also notice that the scatter points are more dispersed for one-feature parameter prediction as shown in subplot 5.9(a). Similarly, two-feature combined prediction already gives the good estimation on actual bit rates.

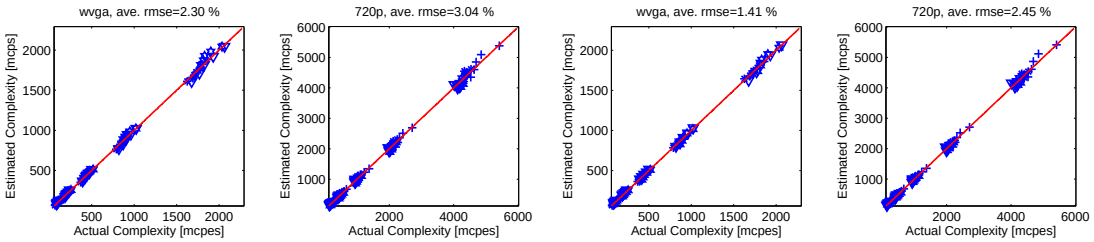
In addition to CIF videos, we also present the prediction accuracy for other 4 WVGA and 4 720p videos. All results show that with the feature predicted parameters, our model can estimate the actual bit rates accurately.

5.4.3 Predicted $g1$, $g2$ and $C_{\max}$ for complexity model



(a) one feature (b) two feature (c) three feature

Figure 5.11: Rate model accuracy using predicted parameters $g_1, g_2, C_{\max}$, scatter points are for all 7 CIF videos.



(a) one feature (b) two feature

Figure 5.12: Rate model accuracy using predicted parameters $g_1, g_2, C_{\max}$, scatter points are for 4 WVGA and 4 720p videos.

As shown in Figure 5.11, we present the prediction accuracy for CIF videos by comparing the actual complexity and model estimated cycles, where the model parameters,i.e., g_1, g_2 and $C_{\max}$, are predicted based on content features. According to our experimental results, complexity prediction has very high PC (over 0.99) for all cases. However, by calculating the average RMSE over all test sequences, we have found that three-feature combined prediction gives the best result, where almost all scatter points are merged to the "perfect line" (i.e., $y = x$). Meanwhile, we can also notice that the scatter points are more dispersed for one-feature parameter prediction as shown in subplot 5.11(a), in particular starting from 400 mega cycles per second (mcps) to the highest value. Instead, two-feature combined prediction already performs well for actual complexity estimation.

Similarly, in addition to CIF videos, we also provide the prediction accuracy for other 4 WVGA and 4 720p videos. All results show that, using the content feature predicted parameters, our model can still estimate the actual complexity accurately.

5.5 Discussion and Summary

In practice, our proposed models will be more useful if we can estimate the model parameters via the underlying video contents instead of explicitly presenting them. Also, according to our extensive simulation results, we have found that the model parameters are indeed content dependent. This chapter explores the model parameter prediction using

content features. Toward this goal, we first abstract useful content features, and develop a lightweight pre-processor to obtain those features. In general, we have considered the features related to the *residual signal* such as frame difference, displace frame difference etc, *motion fields*, such as motion vector magnitude, motion direction activity, etc and *original video signal*, such as video contrast. Different feature combinations are examined in our study to show the parameter prediction accuracy. Simulation results show that three-feature combined prediction works well for CIF videos and two-feature prediction provides the accurate estimation for WVGA and 720p videos. For different video resolution, different feature combination and weighted function will be applied. We also plug the estimated parameters into our proposed model, and find that the actual MOS, rate and complexity can be well estimated using two-feature combined prediction.

Based on our simulations, the content feature set is quite stable for various videos. We also note that, the weighted coefficients varies largely, which brings the large variation when we introduce new test video. It might be helpful to do the feature normalization before conducting the generalized linear regression. However, it is out of the scope of current study and deferred as our future work.

Chapter 6

Applications

In this chapter, we present several popular applications using our proposed models. First, we apply our proposed models to guide the resource constrained scalable video adaptation. In particular, we consider the situation where the network bandwidth is the only limitation. We then analyze the situation where the video receiver has the limited battery capacity and access network bandwidth, which is the typical scenario for video playback capable mobile handhelds. The second application we have studied is using our complexity prediction algorithm to guide the DVFS of the underlying processor so as to conduct the energy efficient video decoding.

6.1 Resource Constrained Scalable Video Adaptation

The first deployment is using our proposed models to guide the scalable video adaptation given the constrained resource (such as access network bandwidth, remaining battery capacity, user preference, etc.) at the receiver. This is typical for mobile handheld, where it is powered by the battery with limited capacity and always has network bandwidth constraint when connecting with the wireless access point. In the following sections, we will first introduce the solution using our proposed quality and rate models to guide the rate constrained bit stream adaption; then we extend our solution to consider the limited remaining battery power as well using our devised power consumption model.

6.1.1 Rate-Constrained Bit Stream Adaptation

Combining the rate and quality models, we draw in Figure 6.1 [1], quality vs. rate curves achievable at different frame rates. We also plot the measured MOS data on the same figure. The model fits the measured data very well for sequences "akiyo", "crew" and "waterfall". But the model is not as accurate at some frame rates for "football", "city", "foreman" and "ice". The inaccuracy is mainly due to the difference between the predicted quality and MOS for these sequences. We suspect that this is partly due to the large spread of the viewer quality ratings to these sequences, due to the difference in the temporal sensitivity among the viewers. It is clear from this figure, that each frame rate is optimal only for a certain rate region, which will be further elaborated as follows.

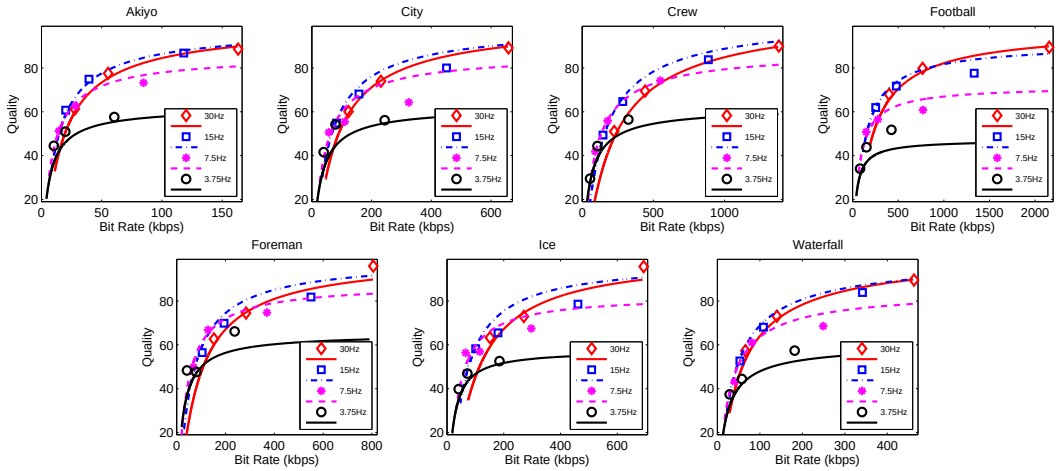


Figure 6.1: Quality vs. rate at different frame rates. Points are measured data, curves are based on the rate model in Eq. (3.28) and the quality model in Eq. (2.4).

In this section, we consider how to apply our proposed rate and quality models to perform rate-constrained SVC bit stream adaptation. Figure 6.2 provides a system view of the adaptation problem. For each video, a single full-resolution scalable stream is available at a media content server, which will be adapted at a network proxy or gateway in response to the user channel conditions and viewing preferences. When a user requests the video

[1] Since we only have results for 7 CIF videos regarding the quality model, we present the adaptation application for those seven sequences. We believe that the same scheme can be applied to other videos as well.

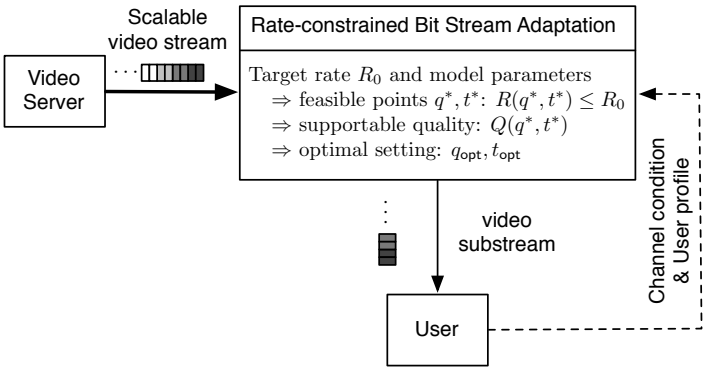


Figure 6.2: Rate-Constrained SVC Video Adaptation

from the server, the adaptor (sitting at the proxy) will determine an appropriate video rate R_0 based on the user's channel condition (e.g. R_0 is the sustainable transmission rate for the given channel condition minus all the overheads for channel error correction and packetization). Based on R_0 and the user's viewing preference setting (embedded in the user profile sent to the adaptor), the adaptor determines the optimal set of temporal and amplitude layers (more generally spatial layers) to extract, so as to provide the best perceptual quality. In Figure 6.2, we assume that the adaptor monitors the channel condition based on some feedbacks from the user. (The user may inform the adaptor its desired rate R_0 in alternative implementations.) Furthermore, it determines the quality model parameters based on the user's preference setting, which describes the user's preferred tradeoff among spatial, temporal, and amplitude resolutions. Recall that parameters a, b, c, d and $R_{\max}$ can be predicted accurately using the content features as discussed in earlier Chapter 5. For a simple implementation, we can embed those features in the full-resolution bitstream as side information, which can be parsed at adaptor to estimate the parameters. Based on the target rate R_0 and the model parameters, the adaptor determines the optimal frame rate $t_{\mathtt{opt}}$ and quantization $q_{\mathtt{opt}}$, and corresponding temporal and amplitude layers. Finally the adaptor extracts these layers from the scalable bit stream and delivers the resulting bit stream to the user.

For a given target rate R_0, the adaptation problem can be formulated as the following

constrained optimization problem,

$$\begin{aligned} &\text{Determine } t, q \text{ to } \text{ maximize } \ Q(q,t) \\ &\qquad\qquad\qquad\qquad \text{subject to } \ R(q,t) \leq R_0. \end{aligned} \tag{6.1}$$

In the following subsections, we employ proposed rate and quality models to solve this optimization problem, first assuming the frame rate can be any positive value, and then considering the discrete set of frame rates afforded by the dyadic temporal prediction structure.

Optimal solution assuming t and q continuous values

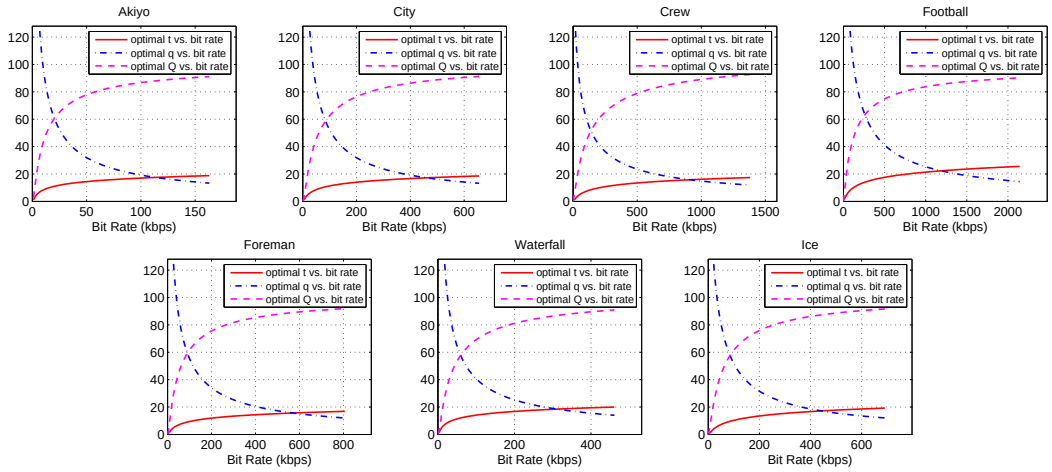


Figure 6.3: Optimal quantization stepsize $q_{\texttt{opt}}$, frame rate $t_{\texttt{opt}}$, and the corresponding quality $Q_{\texttt{opt}}$ versus the bit rate R by assuming q and t can take on any continuous values within their respectives ranges.

We first solve the constrained optimization problem in (6.1) assuming both the frame rate t and quantization stepsize q can take on any value in the range of $t \in (0, t_{\max})$, $q \in (q_{\min}, +\infty)$. To simplify the notation, let $\hat{Q} = \frac{Q_{\max}}{(1-e^{-d})e^{-c}}$, $\hat{t} = t/t_{\max}$, $\hat{q} = q/q_{\min}$, $\hat{R} = R/R_{\max}$, and $\hat{R}_0 = R_0/R_{\max}$, the rate and quality models in (3.28) and (2.4) become respectively

$$\hat{R}\left(\hat{q}, \hat{t}\right) = \hat{q}^{-a}\hat{t}^{b}, \tag{6.2}$$

$$Q\left(\hat{q}, \hat{t}\right) = \hat{Q}e^{-c\hat{q}}\left(1 - e^{-d\hat{t}}\right). \tag{6.3}$$

By setting $\hat{R}\left(\hat{q},\hat{t}\right)=\hat{R}_0$ in (6.2), we obtain

$$\hat{q}=\sqrt[a]{\left(\hat{t}^b/\hat{R}_0\right)}, \tag{6.4}$$

which describes the feasible q for a given t, to satisfy the rate constraint R_0.

Substituting (6.4) into (6.3) yields

$$Q(\hat{t})=\hat{Q}e^{-\frac{c\cdot\hat{t}^{\psi}}{\sqrt[a]{\hat{R}_0}}}\left(1-e^{-d\hat{t}}\right),\ \ \hat{t}\in(0,1) \tag{6.5}$$

where $\psi=b/a$. Equation (6.5) expresses the achievable quality with different frame rate under the rate constraint R_0. Clearly, this function has a unique maximum, which can be derived by setting its derivative with respect to $\hat{t}$ to zero. This yields

$$\hat{R}_0=\left(\frac{c\psi\hat{t}^{\psi-1}(1-e^{-d\hat{t}})}{de^{-d\hat{t}}}\right)^a. \tag{6.6}$$

For any given rate constraint R_0, we can solve (6.6) numerically to determine the optimal frame rate $t_{\texttt{opt}}$. Then using (6.4) and (6.5) we can determine the optimal quantization stepsize $q_{\texttt{opt}}$, and the corresponding maximum quality $Q_{\texttt{opt}}$ at the rate R_0. Figure 6.3 shows $t_{\texttt{opt}}$, $q_{\texttt{opt}}$, and $Q_{\texttt{opt}}$ as functions of the rate constraint R_0. As expected, as the rate increases, $t_{\texttt{opt}}$ increases while $q_{\texttt{opt}}$ reduces, and the achievable best quality continuously improves. Notice that $t_{\texttt{opt}}$ increases more rapidly for the "football" sequence than for the other sequences, because of its faster motion. Based on the parameters derived from our subjective test data, even at the highest bit rates examined, the optimal frame rate is below 20 Hz for the other three sequences. Note that had we used a smaller $q_{\min}$ to allow much higher values for $R_{\max}$, $t_{\texttt{opt}}$ would have increased to 30 Hz beyond some rates.

Optimal solution under dyadic temporal scalability structure

A popular way to implement temporal scalability is through the dyadic hierarchical B-picture prediction structure, where the frame rate doubles with each more temporal layer. With 5 temporal layers, the corresponding frame rates are 1.875, 3.75, 7.5, 15 and 30 Hz. From a practical point of view, it will be interesting to see what is the optimal combination

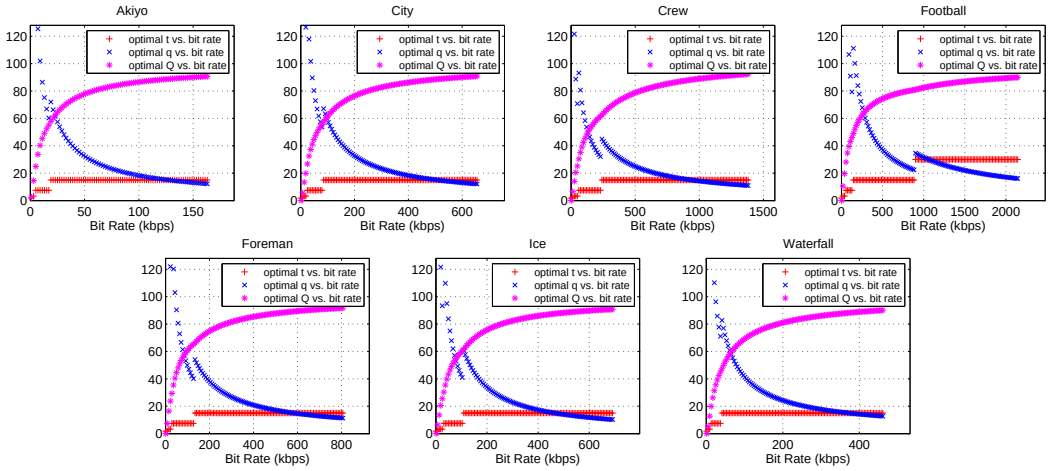


Figure 6.4: Optimal operating points $q_{\mathtt{opt}}$, $t_{\mathtt{opt}}$, and $Q_{\mathtt{opt}}$ versus R by assuming t can only take discrete values allowed by the dyadic prediction structure, whereas q can vary continuously.

of the frame rate and quantization stepsize for different bit rates under this structure. To obtain the optimal solution under this scenario, for each given rate, we determine the quality values corresponding to all five possible frame rates using (6.5), and choose the frame rate (and its corresponding quantization stepsize using (6.4)) that leads to the highest quality. The results are shown in Figure 6.4. Because the frame rate t can only increase in discrete steps, the optimal q does not decrease monotonically with the rate. Rather, whenever $t_{\mathtt{opt}}$ jumps to the next higher value (doubles), $q_{\mathtt{opt}}$ first increases to meet the rate constraint, and then decreases while t is held constant, as the rate increases. Consistent with the previous results in Figure 6.3, for football, the optimal frame rate transition to 30 Hz at an intermediate bit rate; whereas for the other sequences, the optimal frame rate stays at 15 Hz even at the highest bit rates examined. As mentioned earlier, had we used a lower $q_{\min}$, we would have seen transitions to 30 Hz after some rates.

The results in Figure 6.4 can be validated by cross checking with Figure 6.1. For example, for "Crew", in the rate region below 25.0 kbps, 1.875 Hz leads to the highest quality, in the rate range between 25 and 61 kbps, 3.75 Hz gives the highest quality, between 61 and 253 kbps, 7.5 Hz is the best, and beyond 253 kbps, 15 Hz provides the highest

quality. Connecting the top segments for each sequence in Figure 6.1 will lead to the optimal Q vs. bit rate curve in Figure 6.4.

In practice, the SVC encoder with amplitude scalability does not allow the quantization stepsize to change continuously. The finest granularity in quality scalability is a decrement of QP by 1 with each additional quality layer. This means that the quantization stepsize reduces by a factor of $2^{-1/6}$ with each additional layer. In practice, much coarser granularity is typically used, with decrement of QP by 2 to 4 typically. When we constrain q to take only discrete values corresponding to such QP values, in addition to allow only dyadic frame rates, one cannot always meet a rate constraint exactly. One can still solve the optimal t and q for any given rate constraint using the proposed models, by exhaustive search within the finite set of feasible values for t and q.

6.1.2 Power-Rate Constrained Bit Stream Adaptation

In this subsection, we extend our earlier solution regarding the rate-constrained SVC adaptation to power-rate constrained problem by considering an additional constraint introduced by the limited remaining battery capacity of mobile handhelds. The system model for power-rate constrained problem is illustrated in Figure 6.5. As we can see, target bit rate R_0 and remaining power P_0 are provided by the user channel condition and its local profile (such as battery status). For example, the power constraint P_0 can be informed by the user according to its remaining battery capacity (i.e., B_r mAh) and how long he wants his mobile to be active (before shut-down) (i.e., $\tau_{\texttt{on}}$), then we can simply use $B_r/\tau_{\texttt{on}}$ to estimate the required power P_0. Usually, we want the $\tau_{\texttt{on}} > \tau_{\texttt{video}}$, where $\tau_{\texttt{video}}$ indicates the length of video playback, so as to avoid running out the battery. Other researchers have already provided some efficient ways to estimate the remaining battery capacity [53], which is out of the scope of our work. Here, we simply assume we have the knowledge of the remaining battery capacity for video playback, for instance, κ percentile of the total remaining battery power, i.e., $B_{\texttt{video}} = \kappa B_r, 0 < \kappa < 1$. Together with the total playback

length of the video, we can obtain the power limitation to guide the adaptation, i.e.,

$$P_0 = \frac{B_{\texttt{video}}}{\tau_{\texttt{video}}} = \kappa \frac{B_r}{\tau_{\texttt{video}}}, \tag{6.7}$$

where $\kappa \in (0, 1)$ is a known variable which can be configured by user, and $\tau_{\texttt{video}}$ is the length for video playback which can be easily obtained by "user"–"server" communication for live video streaming scenario, or by fast parsing container header field associated with the local buffered video clip.

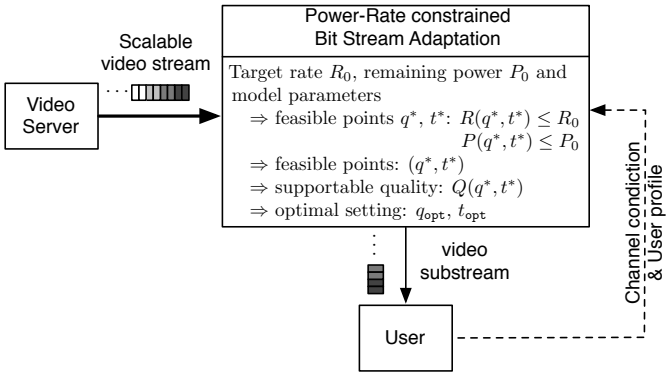


Figure 6.5: Power-Rate constrained scalable bit stream adaptation.

In previous Section 4.3.4, we have devised a power consumption model for scalable video decoding on ARM cortex A8 processor [43], which is a function of the frame rate t and quantization stepsize q. Therefore, we can derive the feasible points for q and t by solving $P(q, t) \le P_0$ with P_0 defined as the limited power bound. Since the same type of ARM processor is commonly deployed on various mobile handhelds, such as SmartPhones, mobile Pads, etc, we believe that our model can be used widely. Even with new ARM processor, such as ARM cortex A9, we believe that our model can be migrated to it smoothly. Recall that all parameters, i.e., a, b, c, d, g_1, g_2, $R_{\max}$ and $C_{\max}$ can be well predicted using the content features in Chapter 5. We can embed those features in the full-resolution bitstream as side information, which can be parsed at adaptor to estimate the parameters as suggested earlier. Based on the target rate R_0, power bound P_0 and the model parameters, the adaptor determines the optimal frame rate $t_{\texttt{opt}}$ and quantization $q_{\texttt{opt}}$, and corresponding

temporal and amplitude layers. Finally the adaptor extracts these layers from the scalable bit stream and delivers the resulting bit stream to the user/receiver.

For a given target rate R_0 and power bound P_0, the adaptation problem can be written as

$$\begin{aligned} \text{Determine } t, q \text{ to } & \text{ maximize } Q(q,t) \\ & \text{subject to } R(q,t) \le R_0, \\ & P(q,t) \le P_0. \end{aligned} \tag{6.8}$$

Because of the unique mapping between power consumption and complexity workload as discussed in previous Section 4.3.4, we can also rewrite the optimization problem as

$$\begin{aligned} \text{Determine } t, q \text{ to } & \text{ maximize } Q(q,t) \\ & \text{subject to } R(q,t) \le R_0, \\ & C(q,t) \le C_0, \end{aligned} \tag{6.9}$$

where C_0 is determined by the P_0 according to Eq. (4.14).

In the following paragraphs, we employ proposed complexity, rate and quality models to solve this optimization problem, first assuming that the frame rate, quantization stepsize and supportable complexity level can be any positive value, second considering the discrete set of the complexity level enabled by the typical processor, and then evaluating the discrete frame rates afforded by the dyadic temporal prediction structure and discrete complexity levels constrained by the underlying processor.

Optimal solution assuming continuous t, q and C_0

Let $\hat{q} = q/q_{\min}$, $\hat{t} = t/t_{\max}$, $\hat{R}_0 = R_0/R_{\max}$, $\hat{Q}_{\max} = Q_{\max}e^c/(1-e^{-d})$, $\hat{C}_0 = C_0/C_{\max}$, by combing Eqs. (2.4), (3.28) and (4.6), we can have the adaptation problem as

$$\max \quad \hat{Q}_{\max}e^{-c\hat{q}}(1-e^{-d\hat{t}}), \tag{6.10}$$

$$\text{s.t.} \quad \hat{q}^{-a}\hat{t}^{b} \le \hat{R}_0, \tag{6.11}$$

$$\hat{q}^{-g_1\hat{t}^{-g_2}}\hat{t} \le \hat{C}_0. \tag{6.12}$$

Obviously, we can apply the constrained optimization tool by introducing the lagrangian multipliers to solve the above problem. However, it is noted that $Q(\hat{q},\hat{t})$, $R(\hat{q},\hat{t})$ and $C(\hat{q},\hat{t})$ are monotonic function with respect to $\hat{t}$ and $\hat{q}$ (and same as the t and q because of the linear scaling), therefore we can always have the solution of $\hat{q}_{\texttt{opt}}$ and $\hat{t}_{\texttt{opt}}$ when we set the reasonable constraint bounds $\hat{R}_0$ and $\hat{C}_0$. Because of the monotonicity of (6.11) and (6.12), we can decompose the original 2-D optimization into two 1-D optimization, i.e., either (6.11) or (6.12) is active at the boundary, then leave the other as the only one constraint. The proof of the monotonicity for those models with respect to frame rate and quantization stepsize can be found in [54].

Particularly, we first assign the rate constraint as active condition, i.e., $\hat{q}^{-a}\hat{t}^{b} = \hat{R}_0$. Thus, we can present $\hat{q}$ as a function of $\hat{t}$, i.e., Eq. (6.4). Then, we plug Eq. (6.4) into (6.12) to determine the optimal $\hat{t}$, i.e.,

$$\left(\hat{t}^{b}/R_0\right)^{\frac{-g_1\hat{t}-g2}{a}} \cdot \hat{t} \leq \hat{C}_0. \tag{6.13}$$

The $\hat{t}_{\texttt{opt}}$ and $\hat{q}_{\texttt{opt}}$ are chosen so as to yield the best video quality using Eq. (6.10).

Figure 6.6, 6.7 and 6.8 show the optimum solution given any network bandwidth R_0 and complexity C_0 constraints. For each pair of input R_0 and C_0, we can find the optimal quantization stepsize $q_{\texttt{opt}}$ in Figure 6.6, optimal frame rate $t_{\texttt{opt}}$ in Figure 6.7 and corresponding best quality $Q_{\texttt{opt}}$ in Figure 6.8.

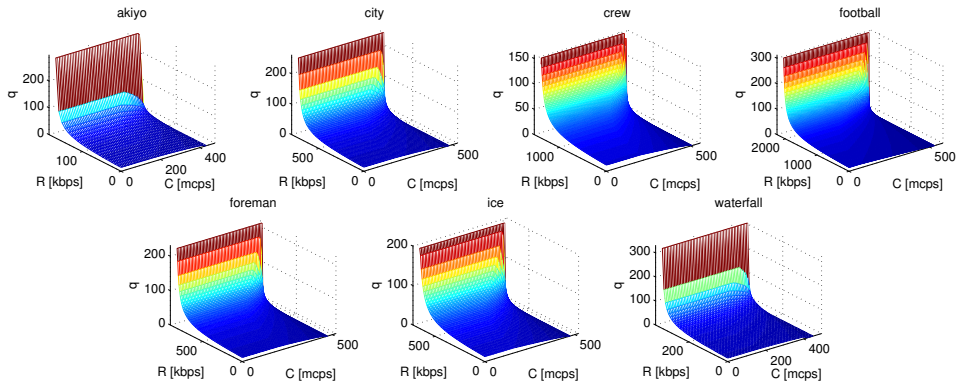


Figure 6.6: Optimal quantization stepsize $q_{\texttt{opt}}$ versus the bit rate R and complexity C assuming q and t can take on any continuous values within their respective ranges.

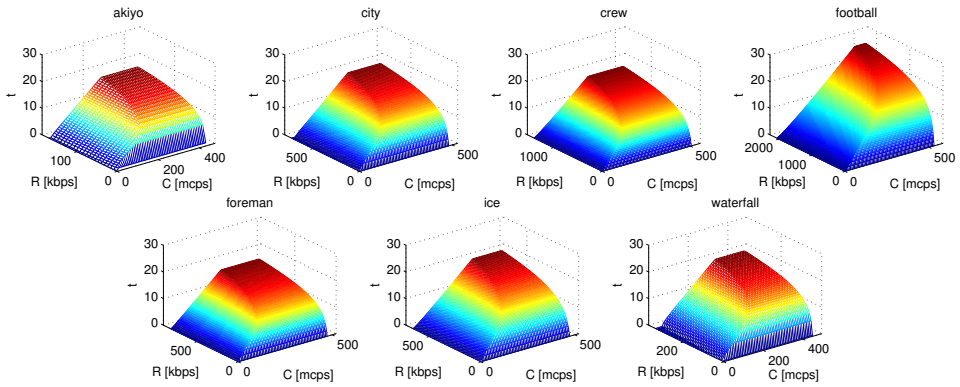


Figure 6.7: Optimal frame rate $t_{\mathtt{opt}}$ versus the bit rate R and complexity C assuming q and t can take on any continuous values within their respective ranges.

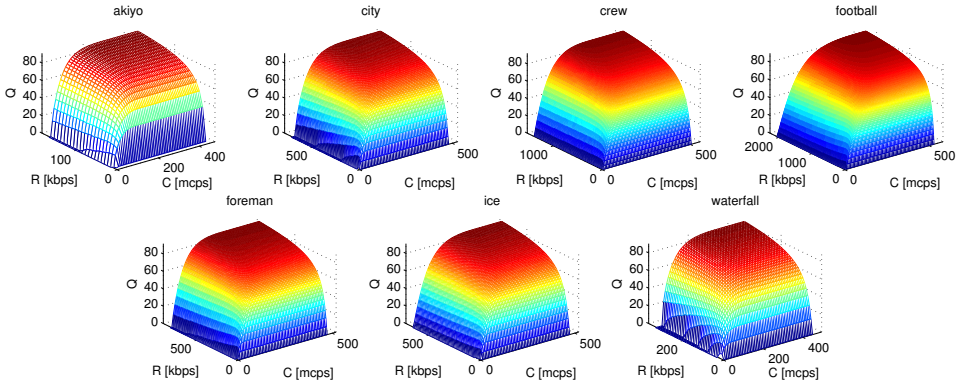


Figure 6.8: Optimal quality $Q_{\mathtt{opt}}$ (corresponding to the $t_{\mathtt{opt}}$ and $q_{\mathtt{opt}}$) versus the bit rate R and complexity C assuming q and t can take on any continuous values within their respective ranges.

Optimal solution assuming continuous t, q and discrete C_0

The complexity constraint is defined to show how many cycles per second is sufficient to provide the smooth video playback, which is the same as the required CPU frequency in Hz. Previously, we assume the continuous complexity, i.e., continuous frequency. However, in practice, the supportable CPU frequency is discrete as shown in Table 6.1 and Table 4.3 for Intel PM 1.6 GHz and ARM cortex A8 600 MHz processor, which means that we can only support several levels of the CPU clock rate. For example, our ARM processor can support 125, 250, 500, 550 and 600 MHz in default. Referring to Table 4.2, even for motion intensive "football" sequence, the required maximum frequency is 530 MHz, therefore, we choose the 550 MHz as the maximum clock rate. Since the 550 MHz is quite close to 500 MHz, we choose to skip the 500 MHz and only select 125, 250 and 550 MHz to examine the discrete complexity levels. We believe that such evaluation can be applied widely, given the discrete clock rates supported by the popular micro-processors.

Figure 6.9 shows the optimal combination of q and t given the bit rate constraint R_0 at a typical discrete complexity level. Let us take "football" sequence as an example. For different complexity levels, we have different optimal qualities. When we have sufficient computing power, we will have the magenta curves for $q(R)$, $t(R)$ and $Q(R)$, which are the same as the results presented in rate constrained problem where complexity is not constrained. The curves for $q(R)$, $t(R)$ and $Q(R)$ will be altered when we add the complexity constraint. From 550 MHz to 250 MHz, we can see the best quality is reduced, and there is a turning point for the blue curve of frame rate t, which means that the complexity bound will break if we further increase the frame rate, and the subsequent bit rate increment is mainly due to the the decreasing of the quantization stepsize. This result suggests that the temporal decoding is more complex than the amplitude decoding (also proves that SVC layered decoding has the quite similar complexity compared with single layer decoding). Because of the decrement of the quantization stepsize, the overall quality still stays stable even we have to reduce the frame rate at 250 MHz. It means that the temporal quality reduction is compensated by the spatial quality improvement by the decreasing quantization stepsize. However, when we only have 125 MHz computing capability, we see that the

quality will decrease as well after a certain bit rate, where the frame rate reduces largely and the decrement of the quantization stepsize can not compensate the temporal quality reduction although the bit rate is still going up. In reality, because of the finite amplitude levels encoded in the bit rate, we do not further decrease the quantization stepsize when the frame rate meets the complexity constraint. We can choose the bit rate R_t corresponding to the maximum supportable quality Q_t to deliver the bit stream with $R_t \leq R_0$.

Optimal solution under dyadic temporal scalability structure and discrete C_0

Furthermore, we apply our solution under the dyadic prediction structure, where we can only support five discrete frame rates, i.e., 1.875, 3.75, 7.5, 15, and 30 Hz given the maximum frame rate at 30 Hz and the 16-picture GOP for temporal scalability. More practically, we follow the assumption discussed earlier to choose the discrete complexity level supported by the ARM processor. Therefore, our solution is more generic and applicable for the practical system. Please note that we still assume the continuous quantization stepsize. Because the frame rate t can only increase in discrete steps, the optimal q does not decrease monotonically. Rather, whenever $t_{\texttt{opt}}$ jumps to the next higher value (doubles), $q_{\texttt{opt}}$ first increases to meet the rate constraint, and then decreases while t is held constant, as the rate increases. Figure 6.10 shows the same plots in comparison to the Figure 6.4 when we set the complexity level at 550 MHz. Consistent with the previous results, for football, the optimal frame rate transition to 30 Hz at an intermediate bit rate; whereas for the other sequences, the optimal frame rate stays at 15 Hz even at the highest bit rates examined. When the mobile doesn't have enough power, the complexity level is reduced to 250 MHz (illustrated using blue curve in Figure 6.10) the optimal t does not go to 30 Hz anymore because of the complexity constraint. Moreover, the optimal frame rate will decrease when we have even lower complexity support, i.e., 125 MHz, which is consistent with previous results shown in Figure 6.9. We also see that the quality, frame rate and quantization curves are truncated for 125 MHz, where the terminating point corresponds to the best quality. If we keep trying to increase the bit rate, the perceptual quality will reduce because of the severe reduction of the temporal video quality. In reality, we can choose to stop at the

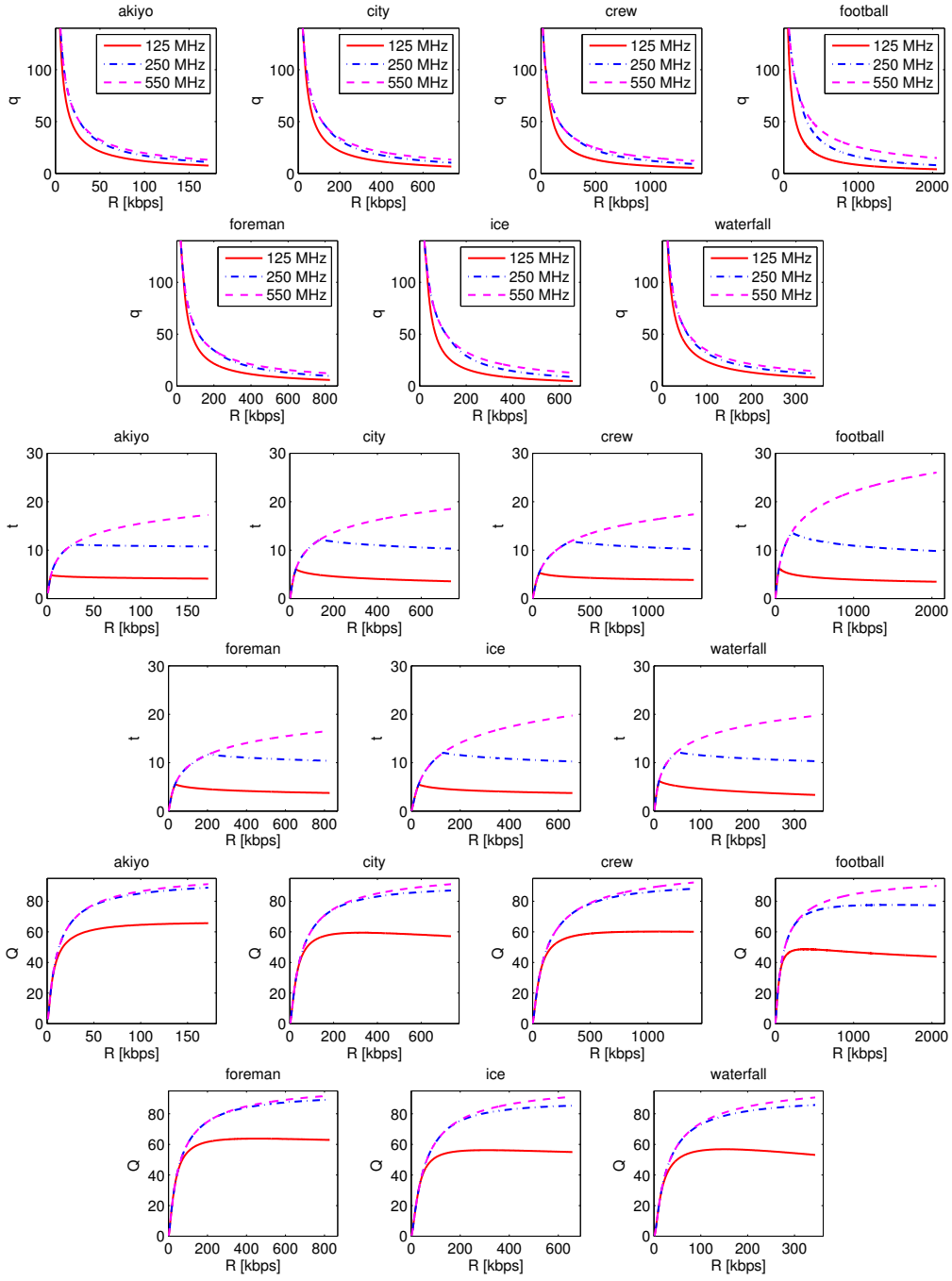


Figure 6.9: Optimal quantization stepsize $q_{\mathtt{opt}}$, frame rate $t_{\mathtt{opt}}$ and corresponding quality $Q_{\mathtt{opt}}$ versus the bit rate R when the complexity takes discrete levels and q and t take on any continuous values within their respective ranges.

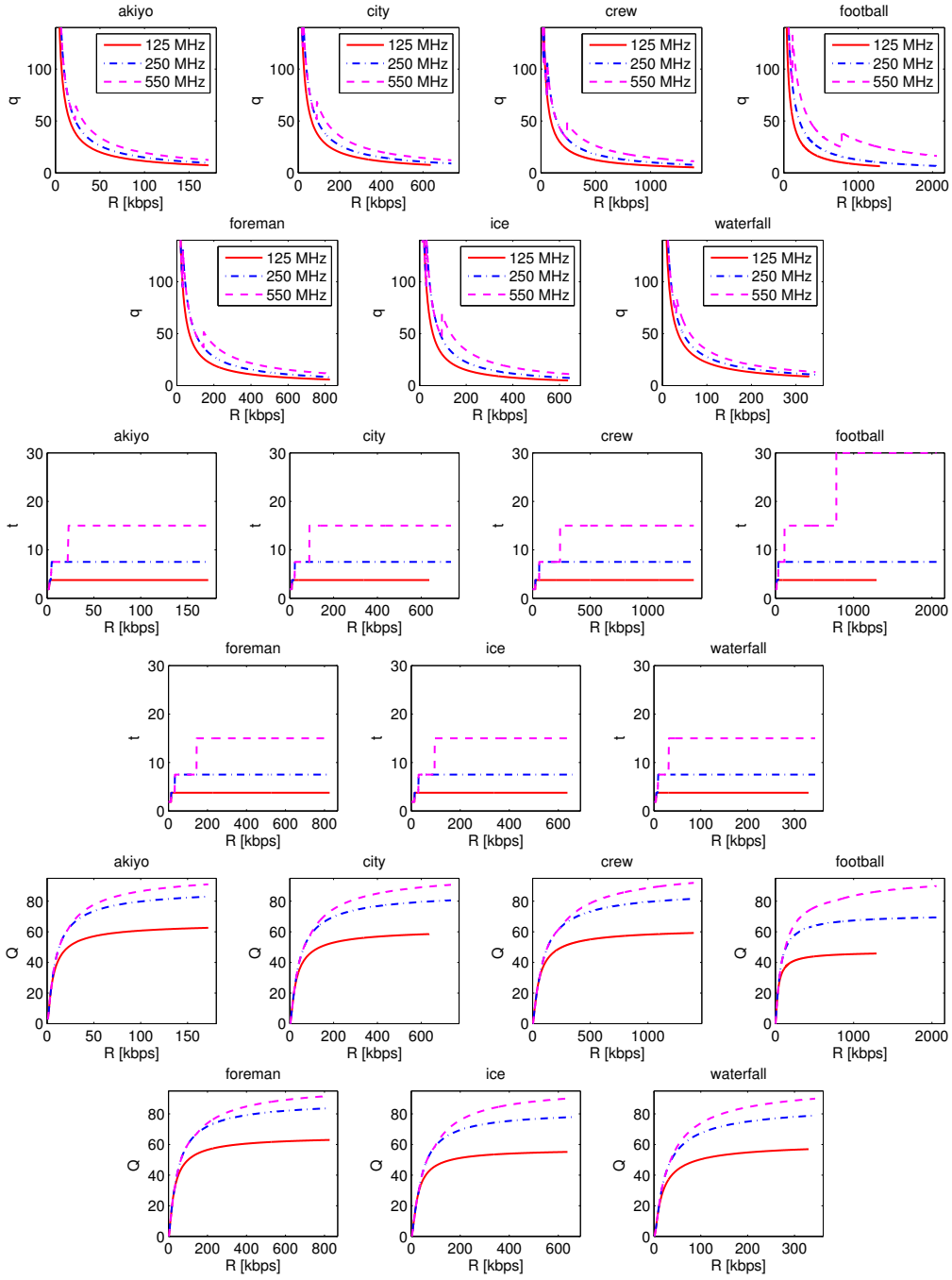


Figure 6.10: Optimal quantization stepsize $q_{\mathtt{opt}}$, frame rate $t_{\mathtt{opt}}$ and corresponding quality $Q_{\mathtt{opt}}$ versus the bit rate R when the complexity takes discrete levels under dyadic prediction structure.

maximum quality point for insufficient computational complexity supply.

6.2 DVFS-enabled Energy Efficient Video Decoding

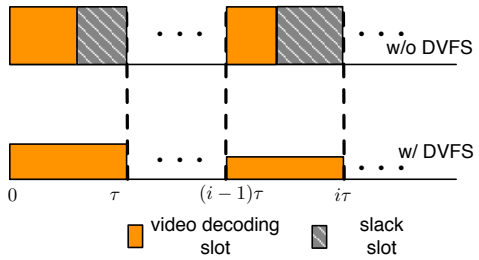


Figure 6.11: DVFS-enabled video decoding, the i-th frame or GOP decoding and rendering is allocated in the slot $[(i-1)\tau, i\tau]$.

Dynamic voltage/frequency scaling (DVFS) is a technique that adjusts the voltage and frequency of a processor based on the required processing cycles C, and completion deadline of a task (with time interval τ). In a traditional processor without DVFS, the processor always runs at a maximum voltage $V_{\max}$ and frequency $f_{\max}$, regardless required CPU cycles, as illustrated in the upper part of Figure 6.11. With DVFS, the CPU frequency f is adjusted according to C, so that in the ideal case $f = C/\tau \leq f_{\max}$, and $V_{dd} \leq V_{\max}$, as depicted in the bottom part of Figure 6.11, thereby reducing the total power consumption.

6.2.1 Proposed DVFS Control Driven by Complexity Prediction

From previous sections, our proposed complexity model can estimate the video decoding complexity accurately for the next frame or GOP, based on certain embedded data in the packetized stream and the measured processing cycles for some DM for the previous frame or GOP. Let us take frame-based video decoding as an example in the following discussion, where each frame must be decoded and rendered within the allocated time slot (e.g. $\tau = 33$ ms. for a 30 Hz video). Note that the discussion can be applied to the GOP-based setting similarly.

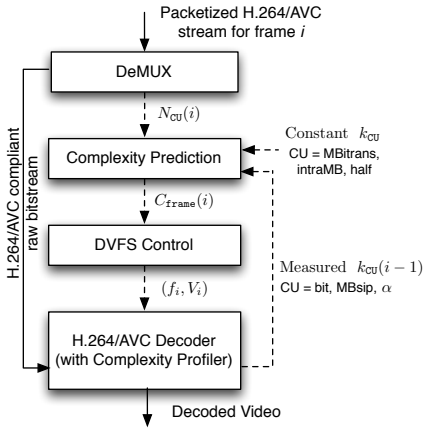


Figure 6.12: Complexity prediction based DVFS for H.264/AVC video decoding, complexity profiler is embedded into video decoder and used to collect cycles for each module.

Figure 6.12 illustrates our DVFS control scheme for H.264/AVC video decoding based on frame level complexity prediction. A similar process applies to GOP level DVFS adjustment as well. Usually, the raw H.264/AVC bitstream is packed into a certain container in popular applications, e.g., FLV, AVI, MKV, etc, for delivery or storage. In our work, we fill the $N_{\mathtt{CU}}$ for each frame into the header field of the container. Then the decomposed raw video bitstream can be decoded by any available H.264/AVC decoder. When complexity prediction is done at GOP interval, the $N_{\mathtt{CU}}$ information only needs to be embedded in the container header of packetized stream for a whole GOP. The packetized H.264/AVC stream is parsed to obtain the complexity metadata $N_{\mathtt{CU}}$ and H.264/AVC compliant raw bitstream (for example, Annex B compliant bitstream). Together with the $k_{\mathtt{CU}}$ which is either constant, or from the prediction using complexity data of the previous decoded DM, the parsed $N_{\mathtt{CU}}$ can be used to estimate cycles required by current DM decoding. Subsequently, the estimated total complexity for the current frame is used to select and set the proper frequency and voltage for the underlying processor prior to conducting current frame decoding. Based on our profiling, we have found that such DVFS control (together with complexity profiling and prediction) only requires cycles on the order of tens, which is far less than the cycles demanded by video decoding. Moreover, the voltage transition due to DVFS is around

70 μs [45] [2], which is far less than the real-time frame decoding constraint, for example, about 0.2% of 33 ms for 30Hz video. Thus such transition latency is acceptable for video decoding.

Typically, there is a set of discrete V_k and f_k supported by a processor to enable DVFS. For Intel PM processor on ThinkPad T42, there are 6-level voltages supported as listed in Table 6.1 [42], while there are 5 achievable voltages and corresponding maximum clock rates for the ARM processor on TI OMAP35x EVM platform, as presented in Table 4.3 [48]. To validate the power saving using DVFS, we create a video stream using concatenated sequences in the order of "News", "Harbour", "Ice" and "Soccer" at QP 24, each of which contains 120 frames [3]. Our experimental data provide that the maximum error between predicted and measured complexity is 8.7%, therefore, we scale the predicted complexity by a factor of 1.1 to avoid underestimation, and use this scaled version to set voltage and frequency of DVFS. On Intel platform, we use the scaled frame or GOP complexity to obtain the analytical power consumption. For the OMAP system, in addition to the analytical power saving, we also conduct real power measurement when doing DVFS enabled video decoding on the OMAP35x board. Two DVFS schemes are conducted for both experimental and analytical power saving investigation, [4] which are

- Discrete DVFS (D-DVFS): only the discrete sets of voltage and frequency listed in Table 6.1 and Table 4.3 are allowed. We choose the frequency f (and its corresponding V), that is the smallest one which is equal or larger than $\min(C/\tau, f_{\max})$, where τ is the frame interval.

- Continuous DVFS (C-DVFS): here we assume that the frequency and voltage can be adjusted continuously. The frequency is set to $f = C/\tau$, while the voltage is determined by the Eq. (4.10).

[2] The actual transition latency for our ARM platform is 78 μs.

[3] Because of the limited internal memory for data recording supported by our scope, we created new concatenated videos with 480 frames in total without using the longer sequences exemplified in previous sections.

[4] Currently, we only use the default discrete V_k and f_k supported by the OMAP system to do experimental power saving investigation, without implementing continuous frequency and voltage adjustment.

In the following paragraphs, we will present power savings by using DVFS through both analysis and measurements.

6.2.2 Intel PM 1.6 GHz

In this section, the DVFS enabled analytical power saving is computed for Intel PM processor on our ThinkPad T42 platform in comparison with traditional CPU operation without DVFS. As known, this 1.6 GHz Intel PM processor is fabricated using 90nm technology, and the dynamic power dominates the total power consumption. According to the discrete voltages and frequencies supported by the processor in Table 6.1 [42], we have found that the voltage V_{dd} is linearly related to the frequency f, with $\phi = 1$, $\omega = 5.6 \times 10^{-10}$, and $\theta = 0.61$, as illustrated in Figure 6.13. Thus, the dynamic power (4.8) can be represented as a function of f, i.e.,

$$P_{dyn} = K_{eff} \left(\omega f^{\phi} + \theta\right)^2 f. \tag{6.14}$$

In Table 6.2 we present the estimated dynamic power saving for two DVFS cases compared to the "Performance" scheme without DVFS. For the "Performance"scheme, we assume the CPU runs at maximum voltage $V_{\max}$ and clock rate $f_{\max}$ regardless of the required CPU cycles, and use $P = K_{eff}(\omega f_{\max}^{\phi} + \theta)^2 f_{\max}$ (in watt) to note the average power consumption for conducting video decoding. Although we separate the frame and GOP based video decoding, the "Performance" power consumption is the same for both, since same voltages are held during the entire video duration.

In comparison to the power consumption required by "Performance" scheme using peak power, the power saving factors of D-DVFS and C-DVFS are up to 2.94 and 3.33 for frame based video decoding, and are 3.03 and 3.45 for GOP based video decoding, as shown in Table 6.2.

6.2.3 ARM Cortex A8 600 MHz

In this section, we investigate the total power consumption required by ARM processor on the OMAP35x board given that the leakage power can not be ignored for our 65nm

Table 6.1: Supported dynamic voltage (volt) and frequency (MHz) of Intel PM 1.6 GHz processor on ThinkPad T42

Voltage	1.484	1.420	1.276	1.164	1.036	0.956
Max. Freq.	1600	1400	1200	1000	800	600

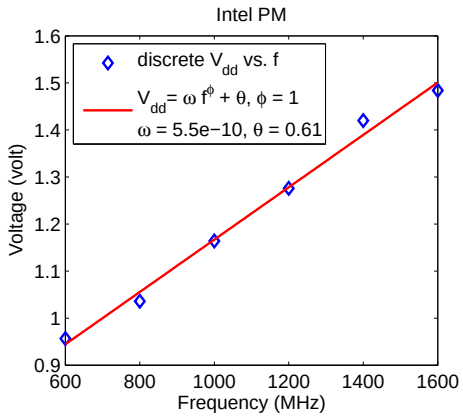


Figure 6.13: Relation between voltage and frequency for Intel PM processor.

Table 6.2: Normalized Dynamic power consumption for Intel PM processor based on analytical power models relative to using peak power

	QP=24		QP=36	
	Frame	GOP	Frame	GOP
D-DVFS	34%	33%	31%	31%
C-DVFS	30%	29%	28%	27%

fabricated ARM processor. Similar as the Intel PM processor, the ARM processor on the TI OMAP35x board only supports a discrete set of voltage and frequency levels, as shown in Table 4.3 [48]. Each pair of voltage and frequency is associated with a CPU operating point (OPP) state. We first experiment with the video decoding on OMAP board using ARM processor for three cases: one is "Performance" which fixes the voltage and clock rate at the maximum value, the second one is "onDemand" which adapts the processor voltage and clock rate at a regular interval (e.g., 156 ms for our OMAP system) based on the measured CPU load [55], and the third one is the DVFS using our proposed complexity prediction method. For the "onDemand" DVFS control on the OMAP system, we have found that the default starting voltage and frequency is 1.27 volt and 550 MHz, which corresponds to OPP 4. If the CPU load is over 80% of the peak load supported by the chosen OPP state in the previous interval, the OPP state will be changed to the next higher level in the current interval. If the CPU load is below 20% of the peak load, the OPP state will be adjusted to the next lower level. Since we only use available discrete voltages (clock rates) supported by ARM processor, the complexity model based DVFS can be treated as experimental D-DVFS (eD-DVFS). Along with video decoding, we measure the voltage and current through ARM processor [5] using Agilent MSO7054A Digital Oscilloscope. This oscilloscope is capable of sampling at 10KHz and storing almost 5 million data points. Figure 6.14 illustrates the test environment where instant voltages are collected by the scope, and the laptop is used to control and command the OMAP board via serial port. The recorded data is transferred via USB from the scope for data post-processing.

Figure 6.15 plots the average power of three experimental cases in video decoding order, for both frame and GOP-based complexity prediction. Note that the DVFS reduces the processor power consumption. According to the simulation results, the power saving factors of our proposed complexity prediction based DVFS are 1.59 compared to the traditional "Performance" scheme, and 1.40 in comparison to the default "OnDemand" solution, for the frame-based complexity prediction, and are 1.61 and 1.42 respectively for

[5] To make our simulation accurate, we disable the DSP core inside OMAP system, and only conduct the video decoding using ARM processor.

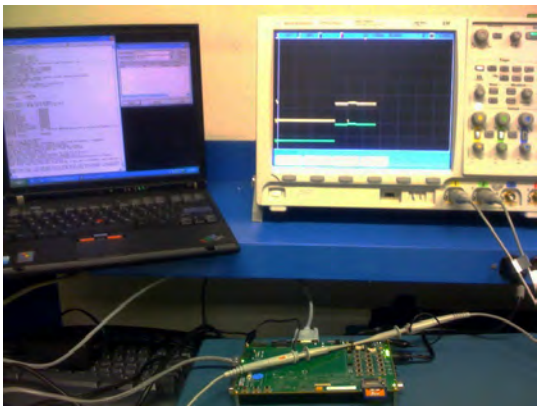


Figure 6.14: Power measurement using Agilent MSO7054A Digital Oscilloscope when conducting video decoding on OMAP board. Voltage probes from scope are connected with jumper J6 on OMAP board to collect the instant voltage and current (via voltage difference over a resistance).

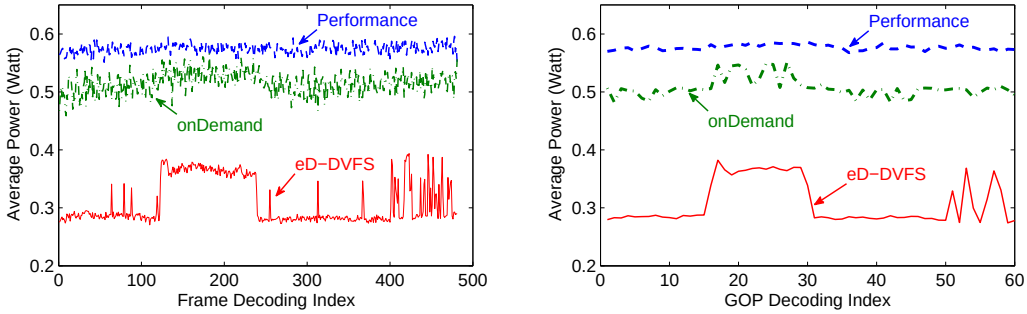
the GOP-based complexity prediction.

To derive analytical power savings, we fit the voltages and clock rates in Table 4.3 for ARM processor, and have found that the voltage and frequency are related by (4.10) with $\phi = 1.69$, $\omega = 6 \times 10^{-16}$ and $\theta = 0.91$, as depicted in Figure 4.6. To evaluate the relation between power and the voltage, we collect the instant power and its corresponding voltage, plot them as scatter points in Figure 4.7, and find its best fit using the power model in (4.11).

Table 6.3 lists the power consumption of analytical D-DVFS and C-DVFS schemes as well as the experimental "OnDemand" and "eD-DVFS" cases relative to the power consumed using "Performance" scheme. Note that our experimental DVFS (eD-DVFS) is very close to the analytical result for D-DVFS [6]. In practice, the processor voltage/frequency transition requires additional power. However, this kind of power dissipation is negligible according to the results of eD-DVFS and analytical D-DVFS in Table 6.3. Ideally, if the processor supports continuous voltage and frequency, and the frequency is set according to the predicted complexity, based on the analytical result obtained with C-DVFS, it is

[6] Here, D-DVFS and C-DVFS are analytical derivations, while eD-DVFS, eD-DVFS(seg) and onDemand are experimental real measurements.

possible to save the power consumption by half approximately, compared with the original "Performance" scheme. The fact that the power savings achievable by experimental measurements (eD-DVFS) and that by analytical derivation (D-DVFS) are very close also suggests that on-chip memory access does not consume significant amount of power. This is because the analytical power saving is derived without including the on-chip memory access energy impact and the measured total power consumption by the CPU, which includes the power consumption due to computation cycles and on-chip memory access.



(a) Average power recorded for frame based video decoding

(b) Average power recorded for GOP based video coding

Figure 6.15: Average power recorded when conducting frame or GOP based video coding on OMAP35x EVM platform for "Performance", "OnDemand" and "eD-DVFS" (experimental D-DVFS) cases.

Table 6.3: Normalized power consumption for ARM processor relative to using peak power

	QP=24		QP=36	
	Frame	GOP	Frame	GOP
onDemand	88%	88%	-	-
eD-DVFS	63%	62%	-	-
D-DVFS	62%	62%	56%	55%
C-DVFS	52%	51%	47%	45%
eD-DVFS(seg)	52%	47%	-	-

As shown above, the difference using DVFS with frame or GOP-based complexity prediction is slight. This is due to the relatively small complexity variation from frame to

frame in the adopted test video. If the decoding complexity changes more rapidly from frame to frame, the GOP based DVFS is expected to provide more power saving. For example, the frame decoding complexity varies more significantly during the "Soccer" period within the simulated concatenated video, i.e., from frame #400 to #480 according to our experimental data. Also, the instant power for the eD-DVFS scheme changes rapidly as presented in Figure 6.15(a). The last row in Table 6.3, i.e., eD-DVFS(seg), provides the average power consumption for this video segment. It is shown that GOP-based method consumes 90% of the power required by the frame-based method. This result is encouraging, as GOP-based complexity prediction and DVFS control not only leads to more power savings, but also requires less computation and bit rate overhead to enable complexity prediction, and involves less frequent adjustment of the processor frequency and voltage. A downside of the GOP-level scheme is that it incurs more delay in video decoding (1 GOP instead of 1 frame, in our case, 1 GOP includes 8 frames). For applications that can accept longer delay, GOP based model is more practical.

The power savings reported so far are for decoding the test video at QP 24. It is expected that at higher QP (and hence lower bit rate), more savings are achievable using DVFS, compared to using the peak power invariably. Specifically, we have coded the same concatenated sequence at QP 36, and estimated the power consumption by the two platforms for decoding this sequence using the same analytical models. The results are also provided in Tables 6.2 and 6.3. The power saving factors obtainable with D-DVFS and C-DVFS increase to 3.23 and 3.7, respectively, for the Intel processor; and become 1.82 and 2.22 for the ARM processor.

6.3 Discussion and Summary

The chapter presents the practical applications using our propose models. First of all, we apply our model to solve the power rate constrained scalable video adaptation, which is a typical problem for wireless video streaming and playback on mobile device, where the network bandwidth and mobile battery capacity are usually bounded. We first

solve the rate constrained problem assuming that the complexity constraint is unlimited (or relaxed). Then, we extend the solution to consider the limited complexity as well. Not only do we provide the theoretical results assuming the continuous quantization stepsize and frame rate, but we also present the results for practical coding system where the temporal scalability is discrete due to the hierarchical dyadic prediction structure. Moreover, in practice, the complexity constraint is also discrete, which is consistent with the reality where the micro-processor only supports limited discrete clock rate (and corresponding voltage). Thus, we believe that our solution can be applied widely to the practical system.

Furthermore, we have presented the DVFS-enabled energy efficient video decoding, where the DVFS is driven by our complexity prediction algorithm. Particularly, we use our video decoding complexity prediction algorithm to guide the voltage and clock rate adjustment of the underlying processor. We have validated our complexity prediction based DVFS algorithm on both Intel PM and ARM processors. Results show that our method has achieved an outstanding performance regarding the power saving. For Intel PM processor, where the dynamic power dominates, we just need 34% power in comparison to the "Performance" method where we assume the maximum clock rate and voltage regardless the actual workload. Meanwhile, we require 62% power to the "Performance" scheme for ARM processor, where the static leakage power can not be ignored. Those results are shown for default limited OPPs of the underlying processor with the frame level complexity prediction. In devices supporting more fine voltages and clock rates, the power saving can be even more. In addition, the saving can be further augmented if we use the GOP level complexity prediction.

Chapter 7

Conclusion and Future Work

7.1 Summary

In this thesis, we model the perceptual quality, rate and power consumption for scalable video with the focus on the joint temporal and amplitude scalability. We apply the "impact separation" methodology to differentiate the impact from temporal resolution (i.e., frame rate t) and amplitude resolution (i.e., quantization stepsize q) so as to construct the analytical models for perceptual quality, rate and power consumption. Each model can be expressed as the product of a function in terms of the frame rate and a function in terms of the quantization stepsize.

First, to develop the perceptual quality metric, we first normalize the perceptual score (i.e., MOS) for each combination for frame rate and quantization by the MOS at the maximum frame rate (i.e., $t_{\max}$) for any given q to obtain the normalized quality versus temporal resolution (NQT), which is also called temporal correction factor for quality (TCFQ). Based on our extensive simulations, we have found a single-parameter inverted exponential function of frame rate that can well predict the NQT curves. On the other hand, curves for normalized quality versus quantization (NQQ) at maximum frame rate can be well captured by another single-parameter exponential function of q as well. Therefore, the overall quality metric considering both temporal and quantization variation can be expressed as a product of maximum quality $Q_{\max}$ (i.e., obtained at maximum frame rate and minimum

quantization stepsize), an exponential function of quantization and an inverted exponential function of frame rate. According to our experiments, we have found that $Q_{\max}$ is fixed for every videos, then our model only has two content dependent parameters. Based on the verification using our test data as well as other data set proposed in the literature, our analytical quality model can well predict the perceptual quality at any combination of temporal and quantization level.

Second, we use two power functions to express the relationship between normalized rate versus temporal resolution (NRT) and normalized rate versus quantization (NRQ), in terms of frame rate and quantization stepsize respectively, where NRT and NRQ can be derived using the same method for quality model development. Specifically, the overall rate model is the product of maximum bit rate $R_{\max}$ encoded using maximum frame rate and minimum quantization, a power function of frame rate and a power function of quantization. In addition to the experimental findings, we also provide the theoretical analysis to show that the power function approximation fits the NRT curves very well. In general, there are three parameters in our rate model including $R_{\max}$ and other two power function related parameters. We have validated our proposed rate model using various videos with different content activities and resolutions. Results show that our model has a very high prediction accuracy.

Third, similar as above, we separate the impact of frame rate and quantization for scalable video decoding complexity model by normalizing the complexity for any combination of frame rate and quantization with respect to points at maximum frame rate and minimum quantization. We have found that normalized complexity versus frame rate (NCT) at any q can be well explained by a linear function. However, normalized complexity versus quantization (NCQ) curves are highly temporal dependent. Therefore, we use the power function with a frame rate dependent parameter to express the NCQ. According to our simulation data using a wide range of videos, our proposed model is very accurate of predicting the actual complexity for scalable video decoding at at any given frame rate and quantization. Furthermore, we extend the complexity model to the power consumption model for the scalable video decoding on ARM processor, by bridging the power consumption as a func-

tion of the instant CPU clock rate, which, in other words, is the the complexity workload demand.

Fourth, in addition to develop the complexity model considering the temporal and amplitude resolution variation. We also propose the complexity prediction model to estimate the frame or GOP decoding cycles along with the video decoding. The estimated cycles are used to dynamically adapt the voltage and clock rate of the underlying DVFS-capable processor so as to save more power. To accurately capture the frame decoding complexity, we decompose the entire decoder into 6 *decoding modules* (DM), each of which is controlled by an unique *complexity unit* (CU). These CUs are defined to ensure that their average complexity (per frame) is either constant or can be easily predicted using previous decoded data. At the current stage, the number of CUs involved (i.e., $N_{\mathtt{CU}}$) are embedded as metadata inside the header field of a certain container (such as FLV, MKV) associated with the video track, so as to inform the decoder to do fine DVFS adjustment. According to our results, such metadata occupy less than 2% for a 96 kbps video stream. The percentage is much smaller for high bit rate video streams. Based on our experimental data, our complexity prediction based DVFS can achieve half power saving for popular ARM processor, which is much better than the default "onDemand" DVFS method provided by many operation systems (OSs).

Fifth, the proposed models will be more useful if we can predict the model parameters accurately using content features, and results show that the parameters are indeed content adaptive. First of all, we abstract a set of content features that can be derived from *residual signal*, e.g., frame difference, displaced frame difference, etc, *motion fields* e.g., as motion vector magnitude and motion direction activity, etc and *original video signal* e.g., as video contrast. To obtain the input raw information, we have implemented a lightweight pre-processor in the video encoding by enabling macroblock based integer motion estimation. Compared with video encoding, the pre-processor is much simplified. According to the results, we have found three-feature combined prediction can estimate the model parameters for CIF videos very well, and two-feature combination is sufficient to estimate the model parameters for WVGA and 720p videos. We do note that the weighted coefficients and fea-

tures are different for different video resolutions even for the same model parameter. We will conduct further investigation in our future research. We also plug the estimated model parameters into our analytical models to compare with the raw data obtained by real experiments. Results show that, with the proper predicted parameters, our model can estimate the simulation data very well.

Moreover, we apply the proposed models to do resource constrained scalable video adaptation. At first, we solve the rate constrained video adaptation problem to maximize the video quality given the limited network bandwidth (i.e., bit rate). We also extend the solution to do power-rate constrained scalable video adaption, where the network bandwidth and mobile battery power are both bounded. Each time, given the bit rate and power consumption constraint, we will calculate a best combination of frame rate and quantization stepsize, which yields the best perceptual quality. Using our models, we can tackle the above problem analytically instead of computational intensive dynamic programming or brute force search. Currently, we can simply embed the features and weighted coefficients [for model parameter prediction] as metadata in the bitstreams to guide the video adaptation at proxy or gateway.

7.2 Future Work

7.2.1 Extension of Proposed Models by Considering the Spatial Scalability

Our proposed models have considered the joint impact of temporal and amplitude scalability provided by the SVC. Practically, we should investigate the quality, rate and complexity variation introduced by spatial scalability, and incorporate the spatial resolution impact into current metrics to establish the complete models $Q(s,t,q)$, $R(s,t,q)$ and $C(s,t,q)$, where s, t and q mean the spatial resolution index (i.e., frame size), temporal resolution (i.e., frame rate) and amplitude level (i.e., quantization stepsize), respectively.

7.2.2 Video Encoder Optimization

As shown in this thesis, we have developed the analytical models, and applied them to guide the video adaptation so as to choose the best extracted point to provide the best video quality. On the other hand, our models can be applied to the encoder as well, where we can use the models to choose the best combination of encoding parameters, such as spatial, temporal and amplitude resolution (STAR), to maximize the perceptual quality of reconstructed video given the network bandwidth R and encoding or decoding power consumption (P) constraints.

7.2.3 Implementation of real-time SVC codec with STAR optimization

Practically speaking, it will be much more useful if we can plug our STAR optimization into the real SVC product, in particular for mobile platform. However, according to our best knowledge, no commercial real-time SVC codec on mobile platform are available in the market so far. Given the popular, fast and well-known open source x264 [56] for H.264/AVC single layer encoder, our group is working on implementing the SVC functionality on top of the x264 code base. Many encoder functions are inherited and reused to improve the encoder efficiency so as to ensure that x264 with SVC tools can still achieve the real-time encoding, in particular on mobile platform, such as ARM. For the decoder part, we have developed C-code based standard compliant SVC decoder for our research. This decoder is platform independent and highly modularized. Our next step is to use the platform dependent instructions, such as "MMX", "SSE" [42] on x386 architecture, and "NEON" [43] on ARM architecture to do the code optimization for speedup. With the real-time SVC codec, we can incorporate our STAR algorithms to guide the encoder optimization and receiver side video adaptation to emulate the real mobile video communication over wireless networks. With our STAR algorithm, we can provide the best video quality under the access network bandwidth and/or remaining battery capacity constraints.

Bibliography

[1] H.264/AVC, *Draft ITU-T Rec. and Final Draft Intl. Std. of Joint Video Spec. (ITU-T Rec. H.264\ISO/IEC 14496-10 AVC) Joint Video Team (JVT)*, Joint Video Team, Doc. JVT-G050, Mar. 2003.

[2] MPEG-2 video, *Generic coding of moving pictures and associated audio information – part 2: video*, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2 video), ITU-T and ISO/IEC JTC 1, Nov. 1994.

[3] H.263, *Video coding for low bit rate communication*, ITU-T Rec. H.263, ITU-T, Version 1: Nov. 1995, Version 2: Jan. 1998, Version 3: Nov. 2000.

[4] MPEG-4 Visual, *Coding of audio-visual objects – Part 2: visual*, ISO/IEC 14492-2 (MPEG-4 Visual), ISO/IEC JTC 1, Version 1: Apr. 1999, Version 2: Feb. 2000, Version 3: May 2004.

[5] M. Wien, H. Schwarz, and T. Oelbaum, "Performance analysis of SVC," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 17, no. 9, pp. 1194–1203, Sept. 2007.

[6] 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*, vol. 17, no. 9, pp. 1103–1120, Sept. 2007.

[7] G. Sullivan, T. Wiegand, and H. Schwarz, *Text of ITU-T Rec. H.264 | ISO/IEC 14496-10:200X/DCOR1 / Amd.3 Scalable video coding*, ISO/IEC JTC1/SC29/WG11, MPEG08/N9574, Antalya, TR, January 2008.

[8] P. Amon, T. Rathgen, and D. Singer, "File format for scalable video coding," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 17, no. 9, pp. 1174–1186, Sept. 2007.

[9] Y.-K. Wang, M. Hannuksela, S. Pateux, A. Eleftheriadis, and S. Wenger, "System and transport interface SVC," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 17, no. 9, pp. 1149–1163, Sept. 2007.

[10] Y.-F. Ou, Z. Ma, and Y. Wang, "Perceptual quality assessment of video considering both frame rate and quantization artifacts," *accepted by IEEE Circuits and Systems for Video Technology*, May 2010.

[11] J. Y. C. Chen and J. E. Thropp, "Review of Low Frame Rate Effects on Human Performance," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 37, pp. 1063–1076, Nov. 2007.

[12] Y. Wang, S.-F. Chang, and A. Loui, "Subjective Preference of Spatio-Temporal Rate in Video Adaptation Using Multi-Dimensional Scalable Coding," in *Proc. of ICME'04*, vol. 3, Jun. 2004, pp. 1719–1722.

[13] G. Yadavalli, M. Masry, and S. S. Hemami, "Frame Rate Preference in Low Bit Rate Video," in *Proc. of ICIP*, vol. 1, Nov. 2003, pp. I–441–4.

[14] J. McCarthy, M. A. Sasse, and D. Miras, "Sharp or smooth ?: Comparing the effects of quantization vs. frame rate for streamed video," in *Proc. of ACM CHI on Human Factors in Computing Systems*, Apr. 2004, pp. 535–542.

[15] Z. Lu, W. Lin, B. C. Seng, S. Kato, S. Yao, E. Ong, and X. K. Yang, "Measuring the Negative Impact of Frame Dropping on Perceptual Visual Quality," in *Proc. SPIE Human Vision and Electronic Imaging*, vol. 5666, Jan. 2005, pp. 554–562.

[16] K.-C. Yang, C. C. Guest, K. El-Maleh, and P. K. Das, "Perceptual Temporal Quality Metric for Compressed Video," *IEEE Trans. on Multimedia*, vol. 9, pp. 1528–1535, Nov. 2007.

[17] H.-T. Quan and M. Ghanbari, "Temporal Aspect of Perceived Quality of Mobile Video Broadcasting," *IEEE Trans. on Broadcasting*, vol. 54, no. 3, pp. 641–651, Sept. 2008.

[18] R. Feghali, D. Wang, F. Speranza, and A. Vincent, "Video quality metric for bit rate control via joint adjustment of quantization and frame rate," *IEEE Trans. on Broadcasting*, vol. 53, no. 1, pp. 441–446, Mar. 2007.

[19] S. H. Jin, C. S. Kim, D. J. Seo, and Y. M. Ro, "Quality Measurement Modeling on Scalable Video Applications," in *Proc. of IEEE Workshop on Multimedia Signal Processing*, Otc. 2007, pp. 131 – 134.

[20] E. Ong, X. Yang, W. Lin, Z. Lu, and S. Yao, "Perceptual Quality Metric For Compressed Videos," in *Proc. of ICASSP*, vol. 2, Mar. 2005, pp. 581 – 584.

[21] Joint Video Team, "http://wftp3.itu.int/av-arch/jvt-site/."

[22] JSVM software, *Joint Scalable Video Model*, Joint Video Team, Doc. JVT-X203, Geneva, Switzerland, 29 June - 5 July 2007.

[23] M. H. Pinson and S. Wolf, "Techniques for evaluating objective video quality models using overlapping subjective data sets," NITA, Tech. Rep. TR-09-457, Nov. 2008.

[24] *ITU-T Rec. P. 910: Subjective video quality assessment methods for multimedia applications*, 1999.

[25] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang, "Modeling The Impact of Frame Rate on Perceptual Quality of Video," in *Proc. of ICIP*, San Diego, Oct. 2008, pp. 689 – 692.

[26] Y.-F. Ou, Z. Ma, and Y. Wang, "A novel quality metric for compressed video considering both frame rate and quantization artifacts," in *Proc. of Intl. Workshop Video Processing and Quality Metrics for Consumer (VPQM)*, Scottsdale, AZ, Jan. 2009.

[27] S. Wolf and M. Pinson, *Video quality measurement techniques*, NTIA, Tech. Report 02-392, June 2002.

[28] W. Ding and B. Liu, "Rate control of MPEG video coding and recoding by rate-quantization modeling," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 6, pp. 12–20, Feb. 1996.

[29] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 7, no. 2, pp. 246–250, Feb. 1997.

[30] T. Chiang, H.-J. Lee, and H. Sun, "An overview of the encoding tools in the MPEG-4 reference software," in *Proc. of IEEE Intl. Symp. Circuit and Systems*, Geneva, Switzerland, May 28 -31 2000.

[31] J. Ribas-Corbera and S. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 9, no. 2, pp. 172–185, Feb. 1999.

[32] Z. He and S. K. Mitra, "A novel linear source model and a unified rate control algorithm for H.264/MPEG-2/MPEG-4," in *Proc. of Intl. Conf. Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, May 2001.

[33] T. Berger, *Rate-distortion theory*. John Wiley & Sons, Inc., 15 April 2003.

[34] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Prentice Hall, 2002.

[35] Y. Wang and S. Lin, "Error-Resilient Video Coding using Multiple Description Motion-Compensation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 438–452, June 2002.

[36] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704–716, July 2003.

[37] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 15, no. 5, pp. 645–659, May 2005.

[38] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proc. of IEEE ICME*, 2006.

[39] Z. Ma, H. Hu, and Y. Wang, "On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding," *submitted to IEEE Trans. Circuit and Systems for Video Technology*, 2010.

[40] IA-32 Optimization, *Intel 64 and IA-32 architectures optimization reference manual*, Intel Coporation, Nov. 2007.

[41] ARM optimization, *ARM-cortex A8 NEON optimization manual*, ARM Limited, 2007.

[42] Intel Pentium Mobile Processor. [Online]. Available: http://www.intel.com/design/intarch/pentiumm/pentiumm.htm

[43] ARM Cortex A8 processor. [Online]. Available: http://www.arm.com/products/CPUs/ARM_Cortex-A8.html

[44] H. Hu, L. Lu, Z. Ma, and Y. Wang, "Complexity profiler design for Intel and ARM architecture," Video Lab, Dept. of ECE, Polytehnic Insititute of NYU, Tech. Rep., 2009.

[45] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571 – 1580, Nov. 2000.

[46] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real Time Embedded Systems," in *Proc. of 41st Annual Conf. on Design Automation*, 2004.

[47] J. M. Rabaey, *Digital Integrated Circuits*. Prentice Hal, 1996.

[48] TI OMAP35x EVM. [Online]. Available: http://focus.ti.com/docs/toolsw/folders/print/tmdsevm3530.html

[49] Z. Ma, Z. Zhang, and Y. Wang, "Complexity Modeling of H.264 Entropy Decoding," in *Proc. of ICIP*, 2008.

[50] M. Zhou and R. Talluri, *Handbook of image and video processing*, 2nd ed. Elsevier Academic Press, 2005, ch. Embedded Video Codec.

[51] H. Schwarz, D. Marpe, and T. Wiegand, *Hierarchical B Pictures*, Joint Video Team, Doc. JVT-P014, Poznan, Poland, July 2005.

[52] P. List, A.Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking fitler," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 614–619, July 2003.

[53] R. Peng and M. Pedram, "An analytical model for predicting the remaining battery capacity of lithium-ion batteries," *IEEE Trans. on VLSI Systems*, vol. 14, no. 5, pp. 441 – 451, May 2006.

[54] Z. Ma, "Proof of monotonicity for $Q(q,t)$, $R(q,t)$ and $C(q,t)$," Video Lab, Polytehnic Insititute of NYU, http://vision.poly.edu/ zma03/model_monotoni_proof.pdf, Tech. Rep., December 2010.

[55] cpuFreq governor. [Online]. Available: http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt

[56] x264 - a free H.264/AVC encoder. [Online]. Available: http://www.videolan.org/developers/x264.html