

# UC San Diego

## UC San Diego Electronic Theses and Dissertations

### Title

Modeling and Enhancing Quality of Service for Cloud Mobile Media Applications

### Permalink

<https://escholarship.org/uc/item/0td32248>

### Author

Liu, Yao

### Publication Date

2015

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

**Modeling and Enhancing Quality of Service for Cloud Mobile Media Applications**

A dissertation submitted in partial satisfaction of the  
requirements for the degree  
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Yao Liu

Committee in charge:

Professor Sujit Dey, Chair  
Professor Pamela Cosman  
Professor Truong Nguyen  
Professor Alex Snoeren  
Professor Geoffrey Voelker

2015

Copyright

Yao Liu, 2015

All rights reserved.

This Dissertation of Yao Liu is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

---

---

---

---

---

Chair

University of California, San Diego

2015

## TABLE OF CONTENTS

<b>Signature Page .....</b>	<b>iii</b>
<b>Table of Contents .....</b>	<b>iv</b>
<b>List of Figures .....</b>	<b>vii</b>
<b>List of Tables.....</b>	<b>xi</b>
<b>Acknowledgements.....</b>	<b>xiii</b>
<b>Vita .....</b>	<b>xv</b>
<b>Abstract of the Dissertation.....</b>	<b>xvii</b>
<b>Chapter 1. Introduction.....</b>	<b>1</b>
1.1 Cloud Mobile Media Architecture and Applications .....	3
1.2 Challenges of Cloud Mobile Media Applications .....	7
1.2.1 Impact of Mobile Network Factors .....	7
1.2.2 Monitor and Guarantee Good User Experience. ....	9
1.3 Contribution and Overview .....	10
<b>Chapter 2. Modeling User Experience in Cloud Mobile Rendering Application .....</b>	<b>13</b>
2.1 Factors Affecting User Experience of Cloud Mobile Rendering Application .....	14
2.2 Derivation and Validation of Impairment Functions for Rendering Factors.....	16
2.2.1 Impairment Function for Each Rendering Factor.....	17
2.2.2 Subjective Quality Assessment Experiment .....	18
2.2.3 Derivation of Impairment Function $I_{VD}$ , $I_{RE}$ , $I_{TD}$ .....	19
2.2.4 Validation of Impairment Function $I_R$ .....	24
2.3 Derivation of Cloud Mobile Rendering User Experience Model.....	25
2.4 Conclusion.....	30
<b>Chapter 3. Content-Aware Adaptive Rendering Algorithm .....</b>	<b>32</b>
3.1 Tradeoff Between Rendering Richness and Video Frame Quality .....	33
3.2 Content-Aware Adaptive Rendering Algorithm .....	35

3.2.1	Offline Step: Automatically Characterizing the $I_E$ - $I_R$ Relation Model.....	36
3.2.2	Online Step I: Determine Optimal $I_R$ .....	40
3.2.3	Online Step II: Determine Optimal Rendering Setting .....	43
3.3	Experimental Result in Real Wireless Network.....	46
3.4	Conclusion.....	50
<b>Chapter 4. Enhancing Video Encoding for Cloud Mobile Rendering Application Using Rendering Information .....</b>		<b>52</b>
4.1	Introduction .....	53
4.2	Related Work.....	55
4.3	Rendering-based Prioritized Encoding Technique.....	58
4.3.1	Overview of Rendering-based Prioritized Encoding Technique .....	59
4.3.2	Extraction of Rendering Information .....	60
4.3.3	Automatic Importance Computing using Rendering Information.....	63
4.3.4	Rate Allocation Algorithm .....	69
4.3.5	Subjective Assessment Experiments .....	72
4.4	Rendering-based Encoding Acceleration Technique .....	75
4.4.1	Direct Calculation of Motion Vectors .....	77
4.4.2	Fast Mode Selection Algorithm .....	81
4.4.3	Assessment Experiment .....	86
4.5	Conclusion.....	94
<b>Chapter 5. Deriving and Validation User Experience Model for DASH Video Streaming .....</b>		<b>96</b>
5.1	Introduction .....	97
5.2	Factors Affecting User Experience for DASH Video .....	101
5.3	Test Video Generation.....	103
5.3.1	DASH Video Streaming Characterization Experiments .....	104
5.3.2	Generated Test Cases for Round I Subjective Tests.....	107
5.4	Derivation of Impairment Functions .....	111
5.4.1	Round I Subjective Experiments.....	111
5.4.2	Impairment Function for Initial Delay .....	112
5.4.3	Impairment Function for Stall .....	113

5.4.4 Impairment Function for Level Variation .....	118
5.5 Overall User Experience Model .....	124
5.5.1 Second Round of Subjective Test.....	125
5.5.2 Model Derivation and Validation.....	127
5.6 Application of DASH-UE Model to Long Videos .....	130
5.7 Conclusion.....	135
<b>Chapter 6. Conclusion and Future Direction.....</b>	<b>137</b>
<b>Bibliography .....</b>	<b>140</b>

## LIST OF FIGURES

Figure 1.1.	Cloud Mobile Media architecture, showing control and data flows. ....	3
Figure 1.2.	Delay and response time measured in a CMG session at different time during a day. ....	8
Figure 1.3.	Mobile bandwidth trace and DASH video quality variation patterns. ....	9
Figure 2.1.	Objective and subjective factors affecting user experience of CMR applications. ....	15
Figure 2.2.	Graphic quality difference of different settings of View Distance and Texture Detail: (a) 300m view and high texture detail; (b) 60m view and high texture detail; (c) 300m view and medium texture detail; (d) 300m view and low texture. ....	15
Figure 2.3.	Experiment framework.....	18
Figure 2.4.	Relation between $I_{VD}$ and view distance for different game scenes: (a) screenshot of scene 1; (b) screenshot of scene 2; (c) relationship between subjects' evaluation $I_{VD}$ and view distance. . ....	20
Figure 2.5.	Relation between subjects' evaluation $I_{VD}$ and $F_{render}$ . ....	22
Figure 2.6.	Relation between subjects' average evaluation $I_{VD}$ and $F_{render}$ . ....	22
Figure 2.7.	Relationship between predicted and subjective $I_R$ value: (a) new model which includes video content information; (b) old model which doesn't consider video content information. ....	24
Figure 2.8.	Relation between predicted and subjective CMR-MOS for game PlaneShift: (a) old model, not including rendering impairment; (b) proposed model, including rendering impairment. ....	28
Figure 2.9.	Evaluating CMR-UE model on game Broadsides: (a) screenshot of game Broadsides; (b) relation between predicted and subjective CMR-MOS. ....	29
Figure 3.1.	Comparison of video frame quality for different rendering setting with 300kbps bit rate: (a) view distance = 150m; (b) view distance = 70m. ....	34



Figure 3.2.	Overview of content-aware adaptive rendering algorithm. ....	35
Figure 3.3.	$I_E$ , $I_R$ and CMR-MOS for different view distances used to render and encode a 200kbps CMR video. ...	38
Figure 3.4.	Relation between $I_R$ and $I_E$ under different bit rate targets....	38
Figure 3.5.	Recursive searching approach to obtain optimal $I_R$ .....	42
Figure 3.6.	Flowchart of the online steps for adaptive rendering algorithm.....	44
Figure 3.7.	User experience calculated using CMR-UE model on a representative 3.5G network for game PlaneShift. ....	47
Figure 4.1.	Overview of rendering-based prioritized encoding technique. ....	59
Figure 4.2.	Game frame and rendering information: (a) game frame of PlaneShift; (b) values stored in Z-buffer; (c) values stored in Stencil buffer; (d) illustration of distance-based saliency.. ....	61
Figure 4.3.	Game frame and combined saliency map ( $S_{combined}$ ) for different $\alpha$ values. ...	68
Figure 4.4.	Encoded video frames without and with rendering-based prioritized encoding, under the video encoding bit rate budget of 1000 kbps. ....	73
Figure 4.5.	Subjective evaluation of two games under different encoding bit rates: (a) for game PlaneShift; (b) for game Broadsides. ....	74
Figure 4.6.	Two consecutive frames of game Broadsides and the corresponding uncovered pixels: (a) the previous frame (reference frame); (b) the latter frame (current frame to be encoded); (c) black pixels correspond to the uncovered pixels detected using the Stencil buffer. ....	79
Figure 4.7.	Different partitions for 16x16 MB:(a) 16x16;(b) 16x8;(c) 8x16;(d) 8x8.....	82
Figure 4.8.	Game frame for Broadsides and the optimal MB mode selected with full mode RDO. ...	84
Figure 4.9.	Flowchart of the proposed fast mode selection algorithm.....	85
Figure 4.10.	ME time per frame using different ME approaches.....	89
Figure 4.11.	Relation between encoding time per frame and bit rate for video sequence BOH. ....	90

Figure 4.12.	Relation between PSNR and bit rate for video sequence BOH. ....	91
Figure 5.1.	Factors affecting user experience of DASH video. ....	102
Figure 5.2.	Level variation pattern. ....	102
Figure 5.3.	Testbed of DASH video streaming characterization experiments.. ....	104
Figure 5.4.	Results for characterization experiments: (1) purple: network bandwidth; (2) green: segments download bit rate; (3) yellow: video bit rate... ....	105
Figure 5.5.	Distribution of initial delay among 20 streaming sessions. ....	106
Figure 5.6.	Results for stall: (a) distribution of total stall duration; (b) distribution of number of stalls. ....	106
Figure 5.7.	Distribution of level variation: (a) average level; (b) number of switches; (c) average switch magnitude. ....	106
Figure 5.8.	Snapshot of test videos. ....	108
Figure 5.9.	Special frames used to simulate initial delay and stall: (a) 'buffering' frame for simulating initial delay; (b) 'loading' frame for simulating stall. ....	109
Figure 5.10.	Test videos for level variation factor. ....	110
Figure 5.11.	Relationship between impairment and initial delay. ....	113
Figure 5.12.	Subjective stall impairment results for different video contents, stall duration and stall number. ....	113
Figure 5.13.	Motion of video content: (a) $\overline{MVM}$ value of each frame; (b) AMVM for the whole video. ....	115
Figure 5.14.	Relation between $I_{ST}$ and video motion for different stall distribution. ....	116
Figure 5.15.	Relationship between VQM value and bit rate.. ....	120
Figure 5.16.	$D_i$ values and VQM values for a 20-second DASH video. ....	121
Figure 5.17.	Relationship between subjective and objective impairments: (a) first round of validation for $I_{LV}$ ; (b) second round of validation for $I_{LV}$ . ....	123
Figure 5.18.	Relation between subjective R scores and $I_{ST}$ and $I_{LV}$ . ....	127
Figure 5.19.	Subjective R score under different initial delay, stall and level variation. ....	128

Figure 5.20.	Relation between predicted and subjective DASH-MOS. ....	129
Figure 5.21.	Impairment values of each minute of video. ....	131
Figure 5.22.	Subjective evaluations of initial delay impairment for different video length. .....	133
Figure 5.23.	Relation between subjective R scores and predicted scores using two approaches for test videos.. ....	134
Figure 5.24.	Trace of predicted R scores for 15-minute Soccer test video. ....	134

## LIST OF TABLES

Table 1.1.	Analysis of different categories of cloud mobile media application.....	5
Table 2.1.	Graphic quality rating criteria and $I_R$ values.. .....	17
Table 2.2.	Parameters for subjective experiment. ....	19
Table 2.3.	Impairment function for realistic effect. ....	23
Table 2.4.	Impairment function for texture detail. ....	23
Table 2.5.	CMR-MOS rating and R values.....	27
Table 2.6.	Compensation coefficients in equation (2.5) for game PlaneShift.....	28
Table 2.7.	Compensation coefficients in equation (2.5) for game Broadsides.....	29
Table 3.1.	Parameters for offline experiments. . ....	37
Table 3.2.	Coefficient values for the $I_E$ - $I_R$ model for game PlaneShift. ....	39
Table 4.1.	Parameters for video coding. ....	71
Table 4.2.	Candidate inter-mode for each MB category. ....	84
Table 4.3.	Description of experiment videos.. . ....	87
Table 4.4.	Average PSNR difference for test videos. . ....	89
Table 4.5.	Average performance comparison of the three fast encoding techniques. ....	93
Table 5.1.	Encoding settings for steaming Vancouver Olympics. . ....	98
Table 5.2.	Video content description. ....	108
Table 5.3.	Test cases for initial delay. ....	109
Table 5.4.	Test cases for stall. . ....	109
Table 5.5.	Rating criteria for video quality. . ....	111
Table 5.6.	Values of coefficients. ....	114
Table 5.7.	Subjective experiment results for different stall duration and stall number for video Bunny.. . ....	116
Table 5.8.	Subjective experiment results for level variation tests.....	118
Table 5.9.	Parameters for second round of subjective test. ....	126

Table 5.10. Two approaches to compute user experience of long video. ....131

## ACKNOWLEDGEMENTS

This dissertation would not have been completed without the guidance and support of many people. I take this opportunity to express my sincere gratitude for them.

The first people I would like to thank is my advisor, Prof. Sujit Dey, who has been a constant source support over these years. I thank him for being extremely patient with me, for teaching me how to conduct research and present ideas, and for being enthusiastic in venturing into new topics of research. His invaluable support and aspiring guidance have provided me with the ability to complete this dissertation.

Secondly, I would like to thank Jason Lu and Shaoxuan Wang, my collaborator and friend, for their contributions, insights and feedback. I also thank the members of my thesis committee, Professors Truong Nguyen, Pamela Cosman, Geoffrey Voelker, and Alex Snoeren for providing valuable suggestions and feedback. I also gratefully acknowledge the financial support provided by the ECE department that made my doctoral studies possible. Throughout my graduate studies, I have been fortunate to have the chance to work with a wonderful group of co-workers in the MESDAT lab at UCSD. I am very grateful to all the past and present members of our lab for providing an interesting and encouraging working experience. Our interactions and discussions made the day-to-day graduate student life enjoyable.

I dedicate this dissertation to my parents. I would not have been able to dream of embarking upon this journey if it were not for their encouragement and support. They have always been there during my weak moments to give me the strength and courage to overcome all the obstacles.

The text of the following chapters, in part or in full, is based on material that has been published in journals or conference proceedings.

Chapter 2 and 3 are based on material that has been published in IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS) (Y. Liu, S. Wang, S. Dey, "Content-Aware Modeling and Enhancing User Experience in Cloud Mobile Rendering and Streaming", JETCAS, Mar. 2014.) and material published in IEEE International Conference on Computing, Networking and Communications (ICNC) (Y. Liu, S. Wang, S. Dey, "Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering", ICNC, Jan. 2012.).

Chapter 4 is based on material that has been published in IEEE Transactions on Circuits and Systems on Video Technology (TCSVT) (Y. Liu, Y. Lu, S. Dey, "Enhancing Video Encoding for Cloud Gaming Using Rendering Information", IEEE TCSVT, Jun. 2015).

Chapter 5 is based on material that has been published in IEEE Transactions on Broadcasting (Y. Liu, S. Dey, M. Ulupinar, M. Luby, Y. Mao, "Deriving and Validating User Experience Model for DASH Video Streaming"), and Packet Video Workshop (Y. Liu, S. Dey, D. Gillies, F. Ulupinar, M. Luby, "User Experience Modeling for DASH Video", IEEE PV, Dec. 2013).

I was the primary researcher and author of each of the above publications, and the co-authors listed in these publications collaborated on, or supervised the research which forms the basis for these chapters.

## VITA

- 2005-2009            B.S., Electrical Engineering,  
                          Huazhong University of Science & Technology, Wuhan, China
- 2009-2011            M.S., Electrical Engineering (Communication Theory and System),  
                          University of California, San Diego
- 2010                    Summer Intern, Qualcomm Inc.,  
                          Beijing, China
- 2011-2015            Research Assistant, Electrical and Computer Engineering,  
                          University of California, San Diego
- 2013                    Summer Intern, Yahoo Inc.,  
                          Sunnyvale, California
- 2015                    Research Scientist, Facebook Inc.,  
                          Menlo Park, California
- 2015                    Ph.D, Electrical Engineering (Computer Engineering),  
                          University of California, San Diego

## PUBLICATIONS

- Y. Liu, S. Dey, F. Ulupinar, M. Luby, Y. Mao, "Deriving and Validating User Experience Model for DASH Video Streaming", *IEEE Transactions on Broadcasting*, Volume PP, Issue 99, Aug. 2015
- Y. Liu, Y. Lu, S. Dey, "Enhancing Video Encoding for Cloud Gaming Using Rendering Information", *IEEE Transactions on Circuits and Systems on Video Technology*, Volume PP, Issue 99, Jun. 2015
- Y. Liu, S. Wang, S. Dey, "Content-Aware Modeling and Enhancing User Experience in Cloud Mobile Rendering and Streaming", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, VOL.4, NO.1, Mar. 2014.



Y. Liu, S. Dey, D. Gillies, F. Ulupinar, M. Luby, "User Experience Modeling for DASH Video", in *Proc. of Packet Video Workshop (PV2013)*, San Jose, Dec. 2013.

S. Dey, Yao Liu, S. Wang, Y. Lu, "Addressing Response Time for Cloud-based Mobile Applications", in *Proc. of ACM MobileCloud 2013*, Bangalore, India, Jul. 2013.

Y. Liu, S. Wang, S. Dey, "Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering", in *Proc. of IEEE ICNC, Maui*, Jan. 2012.

Y. Lu, Y. Liu, S. Dey, "Modeling and Optimizing Cloud Mobile 3D Display Gaming User Experience by Asymmetric Graphics Rendering", in *IEEE Journal of Selected Topics in Signal Processing*, Volume 9, Issue 3, pp 1-16, April, 2015.

Y. Lu, Y. Liu and S. Dey, "A Joint Asymmetric Graphics Rendering and Video Encoding Approach for Optimizing Cloud Mobile 3D Display Gaming User Experience." in *Proc. of IEEE International Symposium on Multimedia (ISM'15)*, Miami, Dec. 2015.

Y. Lu, Y. Liu, S. Dey, "Optimizing Cloud Mobile 3D Display Gaming User Experience by Asymmetric Object of Interest Rendering", in *Proc. of the International Conference on Communications (ICC)*, London, United Kingdom, June 2015.

Y. Lu, Y. Liu, S. Dey, "Enhancing Cloud Mobile 3D Display Gaming User Experience by Asymmetric Graphics Rendering", in *Proc. of IEEE ICNC2014 - Multimedia Application Symposium*, Hawaii, USA, Jan. 2014.

S. Wang, Y. Liu, S. Dey, "Wireless Network Aware Cloud Scheduler for Scalable Cloud Mobile Gaming", in *Proc. of IEEE International Conference on Communications (ICC'12)*, Ottawa, Jun. 2012.

ABSTRACT OF THE DISSERTATION

**Modeling and Enhancing Quality of Service for Cloud Mobile Media Applications**

by

Yao Liu

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2015

Professor Sujit Dey, Chair

Over the past few years, there has been an increased number of media applications that have migrated to the cloud, which will enable mobile users to engage in rich media experiences from any mobile platform. In this research, we develop techniques to model and enhance the quality of service for two cloud mobile media applications: Cloud Mobile Rendering (CMR) application and HTTP-based adaptive video streaming.

In CMR application, compute intensive graphic rendering is performed on cloud servers, and the rendered video is encoded and transmitted through wireless network to mobile

devices. Although promising to address the computation and battery limits of mobile devices, this approach suffers from challenges posed by limited and unstable wireless bandwidth, such that it is difficult to ensure the quality of service. In this research, we first develop a model to quantitatively measure the user experience of CMR application, considering rendering factors, encoding factors and network factors simultaneously. Secondly we derive several techniques to enhance the quality of service, including a) an adaptive rendering technique which dynamically select the optimal rendering factor based on network condition; b) a prioritized encoding technique which allocates bits among different regions of video frame depending on their priorities; c) an encoding acceleration technique which utilizes rendering information to reduce the computation complexity of video encoding while maintaining the video quality.

In HTTP-based adaptive video streaming, video content are split into a sequence of small segments, which are encoded into several versions with different bit rates and quality, and streamed according to the requests from streaming client. The version of video segments will keep varying according to the available bandwidth. Fluctuating bandwidth will lead to temporal artifacts such as video rebuffering (freezing), and spatial artifacts such as visual quality variation. In this thesis, we develop the first user experience model for this application through extensive subjective experiments. We validate with high accuracy the user experience model, and demonstrates its applicability to video with different length and motion characteristics.

# Chapter 1

## Introduction

Over the last several years, there has been a large number of applications that have “migrated to the cloud”, and cloud-based applications have become very popular. Most of the early adopters of cloud are enterprise applications and IT departments; according to Juniper Research, revenue from mobile enterprise cloud-based applications and services is expected to rise from nearly \$2.6 billion in 2011 to \$39 billion in 2016 [JUN11].

Similar motivations that have driven mobile enterprise cloud services are also driving adoption of mobile consumer cloud services: the ability to access media from anywhere: any platform, and network. According to Juniper Research, revenues from consumer cloud mobility services, initially driven by cloud based music and video storage and download services like the ones launched by Apple’s iCloud, are expected to reach \$6.5 billion per year by 2016 [JUN11].

In addition to such download and storage services, a big increase to mobile consumer cloud services will come from a major shift in the mobile applications market, from primarily native applications to ones based on mobile cloud computing: utilizing the computing and storage resources available in the cloud, thereby enabling the use of cutting edge multimedia

technologies that are much more computing and storage intensive than what mobile devices can offer, and thus enabling much richer media experiences than what current native applications can offer. Furthermore, mobile cloud computing based applications can simultaneously avail of not only cloud resources, but also the unique resources of mobile devices, like user location and device sensors, that will make such applications more powerful than either server or PC-based applications, or current native mobile applications.

In this thesis, we focus on Cloud Mobile Media (CMM) applications, which will enable mobile users to not only access rich media from any mobile device and platform, but even more importantly, which will enable mobile users to engage in new, rich media experiences, through the use of mobile cloud computing, that are not possible otherwise from their mobile devices. CMM will also enable service providers and network operators to offer services much more efficiently, with lower cost and better user experience. As more consumers adopt smartphones and tablets as one of their primary media experience platforms, CMM has the potential of significantly boosting the revenue of cloud Software-as-a-Service (SaaS) providers. Some of the media rich CMM applications will require new and richer platform and infrastructure capabilities as explained in the next sections, thereby providing a new set of revenue opportunities for cloud platform and infrastructure providers. And finally, CMM offers new opportunities for mobile network operators to close the growing gap between growth in data usage and data revenue by offering innovative CMM services and experiences, outside of conventional application stores where their participation has not been strong so far.

In this chapter, we first present the overall architecture of CMM applications, and then discuss different types of possible CMM applications, including their advantages. Subsequently we elaborate on a few major challenges of CMM application and briefly explain

the techniques we propose to solve them. Finally, we outline the contributions made by this thesis, and conclude with an overview of the remaining chapters.

## 1.1 Cloud Mobile Mobile Media Architecture and Applications

Utilizing available cloud computing and storage resources, we expect a heterogeneous set of Cloud Mobile Media services and applications to emerge, with different types of consumer experiences and advantage enabled. In this section, we first describe the typical end-to-end control and data flow architecture of CMM applications. Next, we categorize the existing and expected CMM applications, and analyze for each category the cloud infrastructure and platform needs, advantages and user experiences enabled, and challenges to make the applications successful.

Figure 1.1 shows the overall architecture, including end-to-end flow of control and data between the mobile devices and the Internet cloud servers, for a typical CMM application. A typical CMM application has a small footprint client on the mobile device, which provides

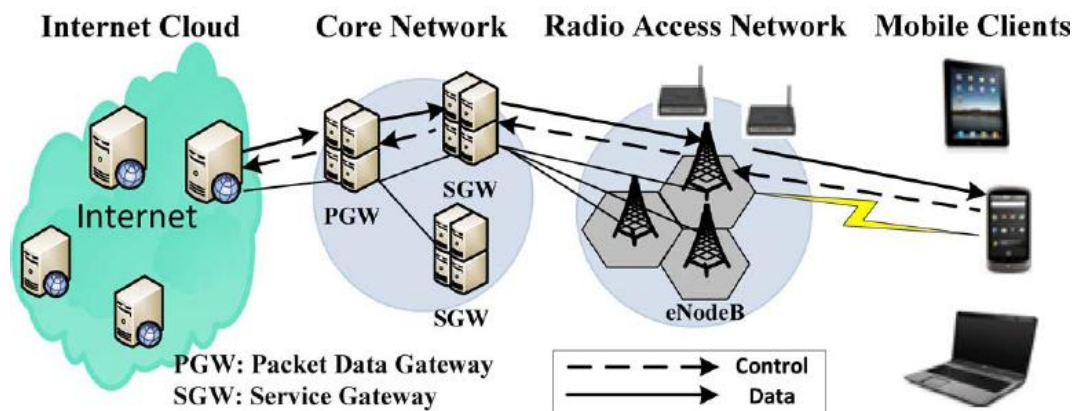


Figure 1.1. Cloud Mobile Media architecture, showing control and data flows.

the appropriate user interfaces (gesture, touchscreen, voice, text based) to enable the user to interact with the application. The resulting control commands are transmitted uplink through cellular Radio Access Networks (RAN) or WiFi Access Points to appropriate gateways located in operator Core Networks (CN), and finally to the Internet cloud. Subsequently, the multimedia data produced by the cloud, either as a result of processing using the cloud computing resources, and/or retrieval from cloud storage resources, is transmitted downlink through the CN and RAN back to the mobile device. The CMM client then decodes and displays the results on the mobile device display. From the above description, and as shown in figure 1.1, a typical CMM application will be highly interactive, with some types of applications needing near real-time response times.

Table 1.1 summarizes three main categories of mobile multimedia applications that already are, or can potentially be, driven by the use of the cloud, including storage based applications, audio and video streaming applications, and rich rendering based applications like cloud mobile gaming (CMG) and augmented reality. For each category of CMM applications, we list the IaaS and PaaS features that will be needed. We also list the advantages of each category of CMM applications, including what multimedia experience can be enabled that cannot be supported currently, and the challenges that need to be addressed to make the application category successful.

As discussed before, **Mobile Cloud Storage** is the most commonly used category of CMM application/service today, with offerings from Amazon, Apple, Dropbox, etc. These services provide diverse capabilities, including storing documents, photos, music and video in the cloud, accessing media from any device anywhere irrespective of the source of the media and/or the device/platform used to generate the media, and synchronizing data/media across multiple devices a typical user owns. To enable mass adoption of such services, the PaaS

providers will need to ensure high availability and integrity of data, and the SaaS provider will need to ensure content security and user privacy.

The **Cloud Mobile Rendering** category is expected to be a rapidly growing segment of mobile multimedia applications, including mobile gaming, virtual reality, telemedicine, augmented reality, etc. Despite the progress in the capabilities of mobile devices, there is still a big gap with the growing requirements of the latest 3D and multi-view rendering techniques and that can be supported by today's and near future mobile devices, including tablets. Cloud based Rendering can bridge this gap by allowing rendering to be executed in the cloud, instead of on the mobile device, thereby potentially enabling mobile users to play the same rich Internet games available to high-end PC users, or participate in rich augmented reality and/or multi-view immersive experiences that are being developed primarily with PC users in mind. Moreover, analogous to transcoding, we expect a new capability to emerge that we term trans-rendering—the ability to automatically adjust rendering to different device and platform capabilities, thus relieving game and augmented reality application developers from the

Table 1.1. Analysis of different categories of potential cloud mobile media applications.

	<b>Mobile Cloud Storage</b>	<b>Cloud Mobile Rendering</b>	<b>Video/audio Streaming</b>
<b>Sample applications/ services enabled on mobile devices</b>	Storing and accessing Photos, Music, Files	Mobile gaming, virtual reality, telemedicine, augmented reality	Streaming video/audio, cloud DVR
<b>IaaS, PaaS features needed</b>	Cloud storage with high availability and integrity	Multiple-core GPU, efficient cloud rendering	Cloud transcoding, Transrating, Caching
<b>Advantages</b>	Ubiquitous access from any device, synchronizing between devices	Enables highest quality rendering, multi-platform	Low capex, high scalability with demand
<b>Challenges</b>	Ensure content security, privacy	Response time, user experience, cloud service cost	Response time, video quality



expensive cycle of developing device and platform specific mobile versions. The ability to enable such rich experiences on all mobile devices and platforms, coupled with the inherent advantages of ubiquity and location information associated with the use of mobile devices, will have the potential to drive a new generation of cloud based mobile media applications.

As summarized in Table 1.1, enabling cloud mobile rendering applications will require additional capabilities from cloud infrastructure and platforms, as well as addressing some significant challenges. Clouds will need to include architectural provisions, like massively multi-core GPUs, and software support for developing highly concurrent cloud based rendering applications like cloud gaming engines, as each cloud rendering session will need to be supported by a separate cloud rendering instance. Moreover, in applications like Cloud Mobile Gaming (CMG), in response to gaming commands from the mobile device, not only does rendering need to be performed in the cloud, but the rendered video will need to be encoded and delivered over wireless networks to the mobile device, all in near real time, as user experience for such applications is highly dominated by fast response time. Also, the need to transmit the rendered video for each rendering/gaming session can mean significant additional bandwidth cost leading to high operating expenses for the service provider, significant additional traffic on the wireless networks possibly leading to overloading of the networks, loss of video quality, and additional cost for mobile subscribers due to tiered data plans.

**Video and audio streaming based services** can benefit by utilizing cloud computing resources to perform compute intensive tasks of encoding, and transcoding and transrating needed to adjust to different devices and networks. For on-demand video, computing costs can be reduced by caching popular videos at different resolutions and bit rates. Besides lower initial capital expenses, the advantage of cloud based video/audio services is the use of

elasticity in cloud computing resources to more cost-effectively handle variable peak demands. However, to support the expected increase in demand for mobile video, the cloud will need to provide server architectures and tools that can enable massively concurrent transcoding/transrating implementations, at efficiencies and cost points that can be enabled today with custom hardware solutions. The concern is particularly critical for the use of public clouds to offer video services, as current public cloud capabilities and price structures, including network bandwidth costs, may make operating expenses of video services very high.

## **1.2 Challenges of Cloud Mobile Media Applications**

Though the CMM applications are promising in terms of enabling mobile users to have rich and immersive media experience on any mobile device, several challenges exist and must be addressed in order to make them feasible.

### **1.2.1 Impact of Mobile Network Factors**

It has been well established that wireless networks are characterized by rapid fluctuations of the network bandwidth experienced by users. For example, in 3G and 4G cellular networks, while various techniques significantly reduce channel errors experienced by applications, they produce rapidly changing channel rates, leading to significant spatial and temporal variations of the mobile network bandwidth.

This inherent characteristic (fluctuating and constrained network bandwidth) of the from cloud servers to mobile devices can be subject to high and unpredictable congestion delay and packet loss, leading to an undesirable increase in response time, besides the adverse impact on the quality of the video streamed. To study the impact of wireless networks on the quality of CMM applications, we conducted experiments with a Cloud Mobile Gaming (CMG)

application we have developed, which as described in Section 1.1 is a highly interactive cloud based rendering application: gaming commands are transmitted uplink from the mobile device to the cloud servers, and the rendered video needs to be streamed downlink from the server to the mobile client in near real time. Since this application is highly sensitive to response time, we measured uplink delay, downlink delay, and round-trip response time, as shown in figure 1.2. The experiments were conducted under different network conditions – figure 1.2 shows data samples collected under three different conditions: when the network was not loaded (data collected at mid night), when the network was loaded (data collected at 5 pm), and when the network was loaded and the signal conditions were not strong (data collected at 6 pm, and inside a building).

From figure 1.2, we can observe that when the network is not loaded and the signal strength is strong, CMG application can achieve low response time. However when the mobile network is loaded, or when the user is in noisy network condition with poor signal strength, there are significant increases in uplink, downlink, and round-trip response time, leading to significant adverse impact on the quality of service. The above experiments indicate that for CMM applications to be successful, serious attention has to be given to (a) address challenges

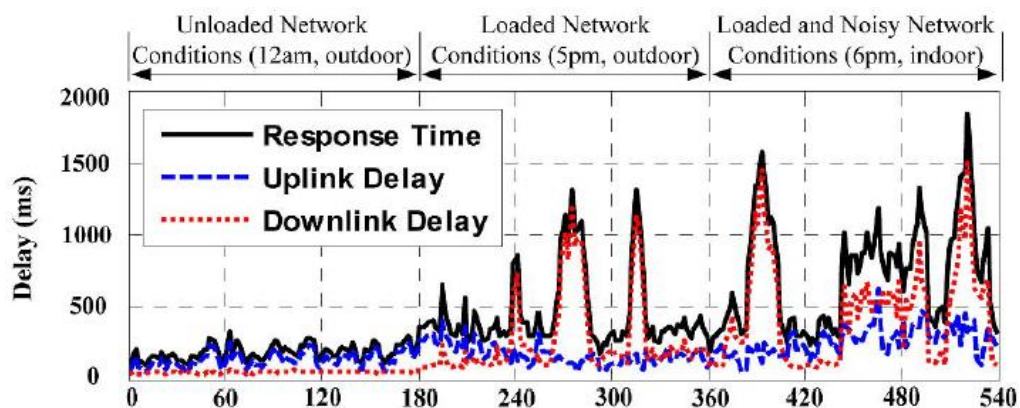


Figure 1.2. Delay and response time measured in a CMG session at different time during a day.

imposed by mobile networks like latency and response time, and (b) ensure good user experience.

### 1.2.2 Monitor and Guarantee Good User Experience

For cloud mobile rendering (CMR) application and video/audio streaming application, in order to address the challenges posed by mobile network, different optimization techniques had been developed to optimize the quality of service by adapting video encoding and graphic rendering factors in real time according to the network condition. For instance, regarding CMR application, in [WD13], the authors have proposed a technique to adapt the richness and complexity of graphic rendering, thereby impacting the bit rate of the rendered video that needs to be transmitted from the cloud server to the mobile device. For video/audio streaming application, DASH, dynamic adaptive streaming over HTTP, is a new world wide standard for adaptive streaming of video, in which the bit rate and quality of transmitted video are adapted to match the available bandwidth and avoid network congestion.

Although these optimization techniques can mitigate the negative impact of the rapid fluctuating mobile bandwidth, they have introduced another level of complexity for CMM

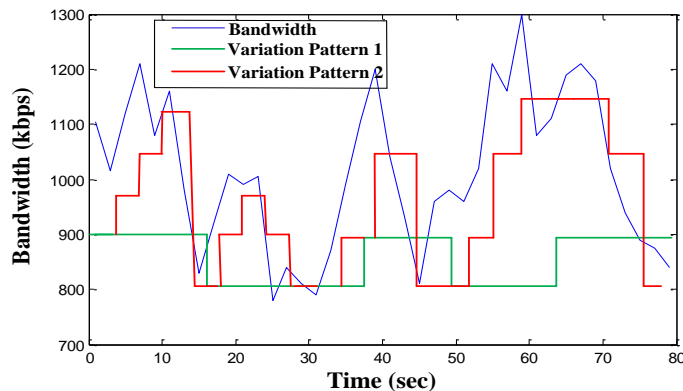


Figure 1.3: Mobile bandwidth trace and DASH video quality variation patterns.

service providers to monitor the user experience. Take DASH video as an example. Figure 1.3 shows a mobile bandwidth trace and two associated video bitrate adaptation patterns produced by using different DASH algorithms. Pattern 1 is more conservative but provides a more stable overall quality. Pattern 2 is more aggressive, and it tries to increase the video quality whenever the available network bandwidth increases. It is difficult to tell which one is more preferable from a user experience perspective. The answer to this question will be helpful for video service providers to optimize their DASH quality adaptation algorithm.

In this thesis, we focus on solving the challenges of two kinds of CMM applications: CMR application and video/audio streaming application. We present a series of innovative techniques to: 1) develop user experience models for CMR application and DASH video streaming; 2) develop several techniques for CMR application to optimize rendering and encoding process to enhance the user experience.

### 1.3 Contribution and Overview

The main contributions made by this thesis are fivefold. First, we develop and validate a **Cloud Mobile Rendering User Experience (CMR-UE)** model through controlled subjective testing. We will first study the effect of rendering factors on user experience, and then combine different factors to derive a complete CMR-UE model for video rendered and streamed from cloud servers. To the best of our knowledge, this is the first work to study how rendering of video will affect user experience, and the first UE model suitable for CMR applications where the video rendering factors are adapted. We have also observed that the impairment caused by rendering is highly dependent on the content information of the rendered video, such as what kind of scene and how many 3D objects are being rendered.

Therefore, in the CMR-UE model, we have incorporated video content information in order to further improve modeling accuracy.

Second, besides being able to quantitatively determine user experience of the rendered and streamed video, another major contribution is to be able to decide the optimal rendering factors and encoding factors such that the CMR user experience is maximized. As will be explained in Chapter 3, given a certain available network bandwidth, and hence the bit rate budget of the rendered and encoded video stream, there is a tradeoff relation between rendering richness and video quality achieved, increasing rendering richness can lead to lower video quality, and vice versa. In this thesis, for the first time, we investigate this tradeoff, and utilize the CMR-UE model as a tool to determine the optimal rendering richness and video quality so as to maximize the overall user experience. Furthermore, we propose a Content-Aware Adaptive Rendering (CAAR) algorithm to dynamically select the optimal rendering factor values according to the network condition such that the user experience (as measured by the CMR-UE model) can be maximized at any time. We will show the effectiveness of the proposed CAAR algorithm through experiments using real mobile network.

Third, for CMR application, we develop ways to use rendering information to identify the importance of different macroblocks (MB) of a video frame, spend more bits on the important MBs, and hence achieve a better perceptual video quality compared to allocating bits equally on the entire video frame.

Fourth, we propose a rendering-based video encoding acceleration technique to exploit the rendering information to accelerate the encoding process. This technique consists of two parts: 1) a method to directly calculate the motion vector between adjacent game frames, and use this calculation method to replace the regular compute intensive search based motion estimation method; 2) a fast mode selection algorithm, which uses the homogeneity of

motion vectors to reduce the number of candidate MB modes that need to be tested in the rate-distortion optimization process, ultimately reducing the computational complexity of video encoding.

Last, we develop a user experience model for another kind of CMM application: Dynamic Adaptive Streaming over HTTP (DASH) video streaming. We investigate three factors which impact user perceived video quality: initial delay, stall (frame freezing), and bit rate (frame quality) fluctuation. We conduct extensive subjective tests in which a group of subjects provide subjective evaluation while watching DASH videos with one or more artifacts occurring. Based on the subjective tests, we first derive impairment functions which can quantitatively measure the impairment of each factor, and then combine these impairment functions together to formulate an overall user experience model for any DASH video. We validate with high accuracy the user experience model, and demonstrate its applicability to long videos.

In the remainder of this dissertation, we take a detailed look at the novel techniques previously introduced. In Chapter 2, we present the methods of how we develop the CMR-UE model. In Chapter 3, we elaborate the adaptive rendering techniques, which dynamically select the optimal trade-off between rendering richness and video quality under a fix bandwidth budget. Chapter 4 introduces the prioritized encoding technique and encoding acceleration technique, which utilize the rendering information improve the visual quality of encoded video and reduce the computation complexity of encoding process. Chapter 5 covers the user experience model we developed for DASH video.

## **Chapter 2**

# **Modeling User Experience for Cloud Mobile Rendering Application**

Cloud Mobile Rendering (CMR), where compute intensive rendering is performed on cloud servers instead of on mobile devices, is a promising approach to enable rich rendering based multimedia applications on battery and CPU constrained mobile devices. However, since the video rendered in the cloud has to be streamed to the mobile device over a wireless network with fluctuating and constrained bandwidth, the resulting user experience can be impacted. In [WD13], adaptive rendering was proposed as a solution, wherein multiple rendering factors can be adapted such that the encoding bit rate of the rendered video is compatible with network bandwidth. However, changing the rendering factors may itself have adverse impact on user experience, which has not been studied earlier. Moreover, the impact can vary depending on the content scene and complexity. In this chapter, we analyze the impairment of rendering factors on user experience while considering the rendering impairments with impairments due to video encoding factors (like bit rate and frame rate) and



network factors (like bandwidth and delay) to propose a UE model to measure user experience. We demonstrate the accuracy of this model through subjective testing.

## **2.1 Factors Affecting User Experience of Cloud Mobile**

### **Rendering Application**

The first step to study and implement the CMR-UE model is to identify the impairment factors, which impacts user experience of a CMR session. In this section, we first introduce the impairment factors of CMR application, and then provide a detailed explanation of how rendering factors affect user experience.

As described in Chapter 1 (figure 1.1), during a Cloud Mobile Rendering session, graphic rendering is performed at a cloud server, with the resulting video subsequently encoded and streamed over the bandwidth constrained and error-prone wireless network. Therefore, as shown in figure 2.1, there are three major categories of objective factors: rendering factors, video encoding factors, and mobile network factors. The user experience is determined by three subjective factors: graphic quality, video quality, and response time. Response time is the duration between the user issuing a command and the effect of this command being displayed on the mobile device. It has significant impact on user experience due to the interactivity nature of CMR applications. As shown in figure 2.1, the objective factors would affect the subjective factors in different ways. Rendering factors will only affect the user perceived visual quality of the graphics, while encoding factors and network factors affect user perceived video quality and response time in a complex manner. For instance, video encoding bit rate will affect user perceived video quality as well as the response time since it will determine the amount of video traffic need to be transmitted. Note both the time

taken for rendering and encoding also contribute to response time. However, assuming scheduling of CMR applications to the CMR servers are done such that there is no over-utilization of the servers, the rendering and encoding time for a certain CMR application can be expected to be fixed, and hence we do not consider them any more in the discussion below.

We next describe the impact of different rendering factors (figure 2.1) on bit rate of

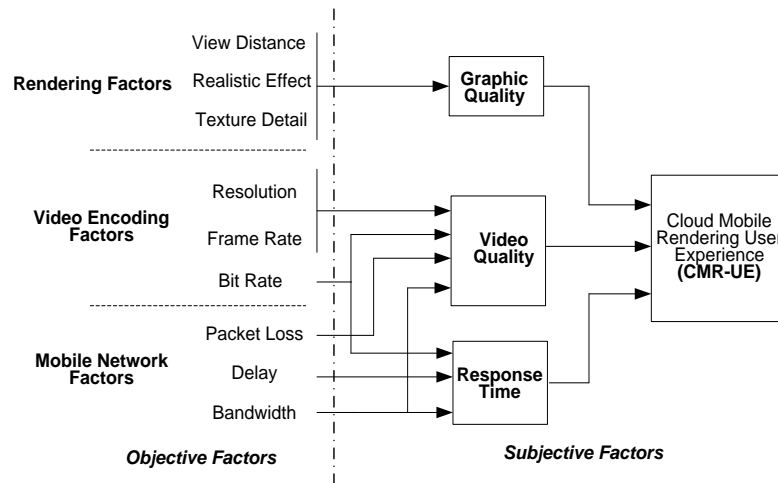


Figure 2.1. Objective and subjective factors affecting user experience of CMR applications.

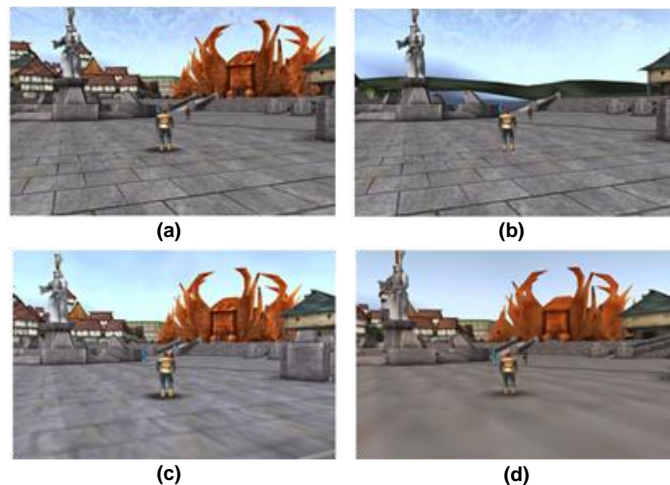


Figure 2.2. Graphic quality difference of different settings of *View Distance* and *Texture Detail*: (a) 300m view and high texture detail; (b) 60m view and high texture detail; (c) 300m view and medium texture detail; (d) 300m view and low texture.

the rendered video as well as the user experience. Figure 2.2 shows graphic quality differences when different settings are used for different rendering factors in a cloud mobile gaming application. Figure 2.2(a) and (b) compare the graphic quality between two different view distance settings (300m and 60m). Figure 2.2(a), (b) and (c) compares the graphic quality for different rendering texture detail levels. A shorter view distance (figure 2.2(b)) or a lower texture detail level (figure 2.2(c) and (d)) can have significantly less bit rate needed to encode the rendered video. For instance, to achieve a PSNR value of 32dB, the required video bit rates of the rendered video shown in figure 2.2(a) ~ (d) are about 700kbps, 590kbps, 530kbps and 420kbps, respectively. We can leverage these characteristics and adapt the graphic quality (rendering richness) and therefore adapt the bit rate of the streamed video to avoid network congestion. However, obviously shorter view distance or lower texture detail level has impairments on user perceived gaming quality. Therefore, we need to study how view distance or texture detail level affects user experience. The above understanding will help guide the CMR system to perform the right tradeoff between increasing view distance/texture detail and possibly affecting graphic visual quality, versus having network congestion affecting video quality and response time. Motivated by this, in this chapter we develop a user experience model (CMR-UE), which takes into account rendering factors in addition to network and video encoding factors, to quantitatively measure the user perceived CMR service quality.

## **2.2 Derivation and Validation of Impairment Functions for Rendering Factors**

In this section, we focus on studying the effects of rendering factors on CMR-UE. Based on a subjective study, we derive an impairment function  $I_R$  which denotes the

impairments caused by graphic rendering factors as discussed in figure 2.1. In the next section, we will derive a complete CMR-UE model with this new  $I_R$ , and video encoding impairment function ( $I_E$ ) and network impairment functions ( $I_D$  and  $I_L$ ) which were derived in [WD12].

### 2.2.1 Impairment Function for Each Rendering Factor

We first define a rendering impairment function, denoted as  $I_R$ , to indicate the impairment caused by graphic rendering.  $I_R$  is a metric that takes value from range [0, 100], and the relationship between  $I_R$  and the perceived graphic quality is listed in Table 2.1 (the lower the  $I_R$ , the better the graphic rendering quality).

As shown in figure 2.1, we consider the effect of 3 common graphic rendering factors, View Distance, Realistic Effect and Texture Detail, on the graphic quality of CMR applications. Each of these factors would affect graphic quality from a different aspect, and the total rendering impairment  $I_R$  can be formulated with three impairment sub-functions:

$$I_R = I_{VD} + I_{RE} + I_{TD} \quad (2.1).$$

where  $I_{VD}$  represents the impairment caused by reducing view distance,  $I_{RE}$  indicates the

Table 2.1. Graphic quality rating criteria and  $I_R$  values.

$I_R$	Description
0	Excellent experience, no rendering impairment at all
0-20	Minor rendering impairment, will continue CMR session
20-40	Noticeable rendering impairment, might quit CMR session
40-60	Clearly rendering impairment, usually quit CMR session
60-100	Annoying experience, unacceptable quality, definitely quit

impairment caused by reducing realistic effect, and  $I_{TD}$  indicates the impairment of reducing texture detail.

In the next subsections, we derive the impairment sub-functions  $I_{VD}$ ,  $I_{RE}$ , and  $I_{TD}$  by conducting a set of subjective quality assessment experiments, and the accuracy of the linear model (equation (2.1)) will be validated with another set of subjective experiments in section 2.3.4.

### 2.2.2 Subjective Quality Assessment Experiment

To derive the impairment sub-functions ( $I_{VD}$ ,  $I_{RE}$  and  $I_{TD}$ ) in equation (2.1), we developed and conducted a subjective quality assessment experiment, using a group of participants comprising of 18 students at UCSD. We used Cloud Mobile Gaming application as a demonstration example of CMR application. We use a MMORPG game called PlaneShift [PS]. The experiment environment is shown in figure 2.3, where the mobile device is connected to the game server through a network emulator, which can be used to control the network conditions, such as delay, packet loss pattern, bandwidth, etc. In the experiment environment, the game logic and rendering tasks are executed by the server, while the mobile device sends control commands and displays gaming video. We control the network emulator



Figure 2.3. Experiment Framework.

to keep the network bandwidth to be sufficiently large, and only change the rendering factors on the server. Each experiment participant plays the game on the mobile device with different combinations of the rendering factors shown in Table 2.2, and provided assessment of the rendering impairments  $I_R$  using an impairment rating system shown in Table 2.1. Finally, the results of the study group were tabulated for further analysis and derivation of impairment functions.

### 2.2.3 Derivation of Impairment Function $I_{VD}$ , $I_{RE}$ and $I_{TD}$

To derive  $I_{VD}$ ,  $I_{RE}$  and  $I_{TD}$ , we use the result of subjective tests where we only change one of the rendering factors while keeping the other two rendering factors at their best values. For example, to derive  $I_{VD}$ , we only change view distance while keeping realistic effect at high level (color depth is 32; multi-sample is 8; texture filter 16; and lighting mode is vertex light) and texture detail (down-sample rate) at 0 (without any down-sampling of texture detail).

In order to derive the impairment due to view distance,  $I_{VD}$ , we first investigate the relation between the average impairment evaluation by the participants and the view distance values. We find that this relation is very different for different game scenes. For example,

Table 2.2. Parameters for subjective experiment.

Parameters		Experiment Values		
Realistic Effect		H(High)	M(Medium)	L(Low)
	color depth	32	32	16
	multi-sample (factor)	8	2	0
	texture-filter (factor)	16	4	0
	lighting mode	Vertex light	Lightmap	Disable
Texture Detail (down sample)		0, 2, 4		
View Distance (meter)		150, 120, 100, 70, 60, 40, 20		

figure 2.4 depicts the players' average evaluation of impairment for different view distances, in the case of two game scenes. Figure 2.4(a) is a screenshot of scene 1, which is a representative outdoor game scene. Figure 2.4(b) is a screenshot of scene 2 which is an indoor scene. For each scene, we plot the average evaluations of  $I_{VD}$  for different view distance values, as shown in figure 2.4(c). Based on the experiment results, it is obvious that the relationship between  $I_{VD}$  and view distance value are very different for different game scenes. In other words, the impairment of view distance,  $I_{VD}$ , is not solely a function of view distance value, but it is also related with the game scene (content) rendered. Since during a gaming session, numerous possible game scenes may be rendered depending on the actions of the gamers, it will be challenging, if not impossible, to develop and use a model for  $I_{VD}$  which depends on the specific scenes.

From the different types of scenes we rendered with different view distances, and the  $I_{VD}$  subjective evaluations we obtained, we observe that the user experience is decreased

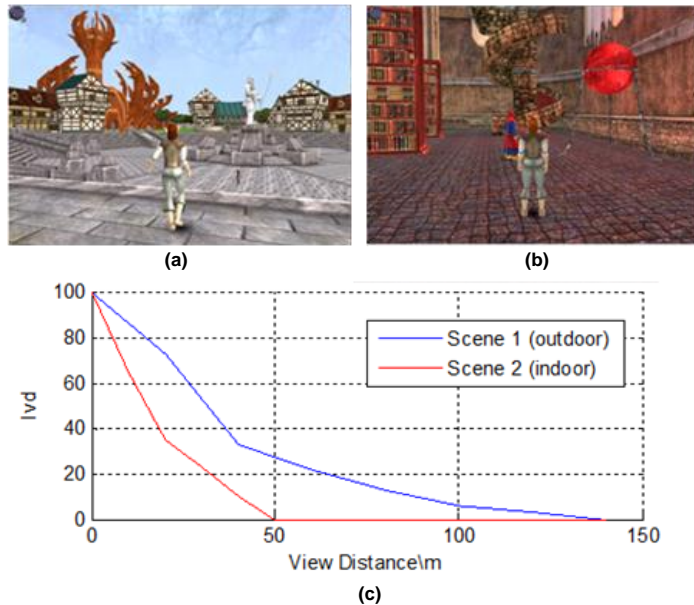


Figure 2.4. Relation between  $I_{VD}$  and view distance for different game scenes: (a) screenshot of scene 1; (b) screenshot of scene 2; (c) relationship between subjects' evaluation  $I_{VD}$  and view distance.

mainly because of the missing objects in the rendered frame. For example, suppose we set the view distance to be 70 meters, in scene 1 some distant objects will be excluded in the rendered frame, whereas in scene 2 there will be no objects excluded. So reducing view distance to 70m will have negative impact on UE for scene 1, but no impact for scene 2. We observe that more the missing objects, the larger the impairment. Therefore, instead of using view distance value as the metric to model  $I_{VD}$ , we propose to use a new metric  $F_{render}$ , which indicates the fraction of objects that are included in the rendered video. For example, if there are totally 300 objects in scene 1, among them 210 objects have distances less than 70 meters to the camera. When the view distance is set to be 70 meters, only 210 objects out of 300 will be rendered and  $F_{render}$  value is 0.7 in this case.

Given a view distance value, the corresponding  $F_{render}$  value can be computed conveniently by exploiting the video content information. In CMR applications, some video content information can be extracted from the graphic engine during the rendering process. For example, we can extract the location information of all the 3D objects (avatars, buildings, trees, etc.) in the game scenes (shown as figure 2.4(a)(b)), as well as the distance values from these 3D objects to the camera. Therefore, for a view distance value  $dist$ , we can check how many 3D objects in the game scene have a smaller distance than  $dist$ . Then  $F_{render}$  can be calculated by dividing this number by the total number of 3D objects.



Having defined  $F_{\text{render}}$ , we conduct the same experiment process as described in section 2.2.2 to derive  $I_{\text{VD}}$  using metric  $F_{\text{render}}$ . As we vary view distance, we collect subjects' evaluations about  $I_{\text{VD}}$  under 20 different game scenes. This time we record subjects' evaluations together with corresponding  $F_{\text{render}}$  values. Among the 20 game scenes, we plot in figure 2.5 the relation between subjects' average evaluation of  $I_{\text{VD}}$  and corresponding  $F_{\text{render}}$  for 4 representative scenes. We can see that although the relationship between subjects' evaluation of  $I_{\text{VD}}$  and  $F_{\text{render}}$  are not identical for these 4 different game scenes (2 outdoor scenes and 2 indoor scenes), their trends are very similar, and we can observe similar trend for the other 16 game scenes. This result indicates that by formulating  $I_{\text{VD}}$  as a function of  $F_{\text{render}}$  can make  $I_{\text{VD}}$  scene-independent.

To derive this scene-independent impairment function  $I_{\text{VD}}$ , for each  $F_{\text{render}}$  value, we

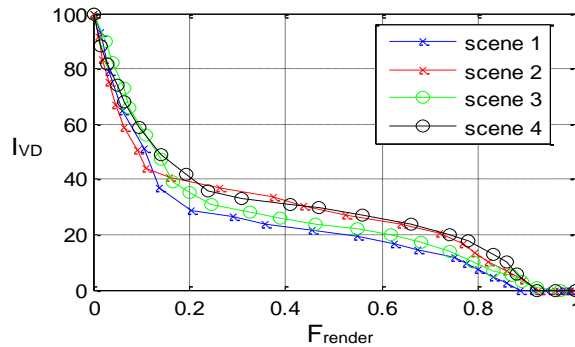


Figure 2.5. Relation between subjects' evaluation  $I_{\text{VD}}$  and  $F_{\text{render}}$ .

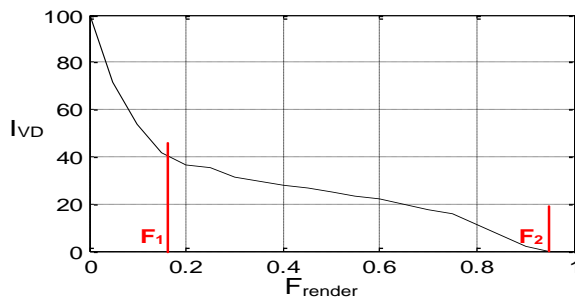


Figure 2.6. Relation between subjects' average evaluation  $I_{\text{VD}}$  and  $F_{\text{render}}$ .

compute the average  $I_{VD}$  score over all the 20 game scenes. The resulting average curve is plotted in figure 2.6. From figure 2.6, we have two observations: first, there is no impairment when  $F_{render}$  is bigger than a certain threshold  $F_2$  (for this specific game PlaneShift,  $F_2= 0.95$ ); second, there is an obvious slope change at the point where  $F_{render}$  equals a certain value  $F_1$  ( $F_1 = 0.15$  for this game).  $F_1$  and  $F_2$  divide the view distance value into different segments. Consequently, the impairment sub-function  $I_{VD}$  can be derived, using linear regression analysis, as the following:

$$I_{vd} = \begin{cases} 0 & (F_{render} > F_2) \\ 40 * [(F_2 - F_{render}) / (F_2 - F_1)] & (F_2 > F_{render} > F_1) \\ \alpha * (F_1 - F_{render}) + 40 & (F_1 > F_{render}) \end{cases} \quad (2.2).$$

In equation (2.2), coefficient  $F_1$  equals 0.15,  $F_2$  equals 0.95 and  $\alpha$  is 30, respectively.

We use a similar method to derive the impairment functions for the other rendering parameters, realistic effect,  $I_{RE}$ , and texture detail,  $I_{TD}$ . We vary only texture detail or realistic effect value according to Table 2.2, while keeping the other two rendering factors at the highest value. Then we use the average subjective score over all the subjects and game scenes as the results for  $I_{RE}$  or  $I_{TD}$ , which are listed in Table 2.3 and 2.4, respectively. Unlike  $I_{VD}$ , the results of  $I_{TD}$  and  $I_{RE}$  are given in table instead of equation, because there are only 3 options

Table 2.3. Impairment function for realistic effect.

Realistic Effect	High	Medium	Low
$I_{RE}$	0	4	15

Table 2.4. Impairment function for texture detail.

Texture Detail (down-sample)	0	2	4
$I_{TD}$	0	12	34

for texture detail and realistic effect. The standard deviation of  $I_{RE}$  and  $I_{TD}$  scores over different game scenes is 0.56 and 1.89 respectively. These small standard deviation values imply that both  $I_{RE}$  and  $I_{TD}$  values have very weak dependencies on game scene/content.

Derivation of impairment sub-functions  $I_{VD}$ ,  $I_{RE}$ , and  $I_{TD}$ , allows derivation of the impairment function  $I_R$  according to equation (2.1). Note that the values shown in Tables 2.3 and 2.4 are specifically for Planeshift, and will be different for other applications; however, the methodology to derive them will be same. Next, we will validate the accuracy of  $I_R$  using another set of subjective assessment experiments.

## 2.2.4 Validation of Impairment Function $I_R$

To validate impairment function  $I_R$ , we conducted another set of experiments with a new set of participants, where the participants now provide assessment of rendering impairment  $I_R$  when multiple rendering factors,  $I_{VD}$ ,  $I_{RE}$ , and  $I_{TD}$  are varied simultaneously. For each such experiment, we compare the subjective  $I_R$  scores the participants give and the predicted  $I_R$  values calculated from our impairment function  $I_R$  (equation (2.1) and (2.2)), Table

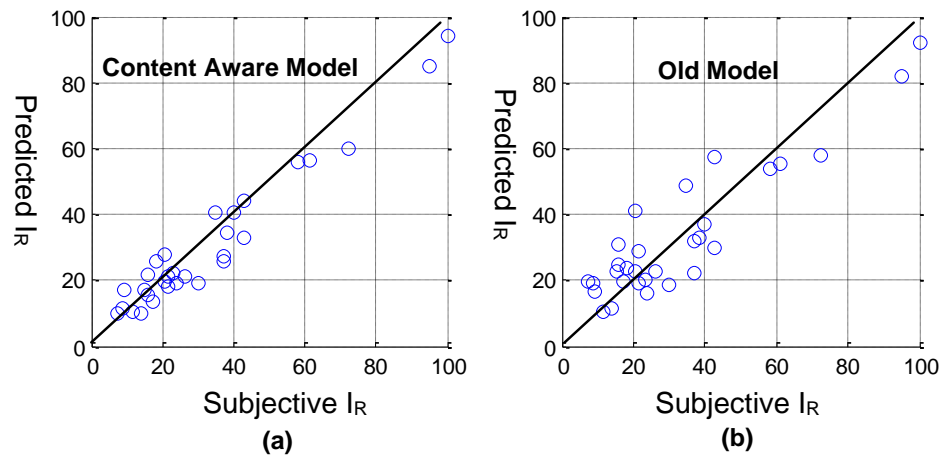


Figure 2.7. Relationship between predicted and subjective  $I_R$  value: (a) new model which include video content information; (b) old model which doesn't consider video content information.

2.3 and 2.4). We plot the results in figure 2.7(a). For comparison, we also plot in figure 2.7(b) the subjective  $I_R$  scores with  $I_R$  values calculated by a preliminary model developed in [LWD12], which does not take into account video content information. From figure 2,7, we see that the correlation between predicted  $I_R$  (y-axis) and subjective  $I_R$  (x-axis) is 0.92 using the proposed content aware  $I_R$  model, and 0.85 using the  $I_R$  model that doesn't incorporate video content information. This result demonstrates that  $I_R$  can predict with high accuracy the total impairment of rendering factors on user experience.

Having established an impairment function  $I_R$  for rendering factors, in the next section we will combine it with the impairment functions of network and encoding factors to formulate a complete CMR-UE model which can quantitatively measure mobile client's user experience during a CMR session.

## 2.3 Derivation of Cloud Mobile Rendering User Experience

### Model

In this section, we develop a Cloud Mobile Rendering User Experience (CMR-UE) model, incorporating the impairment due to rendering factors (developed in section 2.2) in addition to impairments due to video encoding and wireless network factors that we had developed in [WD12]. We present results of subjective experiments conducted to derive and validate the CMR-UE model.

We define CMR Mean Opinion Score (CMR-MOS) as a measurement metric for CMR-UE. Since CMR-MOS is determined by rendering, encoding and network factors as shown in figure 2.1, we attempt to formulate it using the impairment functions of these factors, similar to the framework of ITU-T E-Model [G107]. Although the ITU-T E-model is

originally developed for audio transmission applications, the framework of transmission rating factor  $R$ , which represents the overall user experience, can also be applied in our study. The function of CMR-MOS formulated by  $R$  factor can be found in the ITU-T E-model. We duplicate that function for our CMR-MOS formulation:

$$CMR-MOS = 1 + 0.035R + 7 \times 10^{-6} R(R-60)(100-R) \quad (2.3).$$

In equation (2.3), the transmission rating factor  $R$  takes value from range  $[0, 100]$  (the higher  $R$ , the better CMR-UE). CMR-MOS is related with  $R$  through non-linear mapping, and it is within the range of  $[1, 4.5]$ .

Although the framework of ITU-T E model is helpful for our study, the formula to compute  $R$  factor specified in ITU-T E model is specific to audio transmission and not suitable for CMR applications where graphic rendering and video encoding factors affect user experience. Therefore in this paper we propose to formulate  $R$  factor as equation (2.4):

$$R = 100 - I_E - I_D - I_L - I_R + \sum_{\substack{i,j \in \{E,D,L,R\} \\ i \neq j}} f_{ij}(I_i, I_j) \quad (R > 0) \quad (2.4).$$

In equation (2.4),  $R$  is composed of impairment functions for video encoding,  $I_E$ , network delay,  $I_D$ , network packet loss,  $I_L$  and graphic rendering,  $I_R$ . The impairment functions  $I_E$ ,  $I_D$  and  $I_L$  have been derived in our previous work [WD12]. Function  $f_{ij}(I_i, I_j)$  indicates the cross-effect of two different impairments and is used to compensate and adjust the  $R$  factor, because when several impairments happen simultaneously, the overall impairment will be different from the sum of each impairment.

In order to derive the formula of function  $f_{ij}(I_i, I_j)$ , and also validate the accuracy of the overall CMR-UE model, we conduct another set of subjective quality assessment experiment with a new group of participants. Totally 23 volunteers from UCSD are selected for this round of experiments. The same experiment framework (as described in section III) is utilized, but

this time we change multiple factors (encoding, rendering and network factors) simultaneously and ask participants to evaluate their experience. The encoding factors (frame rate, video bit rate, etc) and rendering factors (view distance, realistic effect, etc.) are varied at the game server, and the network factors are controlled by the network emulator. Each experiment participant played the game with different encoding, rendering and network factors and gave evaluation on his/her user experience using a CMR-MOS score rating system shown in Table 2.5.

During the subjective experiments, we have collected the participants' evaluations for different combinations of encoding, rendering, and network factors. For each combination of these factors, we compute the average evaluation among the participants, and use this average value as the subjective evaluation of user experience for that combination of factors. The subjective evaluations for all different combinations of factors form a set of test results. Next, we randomly select 60% of the test results, and use them to train the model for R (equation (2.4)). During the training, we use different types of functions for  $f_{ij}(I_i, I_j)$ , including  $(I_i + a \times I_j)^n$ ,  $I_i^n \times I_j^m$ , and  $e^{(I_i + a \times I_j)^n}$ , and use linear regression to compute the coefficients for the functions. Then we use the other 40% of test results to validate the proposed R model with all possible  $f_{ij}(I_i, I_j)$  functions. Finally we select the function shown in equation (2.5), since it

Table 2.5. CMR-MOS rating and R values.

CMR-MOS	R	Description
4.5	100	Excellent experience , no impairment at all
4.0—4.5	80-100	Minor impairment, will not quit
3.0—4.0	60-80	Noticeable impairment, might quit
2.0—3.0	40-60	Clearly impairment, usually quit
1.0—2.0	0-40	Annoying environment, definitely quit.

Table 2.6. Compensation coefficients in equation (2.5) for game PlaneShift.

$C_{ED}$	$C_{EL}$	$C_{ER}$	$C_{DL}$	$C_{DR}$	$C_{LR}$
0.4	0.3	0.4	0.2	0.4	0.1

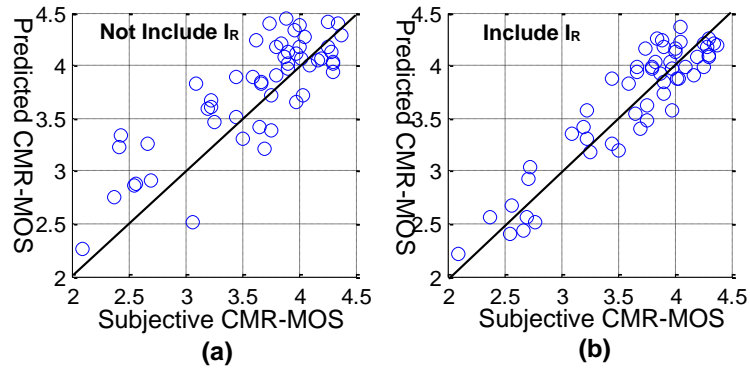


Figure 2.8. Relation between predicted and subjective CMR-MOS for game PlaneShift: (a) old model, not including rendering impairment; (b) proposed model, including rendering impairment.

achieves the highest correlation result in the model validation process.

$$f_{ij}(I_i, I_j) = C_{ij} \sqrt{I_i * I_j} \quad (2.5).$$

The  $C_{ij}$  coefficients can be determined by performing regression analysis using values of  $R$  calculated using equation (2.4) (2.5) and results of subjective tests. The results are shown in Table 2.6.

Having derived the complete CMR-UE model (equations (2.3) ~ (2.5) and Table 2.6), we now use the rest of the subjective tests to validate the model. Figure 2.8(b) shows the correlation between predicted CMR-MOS scores computed by the CMR-UE model (y-axis) and subjective CMR-MOS scores (x-axis) for each of the tests. From figure 2.8(b), we observe a high correlation of 0.90 and hence conclude the proposed model can accurately predict user experience. For comparison purpose, in figure 2.8(a) we also show a set of correlation results using the old UE model we proposed in [WD12] for Cloud Mobile Gaming, which doesn't

Table 2.7. Compensation coefficients in equation (2.5) for game Broadsides.

$C_{ED}$	$C_{EL}$	$C_{ER}$	$C_{DL}$	$C_{DR}$	$C_{LR}$
0.35	0.2	0.4	0.15	0.2	0.45

consider the rendering impairment  $I_R$ . The correlation value for figure 2.8(a) is 0.83. We conclude that the new CMR-UE model which includes  $I_R$  leads to better correlation with people's subjective evaluation.

To demonstrate the applicability of the CMR-UE model on other games, we applied it to another 3D game Broadsides [BRO], which belongs to a different genre than game PlaneShift. While PlaneShift is a role playing game, Broadsides is a first person shooter game. For the new game, we apply the same approach for validating the CMR-UE model. We conduct subjective tests and collect evaluation scores under different combinations of the rendering, encoding and network factors. We use 60% of test results for training the model of R and computing the coefficient values, and use the other 40% of test results for validating the CMR-UE model. Table 2.7 shows the values of coefficients  $C_{ij}$  for game Broadsides. Comparing these  $C_{ij}$  values with the coefficients for game PlaneShift (Table 2.), we can see

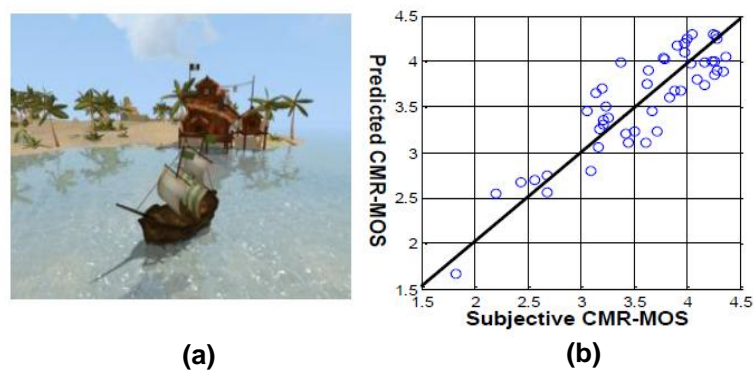


Figure 2.9. Evaluating CMR-UE model on game Broadsides (a) screenshot of game Broadsides; (b) relation between predicted and subjective CMR-MOS.



that for different games, the coefficients  $C_{ij}$  are different. Figure 2.9(a) is a screenshot of the game *Broadsides*, and figure 2.9(b) is the scatter plot of the relationship between subjective and predicted CMR-MOS. For the new game *Broadsides*, the correlation is 0.88. From Table 2.7 and Figure 2.9(b), we can see that for the new game, after training and refitting the coefficients  $C_{ij}$ , the proposed CMR-UE model will also lead to high modeling accuracy.

Note that although for each CMR application (more specifically for each 3D game) the coefficients  $C_{ij}$  need to be derived through the subjective assessment experiments, we still regard the proposed CMR-UE model to be generally applicable because: a) although the coefficients  $C_{ij}$  are different, the framework of CMR-UE model (equations (2.3)~(2.5)) can be applied for different games and have high modeling accuracy according to our experiment results; b) the amount of work of this subjective experiment is manageable, considering the amount of work and money spent on developing a typical 3D game; c) the coefficients  $C_{ij}$  need to be trained only once for each game and can be used forever.

## 2.4 Conclusion

In this chapter, we have studied a cloud based mobile video rendering and streaming approach which is promising to enable mobile users experience advanced rendering-based multimedia applications, such as 3D gaming and augmented reality. We have developed a comprehensive content-aware CMR-UE model which can characterize simultaneous effects of graphic rendering, video encoding and networking factors on the user experience of a CMR application.

After developing and validating the CMR-UE model, in the next chapter, we will apply this model to investigate how to adapt rendering in real time such that the user experience will be maximized.

## **Acknowledgments**

The text of this chapter, in part or in full, is based on material that has been published in IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS) (Y. Liu, S. Wang, S. Dey, “Content-Aware Modeling and Enhancing User Experience in Cloud Mobile Rendering and Streaming,” JETCAS, Mar. 2014) and material published in IEEE International Conference on Computing, Networking and Communications (ICNC) (Y. Liu, S. Wang, S. Dey, “Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering,” ICNC, Hawaii, Jan. 2012). The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

# Chapter 3

## Content-Aware Adaptive Rendering

### Algorithm

Mobile networks are characterized by unpredictable and sometimes fast bandwidth fluctuations, which can seriously affect the rendered video streamed from the CMR servers to the mobile device, in terms of packet loss and latency. An effective way to cope with the throughput fluctuations of mobile networks is to adapt the rendering factors so that the required encoding bit rate for the resulting rendered video can be adapted according to available network bandwidth [WD13]. However, as discussed in Chapter 2 and illustrated in figure 2.2, while lowering rendering settings can reduce the encoding bit rate needed to achieve a certain video quality (PSNR), it can also negatively affect graphics quality, and hence the overall user experience, which was not considered in [WD11]. In other words, given a certain available network bandwidth, and hence a fixed video encoding bit rate budget, there is a tradeoff between rendering richness (graphics quality) achieved by graphics rendering and the video quality achieved by the video encoder. The question that needs to be addressed is

how to select the optimal rendering settings such that the overall user experience is maximized, when taking into consideration both graphics quality and video quality? Fortunately, the proposed CMR-UE model (developed in chapter 2) opens up the possibility to answer this question.

In this chapter, for the first time we study the relationship and tradeoff between rendering richness (graphics quality of rendered video) and video encoding quality. We propose a Content-Aware Adaptive Rendering (CAAR) algorithm, which leverages the CMR-UE model to dynamically select the optimal rendering factors depending on the network conditions. The CAAR algorithm includes an offline step and two online steps. The offline step is used to develop a model of the relation between rendering richness and video frame quality. The online steps will then use this relation together with the CMR-UE model to find the optimal rendering factors for a certain network bandwidth.

This rest of this chapter is organized as follows. In section 3.1, we explain the tradeoff between rendering richness and video frame quality under a fixed bit rate budget, which explains why rendering complexity need to be adapted. In section 3.2, we give an overview of proposed CAAR algorithm. In section 3.3, we introduce the offline step. Sections 3.4 and 3.5 explain in detail the two online steps. The effectiveness of the CAAR algorithm will be shown in section 3.6. .

## **3.1 Tradeoff Between Rendering Richness and Video**

### **Frame Quality**

Given a certain available network bandwidth, the optimal encoding setting would be using Constant Bit Rate (CBR) encoding to make sure the video bit rate is fixed and below the available network bandwidth to avoid network congestion. For a fixed video bit rate, if we reduce the rendering complexity for instance lowering view distance (impairment of rendering  $I_R$  will increase), the content complexity in each video frame will reduce. Thereby the video encoder will have less compression on each video frame, leading to a higher video quality (impairment of encoding  $I_E$  will decrease). Figure 3.1 shows the video quality difference using two different rendering settings (view distances are 150m and 70m) but keeping the same encoding setting (bit rate target is 300kbps and frame rate is 12 fps). It is obvious that reducing the view distance from 150m (left figure) to 70m (right figure) will provide significant improvement in video frame quality while keeping the same video bit rate (the left figure looks much more blurred). This is because the content complexity (the number of 3D objects) in the left figure where the view distance is 150m is much larger than the right figure where the view distance is 70m. Although reducing the content complexity by lowering rendering setting can improve video frame quality, it has negative impact on graphics quality and hence overall user experience. Therefore, to decide the optimal rendering settings which can maximize CMR-UE, we have to investigate the tradeoff between rendering richness



Figure 3.1. Comparison of video frame quality for different rendering setting with 300kbps bit rate: (a) view distance = 150m; (b) view distance = 70m.

(inverse to  $I_R$ ) and video frame quality (inverse to  $I_E$ ) on CMR-UE, which will be discussed in detail in the next sections.

### 3.2 Content-Aware Adaptive Rendering Algorithm

Motivated by the tradeoff between rendering richness (inverse to  $I_R$ ) and video frame quality (inverse to  $I_E$ ), in this section we give an overview of the proposed Content-Aware Adaptive Rendering (CAAR) algorithm, which aims at dynamically selecting the optimal rendering setting (a tuple of rendering factor values) in real time according to the network condition, such that CMR-UE is maximized at any time. However since the number of possible rendering settings may be very large, and the optimal decision depends on many factors such as application scenarios/network conditions, finding the optimal rendering setting will be complex and time-consuming. On the other hand, to be effective, rendering setting

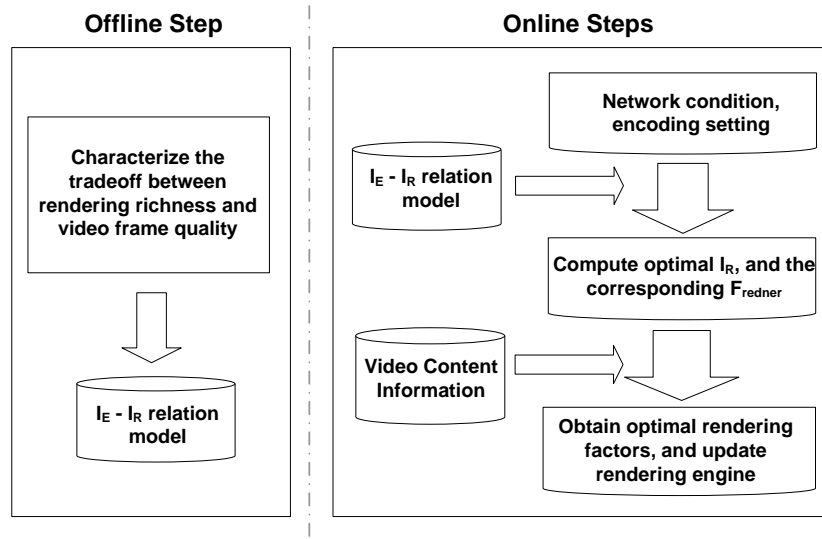


Figure 3.2. Overview of content-aware adaptive rendering algorithm.

should be adapted in real time to cope with rapid network fluctuations. To resolve the above conflict, we propose to partition our approach into two parts: offline step and online step.

Figure 3.2 shows the overview of the proposed Content-Aware Adaptive Rendering approach, including the offline and online steps. The offline step derives an  $I_E$ - $I_R$  relation model for the tradeoff between rendering richness and video frame quality under different available video encoding bit rate budgets. The online steps are executed during CMR sessions; they utilize the  $I_E$ - $I_R$  relation model, together with the measured network condition information as well as the application content information (such as game scene information), to determine the optimal rendering setting in real time. As will be explained later in this section, the first online step uses the  $I_E$ - $I_R$  relation model to find the optimal  $I_R$  value that will maximize CMR-UE. Subsequently, the second online step finds the optimal rendering setting which will lead to the optimal  $I_R$  value, using application content information.

### 3.2.1 Offline Step: Automatically Characterizing the $I_E$ - $I_R$ Relation Model

In order to fully understand the tradeoff relation between rendering richness (inverse to the rendering impairment,  $I_R$ ) and video frame quality (inverse to the encoding impairment,  $I_E$ ), and their impact on CMR-UE, we need to conduct an offline experiment using the same framework shown in figure 2.3. During the entire offline experiments, we configure the network emulator such that the network bandwidth is sufficiently large and there is not any network impairment. Unlike the subjective user experiments described in section 2.2.2 in which we need players to give their subjective evaluations, in this offline experiment, no human subject is needed as we utilize the CMR-UE model to compute the objective  $I_R$  and  $I_E$  values.

The procedure for the offline experiments is as follows. Note that while we describe the procedure using cloud gaming as an example, the same procedure can be followed for any CMR application. First, as the game is executed on the cloud server, we use a script to generate game control commands that automatically control the avatar to move around the virtual game world, including different kinds of game scenes. Second, while the rendered game video is captured and encoded, we vary the values of rendering and encoding factors, and record the corresponding PSNR values. For each possible combination of rendering and encoding setting, we execute the game for 10 seconds and record the average PSNR value during the 10 seconds. For each rendering setting  $i$  and rendering setting  $j$ , we compute the corresponding rendering impairment  $I_R$  using the CMR-UE model, and the encoding impairment  $I_E$  based on the average PSNR value,  $PSNR_{ij}$ , reported by the x264 encoder. All the  $I_E$ - $I_R$  pairs are stored and tabulated in the CMR server to derive the  $I_E$ - $I_R$  relation model. Table 3.1 shows all the values of the rendering and encoding factors used for the offline experiments. Considering a total of 54 rendering settings and 24 encoding settings used, the total duration of offline experiments is:  $54 * 24 * 10 = 12960$  seconds, which is less than 4 hours of play time.

Table 3.1. Parameters for offline experiments.

Factors		Experiment Values		
Rendering Factors	View Distance (meter)	15, 30, 50, 70, 100, 140		
	Texture Detail (down-sample rate)	0	2	4
	Realistic Effect	High	Medium	Low
Encoding Factors	Bit Rate (kbps)	200, 300, 400, 500, 600, 700, 1000, 1200		
	Spatial Resolution	640x480 (VGA), 800x600, 1280x720 (720p)		



Figure 3.3 shows the results of a representative offline experiment, when video bit rate is set to be 200kbps, video spatial resolution is set to be VGA, and rendering factor view distance is varied between 15m to 140m, while keeping texture detail and realistic effect fixed at their highest values. As we fix the video encoding setting and change the view distance value, the corresponding  $I_R$ ,  $I_E$  and CMR-MOS are shown in figure 3.3. Note that the offline experiment is conducted with different kinds of game scenes (indoor and outdoor), hence in the x-axis we show the corresponding  $F_{render}$  value instead of view distance. We can see from figure 3.3, as expected, when  $F_{render}$  increases,  $I_R$  reduces, but  $I_E$  increases. And because the decreasing slope of  $I_R$  is bigger than the increasing slope of  $I_E$ , the overall CMR-MOS

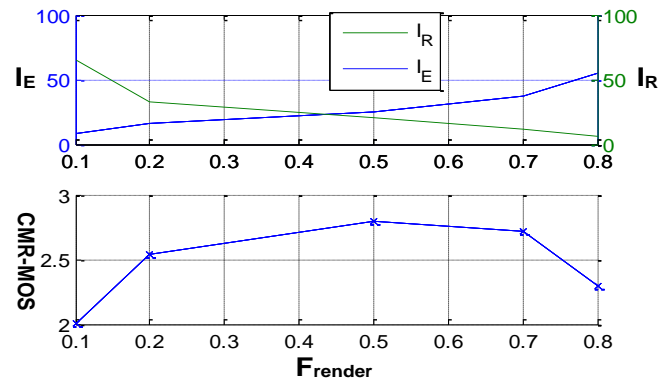


Figure 3.3.  $I_E$ ,  $I_R$  and CMR-MOS for different view distances used to render and encode a 200kbps CMR video.

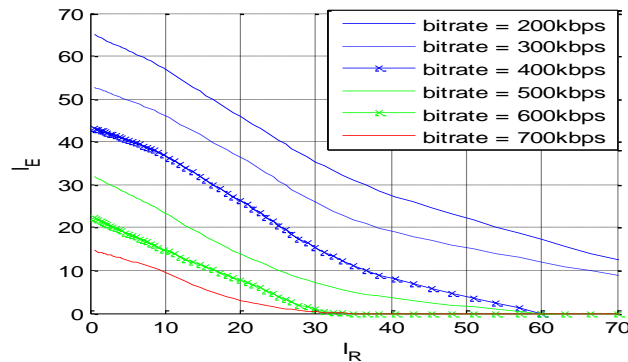


Figure 3.4. Relation between  $I_R$  and  $I_E$  under different bit rate targets.

Table 3.2. Coefficient values for the  $I_E$ - $I_R$  model for game PlaneShift.

Bit Rate/kbps	200	300	400	500	600	700
k	-0.8	-0.8	-0.7	-0.8	-0.7	-0.5
b	65	53	43	32	21	15

increases initially with increasing view distance (and hence  $F_{\text{render}}$ ). However, after the point when  $F_{\text{render}}$  equals 0.5, CMR-MOS starts to decrease as the decreasing slope of  $I_R$  is smaller than the increasing slope of  $I_E$  after that point. From the results shown in figure 3.3, we claim the optimal  $F_{\text{render}}$  for 200kbps bit rate target is 0.5, where it achieves the maximum CMR-MOS.

Figure 3.4 shows the results of tradeoff between  $I_R$  and  $I_E$  when encoding bit rate is varied between 200kbps to 700kbps, when the resolution is set to be VGA. In this case,  $I_R$  is the overall rendering impairment, which includes  $I_{VD}$ ,  $I_{RE}$  and  $I_{TD}$ . We can see that, for a given encoding setting, when  $I_R$  increases,  $I_E$  will decrease. But the slope and shape of the curves are different for different encoding settings. From the results shown in figure 14, we propose to use a linear function to model the relation between  $I_E$  and  $I_R$ , as shown in equation (3.1).

$$I_E = k * I_R + b \quad (3.1).$$

The values of slope  $k$  and coefficient  $b$  can be derived using linear regression. Table 3.2 shows an example of results of  $k$  and  $b$  when the CMR application characterized is the game PlaneShift described before, with the video resolution set to be VGA. Note that for better accuracy, it is recommended that the offline procedure described in this section is repeated for different resolutions that will be supported for the CMR video. While the above may seem cumbersome, in practical terms we envision the number of resolution that needs to be supported to be very limited considering the convergence of most smart phones and tablets to support high resolutions.

Equation (3.1) and the coefficient values (like shown in Table 3.2) form the  $I_E$ - $I_R$  relation model. Note that though the offline experiments and the regression analysis to develop this relation model should be done for each CMR application separately, the amount of required effort for each application is manageable because the procedure as described above is automated and typically takes a few hours of executing the application. In the next subsections, we will apply it to find the optimal rendering setting.

### 3.2.2 Online Step I: Determine Optimal $I_R$

After deriving the  $I_E$ - $I_R$  relation model, as shown in equation (3.1), in this subsection and the next subsection, we will describe the online steps which aim at obtaining the optimal rendering setting (tuple of rendering factor values) under certain network constraints. The online step is divided into two steps, in step I, we find the optimal  $I_R$  values which will maximize the overall user experience CMR-MOS; in step II, we will translate this optimal  $I_R$  value into applicable rendering factor values.

We first formulate the problem of finding the optimal  $I_R$  value as the following optimization problem:

**Given:**

Network statistics including packet loss rate  $L$ , and round-trip delay  $D$

**Find** the optimal  $I_R$  value

$$\begin{aligned} I_R^{opt} &= \arg \max R \\ &= \arg \max \{100 - I_E - I_D - I_L - I_R + \sum_{\substack{i,j \in \{E,D,L,R\} \\ i \neq j}} C_{ij} * \sqrt{I_i * I_j} \} \end{aligned} \quad (3.2).$$

$$\text{s.t.} \quad I_E = k * I_R + b \quad (3.3).$$

$$0 \leq I_E, I_R, I_L, I_D \leq 100 \quad (3.4).$$

During a CMR session, the network packet loss  $L$  and the round-trip delay  $D$  are periodically measured by a network probing technique proposed in [WD12]; therefore the impairment due to packet loss,  $I_L$ , and the impairment due to round-trip delay,  $I_D$ , can be computed as described in our prior work [WD12], and can be treated as known values. Hence the expression for  $R$  becomes:

$$R = M_1 - I_E - I_R + M_2 * \sqrt{I_E} + M_3 * \sqrt{I_R} + C_{ER} * \sqrt{I_E * I_R} \quad (3.5).$$

$$\text{where } \begin{cases} M_1 = 100 - I_L - I_D + C_{LD} \sqrt{I_L * I_D} \\ M_2 = C_{ED} \sqrt{I_D} + C_{EL} \sqrt{I_L} \\ M_3 = C_{RD} \sqrt{I_D} + C_{RL} \sqrt{I_L} \end{cases} \quad (3.6).$$

Furthermore, given the measured  $D$  and  $L$  values, a suitable video encoding bit rate is selected such that network delay  $D$  is reduced and response time constraints are satisfied, using the technique proposed in our previous work [WD09]. Knowing the video encoding bit rate, we can use Table 3.2 to find the corresponding  $k$  and  $b$  values in the  $I_E$ - $I_R$  relation model (equation (3.1)). Substituting for  $I_E$  using equation (3.1),  $R$  in equation (3.5) can be expressed in terms of only one unknown factor,  $I_R$ , and hence the optimization problem stated above becomes a one-dimensional optimization problem.

$$R = M_1 - (k * I_R + b) - I_R + M_2 * \sqrt{k * I_R + b} + M_3 * \sqrt{I_R} + C_{ER} * \sqrt{(k * I_R + b) I_R} \quad (3.7).$$

To solve the one-dimensional optimization problem, one way would be to compute the derivative of function  $R$  (equation (3.7)), and find out at which point the derivative equals 0. However, the format of derivative of  $R$  is too complex for an analytical solution (closed-form solution). Therefore we turn to using computer simulation to find a numerical solution. We have done experiments to compute the derivative with Matlab; it takes about 800ms to

solve the equation on a computer with Intel I5 processor, for granularity of e-5. However, the above computation time is not low enough to be used for the CAAR algorithm, considering the need to apply CAAR in real time, as well as the low response time needed for the CMR applications themselves.

Hence we propose a time-efficient alternative approach of using recursive searching to find the optimal  $I_R$ . The approach uses the following property of function R: R is a concave function of  $I_R$  and there exists one and only one maximum value. We have provided proof of the property in [SUP1]. The recursive searching approach exploits this property of R, and instead of trying all possible values and comparing them, it perform a coarse-to-fine search and reduce the search space by half at each iteration, therefore achieve a low computation time.

Figure 3.5 shows the procedure of the proposed recursive search algorithm. It exploits the concave property of R function and it is similar to the classic Dichotomous search algorithm [DSA]. We want to search for the optimal  $I_R$  value from a certain search range S,

<p><b><i>Recursive Searching Approach</i></b></p> <p><b><i>Input:</i></b> packet loss impairment <math>I_L</math>, network delay impairment <math>I_D</math>, coefficient k and b, and granularity threshold <math>\epsilon</math></p> <p><b><i>Output:</i></b> optimal <math>I_R</math> value, i.e. <math>I_R^{opt}</math></p>
<p>Initial search range <math>S = [s_1, s_2]</math>, <math>s_1 = 0</math>, <math>s_2 = 100</math>;</p> <p><b>Step 1:</b> determine the number of iterations for granularity <math>\epsilon</math>:  <math>n = \text{Ceiling}(\log[(s_2-s_1)/\epsilon])</math></p> <p><b>Step 2:</b> For <math>k = 1</math> to <math>n</math>, repeat Step 3 and 4:</p> <p style="padding-left: 2em;"><b>Step 3:</b> <math>p_1 = (s_1+s_2)/2 - \epsilon</math>, <math>p_2 = (s_1+s_2)/2 + \epsilon</math>;</p> <p style="padding-left: 2em;"><b>Step 4:</b> <b>If</b> <math>R(I_R = p_1) \geq R(I_R = p_2)</math>  Update the search range S, <math>s_2 = p_2</math>  <b>Else</b>  Update the search range S, <math>s_1 = p_1</math>  <b>Endif</b></p> <p><b>Step 5: Return</b> <math>(s_1+s_2)/2</math></p>

Figure 3.5. Recursive searching approach to obtain optimal  $I_R$ .

denoted as  $[s1, s2]$ . The function of  $R$  (equation (3.7)) is evaluated at two points close to the middle point of the search range:  $p1 = (s1+s2)/2 - \epsilon$  and  $p2 = (s1+s2)/2 + \epsilon$  where  $\epsilon$  is a predefined small positive number. Then depending on whether  $R(p1) > R(p2)$  or  $R(p1) < R(p2)$ , we decide that the optimal  $I_R$  lies in the left half or the right half of the search range respectively. Then we reduce the search range by half and repeat this process. This process will end when the size of the search range is smaller than  $\epsilon$ , and the middle point of the last search range is returned as the optimal  $I_R$ . When we apply this recursive searching approach in Matlab, the running time can be reduced to 68 ms for granularity of  $\epsilon=e-5$ .

### 3.2.3 Online Step II: Determine Optimal Rendering Setting

As stated in equation (2.1), the rendering impairment  $I_R$  equals to the sum of the impairments caused by the three rendering factors. Although impairment functions for realistic effect and texture detail are in form of discrete tables (since there are limited number of options for these two rendering factors), the impairment function of view distance,  $I_{VD}$ , is a continuous function which takes value between 0 and 100. This ensures that we can always find a combination of rendering factors to meet any  $I_R^{opt}$  target. For a specific  $I_R^{opt}$  target, we can select the realistic effect and texture detail first, and then subtract  $I_{RE}$  and  $I_{TD}$  from the  $I_R^{opt}$  to compute  $I_{VD}$ , and finally determine the view distance. For some  $I_R^{opt}$  values, we may have multiple choices for the 3 rendering factors. For example, if  $I_R^{opt} = 10$ , either of the two options below can be used to achieve it:  $(I_{VD}=10, I_{RE}=0, I_{TD}=0)$ ,  $(I_{VD}=6, I_{RE}=4, I_{TD}=0)$ . From the perspective of maximizing the CMR-UE, either of these options can be selected. In the following discussion, we will assume the realistic effect and texture detail are set to be their highest values. With this selection, the optimal  $I_{VD}$  will equal to  $I_R^{opt}$ , as shown in equation (3.8):

$$I_{VD}^{opt} = I_R^{opt} \quad (3.8).$$

Note that setting realistic effect and texture detail to be the highest level is just an assumption that we make to simplify the description of the steps below. If we don't want to set them to be the highest level, the approach we describe below can be still applied. We just need to subtract  $I_{RE}$  and  $I_{TD}$  from  $I_R^{opt}$  to get  $I_{VD}^{opt}$ .

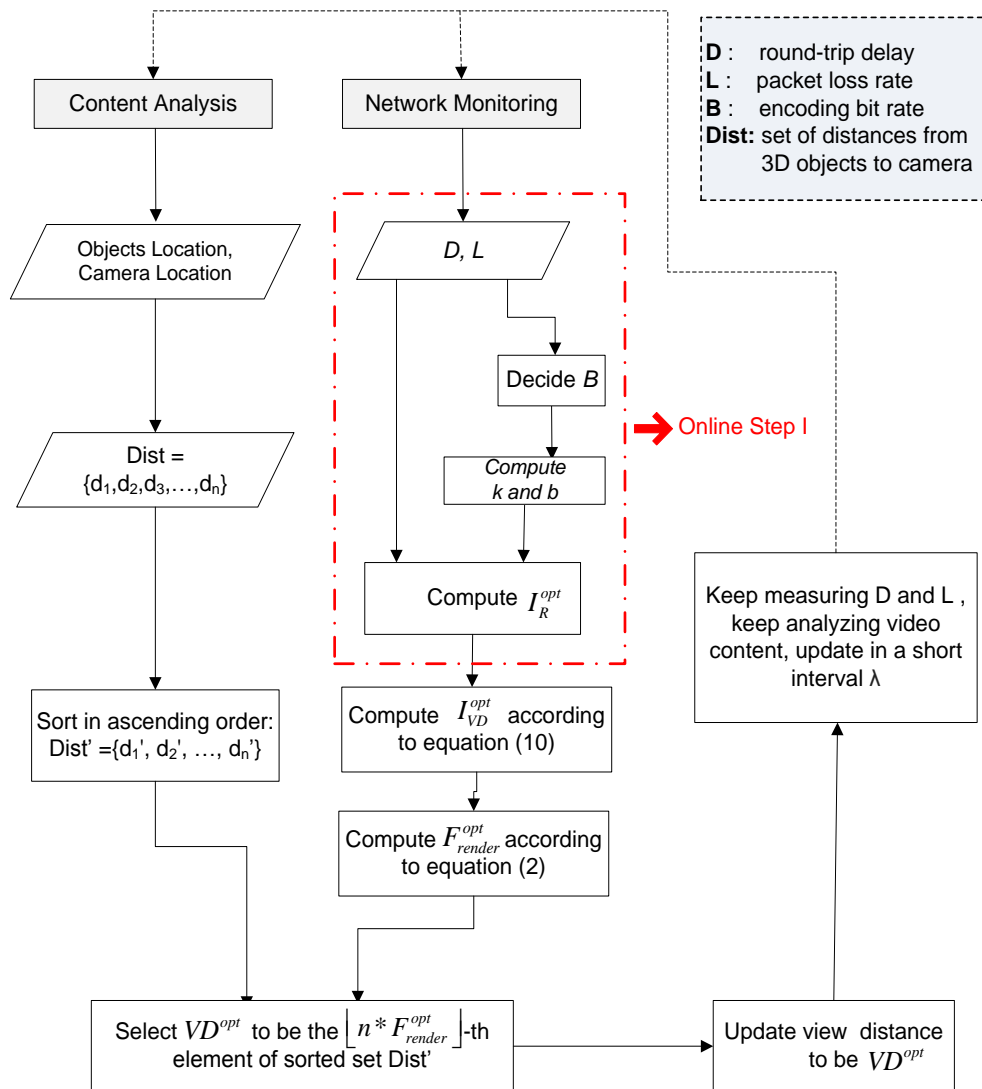


Figure 3.6. Flowchart of the online steps for adaptive rendering algorithm.

The remaining problem is how to find the view distance, such that  $I_{VD}$  will equal to  $I_R^{opt}$ . Firstly, we need to compute the corresponding  $F_{render}^{opt}$ . Secondly, in order to achieve  $F_{render}^{opt}$ , we need to analyze game video content, and obtain the location information of all the 3D objects and the camera. With these location information, we can obtain a distance set,  $Dist = \{d1, d2, d3, \dots, dn\}$ . Each element of the set  $Dist$ ,  $d_i$ , is the distance value from the  $i$ -th 3D object to the camera. We then sort these elements in ascending order and select the  $\lfloor n * F_{render}^{opt} \rfloor$ -th smallest element as the optimal view distance,  $VD^{opt}$ , since this  $VD^{opt}$  value will ensure that the fraction of objects that are rendered is equal to  $F_{render}^{opt}$ , therefore will ensure the optimal  $I_R$  value,  $I_R^{opt}$ , is achieved.

Figure 3.6 illustrates the flow of the online steps of CAAR algorithm, including online step I (highlighted using red box) and II. At short time intervals  $\lambda$ , the adaptation engine will check the network status (including D and L values), as well as video content information. On one hand, using the value of D and L, we can compute the optimal  $I_R$  value,  $I_R^{opt}$ , using the approach explained in online step I. After we get  $I_R^{opt}$ , we can compute  $I_{VD}^{opt}$  and the corresponding  $F_{render}^{opt}$ . On the other hand, using the 3D objects and camera location information, we can obtain a distance set,  $Dist$ . We then sort these elements in ascending order and select the  $\lfloor n * F_{render}^{opt} \rfloor$ -th smallest element as the optimal view distance,  $VD^{opt}$ . Finally we will inform the graphic engine to update the view distance value to be  $VD^{opt}$ .

In the next section, we describe the experiments conducted which demonstrate the effectiveness of CAAR to enable cloud video rendering and streaming over a real mobile network while ensuring high quality of experience.



### 3.3 Experimental Result in Real Wireless Network

In this section, we report on experiments conducted using a real commercial mobile network to verify the performance improvement possible by applying the proposed Content-Aware Adaptive Rendering (CAAR) algorithm. The experiments are conducted using 2 games, PlaneShift and Broadsides.

We have conducted extensive experiments to capture the bandwidth of a commercial 3.5G (1x-EVDO) network, using network measuring tool Iperf. We use a laptop as a mobile client, which is accessing the network through 3.5G network. We execute the Iperf software on the laptop, and hold it and roam around the UCSD campus at different times. The Iperf software will keep capturing the bandwidth trace which will be used later by the network emulator. Next, we use the experiment testbed shown in figure 3.7 to test the performance of CMG application in a 3.5G network. On the CMG server side, we use x264 codec for video encoding, and use UDP as the transport protocol. Encoding adaptation technique [6] is applied to adapt the bit rate to avoid network congestion. The mobile client is connected to the server through the network emulator, which is running the captured bandwidth trace of 3.5G network.

For comparison purposes, we apply three different rendering strategies while running the same bandwidth trace on the network emulator: (1) **non-adaptive rendering**: adapt video encoding bit rate according to network condition, but fix view distance to be the largest value as decided by the rendering engine (hence no rendering impairment); (2) **non-content-aware adaptive rendering**: adapt view distance purely based on encoding bit rate decided based on network condition, without considering video content information[WD13]; (3) **Content-Aware Adaptive Rendering (CAAR)**: dynamically adapt the view distance using the CAAR algorithm discussed in section 3.2. Note that for each of the rendering strategies above, video encoding bit rate adaptation is applied first depending on the measured delay and packet loss.

The non-content-aware adaptation algorithm is based on our previous work [WD13]; it is different from the proposed CAAR algorithm in two aspects. 1) For each encoding setting, it selects the rendering setting that maximizes the video quality (PSNR), whereas our CAAR algorithm selects the rendering setting that maximizes CMR-MOS, which include both video quality and rendering richness. 2) It will select rendering setting only based on encoding setting, and not consider the differences in content among different scenes as done by CAAR.

Next, we present in details results of applying CAAR on the game PlaneShift. Figure

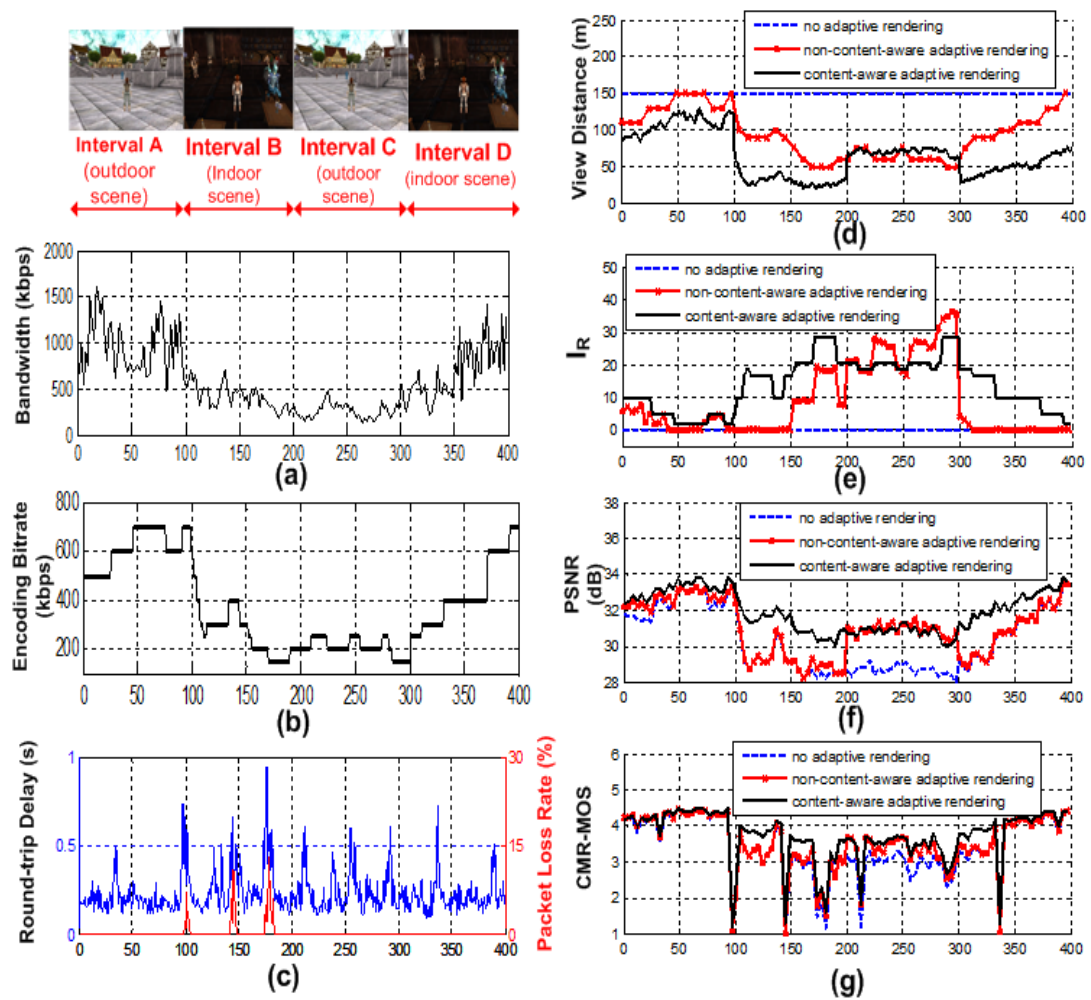


Figure 3.7. User experience calculated using CMR-UE model on a representative 3.5G network for game PlaneShift.

3.7(a) presents a representative sample of the 3.5G mobile network bandwidth traces we collected and used in the experiments. It shows that the bandwidth of the mobile network can rapidly fluctuate. The duration of the network trace is 400 seconds, which we consider equally divided into 4 intervals A to D. We control the avatar to move in different game scenes during these 4 intervals: during interval A and C, the avatar is running in an outdoor game scene; during interval B and D, we control the avatar to move in an indoor game scene. By controlling the avatar in this manner, we can ensure that the game is played in indoor and outdoor scenes under both good and bad network conditions. Then in figure 3.7 from top to bottom, we sequentially present the results of encoding bit rate (after applying encoding adaptation technique), network round-trip delay, packet loss rate, view distance, rendering impairment  $I_R$  (inversely related to rendering richness), PSNR (indicating video frame quality), and resulting CMR-MOS score calculated by CMR-UE model.

Figure 3.7(b) shows the encoding bit rates selected according to the available network bandwidth. The resulting round-trip network delay and packet loss rate are shown in figure 3.7(c). The results show that video encoding adaptation by itself can ensure the round trip delay is well maintained, most of the time to below 250ms, only rising above 500ms very occasionally. Same is true for the packet loss performance, with only a few spikes of high packet loss. However, as will be seen later, the video quality achieved (PSNR) and the overall user experience (CMR-MOS score) can be unacceptably low without applying rendering adaptation. In comparison, we will demonstrate that applying rendering adaptation, specifically content aware rendering adaptation, in conjunction with encoding bit rate adaptation, can significantly enhance not only the PSNR of the resulting encoded video, but also the overall user experience in spite of potentially compromising rendering richness.

Figure 3.7(d) shows the view distance used by the graphics rendering engine to

produce the rendered video frames. When no adaptive rendering is applied, the view distance is fixed at 150m. When non-content-aware adaptive rendering [6] is applied, the view distance is adapted according to the encoding bit rate shown in figure 3.7(b). When CAAR is applied, the view distance is adapted according to the encoding bit rate as well as the game content information: for interval B and D, as the avatar is in an indoor scene, the view distance selected and used is much smaller than that of interval A and C. The average PSNR values for the 3 strategies during the 400 seconds game session reported here are 30.28 dB, 31.02 dB and 31.92 dB, using no rendering adaptation (only encoding bit rate adaptation), non-content-aware rendering adaptation, and CAAR respectively. We can see that CAAR algorithm can achieve a much better PSNR than the other 2 rendering strategies.

Figure 3.7(e) shows the  $I_r$  values for different strategies. Higher IR value indicates lower rendering richness. With no adaptive rendering, IR is always 0; whereas for the other two adaptive rendering strategies, the rendering richness is adapted with the encoding bit rate. Figure 3.7(f) shows the resulting PSNR performance. We can see that: (1) adapting encoding bit rate without adapting rendering (non-adaptive rendering) produces the worst PSNR performance since it does not reduce the content complexity when the encoding bit rate is low; (2) CAAR performs significantly better than non-content-aware adaptive rendering by considering the content of the different scenes. For instance, let's compare the two adaptive rendering strategies during intervals B and C, in which the available network bandwidth are similar, but in which two different types of game scenes are rendered. CAAR assigns smaller view distance values during interval B (indoor scene) than during interval C (outdoor scene), thereby reducing content complexity of rendered video frames in interval B similar to frames rendered in interval C, hence enabling a satisfactory PSNR performance during both interval B and C. On the other hand, when non-content-aware adaptive rendering is applied, it assigns

similar view distance values for indoor and outdoor scenes, thereby rendering frames with larger content complexity for indoor scenes than outdoor scenes. Therefore while providing satisfactory PSNR performance during interval C, it produces low PSNR during interval B.

Figure 3.7(g) demonstrates that the CAAR algorithm outperforms the other two strategies and yields the best CMR-UE performance during all the 4 intervals. This is because CAAR algorithm always selects the best tradeoff between rendering richness (inverse to  $I_R$ ) and video frame quality (indicated by PSNR), and consider game content information to take care of all the scenes. The average CMR-MOS values for the 3 strategies during the 400 seconds game session reported here are 3.19, 3.34 and 3.55, using no rendering adaptation, non-content-aware rendering adaptation, and CAAR respectively.

Next, we briefly summarize results of conducting the same experiments as above using another game, *Broadsides*, which is of a very different genre. Similar to the results for game *PlaneShift*, the CAAR algorithm performs significantly better than the other two strategies. The average PSNR for the 3 strategies during the 400 seconds game session are 30.02 dB, 31.17 dB and 32.05 dB, using no rendering adaptation (only encoding bit rate adaptation), non-content-aware rendering adaptation, and CAAR respectively. Similarly, the average CMR-MOS scores for the 3 strategies are 3.02, 3.28, and 3.50 respectively. The above results demonstrate that the CAAR algorithm can achieve better CMR-MOS scores than the other 2 strategies, and is applicable and effective to other 3D games of different genres.

### **3.4 Conclusion**

In this chapter, we develop a Content-Aware Adaptive Rendering (CAAR) algorithm, to dynamically select the optimal rendering setting according to network conditions such that the right tradeoff between rendering quality and encoding quality is achieved, and user

experience is maximized. Experiments conducted in a real 3.5G network demonstrate the effectiveness of the proposed method. Although we have demonstrated the use of adapting rendering richness and encoding quality to mitigate network conditions, too frequent or steep fluctuations in rendering richness may also impact user experience. In the future, we will extend the CMR-UE model to consider the impairment caused by the variations of rendering richness and video quality, and modify our CAAR algorithm accordingly.

## **Acknowledgments**

The text of this chapter, in part or in full, is based on material that has been published in IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS) (Y. Liu, S. Wang, S. Dey, “Content-Aware Modeling and Enhancing User Experience in Cloud Mobile Rendering and Streaming,” JETCAS, Mar. 2014) and material published in IEEE Conference on Computing, Networking and Communications (ICNC) (Y. Liu, S. Wang, S. Dey, “Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering,” ICNC, Hawaii, Jan. 2012). The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

## **Chapter 4**

# **Enhancing Video Encoding for Cloud Mobile Rendering Application Using Rendering Information**

In CMR application, graphic rendering happens on cloud server and the rendered videos are encoded and streamed in real-time to the mobile devices. Compared with other video streaming applications, CMR application offers a unique opportunity to enhance the video encoding process by exploiting rendering information. In this chapter, we propose two techniques to improve CMR video encoding, aiming at enhancing the perceived video quality and reducing the computation complexity, respectively. Firstly, we develop a rendering-based prioritized encoding technique to improve the perceived video quality according to network bandwidth constraints. We first propose a technique to generate a macro-block level saliency map for every rendered video frame using rendering information. Further, based on such a saliency map, a prioritized rate allocation scheme is proposed to dynamically adjust the value of quantization parameter (QP) of each macroblock (MB). Experimental results indicate that

the perceptual quality can be greatly improved using the proposed technique.

We also develop a rendering-based encoding acceleration technique which utilizes rendering information to reduce the computation complexity of video encoding. This technique mainly consists of two parts. Firstly, we propose a method to directly calculate the motion vectors without employing the compute intensive motion search procedure. Secondly, based on the computed motion vectors, we propose a fast mode selection algorithm to reduce the number of candidate modes of each MB. Experimental results show that the proposed technique can achieve more than 42% saving in encoding time with very limited degradation in video quality.

## 4.1 Introduction

Although the CMR approach is promising in enabling rich 3D media applications (such as 3D game and augmented reality) on thin client devices, there exist two challenges for this approach:

(1) **Video Quality:** since the encoded video has to be streamed through the network which may have a limited and fluctuating bandwidth (especially mobile network), the video bit rate has to be adapted to cope with the available bandwidth. It is very challenging to maintain good video quality when network bandwidth is low. Moreover, unlike other video streaming applications such as video conferencing, CMR application has a high requirement for frame rate in order to ensure the fluidness of user experience. The high frame rate requirement makes it even more difficult to ensure a satisfactory video quality under limited bandwidth.

(2) **Real-time Encoding:** CMR is a highly interactive application and the rendered video frames need to be encoded in real-time. For example, consider playing a game with a



frame rate of 30 fps, which is believed to be sufficiently high to provide a smooth playing experience. Then in order to maintain a highly interactive playing experience, the encoding of each frame needs to be finished in 33.3 ms. The short encoding time budget is not easy to achieve if the game video is captured with large spatial resolution.

Unlike other video streaming applications such as Youtube or Skype, in CMR application, we have control over the rendering process (the generation of video content) on the cloud server; hence, we can exploit some rendering information such as the camera location and angle, the location of each object, the distance of each object to the camera, etc. to help address the above two challenges.

In regards to the video quality challenge, we can leverage rendering information to exploit the perception redundancy in video encoding to provide a better perceptual video quality. Previous research [WAN95] has shown that humans focus on small regions of the video rather than the entire video as a whole. Therefore, in this paper we develop ways to use rendering information to identify the importance of different regions, spend more bits on the important regions, and consequently, achieve a better perceptual video quality compared to allocating bits equally on the entire video frame.

About the real-time encoding challenge, we propose a rendering-based video encoding acceleration technique to exploit the rendering information to accelerate the encoding process. This technique consists of two parts: 1) we propose a method to directly calculate the motion vector between adjacent game frames, and use this calculation method to replace the regular compute intensive search based motion estimation method; 2) we propose a fast mode selection algorithm, which uses the homogeneity of motion vectors to reduce the number of candidate MB modes that need to be tested in the rate-distortion optimization process, ultimately reducing the computational complexity of video encoding.

The rest of the chapter is organized as follows: section 4.2 summarizes the related work. Section 4.3 explains the rendering-based prioritized encoding technique, including the extraction of macroblock importance and the bit rate allocation algorithm, followed by validation results. Meanwhile, in section 4.4 we explain in detail how to use rendering information to help reduce the encoding complexity for cloud gaming, including how to use rendering information to compute motion vectors, and how to use rendering information to efficiently select the encoding mode for each MB. Lastly, section 4.5 concludes this chapter and proposes some future work.

## 4.2 Related Work

In recent years, there has been a great deal of interest in CMR applications [WD12][LWD14][HJS13]. In CMR application, since the rendered video needs to be encoded in real-time and streamed through networks, it is vital to maintain good perceptual video quality and achieve fast encoding speed in order to ensure a satisfactory interactive user experience. There have been some studies aiming at enhancing the video quality or encoding speed, but most of them are based on adapting/optimizing the content complexity of the rendered video. For example, Wang and Dey analyzed in [WD13] the relation between video bit rate and rendering richness and based on that they proposed a rendering adaptation technique which adapts the rendering richness of video frame according to the network condition. In [SHI13], the authors proposed to adapt the scene complexity by omitting some less important objects in the game frame when the network bandwidth is low. However, in this paper, we address the challenges of video quality and encoding speed from a different angle: by optimizing the encoding process and keeping the graphic rendering richness untouched.

In order to enhance perceptual video quality under a given network constraint (bit rate limit), prioritized encoding can be a promising solution. There have been various works on prioritized video encoding which allocates more bits to regions of high importance in order to improve the video perceptual quality. However, for regular video, accurately extracting the importance information for different regions within a video frame is not a trivial task. In [LL08][WFC12][LGW04][LQI11], techniques have been developed to extract the importance information using either spatial or temporal domain characteristics, such as luminance or chrome values. These techniques are very compute intensive and therefore not applicable for cloud game application which requires real-time encoding.

Prioritized encoding techniques specific for CMR videos have not been very widely studied. In [TMP11], authors have proposed a method to identify the importance of MBs based on the depth values (under the assumption that the closer objects have higher priority from viewers' perspective). However, this assumption is not always true since the most important 3D object may not be the closest object to the camera. In this paper, we propose a more general and flexible framework that allows object importance to be defined according to the depth as well as the game scene information, and therefore can correlate better with human perception and achieves better video quality.

Video encoding acceleration techniques specific for CRM application have not been widely investigated either. In [TAH14], the authors propose a high level method to accelerate encoding by efficiently configuring the most important encoding parameters which can reduce the complexity while maintaining acceptable quality. However, instead of tuning the high level parameters like in [TAH14], our method takes a different approach, i.e., to use rendering information to optimize the motion estimation and mode selection process of H.264/AVC encoding.

On optimizing the motion estimation process, Fichtler [FE10] has proposed a rendering-based method to calculate the motion vector using the motion information of every MB's central pixel. Similarly, Semsarzadeh [SHJYS14] has proposed to use game objects' motion information to bypass the regular motion estimation process. These two methods [FE10][SHJYS14] operate at MB level and all the blocks within a MB have the same motion vectors, therefore may not achieve the best performance when encoding the MBs which contain multiple objects moving to different directions. In our work, we propose to generate pixel-level motion vectors which can lead to higher motion estimation accuracy and encoding quality.

On optimizing the mode selection process, the authors of [COS08] have also used depth information to accelerate the mode selection process. The method in [COS08] is based on the assumption that regions of similar depth are likely to correspond to regions of homogenous motion and, hence, are suitable for encoding using large partition. However, this assumption may not be true for MBs containing multiple small objects, which have similar depth but move in different directions. The encoding acceleration technique proposed in this paper uses the motion homogeneity instead of depth, therefore is not constrained by the above assumption.

Furthermore, Shi [SHC11] has proposed a 3D image warping assisted video coding method, which selects a set of key frames in a video sequence, uses the 3D image warping algorithm to interpolate the other non-key frames, and encodes the key frames and the residual frames separately. Although Shi's coding method achieves good encoding performance, there exist two limitations: 1) it works very differently from regular H264/AVC encoder, and hence cannot benefit from H264/AVC's features like motion estimation, mode selection, and variable block sizes, etc.; 2) a 3D warping engine needs to be implemented both on the

encoder and decoder, which increases the complexity. On the other hand, our proposed technique follows the framework of regular H264/AVC encoder, and hence is much easier to implement and adopted by the cloud gaming providers.

### 4.3 Rendering-based Prioritized Encoding Technique

In this section, we present a rendering-based prioritized encoding technique which aims at enhancing the video perceptual quality by adapting the encoding quality of the different regions of the video frame according to their importance. The proposed rendering-based prioritized encoding technique will firstly utilize rendering information such as pixel depth to compute the importance of different regions of game frame, convert this rendering information to a Macro-Block (MB) level saliency map, and finally find the optimal encoding parameter (in this paper we choose Quantization Parameter, QP) of each MB. The task of finding the optimal QP values for each MB is formed as an optimization problem. The optimal QP values are selected such that given the available bandwidth limit (bit rate budget), the perceptual video quality is maximized and the resulting video bit rate does not exceed the bandwidth limit.

This section is organized as follows. Firstly, in section 4.3.1, we give a high-level overview of the rendering-based prioritized encoding technique including two key components, the importance computing module and the bit rate allocation module. In section 4.3.2, we first introduce how to extract the rendering information from the rendering engine. In section 4.3.3, we introduce how to use rendering information to calculate the importance of each MB. In section 4.3.4, we explain the approach to determine the optimal QP for each MB according to its importance and the total bit rate budget. Lastly, in section 4.3.5, we discuss the experiment conducted to validate the performance of the proposed technique.

### 4.3.1 Overview of Rendering-based Prioritized Encoding Technique

Figure 4.1 shows an overview of the proposed rendering-based prioritized encoding technique. Firstly, the rendering engine generates a game frame coupled with depth information (a pixel-wise map which indicates the depth of each pixel), and scene composition information (which pixel corresponds to which 3D object and the importance of the object). Then the rendering information will be fed to an importance computing module, which is responsible for combining the depth information and scene composition information together to generate a macro-block based saliency map. Meanwhile, we use a network monitor proposed in our previous work [YWD14] to periodically measure the interactive latency by sending probe packets and then adapting the encoding bit rate according to the measured latency based on the bit rate adaptation algorithm also proposed in our previous work [WD13]. This bit rate adaptation algorithm will dynamically select the appropriate encoding bit rate such that the video bit rate will not exceed the network bandwidth and cause network congestion. Moreover, the selected bit rate target and the generated MB-level saliency map will be used as inputs to a bit rate allocation algorithm, which is responsible for deciding the optimal Quantization Parameter (QP) of each MB, such that the overall perceptual video quality is maximized and the encoding bit rate target is met. Finally the output of the bit rate

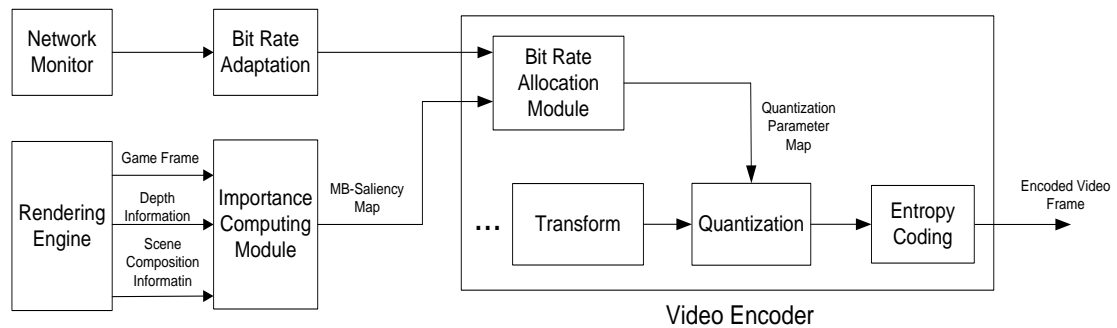


Figure 4.1. Overview of rendering-based prioritized encoding technique.

allocation module, a set of the QP values for each macro-block, will be passed to the quantization module of the encoder and the game frame will be encoded using these QP values.

From implementation perspective, the proposed rendering-based prioritized encoding process will be executed in a frame-by-frame manner. The bit rate adaptation algorithm will be executed periodically with a period of 1 second, which allows enough granularity for bit rate switching to cope with network bandwidth fluctuations.

### **4.3.2 Extraction of Rendering Information**

In order to develop a prioritized encoding technique, which utilizes rendering information to determine the importance of different regions of a video frame, the first step is to clarify which rendering information we need, how they correlate with importance of regions in the rendered frames, and how we extract them. In this subsection, we will explain these three topics.

From a user's perspective, the importance of a region is defined by the probability/likelihood that region will attract the user's attention. In other words, regions with high importance are the regions that users will pay more attention to. In this chapter, we suggest two kinds of rendering information which play significant roles in attracting human attention and can be conveniently extracted from the rendering engine: *depth information* and *scene composition information*.

Firstly, in any CMR application, there is a virtual camera that generates the video. One can easily make the observation that the objects closer to the camera (less depth) are more likely to attract player's attention [TMP11]. Hence, the depth value would be a good indicator for the importance of each pixel (smaller depth corresponds to higher importance). Moreover, the depth value of each pixel can be conveniently obtained using the well-known Z-buffer. During the rendering process, when 3D objects are mapped to a 2D plane, a depth value for

each pixel is computed to determine object occlusion. After a game frame is rendered, the depth value of each pixel will be stored in the Z-buffer. One can easily read the depth values stored in the Z-buffer without interfering with the rendering pipeline.

Secondly, besides the depth value, another important factor that determines the amount of attention a user pays to a particular region is the scene composition information, i.e., which region contains which object and the priority of the object. For example, figure 4.2 (a) shows a frame for the game PlaneShift [PS], which is a typical 3D role playing game. In this kind of game, the player controls the main avatar moving in a 3D world and performing a set of tasks such as fighting against enemy avatars. In this game frame, the main avatar and the enemy avatar would undoubtedly be more important than the other objects such as the buildings and trees. Within the video frame, the regions containing the avatars will draw more

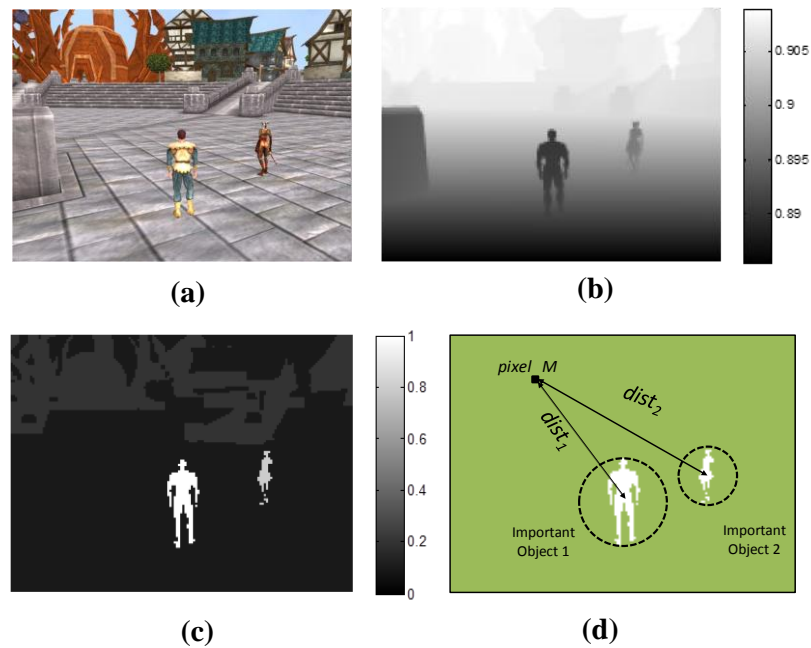


Figure 4.2. Game frame and rendering information: (a) game frame of PlaneShift; (b) values stored in Z-buffer; (c) values stored in Stencil buffer; (d) illustration of distance-based saliency.



attention from players.

The scene composition information is very helpful for identifying the importance of different regions. In general, this information is not available for regular video encoding and streaming applications, but CMR application offers the unique advantage for conveniently extracting scene composition information from the rendering engine. Similar to the depth values, we can obtain the scene composition information using the well-known Stencil buffer by following these two steps:

1) Firstly, we need to define a priority value for objects in the 3D world. Take the game frame shown in figure 4.2(a) as an example; we can define the priority of the main avatar (the closer avatar) to be 1, the priority of the enemy avatar (the further avatar) to be 0.8, the priority of buildings to be 0.2, and other objects (like the sky, floors, etc.) to be 0.1.

2) After defining the priority value for each object, we can store this priority value into the Stencil buffer during rendering. Stencil buffer is a pixel-wise buffer supported by all the mainstream graphic libraries such as OpenGL and DirectX. Both of these graphic libraries have API functions, which allow game developers to control what data to write into the Stencil buffer during rendering. By configuring the Stencil buffer appropriately, whenever an object is being rendered, its corresponding pixels in the Stencil buffer will be written with its priority value. The values stored in the Stencil buffer can be easily read out to help determining the importance of different regions.

Note that although we have proposed prioritization of 3D objects, it is a very flexible step in our framework and it can be implemented in several ways:

1) **Prioritize all objects:** if possible, the CMR application developer can assign a saliency value to each object.

2) **Prioritize some objects:** the application developer can define some objects, which

have certain functionalities or may affect the application logic, as important objects. Examples of these important objects are enemies, weapons, doors, etc. The application developers can only assign priority values to these objects, and set the other objects' priority to be 0.

3) **Prioritize only the main avatar:** the application developer can regard the main avatar as the only important object, set its priority to be 1, and set all the other objects' priority to be 0.

4) **No object prioritization:** in the case when application developers do not specify any object as important object, we will only use the depth information to determine saliency of a certain region.

Furthermore, the object prioritization step can be done offline and only once for a CMR application. For a new application, it can be done during the development period. Considering the amount of effort taken to develop a 3D game, the additional effort for assigning object priorities should be negligible. Also note that the priority value associated with an object will be static and not dependent on rendered scenes.

Another scenario that should be taken into consideration is that when a rendered frame contains transparent/semi-transparent objects. When a transparent/semi-transparent object is rendered, for all the pixels covered by this object, their Stencil buffer values need to be updated to be the maximum value of this transparent/semi-transparent object's priority and the old value stored in the Stencil buffer. For example, if in a rendered game frame there is an enemy hiding behind a semi-transparent glass, then the priority values stored in Stencil buffer will be the maximum value of the enemy object's priority and the glass object's priority.

### **4.3.3 Automatic Importance Computing using Rendering Information**

After explaining how to extract the rendering information (depth information and scene composition information), this subsection introduces the automatic importance

computing module, which takes rendering information as input and computes the saliency for each macro-block as output.

As explained in section 4.3.2, we can configure the Z-buffer and Stencil buffer such that, after a frame is rendered, the Z-buffer and Stencil buffer will contain the depth value and object priority value for each pixel. As an example, figure 4.2(b) and figure 4.2(c) show the values stored in the Z-buffer and Stencil buffer, respectively, after the frame shown in figure 2(a) is rendered.

The values stored in Z-buffer and Stencil buffer can be utilized to determine the importance of each pixel within the frame. In order to do that, we first define Important Object (IO) as the object whose priority value is larger than a certain threshold. As an example, in the frame shown in figure 4.2(a), if we set a priority threshold of 0.6, then among all the objects in the frame, only the two avatars have priorities higher than 0.6 (according to figure 4.2(c)). Hence, there are only two important objects in the frame, as shown in figure 4.2(d).

We now propose two metrics to quantify the importance of each pixel using the depth information from the Z-buffer and the important region information derived using the Stencil buffer:

- **Depth-based saliency** ( $S_{depth}$ ): for each pixel, its depth-based saliency is a metric related with the depth of the pixel. The depth-based saliency for pixel  $M$ ,  $S_{depth}(M)$ , can be computed using equation (4.1). In equation (4.1),  $Z(M)$  is the value stored in the Z-buffer (a number between 0 and 1, as shown in figure 4.2(b)). Larger  $S_{depth}$  value corresponds to lower depth and higher saliency.

$$S_{depth}(M) = 1 - Z(M) \quad (4.1).$$

- **Distance-based saliency** ( $S_{distance}$ ): for each pixel, its distance-based saliency is a value related to its location and the location of all the important objects within the frame.

For a pixel  $M$ , in order to compute its  $S_{distance}$  value,  $S_{distance}(M)$ , two scenarios need to be considered:

1) if pixel  $M$  is inside an important object, then  $S_{distance}(M)$  is equal to the priority of that important object.

2) if pixel  $M$  is not inside any important object, then  $S_{distance}(M)$  is dependent on the distances from pixel  $M$  to the important objects of the game frame. Take figure 4.2(d) as an example. There are two important objects within that frame (the two avatars), and the distance-based saliency for pixel  $M$  should be computed as:

$$S_{distance}(M) = \frac{1}{2 \times \log D} (P_1 \times \log \frac{D}{dist_1} + P_2 \times \log \frac{D}{dist_2}) \quad (4.2).$$

In equation (4.2),  $P_1$  and  $P_2$  are coefficients that represent the object priority of each avatar. Coefficient  $dist_1$  and  $dist_2$  are the distances from pixel  $M$  to the centers of the two important objects' minimum bounding circles (in unit of pixels), as shown in the figure.  $D$  is the diagonal length of the video frame (in unit of pixels) and it can be regarded as constant in equation (4.2). Note that we use the diagonal length of video frame,  $D$ , to divide the distance values,  $dist_1$  and  $dist_2$ , in order to get rid of the influence of video spatial resolution. We use the log function in equation (4.2) because it can distribute importance in a similar manner the human visual system works, such that the quality degradation starts to be perceived by the user only when the distance,  $dist_1$  and  $dist_2$ , increase to a certain value. In [WFC12], the authors have compared the performance of modeling this relation using a linear function and a logarithmic function. The experimental results show that the logarithmic function achieves a much better correlation with human perception than the linear function.

Equation (4.2) is for the frames containing two important objects. In general, suppose a video frame contains  $T$  important objects, then  $S_{distance}(M)$  is represented by:

$$S_{distance}(M) = \begin{cases} P_i & (\text{if } M \in IO_i) \\ \frac{1}{T} \sum_{i=1}^T P_i \times \frac{\log \frac{D}{dist_i}}{\log D} & (\text{if } M \notin IO_i, \forall i \in [1, T]) \end{cases} \quad (4.3).$$

In equation (4.3),  $T$  is the number of important objects (IO),  $P_i$  is the priority of the  $i$ -th IO, and  $dist_i$  is the distance from pixel  $M$  to the center of  $i$ -th IO's minimum bounding circle (only when pixel  $M$  is not inside of any IO).

After defining the *depth-based saliency* and *distance-based saliency* for each pixel, we combine them together to generate a macro-block (MB) level saliency map. The procedure to compute the MB-level saliency map consists of the following steps:

1) Firstly, normalize the depth-based and distance-based saliency, respectively. Let  $S_{depth}(M)$  be the depth-based saliency for pixel  $M$ . We first normalize  $S_{depth}(M)$  as:

$$S_{depth}^N(M) = \frac{S_{depth}(M)}{\overline{S_{depth}}} \quad (4.4).$$

where  $S_{depth}^N(M)$  indicates the normalized depth-based saliency for pixel  $M$  and  $\overline{S_{depth}}$  indicates the average depth-based saliency of all the pixels in the current frame. Similarly, we normalize  $S_{distance}(M)$ , the distance-based saliency for pixel  $M$ , as:

$$S_{distance}^N(M) = \frac{S_{distance}(M)}{\overline{S_{distance}}} \quad (4.5).$$

Note that during the normalization step (equation (4.4)(4.5)), to compute  $S_{distance}^N(M)$  and  $S_{depth}^N(M)$  for pixel  $M$ , we have already taken other pixels' distance-based saliency and depth-based saliency into consideration. Hence, after the normalization step, the  $S_{distance}^N(M)$  and  $S_{depth}^N(M)$  values are good indicators for the relative importance of pixel  $M$ .

Furthermore, this normalization step (equation (4.4)(4.5)) will execute frame by frame and it can automatically handle new emerging important objects. For the same pixel  $M$ , its

$S_{depth}^N(M)$  and  $S_{distance}^N(M)$  values will keep varying every frame, depending on how the current rendered frame is composed.

2) The normalized depth-based and distance-based saliencies are next bounded as

$$S_{depth}^N(M) = \max\{S_L, \min\{S_{depth}^N(M), S_U\}\} \quad (4.6).$$

$$S_{distance}^N(M) = \max\{S_L, \min\{S_{distance}^N(M), S_U\}\} \quad (4.7).$$

where  $S_L$  and  $S_U$  are two constants to denote the lower and upper boundaries of  $S_{depth}^N(M)$  and  $S_{distance}^N(M)$ . The thresholds  $S_L$  and  $S_U$  are used for calibrating saliency values in a similar manner the human visual system works, such that if the computed  $S_{depth}^N(M)$  and  $S_{distance}^N(M)$  values are too large or too small, we bound them to a reasonable range. The values of  $S_L$  and  $S_U$  are determined based on a study where we record multiple CMR video sequences and then analyze the distribution of  $S_{depth}^N(M)$  and  $S_{distance}^N(M)$  for each rendered frame. We find that for all the frames, more than 90% of the pixels have their  $S_{depth}^N(M)$  values distributed within [0,4], and more than 90% of the pixels have their  $S_{distance}^N(M)$  values within [0, 4]. Hence we choose 0 and 4 as the default values for  $S_L$  and  $S_U$ , respectively. The details of the study can be found in [SUP2].

3) Because the video is encoded at the macro-block level, we combine the pixel level saliencies to compute the saliency at the MB level:

$$S_{combined}(i) = \sum_{j=1}^{16} \sum_{k=1}^{16} [\alpha \times S_{distance}^N(i, j, k) + (1 - \alpha) \times S_{depth}^N(i, j, k)] \quad (4.8).$$

where  $i$  is the MB index,  $j$  and  $k$  are the pixel index within one MB. As shown in equation (4.8),  $S_{combined}(i)$  is a weighted average of  $S_{depth}^N(i, j, k)$  and  $S_{distance}^N(i, j, k)$ . Coefficient  $\alpha$  is used to control the relative weight between the depth-based saliency and distance-based saliency, and it takes value between 0 ~ 1.

4) After steps 1~3, the  $S_{combined}(i)$  values of the spatial neighboring MBs within a local area would sometimes vary intensively. If the encoder allocates QP according to the  $S_{combined}(i)$  values, the resulting visual quality of the neighboring MBs may vary significantly, leading to block artifacts. To solve this problem, a weighted 3x3 mean filter MF is applied to all the MBs in the rendered frame to smoothen the saliency map  $S_{combined}$ :

$$M F = \begin{bmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{3} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \end{bmatrix} \quad (4.9).$$

After steps 1~4, we have converted the pixel level saliency values ( $S_{distance}$  and  $S_{depth}$ ) to a MB level map  $S_{combined}$ . Figure 4.3 shows an example of a game frame, and the associated  $S_{combined}$  with  $\alpha$  varying between 0 and 1. From figure 4.3(b) ~ (f) we can clearly see how the coefficient  $\alpha$  can be used to control the relative weight between  $S_{distance}$  and  $S_{depth}$ . When  $\alpha$  equals 0, we can tell from equation (4.8) that  $S_{combined}$  will be equal to  $S_{depth}$ , which is shown in

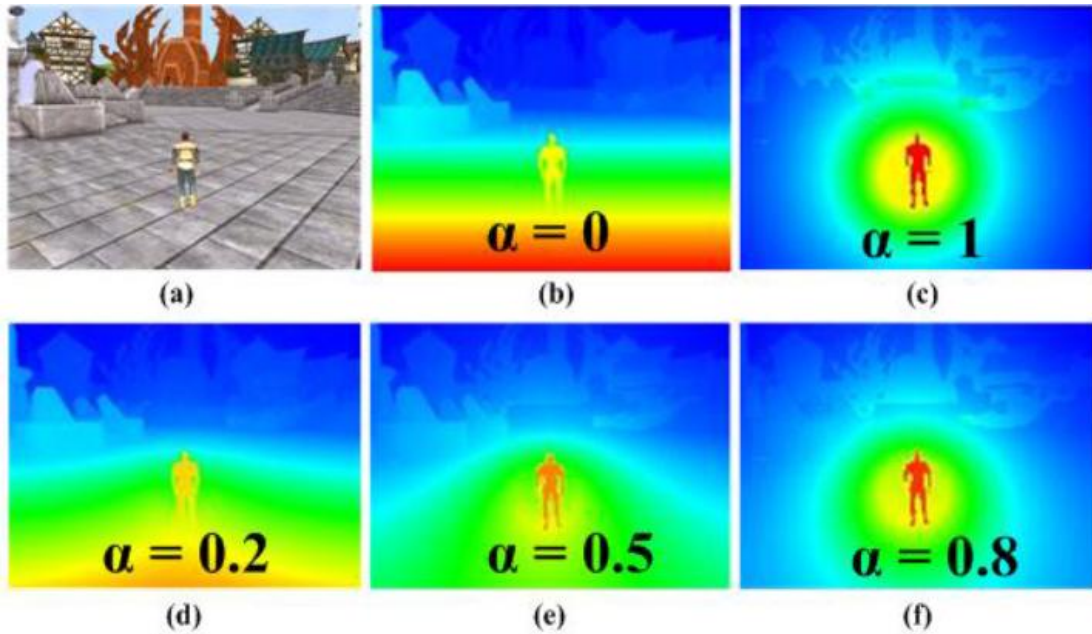


Figure 4.3. Game frame and combined saliency map ( $S_{combined}$ ) for different  $\alpha$  values.

figure 4.3(b); on the other hand, when  $\alpha$  equals 1,  $S_{combined}$  will be simplified to  $S_{distance}$ , which is shown in figure 4.3(c). Figure 4.3(d)(e)(f) can be regarded as a weighted average of figure 4.3(b) and (c), depending on the  $\alpha$  value.

In order to determine the optimal  $\alpha$  value for game PlaneShift, we have performed a study where we invite a group of subjects to play cloud game. As a subject is playing, we vary the  $\alpha$  value from 0 to 1, and ask the subject to select the best  $\alpha$  value. The best  $\alpha$  value means the one which assigns the MB saliency closest to the subject's perception. According to the results, we find that most subjects choose 0.5 as the optimal value and hence, in this study we heuristically select 0.5 as the default  $\alpha$  value. The detail of the experiment result can be found in [SUP2]. Note that the optimal  $\alpha$  value may be different for different kinds of CMR applications. If the CMR service provider wants to experiment with different CMR applications, they can easily repeat this experiment and analyze the distribution of subjects' evaluations.

#### 4.3.4 Rate Allocation Algorithm

In this subsection, we describe a bit rate allocation algorithm which aims at determining the optimal QP values of each macro-block, such that a specific bit rate budget is met and the overall visual quality is maximized.

Firstly, for any macro-block  $MB_i$ , we adopt the linear relation model [DL96] between the compression distortion in terms of Mean Square Error (MSE), and the Quantization Step  $q_i$ :

$$MSE_i = k \times q_i + b \quad (4.10).$$

Secondly, we adopt the rate model proposed in [CZ00], that the encoding bit rate can be modeled using the quantization step  $q_i$  as:

$$R_i = \frac{\theta}{q_i^\gamma} \quad (4.11).$$



After introducing the relation between the distortion and quantization step (equation (4.10)) and the relation between bit rate and quantization step (equation (4.11)), we formulate the rate allocation task as the following optimization problem:

**Given:**

Video bit rate target  $B_T$ ,

**Find** the optimal quantization step for each MB,  $q_i$ , so as to

**Minimize:**

$$WMSE = \frac{1}{N} \sum_{i=1}^N S_{combined}(i) \times MSE_i \quad (4.12).$$

$$\text{Subject to} \quad R = \frac{1}{N} \sum_{i=1}^N R_i \leq B_T \quad (4.13).$$

In this problem formulation, we use the Weighted Mean Square Error ( $WMSE$ ) as the object to minimize because  $WMSE$  is the weighted average of the distortion for all the MBs incorporating the saliency value of each MB. Therefore minimizing  $WMSE$  is equivalent of maximizing the human perceived video quality.

In the objective function  $WMSE$ , the saliency map of each MB,  $S_{combined}(i)$ , can be extracted from rendering engine and can be regarded as known parameters. Furthermore, according to equation (4.9) and (4.10), both  $MSE_i$  and  $R_i$  are functions of the quantization step  $q_i$ . Hence both the objective function and constraint function can be expressed as function of  $q_i$ , and we can use the well-known Lagrange method to solve this optimization problem.

Firstly, according to the problem formulation, the corresponding Lagrange function can be stated as:

$$f(q_1, \dots, q_N) = \frac{1}{N} \sum_{i=1}^N S_{combined}(i) \times MSE_i + \lambda \times \left( \frac{1}{N} \sum_{i=1}^N R_i - B_T \right) \quad (4.14).$$

where  $\lambda$  is the Lagrange multiplier that needs to be determined. Substituting equations (4.10) and (4.11) into equation (4.14) yields:

$$f(q_1, \dots, q_N) = \frac{1}{N} \sum_{i=1}^N (k \times q_i + b) S_{combined}(i) + \lambda \left[ \frac{1}{N} \sum_{i=1}^N \frac{\theta}{q_i^\gamma} - B_T \right] \quad (4.15).$$

To obtain the optimal solution for  $\{q_i\}$ , we use the Lagrange method:

$$\frac{\partial f}{\partial q_1} = \frac{\partial f}{\partial q_2} = \frac{\partial f}{\partial q_3} = \dots = \frac{\partial f}{\partial q_N} = \frac{\partial f}{\partial \lambda} = 0 \quad (4.16).$$

By solving equation (4.15), we obtain the optimal  $q_i$  value:

$$q_i^* = \left( \frac{\theta}{B_T N} \times \frac{\sum_{i=1}^N S_{combined}^{\frac{\gamma}{1+\gamma}}(i)}{S_{combined}^{\frac{\gamma}{1+\gamma}}(i)} \right)^{\frac{1}{\gamma}} \quad (4.17).$$

We can see from equation (4.16) that the optimal quantization step  $q_i^*$  is dependent on  $MB_i$ 's importance,  $S_{combined}(i)$ . For a given MB, a high importance value will correspond to a low quantization step and good encoding quality. The  $q_i$  value is also dependent upon the model coefficients  $\theta$  and  $\gamma$  (shown in equation (4.11)), which are determined by the video content characteristics and encoding parameters such as frame rate.

Finally we convert the quantization step  $q_i$  to the quantization parameter  $QP_i$ , since QP is the parameter that adjustable in the encoder. The relation between QP and  $q_i$  specified in the H.264/AVC [X264] standard is

Table 4.1. Parameters for video coding.

Codec	H.264/AVC
Total number of frames	1200 frames
Frame Rate	30 fps
Spatial Resolution	800x600 (PlaneShift) and 1280x720 (BroadSides)
GOP size	30 frames
Enable B-frames	No
Bit Rate Target	{ 600, 800, 1000, 1500, 2000 } kbps

$$QP_i^* = \text{round}[6 \times \log(q_i^*) + 4] \quad (4.18).$$

The round function is used to ensure the  $QP_i^*$  value is an integer such that the encoder can set it.

### 4.3.5 Subjective Assessment Experiments

In order to demonstrate the effectiveness of the proposed rendering-based prioritized encoding technique, in this section we present the procedure and results of a subjective assessment experiment where we asked subjects to compare the quality of two sets of pre-encoded game videos. One set of test videos are encoded with regular H.264/AVC encoder, and the other set are encoded with the proposed prioritized encoding technique.

In order to demonstrate the generality and robustness of our proposed technique, our experiment – in addition to the previously introduced game PlaneShift - includes another game called Broadsides [BRO], as introduced in section II.

We captured two raw game videos for PlaneShift and Broadsides, respectively. Then we encode them using the typical parameter settings for H.264/AVC real-time encoding, as tabulated in Table 4.1. As shown in Table 4.1, each of these 2 raw videos has a length of 40 seconds. The spatial resolution is chosen to be 800x600 and 1280x720. In addition, the bit rate target varies between 600 kbps to 2000 kbps. The model coefficient  $\alpha$  (in equation (4.8)) is set to be 0.5. The coefficients  $\theta$  and  $\gamma$  (in equation (4.11)) are set to be 7800 and 0.68.

For each raw video and bit rate target, two versions of encoded videos are created by encoding the same raw video twice. The first video is encoded in the usual manner such that the entire frame is encoded with the same quality. The second video is encoded using the proposed technique.

We recruited 22 subjects to participate in this experiment, most of whom are students from UCSD. The subject group included 14 males and 8 females with an age distribution ranging from 18 to 28. All of them have some prior experiences of playing video games.

The test procedure is conducted in a similar fashion to the recommendation from ITU-R BT.500-12, double-stimulus continuous quality-scale (DSCQS) [P800] in order to evaluate the quality of the encoded video with and without the proposed technique under a fixed bit rate target. The test was conducted indoor with good light condition. An iPad Air tablet with a customized application was used to play videos and collect participants' evaluations. The test procedure takes approximately 25 minutes, including 5 minutes of briefing and 20 minutes of user study. During the briefing, the subject was presented with a training video with

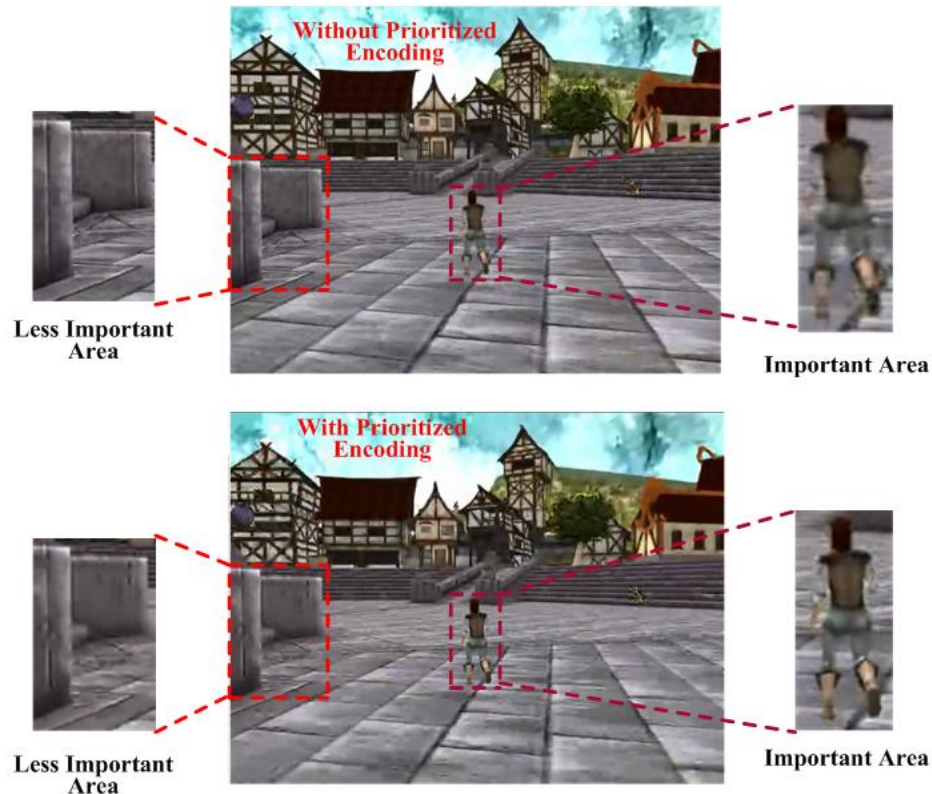


Figure 4.4. Encoded video frames without and with rendering-based prioritized encoding, under the video encoding bit rate budget of 1000 kbps.

instructions on how assessment will be conducted. The outlines of the 20 minutes user study are as follows:

1) For each game, five pairs of test videos are presented to the subject. Each pair corresponds to a bit rate target and consists of a normally encoded video and a prioritized encoded video.

2) The subject watches the test videos one pair at a time. Within each pair, the normally encoded video and the prioritized encoded video are presented in a random order, and the subject is unaware of the order.

3) After watching a pair of videos, the subject is asked to rate each of the two video's quality according to a five-point rating scale ranging from 1 (bad), 2 (poor), 3 (fair), 4 (good), and 5 (excellent). For each pair, the subject is able to switch between the two videos until he/she makes a decision.

As an example, figure 4.4 shows two encoded video frames without and with prioritized encoding, with the bit rate target set to be 1000 kbps. As we can see, for important

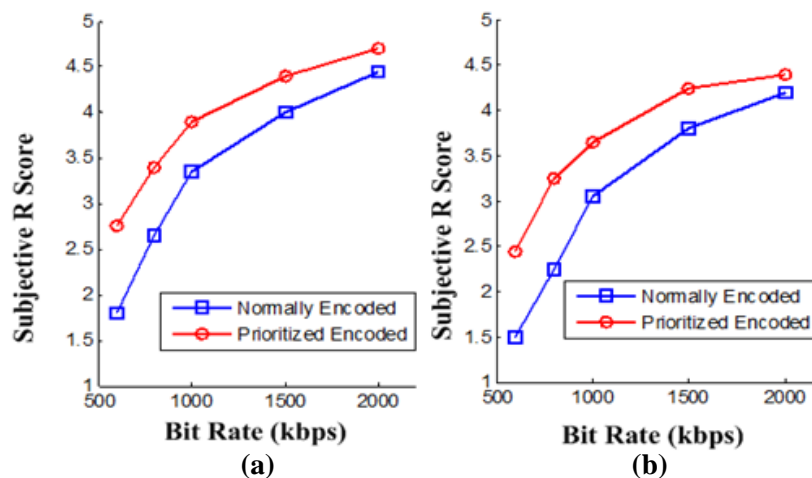


Figure 4.5. Subjective evaluation of two games under different encoding bit rates: (a) for game PlaneShift; (b) for game Broadsides.

areas such as the avatar, the video encoded without prioritized encoding has a much lower quality than the one encoded with prioritized encoding. Concurrently, for the less important areas of the game frame, rendering-based prioritized encoding leads to lower quality. This figure demonstrates the idea that the prioritized encoding increases the quality of important areas at the cost of quality degradation in less important areas.

The average subjective evaluations over all the subjects are shown in figure 4.5. We can see that for both games, the prioritized encoded videos lead to higher user experience than the normally encoded videos (without prioritized encoding). The gap between with and without prioritized encoding is big when bit rate target is low, and it will decrease as the bit rate target increases. This is due to the fact that, when the bit rate target increases to a certain value, the visual quality of the important areas are already sufficiently good and spending more bits on these areas will not further increase the visual quality, hence the advantage of prioritized encoding will not be as noticeable compared to the low bit rate cases.

Furthermore, in the experiment described above, except for video quality comparison, we have also compared the encoding time for the normal encoder and the proposed encoder. We find that the encoding time of the two encoders are very close, which means that the additional computation overhead introduced by our proposed technique is negligible. The details of the experiment results and explanation can be found in [SUP2].

## **4.4 Rendering-based Encoding Acceleration Technique**

The rendering-based prioritized encoding technique described in section 4.3 aims at increasing the perceptual video quality under a fixed bit rate budget. We can think about this approach from another perspective, that in order to achieve the same perceptual video quality, the prioritized encoding technique can help reduce the required video bit rate that needs to be

streamed from the cloud servers. Hence, the proposed technique will be of interest for CMR service providers since it can reduce the bandwidth consumption of the cloud server and hence bring down the operational cost.

In this section, we propose another technique, referred to as rendering-based encoding acceleration (REA) technique, to further reduce the operational cost for CMR application by reducing the computation complexity of video encoding.

It is well known that the most widely used video coding standard H.264/AVC achieves significantly higher encoding efficiency than the previous standards by adopting a set of new features such as adaptive motion compensation with variable block sizes, rate-distortion optimization (RDO) technique, and so on. However, although H.264/AVC has a much higher encoding efficiency compared to the other standards, it also induces significantly higher computation complexity which poses a challenge for applications like cloud gaming which requires real-time encoding and high interactivity.

The high computation complexity of H.264/AVC standard is mainly because of the search-based motion estimation and the large number of candidate modes. In order to address these two bottlenecks of encoding complexity, in this chapter, we propose an encoding acceleration technique, which mainly consists of two parts:

- 1) We propose a direct motion vector calculation technique to utilize rendering information to directly calculate the motion vector for each pixel in a video frame, therefore bypassing the compute intensive search-based motion estimation process.

- 2) We propose a fast MB mode selection algorithm which exploits the motion homogeneity of the video frame and other rendering information, to efficiently determine a subset of candidate modes for each MB and skip the unlikely and unnecessary encoding modes.

This chapter is organized as following. In section 4.4.1, we described in detail how to use rendering information to directly calculate the motion vectors. In section 4.4.2, we further reduce computational complexity by introducing a fast MB mode selection algorithm. Section 4.4.3 presents the experimental results demonstrating the performance improvement achieved by the proposed technique.

#### 4.4.1 Direct Calculation of Motion Vectors

In H.264/AVC standard, temporal correlation in video sequence is exploited by block-based motion compensation, where the MBs of the current frame are predicted from previously encoded frames. The removal of the correlation redundancy leads to high compression efficiency but the search for the best motion vectors is a compute intensive process. Although researchers have developed various fast search methods for motion vectors, the search-based motion estimation still contributes to a big portion of the time taken by the video encoding process.

Fortunately, CMR application offers a unique opportunity to make the motion vectors much easier to be obtained. In CMR application, we have control over the rendering process (video content generation process), and we can use some rendering information to directly calculate the motion vectors with a set of calculations such as matrix multiplication, instead of using the compute intensive search-based approach specified in the H.264/AVC standard. In this subsection, we will explain this direct calculation method for obtaining the motion vectors.

The basic idea for direct calculation method is that, for each pixel in the current frame (assume the corresponding screen coordinates is  $S_c=(sx_c, sy_c)$ ), we want to find the corresponding pixel location ( $S_p=(sx_p, sy_p)$ ) in the previous frame (reference frame). Then the motion vector for this pixel is the difference between these two locations ( $S_c$  and  $S_p$ ) multiplied by four to account for the quarter-pixel resolution of H.264/AVC [X264].



In order to calculate the pixel location in the previous frame ( $S_p$ ), we need to obtain the following rendering information: the camera projection matrix for the current frame ( $PM_c$ ) and the previous frame ( $PM_p$ ) as well as the depth value for the pixel in the current frame ( $z_c$ ), which can be extracted conveniently from the Z-buffer. The procedure to compute  $S_p$  is the following:

Firstly, convert the pixel location (screen coordinates) in the current frame ( $S_c = (sx_c, sy_c)$ ) to the 3D world coordinates ( $W_c = (wx_c, wy_c, wz_c)$ ) using the inverse of the projection matrix  $PM_c^{-1}$ :

$$W_c = PM_c^{-1} \times [sx_c, sy_c, z_c]^T \quad (4.19).$$

Secondly, after obtaining the 3D world coordinates in the current frame ( $W_c$ ), we need to find the corresponding 3D world coordinate in the previous frame:

$$W_p = \begin{cases} W_c & \text{(if } W_c \text{ belongs to a stationary object)} \\ W_c + (-V_o)T_f & \text{(if } W_c \text{ belongs to a moving object)} \end{cases} \quad (4.20).$$

As shown in equation (4.20), if the 3D coordinates  $W_c$  belongs to a static object, then its 3D coordinates (coordinates in the 3D world) in the previous frame,  $W_p$ , will be the same as the location in the current frame,  $W_c$ . On the other hand, if  $W_c$  belongs to a moving object, such as a moving enemy avatar, we need to calculate  $W_p$  by considering the velocity of this moving object ( $V_o$ ), as well as the time difference between two consecutive frames ( $T_f$ ). The information of  $V_o$  and  $T_f$  can be conveniently obtained from the rendering engine.

Thirdly, the 3D world location  $W_p$  is re-projected to 2D pixel location using the projection matrix of the previous frame:

$$S_p = PM_p \times W_p \quad (4.21).$$

Finally, the motion vector is calculated as:

$$MV = 4(S_p - S_c) \quad (4.22).$$

Hence, for each pixel in the current rendered frame (at screen coordinate  $S_c$ ), we can compute its motion vector ( $MV$ ) through a few simple calculations such as matrix multiplication. This method is much faster than the regular search-based motion estimation process.

Although this direct calculation method is convenient, there exists two kinds of problematic cases where the proposed method cannot be applied:

1) *New emerging objects*: it is common in CMR videos that some objects will suddenly appear in a frame but did not exist in the previous frame. For the pixels and MBs that cover these new emerging objects, there exists no motion vector. We need to disable the proposed direct calculation method once we detect these pixels.

These pixels can be detected by using the Stencil buffer. Each object in a rendered frame has a unique object ID. During the rendering process, as an object is being rendered, we

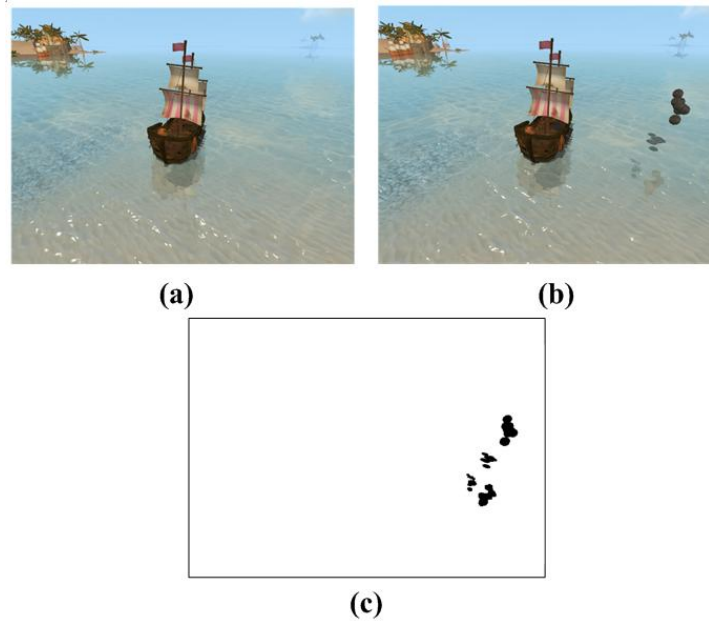


Figure 4.6. Two consecutive frames of game *Broadsides* and the corresponding uncovered pixels: (a) the previous frame (reference frame); (b) the latter frame (current frame to be encoded); (c) black pixels correspond to the uncovered pixels detected using the Stencil buffer.

can store the associated object ID in the correct locations of Stencil buffer. As the videos are rendered, whenever we detect a new object ID appearing in the Stencil buffer that does not exist in the previous frame's Stencil buffer, we can mark this object as a new emerging object. For example, figure 4.6 shows an example of two consecutive frames of game *Broadsides*. Figure 4.6(a) is the reference frame (previous frame) and figure 4.6(b) is the latter frame (current frame that needs to be encoded). The main difference between these two frames is that in the current frame, the ship has shoot some black shells out, and these shells are the new emerging objects that do not exist in the reference frame. We show in figure 4.6(c) the detected new emerging objects using Stencil buffer.

**2) *Pixels pointing out of the frame*:** during the direct MV calculation process (equation (4.19) ~ (4.22)), it is possible that the compute pixel location in previous frame ( $S_p$ ) is not within the frame's spatial range. An intuitive interpretation for this kind of situation is that the motion vector points out of the frame, and in this case we need to disable the proposed direct calculation method as well.

In this thesis, we will name the pixels that point out of the frame or cover new emerging objects as uncovered pixels. For these pixels, there exists no corresponding pixel in the reference frame and our proposed rendering based direct calculation method will not work. Furthermore, if a MB contains uncovered pixels, we name it as ***uncovered MB***. As will be explained later, in our proposed encoding acceleration technique, we will treat these uncovered MBs differently from other MBs. For these uncovered MBs, we need to use regular search-based ME methods to obtain their MVs.

Note that the output of the direct calculation technique will be pixel-level motion vectors (MV). As will be explained in the next subsection, the pixel-level MV will first be exploited to determine the candidate mode set for each 16x16 MB. Then in the following step

of computing the R-D cost for each possible mode, the pixel-level MV will be combined into block-level MV. In this study, we propose to use the average vector of all the associated pixels' pixel-level MV as the block-level MB.

#### 4.4.2 Fast Mode Selection Algorithm

As introduced in section 4.4.1, H.264/AVC standard uses RDO technique to select the optimal mode from a set of candidate modes with block size ranging from 16x16 to 4x4.

In the scope of intra-encoding, large size MB partition is suitable for smooth areas with low spatial complexity, and small partition is suitable for detailed areas with high spatial complexity. In the scope of inter-modes where motion compensation is used to reduce temporal redundancy, large partition is suitable for the MBs exhibiting homogeneous motion properties, and small size MB is suitable for MBs with complex motion. For example, if a MB contains multiple objects that move in different directions, than using large partition (such as 16x16) for motion compensation will lead to large motion compensated residual, which severely degrades the encoding efficiency.

Since CMR application provides us the unique advantage for conveniently calculating pixel-wise motion vectors, in this section we propose a fast mode selection algorithm which exploits these motion vector characteristics to help select the appropriate block size. The basic idea is that by analyzing the homogeneity of the pixel-wise motion vectors within a 16x16 MB, we propose a set of candidate coding modes and eliminate unlikely modes in order to reduce the modes that need to be tested in rate-distortion optimization.

Inspired by the tree-structure block sizes defined in H.264/AVC [X264], we propose 4 metrics to indicate the homogeneity for different regions within a 16x16 MB.

The first metric,  $\text{VAR}_{16 \times 16}$  is the variance of all the 256 pixels' motion vectors:

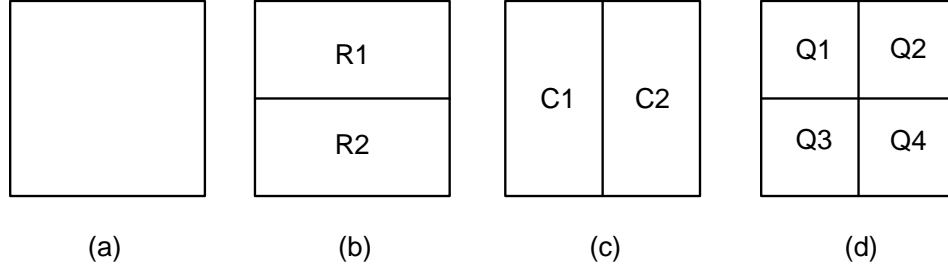


Figure 4.7. Different partitions for 16x16 MB: (a) 16x16; (b) 16x8; (c) 8x16; (d) 8x8.

$$VAR_{16 \times 16} = \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} \left| PMV_{i,j} - \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} PMV_{i,j} \right|^2 \quad (4.23).$$

where  $i$  and  $j$  are the row and column index of a pixel within a 16x16 MB, and  $PMV_{i,j}$  is the pixel-level motion vector for the pixel located at row  $i$  and column  $j$ .

The second metric,  $VAR_{16 \times 8}$  is defined as the average variance of the top half and bottom half's pixel-level motion vectors. As shown in figure 4.7(b), we denote the top half of a MB as region  $R_1$ , and denote the bottom half as region  $R_2$ . Then  $VAR_{16 \times 8}$  is computed as:

$$VAR_{16 \times 8} = \frac{1}{2} \sum_{k=1}^2 \frac{1}{128} \sum_{(i,j) \in R_k} \left| PMV_{i,j} - \frac{1}{128} \sum_{(i,j) \in R_k} PMV_{i,j} \right|^2 \quad (4.24).$$

Similarly, the other two terms  $VAR_{8 \times 16}$  and  $VAR_{8 \times 8}$  are defined as the average variance of different regions' pixel-level motion vectors, and the corresponding region partitions are shown in figure 4.7(c) and (d).

$$VAR_{8 \times 16} = \frac{1}{2} \sum_{k=1}^2 \frac{1}{128} \sum_{(i,j) \in C_k} \left| PMV_{i,j} - \frac{1}{128} \sum_{(i,j) \in C_k} PMV_{i,j} \right|^2 \quad (4.25).$$

$$VAR_{8 \times 8} = \frac{1}{4} \sum_{k=1}^4 \frac{1}{64} \sum_{(i,j) \in Q_k} \left| PMV_{i,j} - \frac{1}{64} \sum_{(i,j) \in Q_k} PMV_{i,j} \right|^2 \quad (4.26).$$

$VAR_{16 \times 16}$ ,  $VAR_{16 \times 8}$ ,  $VAR_{8 \times 16}$  and  $VAR_{8 \times 8}$  are used to measure the homogeneity of the 256 pixel-wise motion vectors in different regions (higher value corresponds to lower homogeneity). We can use this information to estimate what mode will have the higher

probability leading to a low R-D cost. For example, for a 16x16 MB, if the metric  $VAR_{16x8}$  is very low, then we can infer that the pixels in the upper half (region  $R_1$  in figure 4.7(b)) have homogenous motion and the pixels in the bottom half (region  $R_2$  in figure 4.7(b)) have homogenous motion as well. Then encoding region  $R_1$  and  $R_2$  separately (encoding this MB using Inter\_16x8 mode) may lead to small motion compensated residuals and low R-D cost; hence, Intra\_16x8 is very likely to be selected as the optimal mode after the RDO process.

Inspired by the above analysis, we classify each 16x16 MB into one of the following categories when the specified condition is satisfied.

**Category A:** the entire 16x16 MB is homogeneous:

$$VAR_{16x16} \leq T_c \quad (4.27).$$

**Category B:** the motion within the 16x16 MB is complex and exhibits no obvious homogeneity in any sub-partition:

$$VAR_{16x16} > T_c \text{ and } VAR_{16x8} > T_c \text{ and } VAR_{8x16} > T_c \text{ and } VAR_{8x8} > T_c \quad (4.28).$$

If the MB doesn't satisfy the above two conditions, it is further classified into one of the following categories:

**Category C:** the motion in the MB is more likely to be homogeneous within the top half and the bottom half, therefore it is suitable to be encoded using 16x8 partition.

$$VAR_{16x8} < VAR_{8x16} \text{ and } VAR_{16x8} \leq T_c \quad (4.29).$$

**Category D:** the motion in the MB is more likely to be homogeneous within the left half and the right half, therefore it is suitable to be encoded using 8x16 partition.

$$VAR_{8x16} < VAR_{16x8} \text{ and } VAR_{8x16} \leq T_c \quad (4.30).$$

**Category E:** the MB is more suitable to be encoded using 8x8 partition.

$$VAR_{8x8} \leq T_c \text{ and } VAR_{16x8} > T_c \text{ and } VAR_{8x16} > T_c \quad (4.31).$$

Table 4.2. Candidate inter-mode for each MB category.

MB category	Candidate Inter-modes
A	16x16
B	16x16, 16x8, 8x16, 8x8
C	16x16, 16x8
D	16x16, 8x16
E	16x16, 8x8

The value of the threshold TC is selected to be 0.25 by extensive experiments. This value achieves a good and consistent performance on a variety of videos including different game scenes and activities.

Based on the above MB classification according to the pixel-level motion homogeneity, the candidate inter-modes for each MB category is summarized in Table 4.2 (the prefix Inter\_ is omitted).

By utilizing the motion homogeneity, we have established a heuristic to generate candidate inter-modes. Further optimization can be achieved by using the detection of uncovered pixels to eliminate the inter-modes for some MBs. This idea is inspired from the

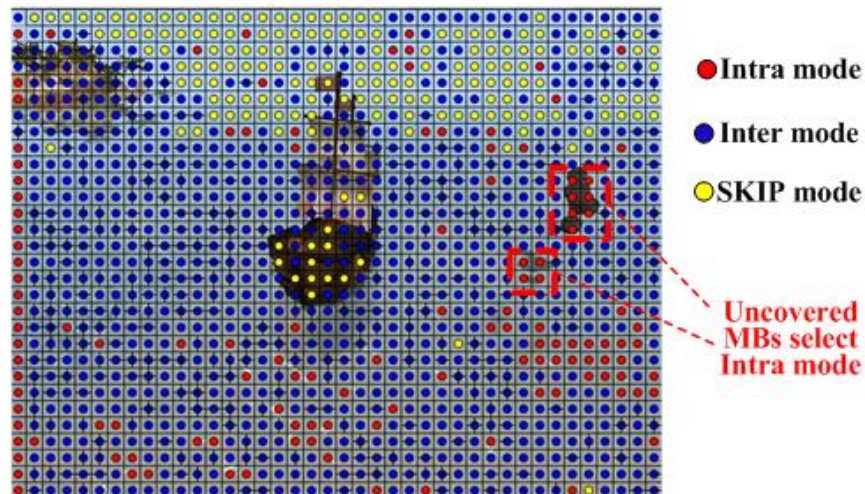


Figure 4.8. Game frame for BroadSides and the optimal MB mode selected with full mode RDO.

intuition that for the MBs that contain the uncovered pixels (namely uncovered MBs), we cannot find corresponding co-located MBs in the previous reference frame, therefore there would be large motion compensated residual if we encode these MBs using Inter-mode. Hence, we propose to directly bypass all the inter-modes for all the MBs that contain uncovered pixels.

Figure 4.8 illustrates an example that supports the idea of eliminating all inter-modes for uncovered MBs. Figure 4.8 shows the optimal modes for each MB selected by the regular H.264 encoder which performs RDO on all the possible modes (including intra-modes, inter-

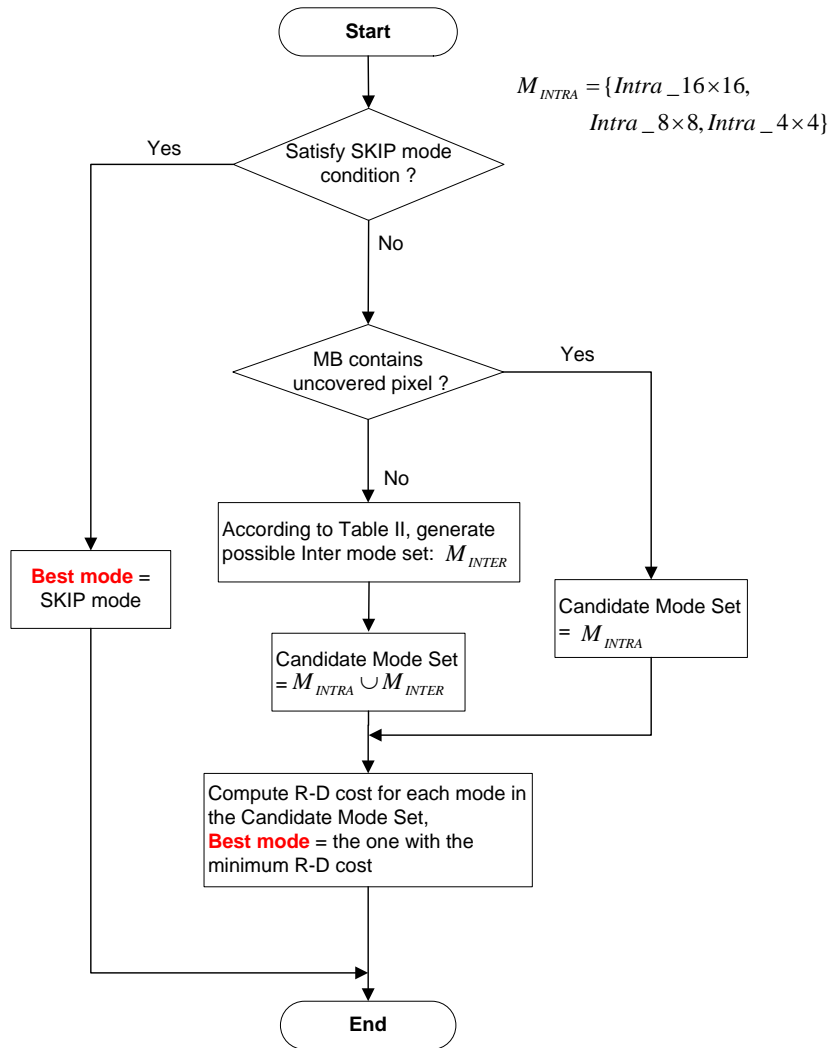


Figure 4.9. Flowchart of the proposed fast mode selection algorithm.



modes and SKIP mode). Note that the game frame shown in figure 4.8 is identical to the frame shown in figure 4.6(b). We can see that the uncovered MBs which contain pixels that did not appear in the reference frame (shown previously in figure 4.6(c)), are mostly encoded using intra-mode.

As a summary, the overall flowchart of the fast mode selection algorithm is shown in figure 4.9. To encode a 16x16 MB, we first test whether it satisfies the SKIP condition. If yes, we just encode it with the SKIP mode. Otherwise, we then test if this MB contains uncovered pixels. If yes, then we claim that encoding this MB using inter-mode is not suitable and we only consider Intra-mode encoding. If the MB does not consider uncovered pixels, we firstly generate candidate inter-mode set according to Table 4.2 and then consider both inter-mode and intra-mode as possible modes for R-D Cost calculation. Finally we calculate the R-D cost of each possible inter- or intra-mode, and select the one with the minimum R-D cost as the optimal mode.

Note that the proposed algorithm is only used for MB that belongs to P slice. For those MBs that belongs to I slice, it must be encoded using intra-mode, and we just use the regular H.264/AVC encoding process to maintain high encoding quality for I slices. Furthermore, we assume the B frame will be disabled since cloud gaming requires real-time video encoding and has strict requirement about encoding delay, and hence the typical encoding GOP structure is IPPP.

### **4.4.3 Assessment Experiment**

In this section, we present the results of two rounds of assessment experiments, where we apply the proposed rendering-based encoding acceleration (REA) technique on a variety of game sequences. The first round of experiments is for studying the performance (speed and quality) of the proposed direct motion vector calculation method (explained in section 4.4.1)

and the second round of experiments is for studying the performance of the entire REA technique (including the direct motion vector calculation method and the fast mode selection algorithm).

We have captured 6 representative game videos as the source for the test videos: 4 sequences for game PlaneShift and 2 sequences for game Broadside. The video sequence name and descriptions are summarized as Table 4.3. These 6 videos are selected such that our experiment has covered different game scene types and different motion properties.

To encode these video sequences, the following encoding setting is applied: GOP structure is IPPP, GOP size is set to be 30, video frame rate is 30 fps, video resolution is 800x600, reference frame number is set to be 1 (for low encoding delay), rate-distortion optimization and CABAC is used in Main profile, motion vector resolution is  $\frac{1}{4}$  pel. A group of QP values from 24 to 36 is used to encode. We use the open source X264 encoder software [X264] as the implementation of the H.264/AVC standard. The version number of the X264 software was R2389. The cloud game server and the encoder is executed on a PC with Intel i7 2.7GHz CPU and 8 GB RAM.

Table 4.3. Description of experiment videos.

Video Sequence	Description
POH	Game PlaneShift, outdoor scene, high motion
POL	Game PlaneShift, outdoor scene, low motion
PIH	Game PlaneShift, indoor scene, high motion
PIL	Game PlaneShift, indoor scene, low motion
BH	Game Broadside, high motion
BL	Game Broadside, low motion

### First Round of Experiment

Firstly, we conducted an experiment to compare the computation time of our proposed direct motion vector calculation method with full search method and two classic fast ME methods: Diamond search [ZM00] and UMHexagon search [ZLC04]. To compare these three ME methods with our proposed method, we configure the x264 encoder to use full mode selection, encode the video sequences listed in Table 4.3, and record the time spent on motion estimation. Note that the other three ME methods (full search, Diamond and UMHexagon) are included in the X264 source code, so we do not need to implement them by ourselves.

From the results of ME time shown in figure 4.10, we can see that full search method takes the most time, Diamond search takes least time, and our proposed method takes slightly longer than Diamond, but much less time than UMHexagon. The results tell us that although our proposed motion vector calculation method performs pixel-level calculation which seems to be computationally intensive, its actual complexity is similar to the Diamond search. The analytical explanation is shown in the supplementary material [SUP2].

Secondly, in order to evaluate the quality degradation caused by the proposed direct motion vector calculation method (not by the fast mode selection), we conduct the second experiment, which is outlined as following:

1) We first encode the six game videos listed in Table 4.3 of the revised manuscript using a regular x264 encoder, with full-search ME and full model selection. The MV resolution is  $\frac{1}{4}$ -pel, MV search range is 16 pels.

2) We then encode the same videos again with a customized x264 encoder, in which we have modified the source code to use the motion vectors calculated with our approach to substitute the motion vectors derived with normal ME method. During this second encoding

process, the customized encoder still uses full mode selection in order to make sure the mode selection method is the same with the first encoding process.

Table 4.4 listed the average PSNR difference between the two encoders for each test video sequence over a group of QP values from 24 to 36. The negative values in Table 4.4 indicate that our proposed ME approach achieves lower PSNR than regular ME method. But as we can see, among the six test videos, the PSNR degradation varies between 0.065 dB to 0.225 dB. The small PSNR difference values demonstrate that the quality degradation of our proposed ME method is very slight compared with normal ME method.

As a summary, from figure 4.10 and Table 4.3 we can conclude that: 1) our proposed motion vector calculation method has a similar computational complexity with other fast ME methods such as Diamond search; 2) the proposed method has very limited quality degradation compared with normal ME (full-search). However, the advantage of our proposed method is obvious: with the same complexity, it can produce pixel-level motion vectors,

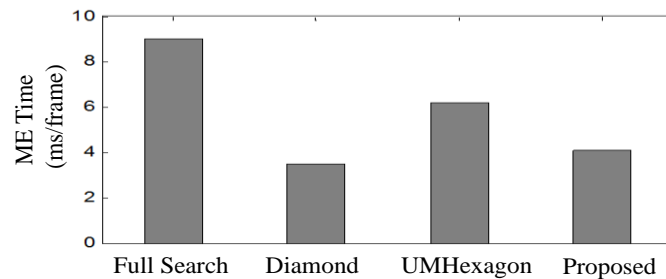


Figure 4.10. ME time per frame using different ME approaches.

Table 4.4. Average PSNR difference for test videos.

Video Sequence	POH	POL	PIH	PIL	BH	BL
$\Delta PSNR$ (dB)	-0.197	-0.078	-0.225	-0.086	-0.114	-0.065

which provides more information than the regular block-level motion vectors, and can benefit the subsequent mode selection step.

### Second Round of Experiment

In the second round of experiment, we study the performance (encoding speed and video quality) of the entire REA technique, including both the motion vector calculation method and the fast mode selection algorithm.

We first use the regular X264 encoder with UMHexagon ME and full mode selection as the reference encoder. We then implement three fast encoding techniques, including our proposed rendering-based encoding acceleration (REA) technique, Fechteler's technique [FE10], and Liu's technique [LSZ09], respectively, for performance comparison.

Our proposed REA technique is different from Fechteler's and Liu's technique in the following aspects: (1) Fechteler's technique uses rendering information to compute motion vectors to bypass the ME process, but it does not optimize the mode selection process; (2) Liu's technique is a fast mode selection technique which utilizes the motion homogeneity to eliminate some unlikely block modes during the RDO process, it can optimize the mode

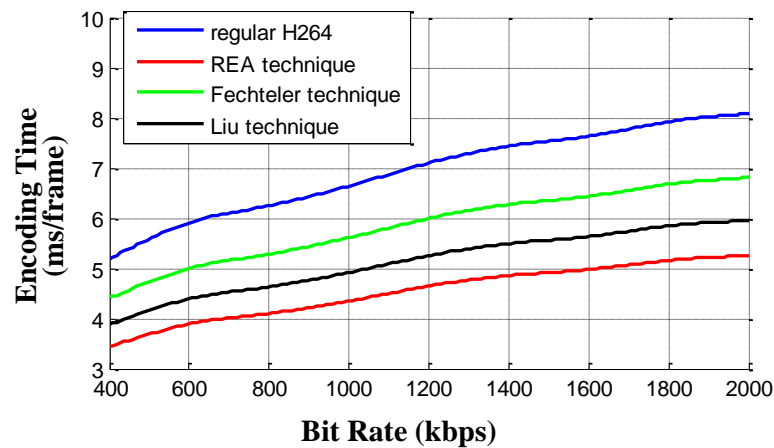


Figure 4.11. Relation between encoding time per frame and bit rate for video sequence BOH.

selection process but it requires to perform block-level ME for every 4x4 block; (3) our technique firstly utilizes rendering information to bypass regular ME, and subsequently uses motion vectors' properties to optimize the mode selection process as well.

Figure 4.11 and 4.12 show an example of the experiment results using video sequence BOH. Figure 4.11 shows the relation between encoding time per frame and the video bit rate. Figure 4.12 shows the encoding efficiency: the relation between PSNR and bit rate. We can see from figure 4.11 and 4.12 that our proposed approach can achieve much lower encoding time compared with the other three approaches, without noticeable video quality degradation (decrease in PSNR). Comparing our proposed technique with Fechteler's technique, we can see that our technique leads to a lower computation time (shown in figure 4.11) while keeping a higher encoding efficiency (shown in figure 4.12). This is because our proposed fast mode selection algorithm reduces the possible tested modes for the rate-distortion optimization process.

Furthermore, if we compare our proposed REA technique with Liu's fast mode

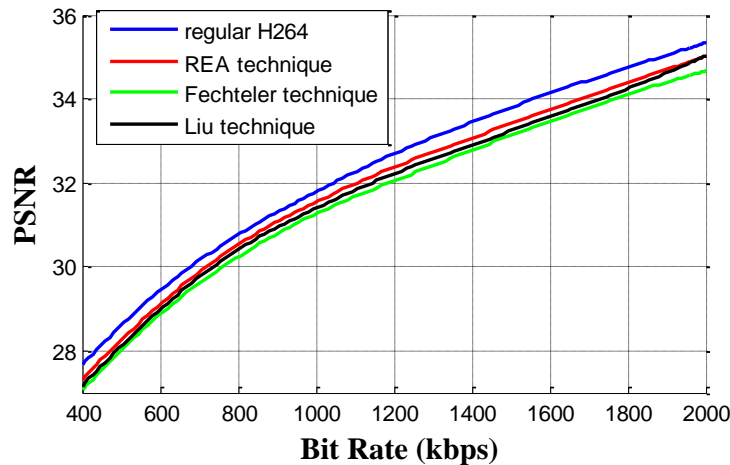


Figure 4.12. Relation between PSNR and bit rate for video sequence BOH.

selection technique, we find that our technique can also achieve a faster encoding speed while maintaining a slightly higher video quality. This is because of two reasons: 1) we have proposed in our technique that for those uncovered MBs (defined in section IV.A), we will disable inter-prediction mode because for these blocks there is no corresponding blocks in the reference frame and inter-prediction might lead to large motion compensated residual. Liu's method [LSZ09], on the other hand, does not have this mechanism because in Liu's method the encoder does not have any information about which blocks belongs to which objects and which objects are new emerging objects; 2) to encode a 16x16 MB, in our proposed technique the mode selection heuristic is based on 256 pixel-level motion vectors while in Liu's technique it is based on 16 block-level motion vectors (each one corresponds to a 4x4 block). Given that the computation complexity for generating the motion vectors are approximately the same for our technique and Liu's technique, our technique provides fine-grained motion vector information and hence may achieve a higher accuracy in mode selection and therefore higher encoding efficiency.

Table 4.5 shows the experiment results from a different perspective; it lists the results of all the 6 video sequences (for each video sequence, the average results over all the QP values is shown). To evaluate the average encoding performance (efficiency and complexity) for each video, we calculate Ejøntegaard delta peak signal-to-noise ratio (BDPSNR) specified in [BJO01] between each fast encoding technique (ours, Fechteler's and Liu's) and the reference encoder (the regular X264 encoder). BDPSNR is a metric used to measure the average PSNR difference between two rate-distortion (RD) curves. Compared with the reference encoder, the average encoding time saving in percentage is defined as:

$$TS = \frac{T_r - T_f}{T_r} \times 100\% \quad (4.32).$$

where  $T_r$  and  $T_f$  indicate the encoding time per frame for reference encoder and fast encoder.

It can be seen from Table 4.5 that our proposed fast encoding technique can achieve on the average 42.6% reduction in encoding time compared with the regular X264 encoder over all the 6 game video sequences, at the cost of a PSNR degradation of only 0.388 dB on average. We can see that the proposed fast encoding algorithm achieves a consistent gain in encoding time savings for all sequences with the least gain of 33.3% in video sequence BH and the greatest gain of 52.0% in video sequence PIL. Compared with Fechteler's algorithm, our proposed technique is faster and can reduce an extra 20.3% encoding time, and the PSNR degradation of our technique is 0.229 dB smaller than that of Fechteler's algorithm. Compared with Liu's algorithm, our proposed REA technique can reduce an extra 9.5% encoding time. Further, the PSNR degradation of our technique is 0.027 dB smaller than that of Liu's technique.

In Table 4.5 we have listed another metric called Coverage, which is defined as the ratio of the number of MBs whose optimal encoding modes are covered by our proposed fast mode selection algorithm (figure 4.9) to the total number of MBs. We can see that our proposed fast mode selection algorithm can make the optimal choice in 88.9% of all the MBs. This high accuracy of mode selection is the main reason that our proposed algorithm only has

Table 4.5. Average performance comparison of the three fast encoding techniques.

Video Sequence	Propose REA technique			Fechteler's technique		Liu's technique	
	BDPSNR	TS	Coverage	BDPSNR	TS	BDPSNR	TS
POH	-0.439	38.2%	83.2%	-0.635	22.8%	-0.465	31.6%
POL	-0.353	45.4%	90.7%	-0.603	22.1%	-0.357	33.0%
PIH	-0.382	36.9%	85.3%	-0.669	22.5%	-0.440	33.5%
PIL	-0.365	52.0%	91.9%	-0.595	23.5%	-0.378	35.2%
BH	-0.414	33.3%	88.7%	-0.604	20.8%	-0.434	31.8%
BL	-0.375	49.5%	93.7%	-0.596	23.8%	-0.415	33.6%
Ave.	-0.388	42.6%	88.9%	-0.617	22.3%	-0.415	33.1%



a 0.388 dB PSNR degradation compared with the regular H.264/AVC encoder with full-search RDO.

Although the proposed REA technique can significantly reduce the encoding time, there exist one limitation: the encoding efficiency may drop if the game has complex lighting and rendering effect such as shadowing and reflection, since in these scenarios, for the same pixel, its value (colors and luminance) may not be the same in consecutive frames. If we use the collocated block in the previous frame as the reference block for inter-prediction, the resulted motion compensated residual might be large, leading to lowering of the encoding efficiency.

## **4.5 Conclusion**

In this chapter, we propose two rendering based techniques which utilize rendering information to enhance video encoding for cloud gaming application to achieve: 1) better video perceptual quality under a fixed bit rate budget; 2) lower encoding complexity and encoding time for every video frame.

The rendering-based prioritized encoding technique exploits rendering information including depth map and scene composition information to prioritize different regions of a video frame, and adapts the QP values for each MB according to their importance and the bit rate budget. Assessment experiments have been carried out to validate the effectiveness of this prioritized encoding technique. The experimental results show that this technique can significantly increase the player's perceptual video quality under a given bit rate budget.

The proposed rendering-based encoding acceleration technique uses rendering information to reduce the computational complexity of the encoding process of H.264/AVC technique by: 1) directly calculating the motion vectors and eliminating the need for regular

search-based motion estimation; 2) using rendering information to reduce the number of candidate encoding modes for ROD process. Experimental results show that the proposed rendering-based encoding acceleration technique can achieve a significant and consistent encoding time reduction compared with regular H.264/AVC standard over a variety of test videos.

The two techniques proposed in this chapter have utilized rendering information that can be very conveniently obtained, such as the depth map and object ID. In the future, in addition to using the above rendering information, we plan to investigate the use of pixel-domain features of the rendered video frame, such as luminance and chrome values of every pixel, to increase the perceptual video quality as well as accelerate video encoding. Furthermore, these pixel-domain features can be used to detect the object boundaries or to estimate the spatial complexity of a MB, and hence can help in optimizing the mode selection algorithm by eliminating some unlikely intra-modes. These are interesting topics which we plan to investigate in the future for further enhance the quality and efficiency of video encoding of CMR application.

## **Acknowledgments**

The text of this chapter, in part or in full, is based on material that has been published in IEEE Transaction on Circuits and Systems of Video Technology (TCSVT) (Y. Liu, S. Dey, Y. Lu, “Enhancing Video Encoding for Cloud Gamin Using Rendering Information,” IEEE TCSVT, Jun. 2015). The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

# Chapter 5

## Deriving and Validation User Experience Model for DASH Video Streaming

Ever since video compression and streaming techniques have been introduced, measurement of perceived video quality has been a non-trivial task. DASH, Dynamic Adaptive Streaming over HTTP, is a new worldwide standard for adaptive streaming of video. DASH has introduced an additional level of complexity for measuring perceived video quality, as it varies the video bit rate and quality. In this paper, we study the perceived video quality using DASH. We investigate three factors which impact user perceived video quality: initial delay, stall (frame freezing), and bit rate (frame quality) fluctuation. For each factor, we explore multiple dimensions that can have different effects on perceived quality. For example, in the case of the factor stall, while most previous research have studied how stall duration correlates with user experience, we also consider how the stalls are distributed together with the amount of motion in the video content, since we believe they may also impact user

perceived quality. We conduct extensive subjective tests in which a group of subjects provide subjective evaluation while watching DASH videos with one or more artifacts occurring. Based on the subjective tests, we first derive impairment functions which can quantitatively measure the impairment of each factor, and then combine these impairment functions together to formulate an overall user experience model for any DASH video. We validate with high accuracy the user experience model, and demonstrate its applicability to long videos.

## 5.1 Introduction

The wide adoption of more capable mobile devices such as smart-phones and tablets, together with the deployment of higher capacity mobile networks and more efficient video compression techniques, are making mobile video consumption very popular. According to Cisco's mobile traffic forecast [CIS13], mobile video consumption will increase 14-fold between 2013 and 2018, accounting for 69 percent of total mobile data traffic by the end of 2018. However, the success of mobile video streaming will largely depend on meeting user experience expectations. Therefore, it is highly desirable for video streaming service providers to be able to define, measure and, if possible, ensure mobile video streaming user experience. Recently, a new class of video transport techniques has been introduced for transmission of video over varying channels such as wireless network. These transport techniques, called adaptive streaming, vary the bit rate and quality of the transmitted video to match the available channel bandwidth and alleviate the problems caused by network congestion, such as large latency and high packet loss rate. DASH, Dynamic Adaptive Streaming over HTTP, is a new international standard for adaptive streaming [ISO10], which enables delivering media content from conventional HTTP web servers. DASH works by splitting the media content into a sequence of small segments, encoding each segment into several versions with different bit

rates and quality, and streaming the segments according to the requests from streaming client. On the client device side, the DASH client will keep monitoring the network and dynamically select the suitable version for the next segment that need to be downloaded, depending on the current network conditions.

On the DASH server side, each media segment is made available at a variety of bit rates. Each bit rate will be associated with a set of other encoding factors such as frame rate and resolution. Different streaming service providers might use different encoding options for a given bit rate. As an example, Table 5.1 shows the bit rate options and the associated frame rates and resolutions that were used for streaming the Vancouver Olympics [OZE11] videos using DASH. In this paper, we use the term level to represent a bit rate and the associated frame rate and resolution. As shown in Table 5.1, the video segments are encoded using any of the 8 levels; each of them has a specific bit rate, frame rate, and resolution.

It is well known that DASH video streaming is based on HTTP (Hypertext Transfer Protocol) and TCP (Transmission Control Protocol) which assure reliable video packets delivery and retransmission of lost packets. Using TCP retransmission and buffering mechanism can avoid audiovisual distortions caused by network artifacts such as jitter or

Table 5.1. Encoding settings for streaming Vancouver Olympics.

Level	Bit Rate (kbps)	Resolution	Frame Rate
1	400	312 x 176	15
2	600	400 x 224	15
3	900	512 x 288	15
4	950	544 x 304	15
5	1250	640 x 360	25
6	1600	736 x 416	25
7	1950	848 x 480	25
8	3450	1280x720	30

packet loss. Instead, these network artifacts would lead to rebuffering interruptions and additional initial delay, which would deform the video's temporal structure and impact user experience. Furthermore, unlike regular TCP-based video streaming, DASH has introduced an additional level of difficulty for measuring video quality, since it varies the video quality during streaming. Although the video quality adaptation scheme can mitigate the temporal impairments such as rebuffering, the quality variation during streaming may also impact the user experience of the viewers.

Therefore, the aim of this paper is to derive a model to quantitatively measure the user experience of a video streamed using DASH, considering both temporal artifacts (like rebuffering) and spatial artifacts (like video quality variation). We first identify three factors that will impact the user experience: initial delay, stall and level variation. We show that each of these factors have multiple dimensions which may impact user experience differently. We design and conduct subjective experiments by which viewers evaluate the effect on viewing experience when one or more of the three factors are varied. Based on the evaluations given by the participants of the subjective experiments, we derive impairment functions for each of the factors, and then combine them together to form an overall user experience model. Note the proposed user experience model is a non-reference model, and no access is needed for the original video source. Hence, the proposed user experience model can be conveniently incorporated into DASH clients on mobile devices to measure the impairments during a live video session.

The impairment of the three factors on user experience may vary depending on the video content, such as the amount of motion, or the duration of the video. For instance, the impact of stalls on user experience may be higher for a high motion video and less for a low motion video. Similarly, the impact of initial delay may be higher for a video with short duration and less for a video with long duration. Our proposed user experience model considers the amount of motion in the video content, and as we later demonstrate, can be applied to short to medium length videos covering most of online videos. Note that the impairment on user experience may also depend on other factors like how much a user likes the video or the type of mobile device (screen size/resolution) used. Our research and the proposed model do not consider these factors, which can be possibly investigated further in our future work.

Numerous video quality assessment methods have been proposed over the past years. Most of them [HR10][RNR08][LVG09][KS12][CER09][MSS10][SSB10] focus on measuring the video spatial quality (visual quality of video frame) and ignore the temporal artifacts such as stalls. In [MC11][SHR12][SSB10][MCC11], models have been proposed to study the video temporal quality, but they don't include the variation of bit rate (visual quality) during the streaming session, and are therefore not suitable for DASH video. In [NEE11][NEE12], the authors have studied the impact of bit rate variation on user experience. While they derive interesting observations about how variation frequency and amplitude affect user experience, they do not develop ways to quantitatively measure the effects. Moreover, they do not consider temporal artifacts such as stall. To the best of our knowledge, this paper is the first study to develop a quantifiable measure of user experience for DASH video, considering both spatial and temporal quality.

The remainder of the chapter is organized as following: in section 5.2, we introduce the factors that will affect user experience of DASH video. In section 5.3, we first explain the characterization experiments we conducted to study how DASH performs in various mobile network conditions, and then we explain how we use the characterization experiments as a guideline to generate the test videos for subjective experiments. In Section 5.4, we describe the first round of subjective experiments, and derive impairment functions for the different factors based on experiment results. Section 5.5 describes a second round of subjective tests and the derivation of the overall user experience model. Section 5.6 demonstrates application of the proposed model to long videos. Section 5.7 concludes this chapter and points out future work.

## **5.2 Factors Affecting User Experience for DASH Video**

The first step to study and model user perceived video quality is to identify the impairment factors which impact user experience. In this section, we propose and explain three impairment factors that will affect the user experience for DASH video.

During a DASH video watching session, video will be transmitted over wireless network, which is characterized by quickly fluctuating and unpredictable bandwidth. In this streaming process, there are mainly three kinds of events which may affect the user perceived video quality: 1) there is an initial delay before the first frame of the video can be displayed, due to the need for the video client to buffer a certain amount of video data; 2) during a video session, it is possible that the bit rate adaptation cannot keep up with the network bandwidth fluctuation, leading to buffer underflow and stalling (rebuffering); 3) during a video session, the video quality might keep switching, reducing the video quality will cause impairment to user experience, and continuous video quality switches will also harm user experience.



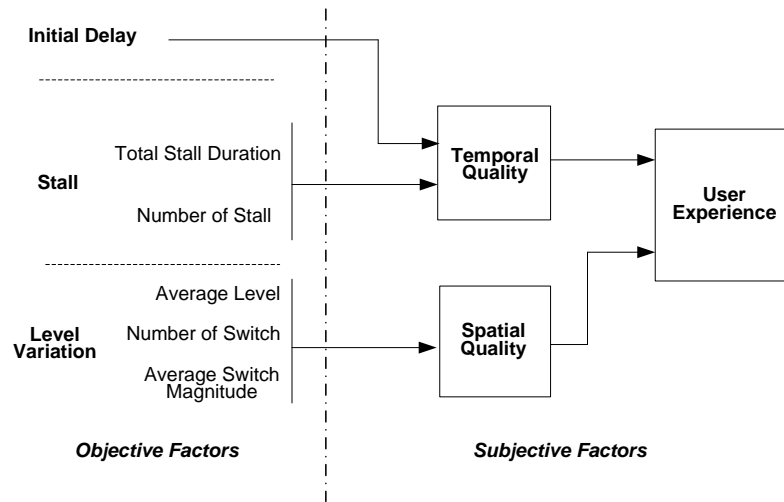
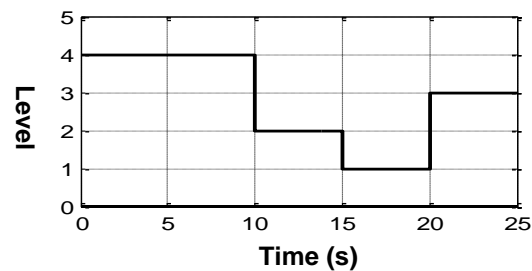


Figure 5.1. Factors affecting user experience of DASH video.



$$\text{Average Level} = (4+4+2+1+3)/5 = 2.8$$

$$\text{Number of Switch} = 3$$

$$\text{Average Switch Magnitude} = [(4-2)+(2-1)+(3-1)]/3 = 1.67$$

Figure 5.2. Level variation pattern.

As shown in figure 5.1, we investigate three objective factors: *initial delay*, *stall* (rebuffering) and *level variation*. The user experience for DASH video mainly depends on two subjective factors: temporal quality and spatial quality. The *initial delay* and *stall* will determine the temporal quality of the perceived video, and the *level variation* will determine the spatial quality of video.

Unlike *initial delay*, the factors *stall* and *level variation* are more complex and have multiple dimensions associated with them. For the *stall* factor, the *total stall duration* (in

seconds) is crucial. Most of the previous research only studied how stall duration correlates with user perceived quality. However, we think the number of stalls is also an important dimension. For example, consider total stall duration of 5 seconds: the effect on user experience may be different if there is a single stall of 5 seconds duration, versus five 1-second stalls. Hence, besides the total stall duration, we would like to also consider the number of the stalls as a second dimension of the factor *stall*.

Similarly we propose three dimensions for factor *level variation*: 1) *average level*, which indicates the average quality of the entire video session; 2) *number of switches*, which indicates the frequency of quality switch; 3) *average switch magnitude*, which indicates the average amplitude of quality change. For instance, for a level variation pattern as shown in figure 5.2, the average level is 2.8, number of switch is 3, and the average switch magnitude equals 1.67. Noted that in figure 5.2 we haven't differentiated increasing level switch (when the bit rate increases) and decreasing level switch (when the bit rate decreases). But as explained in section 5.4, we will decide whether to treat increasing switch and decreasing switch differently based on the results we obtain from subjective tests.

### 5.3 Test Video Generation

In order to derive functions to quantitatively measure the impairment of the 3 factors proposed in section 5.2, we need to conduct extensive subjective tests, where each participant watches DASH video while one of the three factors varies. However, due to the multi-dimensional nature of the factors stall and level variation, there may be numerous cases we need to cover in the test videos. On the other hand, we need to constraint the number of test videos a subject can watch before loss of focus and fatigue can affect the quality of the testing. Motivated by this tradeoff, we designed test videos in an efficient way such that they cover a

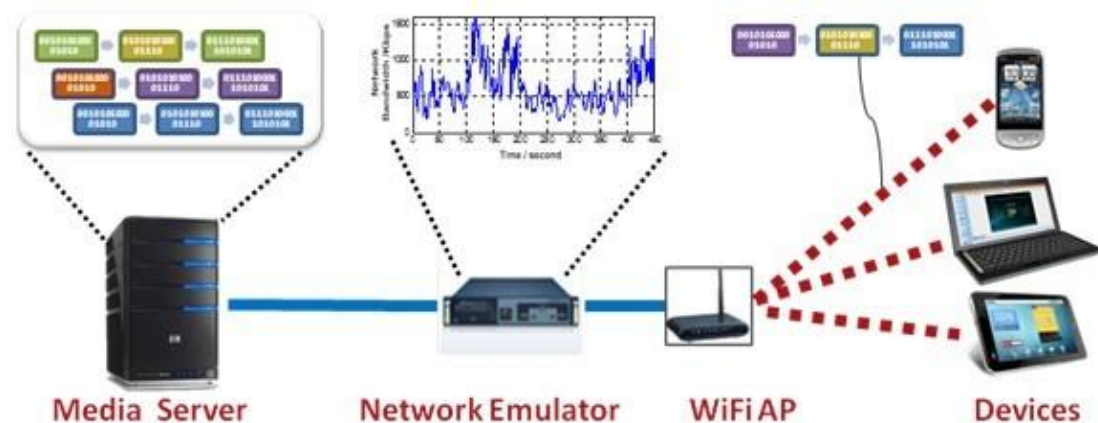


Figure 5.3. Testbed of DASH video streaming characterization experiments.

wide and representative range of the 3 factors, and we are able to derive impairment functions from a limited number of test videos. In this section, we describe how we generate the test videos for the subjective tests.

### 5.3.1 DASH Video Streaming Characterization Experiments

In order to generate meaningful and representative test videos, we first conduct a set of DASH video streaming experiments to characterize how DASH performs under real mobile network conditions. From the streaming experiments, we can understand what will be the possible range and distribution for the 3 factors under various network conditions. This range and distribution information will be used as a guideline to generate the test videos.

Figure 5.3 shows the testbed for the DASH characterization experiments. DASH videos are pre-encoded and stored at the media server. The media server and the mobile devices are connected through a network emulator, which can be used to control network parameters, such as bandwidth, latency and packet loss rate. On the network emulator, we can apply different mobile network bandwidth traces. At the mobile device side, a DASH player displays the received video and makes video level switch decisions. After each video

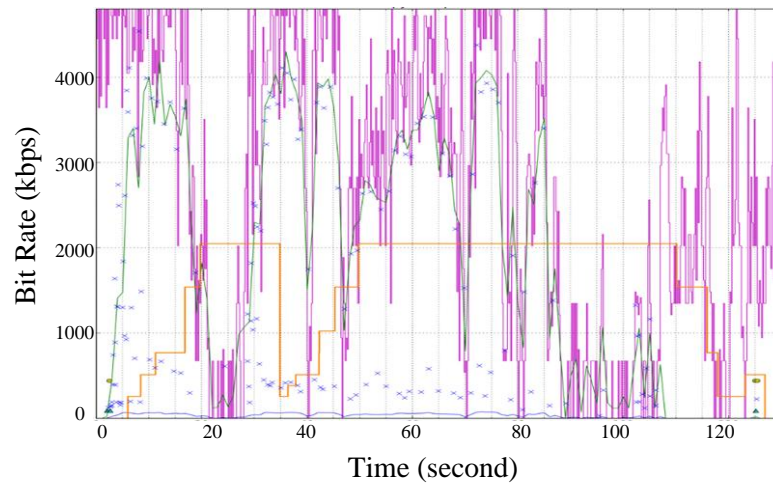


Figure 5.4. Results for characterization experiments: (1) purple curve: network bandwidth; (2) green curve: segments download bit rate; (3) yellow curve: video bit rate.

streaming session, a log file is generated on the mobile device, including information about the 3 factors for this streaming session. For instance, this log file will tell during the streaming session, what the level variation pattern is, how many stalls occurred and when they occurred.

This testbed offers the flexibility for us to stream under different network conditions, and records the values of the 3 factors. In the characterization experiments, we stream a DASH video to an Android tablet, under 20 different mobile network conditions. This selected DASH video is 2-minute long and has medium amount of motion. It is split into 2-second segments and pre-encoded into 7 levels (with encoding bit rate of 256, 384, 512, 768, 1024, 1536, 2048 kbps, respectively). We use 20 mobile network traces that are wide-ranging and representative, captured with different mobile operator networks at different geographical locations, and include stationary as well as different mobility scenarios, such as pedestrian, car, train, etc. The bandwidth of the network traces varies between 4Mbps and 150kbps. Among the 20 network traces, the average bandwidth of each trace varies between 750kbps ~ 1850 kbps.

Figure 5.4 shows a representative result of the characterization experiments. The purple curve represents available mobile network bandwidth, the green curve shows the video segments downloading rate, and the yellow curve shows the actual adaptive video bit rate. We can see that the DASH adaptive bit rate (yellow curve) will switch up and down between several discrete steps due to the fast fluctuation of mobile network bandwidth (purple curve).

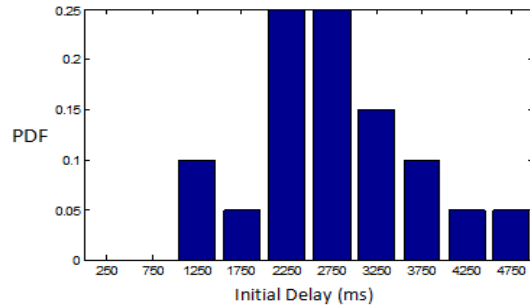


Figure 5.5. Distribution of initial delay among 20 streaming sessions.

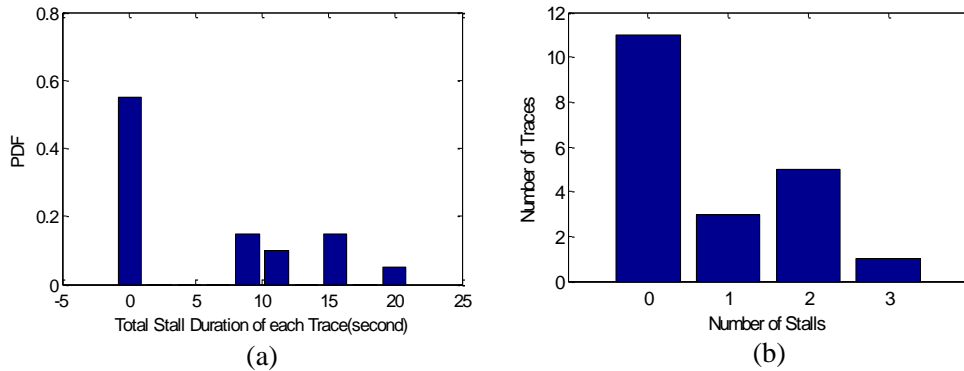


Figure 5.6. (a) distribution of total stall duration; (b) distribution of number of stalls.

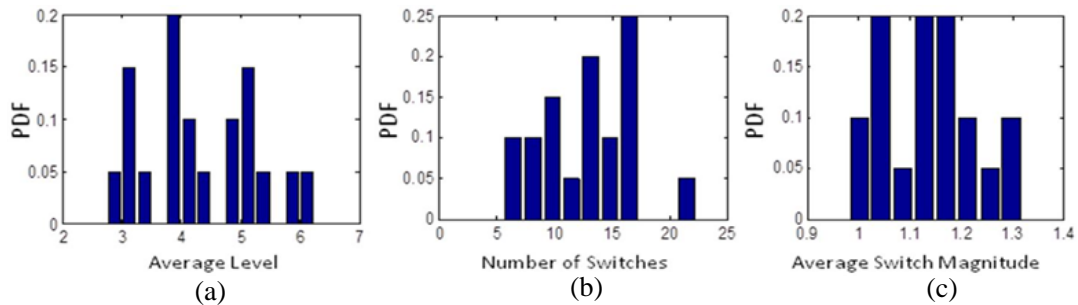


Figure 5.7. Distribution of level variation: (a) average level; (b) number of switches; (c) average switch magnitude.

Figure 5.5 shows the distribution of initial delay among the 20 streaming sessions, each using one of the 20 different network traces. The 20 initial delay values are between 1280ms and 4890ms. Figure 5.6(a) and 5.6(b) show the distribution of total stall duration and stall number among the 20 streaming sessions (each of them is 2-minute long). We find that in 55% of the streaming sessions there is no stall happening. In the other sessions, the stall number is less than 3. And the stall duration of a video session can be as long as 20 seconds. Figure 5.7(a), (b) and (c) show the distribution of the average level, number of switches, and average switch magnitude respectively. We can see that during a 2-minute streaming session, the number of level switches can vary from 6 to 21. The average switch magnitude is between 1 and 1.3, which indicates that the current DASH technique mainly utilizes small magnitude switches to avoid impairment caused by large quality change.

### **5.3.2 Generated Test Cases for Round I Subjective Tests**

After presenting the ranges and distribution of the 3 factors, in this subsection we will use them as a guideline to generate the test videos for round I subjective tests. We may also include test videos whose characteristics are outside of what was observed in the DASH characterization tests to cover more extreme cases. For instance, although the initial delay values we obtain from all real experiments are less than 5 seconds (figure 5.5), we will also have test video with very long initial delay, like 15-second initial delay.

We design 40 test videos for subjective tests. Each of them is 1 minute long. In each test video, we only vary one factor and keep the other two factors at their best values. For instance, we have 5 test videos for deriving the impairment function for initial delay. In these 5 test videos, there is only initial delay impairment; no stall occurs and the video level remains at the highest value. When people watch these 5 videos and give evaluations, they are only evaluating the impairment caused by initial delay. By generating test videos in this manner, we can separate the 3 factors, and be able to derive impairment function for each of them. Figure 5.8 shows a snapshot of the video contents we use, and Table 5.2 lists their descriptions. As can be seen, the 6 videos contents are selected to cover different video genres (news, animation, movie, sports) and different motion characteristics.

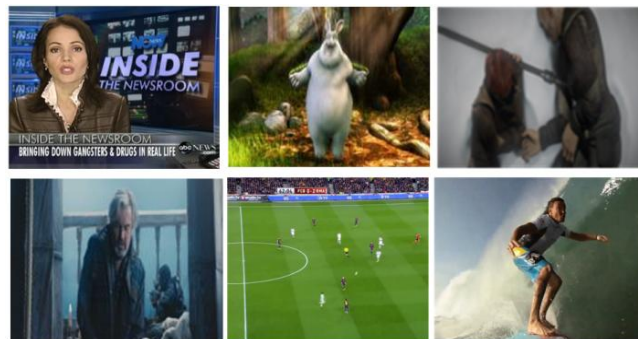


Figure 5.8. Snapshot of test videos.

Table 5.2. Video content description.

Video Name	Description
News	A woman reading news, low motion
Bunny	Cartoon about animals, medium motion
Sintel	Animated movie, medium motion
Steel	Human action movie , medium motion
Soccer	A soccer match, high motion
Surfing	Surfing, high motion, with multiple scene changes

In order to simulate one of the three impairments in each test video, in this study we use ffmpeg [FFM] software as the tool for video encoding and processing. The version of ffmpeg we used is 0.8.15 and the selected video codec is H.264/AVC. To simulate the initial delay, we insert some identical 'buffering' frames (shown in figure 5.9(a)) in front of the raw video frames and then encode all the frames. The duration of the initial delay is controlled by the number of 'buffering' frames inserted. Similarly, the stall impairment is simulated by inserting some identical 'loading' frames (shown in figure 5.9(b)) in the middle of raw video frames. As shown in figure 5.9(b), the inserted 'loading' frame is the last raw video frame with a watermark of word 'Loading'. Moreover, the level variation impairment is simulated by

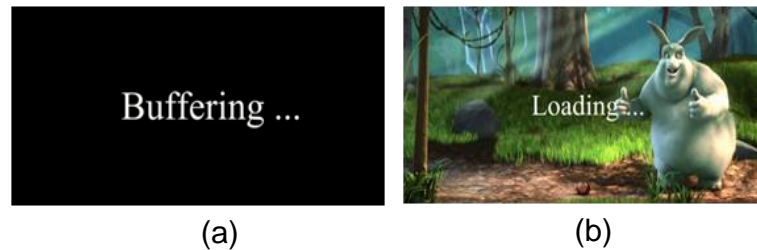


Figure 5.9. Special frames used to simulate initial delay and stall: (a) 'buffering' frame for simulating initial delay; (b) 'loading' frame for simulating stall.

Table 5.3. Test cases for initial delay.

Video ID	1	2	3	4	5
Initial Delay Value (s)	2	4	6	10	15

Table 5.4. Test cases for stall.

Video ID	6	7	8	9	10	11	12	13	14	15
Total Stall Duration	4 sec			8 sec				12 sec		
Stall Number	1	2	4	1	2	3	8	1	3	12



encoding different groups of raw video frames with different encoding parameters and concatenating all the encoded video streams together.

We use video #1~ #5 for deriving the impairment function of initial delay. As shown in Table 5.3, we investigate the initial delay between 2 to 15 seconds.

Videos #6~ #15 are used to investigate the impairment due to stall. As shown in Table 5.4, the total stall duration values we investigated are [4, 8, 12] seconds. Since we observed stall duration between 0 and 20 seconds for the 2-minute video sessions in the DASH characterization tests (figure 5.6(a)), we assume that considering stall durations between 4 and 12 seconds will be reasonable for the subjective tests conducted with videos of 1-minute duration. Similarly, in video 6~15, we consider the number of stalls of 1~3, which corresponds to the result shown in figure 5.6(b). We also want to study the extreme cases where there are a lot of very short stalls and the stall number is bigger than 3. Video #8, #12 and #15 are videos with lots of 1-second stalls, and we want to understand how people feel with these frequent short stalls.

Videos #16~#40 are designed for deriving impairment function of level variation

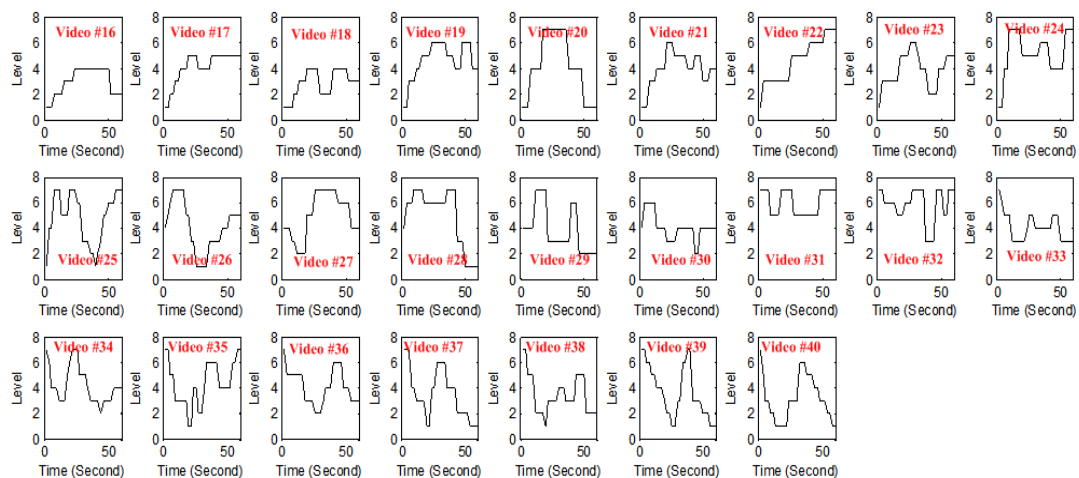


Figure 5.10. Test videos for level variation factor.

factor. Figure 5.10 shows the level variation pattern of these 25 test videos. These 25 level variation patterns are designed to guarantee that: 1) the experiment results (range and distribution) shown in figure 8 are met; 2) include plenty of different switch frequencies and magnitudes, both increasing switches and decreasing switches; 3) include different video starting levels and ending levels.

## 5.4 Derivation of Impairment Functions

After generating these 40 test cases, in this section, we will describe the first round of subjective tests, and then derive the impairment functions for the 3 factors according to the test results.

### 5.4.1 Round I Subjective Experiments

The subjective quality assessment experiments follow ITU-T Recommendations [P800]. Each test video is presented one at a time, and each subject gives individual evaluation about the perceived video quality with a 100 point quality scale, as shown in Table 5.5. As the subjects are evaluating the perceived video quality, denoted as  $R$ , the corresponding

Table 5.5. Rating criteria for video quality.

Quality Evaluation	Description
100	Excellent experience, no impairment at all
80-100	Minor impairment, will not quit
60-80	Noticeable impairment, might quit
40-60	Clearly impairment, usually quit
0-40	Annoying experience, definitely quit.

impairment will be  $100-R$ . The experiment is conducted in a lab environment with good light condition. A Qualcomm MSM8960 tablet with  $1280 \times 768$  display resolution is used to watch the test videos.

30 subjects from UCSD, with age ranging from 18 to 28, were selected for the study, satisfying the requirement of number of viewers specified by ITU-T Recommendations [P800]. To ensure their evaluations are not biased, the selection of the subjects is done so that they don't have prior knowledge or watching experience of DASH video. Each subject is first presented with a training sequence which is different from the test videos to help him/her get familiar with the experiment environment and adjust his/her comfortable viewing distance and angle.

The evaluations for each test video  $i$  are averaged over all subjects to obtain an average video quality value, denoted by  $R_i$ . Correspondingly, the average impairment value of video  $i$  will be  $100 - R_i$ . In the next subsection, we will use these average impairment values to derive impairment functions for all the 3 factors.

#### 5.4.2 Impairment Function for Initial Delay

In test videos #1~ #5, we add different length of initial delay in the beginning of the video. The relation between the initial delay value and the average subjective impairment values is shown in figure 5.11. We can see that the average subjective impairment of the 30 subjects is almost linear with the initial delay. Therefore, the impairment function for initial delay can be formulated as the following linear equation:

$$I_{ID} = \min\{\alpha * L_{ID}, 100\} \quad (5.1).$$

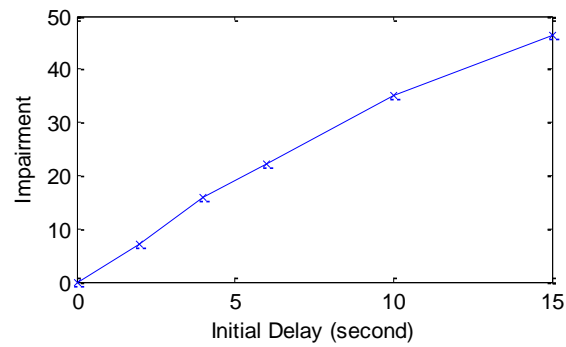


Figure 5.11. Relationship between impairment and initial delay.

where  $I_{ID}$  stands for the impairment due to initial delay,  $L_{ID}$  is the length of initial delay (in seconds). The coefficient is computed by linear regression and is listed in Table 5.6. We have also used a min function to limit the impairment when it reaches its maximum.

### 5.4.3 Impairment Function for Stall

We have prepared test videos with different combinations of stall duration and stall number, for different video content. Figure 5.12 shows an example of the stall impairment results of 5 different stall distributions on 6 different videos. We find that besides stall number and stall duration, video content, more specifically the amount of motion in the video, also

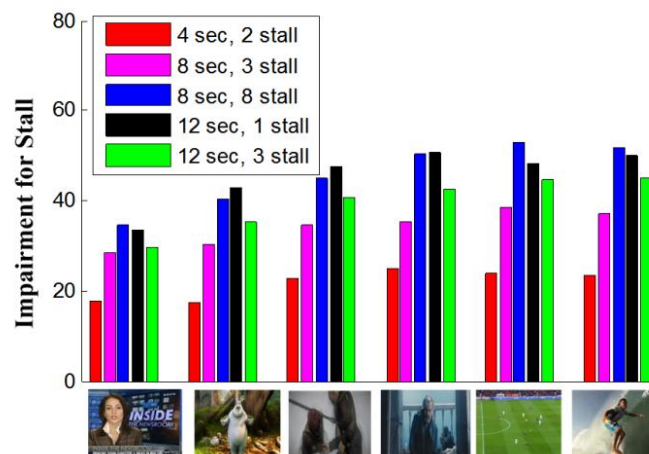


Figure 5.12. Subjective stall impairment results for different video contents, stall duration and stall number.

plays a crucial role in determining impairment value. We can see that for the same stall duration and stall number, high motion video has bigger impairment than low motion video. This may be due to the higher expectation/requirement of fluidness for high motion video content such as sports video.

Therefore in order to model the stall impairment, we first characterize the amount of motion in video. Then we will develop a function to model stall impairment.

### ***Video Motion Characterization***

We propose to use the average magnitude of motion vectors to characterize the amount of motion of a certain video content. In any regular video application, motion vector can be directly extracted from the encoded video bitstream without further computation. Furthermore, from the motion vectors in  $x$  and  $y$  direction, we compute the Motion Vector Magnitude (MVM) for each  $16 \times 16$  macroblock,  $MB_{ij}$ , which we define as:

$$MVM_{ij} = \sqrt{\left(\frac{m_{ij,x}}{N_x}\right)^2 + \left(\frac{m_{ij,y}}{N_y}\right)^2} \quad (5.2).$$

where  $N_x$ ,  $N_y$  are the number of  $16 \times 16$  MBs in the horizontal and vertical directions; and  $m_{ij,x}$ ,  $m_{ij,y}$  are the projection of motion vector on  $x$  and  $y$  directions for  $MB_{ij}$ . In equation (5.2) we have normalized the  $MVM$  by the width and height of video frame to get rid of the influence of video spatial resolution on the motion characteristic of video content.

Table 5.6. Values of coefficients.

$\alpha$	a	b	c	d	k	$B_1$	$B_2$
3.2	3.35	3.98	2.50	1800	0.02	73.6	1608

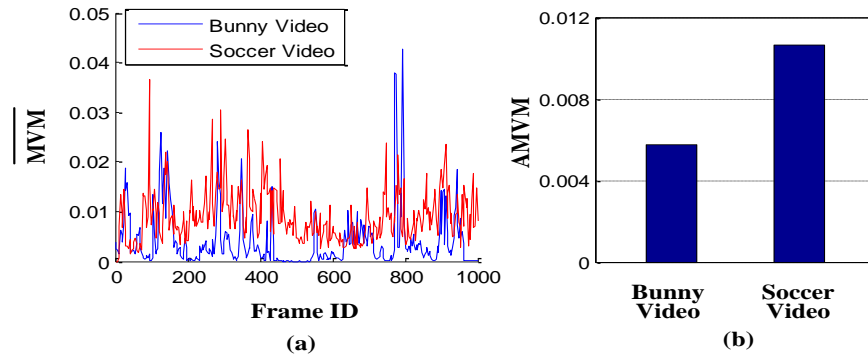


Figure 5.13. Motion of video content: (a)  $\overline{MVM}$  value of each frame; (b) AMVM for the whole video.

We then take the average of all the MBs to obtain the average MVM value of a video frame, denoted as  $\overline{MVM}$  :

$$\overline{MVM} = \frac{1}{N_x * N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} MVM_{ij} \quad (5.3).$$

Let us use  $\overline{MVM}_k$  to denote the average MVM value for the k-th frame, then the Average Motion Vector Magnitude (AMVM) of the whole video can be computed as:

$$AMVM = \frac{1}{M} \sum_{k=1}^M \overline{MVM}_k \quad (5.4).$$

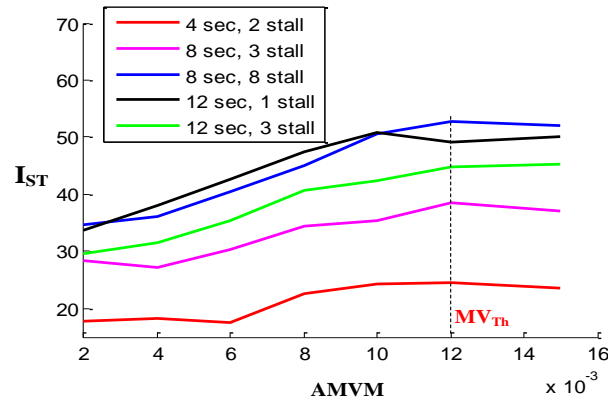
where  $M$  is the number of frames in a video.

We will use  $AMVM$  as the metric to characterize the amount of motion of a video. As an example, figure 5.13(a) shows the motion vector magnitude of video Bunny and Soccer frame by frame. Figure 5.13(b) shows the average motion vector magnitude ( $AMVM$ ) of the two videos. We can see that using  $AMVM$  we can clearly differentiate high motion video and medium motion video. Hence,  $AMVM$  is an effective and easy-to-obtain metric to characterize the amount of motion in video.

### ***Stall Impairment Function Derivation***

Table 5.7. Subjective experiment results for different stall duration and stall number for video Bunny.

Total Stall Duration	4 sec			8 sec				12 sec		
Stall Number	1	2	4	1	2	3	8	1	3	12
Impairment Value	16.5	21.8	31.3	31.1	27.3	33.3	47.5	40.8	37.5	58.5

Figure 5.14. Relationship between  $I_{ST}$  and video motion under different stall distributions.

After being able to quantify the amount of motion in a video, we then derive a stall impairment function,  $I_{ST}$ , based on the test results under different combinations of stall number, stall duration and  $AMVM$ .

First, we investigate for a given  $AMVM$  value, how stall number and stall duration affect the impairment due to stall ( $I_{ST}$ ). Table 5.7 shows the average impairment values for a certain video (we choose video Bunny as an example, it is a cartoon video with medium motion), where stall duration and stall number vary but  $AMVM$  is fixed. From the results listed in Table 5.7, we make the following observations:

**Observation (a):** When stall number is fixed, the impairment value increases monotonically with stall duration.

**Observation (b):** When stall duration is fixed, the impairment value does not increase monotonically with stall number. We also observe that the impairment value is highest with

the highest stall number, which indicates that frequent stalls will cause high impairment on user experience.

Secondly, we consider how motion information (AMVM) will affect  $I_{ST}$ . Figure 5.12 shows the stall impairment values with different video contents (from low motion news video to fast moving sports video). We can see that for a given video content, the observations (a) (b) still hold. Moreover, figure 5.14 shows the relation between stall impairment and the AMVM value. From figure 5.12 and 5.14, we have the following observation:

**Observation (c):** For the same stall duration and stall number, the impairment due to stalling will increase as the motion (AMVM) increases. But after AMVM reaches a certain threshold (when the motion level is high enough), the impairment will not further increase.

Observations (a), (b) and (c) tell us that we cannot use a linear equation to model the relationship between stall impairment with stall number, stall duration and AMVM. Therefore, we propose to use equation (5.5) as the impairment function for stall:

$$I_{ST} = \begin{cases} a * D_{ST} + b * N_{ST} - c * g(D_{ST}, N_{ST}) + d * AMVM & (\text{if } AMVM < MV_{Th}) \\ a * D_{ST} + b * N_{ST} - c * g(D_{ST}, N_{ST}) + d * MV_{Th} & (\text{if } AMVM \geq MV_{Th}) \end{cases} \quad (5.5).$$

In equation (5.5),  $I_{ST}$  stands for the impairment due to stall,  $D_{ST}$  indicates the total duration of stall,  $N_{ST}$  stands for the number of stall. Function  $g(D_{ST}, N_{ST})$  is used to compensate the simultaneous effects of stall duration and stall number and to match the phenomenon explained in observation (b). We use a piecewise function to ensure that once the  $AMVM$  exceeds threshold  $MV_{Th}$ , the stall impairment will not further increase. According to the results shown in figure 5.14, the threshold  $MV_{Th}$  is set to be 0.012.

In order to derive  $g(D_{ST}, N_{ST})$  and the coefficients in equation (5.5), we start with randomly selecting 60% of the test results associated with stall impairment, and use them to train the model for  $I_{ST}$  (equation (5.5)). During the training, we use different types of formulas



for  $g(D_{ST}, N_{ST})$ , including  $k_1 * D_{ST} + k_2 * N_{ST}$ ,  $D_{ST}^{k_1} * N_{ST}^{k_2}$  and  $D_{ST}^{k_1} + N_{ST}^{k_2}$ , and use non-linear regression to compute the coefficients in equation (5.5). Then we use the remaining 40% of the test results associated with stall impairment to validate the proposed  $I_{ST}$  function with all possible  $g(D_{ST}, N_{ST})$  formulas. Finally we select the formula shown in equation (5.6), since it achieves highest correlation in the validation process.

$$g(D_{ST}, N_{ST}) = \sqrt{D_{ST} * N_{ST}} \quad (5.6).$$

The values of coefficients  $a$ ,  $b$ ,  $c$  and  $d$  in equation (5.6) are listed in Table 5.6.

#### 5.4.4 Impairment Function for Level Variation

Level variation is the most complex factor to study, since it is difficult to characterize the complex patterns of level variations during a video session. As introduced in section 5.2, there are 3 dimensions for the level variation factor: average level, number of switches, and average switch magnitude. We need to derive an impairment function which can cover and reflect all 3 dimensions.

Table 5.8 shows the average evaluation of the impairment for test videos #16 ~ #40 (shown in figure 5.10). From the results we have the following observations:

**Observation (d):** All 3 dimensions of level variation factor will together affect user

Table 5.8. Subjective experiment results for level variation tests.

TEST ID	16	17	18	19	20	21	22	23	24
IMPAIRMENT	33.5	24.4	32.9	21.3	44	28	13.6	19.6	12.2
TEST ID	25	26	27	28	29	30	31	32	33
IMPAIRMENT	29	34.3	22.7	41.5	24.8	23.8	13	10.2	19.3
TEST ID	34	35	36	37	38	39	40		
IMPAIRMENT	18.5	36.5	25.2	40.2	30.2	44.6	47.5		

experience in a complex manner. For instance, comparing video #17 with video #20, both of them have an average level of 4.1, but the impairment of video #20 is significantly larger than that of #17. The same average level may lead to a completely different user experience, depending on the level fluctuation pattern. Therefore it may be difficult to reuse the method used for deriving  $I_{ID}$  and  $I_{ST}$  to also derive the impairment due to level variations.

**Observation (e):** The annoyance of staying at a low level (low quality) will grow exponentially with the duration that the low level is maintained. Comparing video #25 with #28, both of them have average level of 4.9 and similar amount of level switch magnitude, but video #25 has much smaller impairment than #28. This is because in video #25, when the level drops to the lowest value (level 1), it only lasts for about 2 seconds and then jumps up; in video #28, level stays at 1 for more than 10 seconds. If the low level (bad quality) just lasts for a short period of time, the viewer might not complain as much. But if a low level is maintained for a long time (such as more than 10 seconds), people will feel great annoyance.

**Observation (f):** The impact of decreasing level switch is much larger than that of increasing switch. Comparing video #17 with video #36, they both have an average level of 4.1, but video #36 has much more impairment than video #17. This is because the level switches in video #17 are mostly increasing switches, while the switches in video #36 are mostly decreasing switches. Therefore, we cannot treat increasing switches and decreasing switches equally when we derive the impairment function.

Based on the results and observations, we next discuss how to derive an impairment function for the factor level variations. Firstly, we need to point out that we cannot use “level” directly in the impairment function. Different streaming service providers will have different encoding settings for each level. For the same level, different service providers will specify

different frame rates and resolutions associated with it. If we derive an impairment function based on the level value, then this impairment function cannot be applied generally.

Therefore, we propose to use VQM [PW04] instead of level in impairment function. VQM is a widely accepted objective video quality metric, which has been proven to have good correlation with human perception. But VQM cannot be applied directly to DASH video because it doesn't consider the impairment due to level variation. The VQM value is a number between 0 and 1. A lower VQM value indicates a better video quality. In this thesis, we use  $VQM_i$  to indicate the amount of annoyance of video segment  $i$ . A lower  $VQM_i$  value means better quality and less annoyance.

The need to use VQM will not cause too much additional effort for content providers. On the DASH media server, the video sources are split into fixed-length segments and encoded into different levels. For each video segment  $i$  encoded at level  $j$ , we can obtain its VQM value,  $VQM_{ij}$ . The process of obtaining VQM value for each segment at each layer can be conducted offline on the media server, and it only needs to be carried out once. Once this process is done, the VQM values can be utilized to measure experienced impairments for all the future users.

Figure 5.15 shows an example of the VQM values for different levels for the encoding

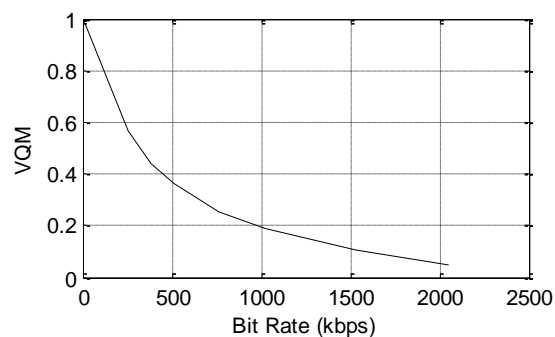


Figure 5.15. Relationship between VQM value and bit rate.

settings we used in our study. We can see that increasing the bit rate will cause a sharp decrease in VQM when bit rate is low. When bit rate becomes higher, further increasing bit rate will not lead to significant decrease in VQM.

Next we will derive an impairment function using metric VQM. Basically, the impairment caused by level variation during a DASH video session consists of 2 parts: 1) the impairment caused by low level (bad video spatial quality); 2) the impairment caused by level fluctuations.

In order to derive the impairment function, we first define the following terms: assuming in a video session, totally  $N$  video segments are being transmitted. All the video segments have the same duration  $T$ . Depending on the DASH implementation, the value of  $T$  can be 2 seconds, 5 seconds or 10 seconds, etc. For each segment  $i$ , we define a term  $D_i$ , which indicates the number of consecutive segments that are right before segment  $i$  and have VQM value within range  $[VQM_{i-\mu}, VQM_{i+\mu}]$ . Parameter  $\mu$  is heuristically set to be 0.05.

Figure 5.16 shows an example of bit rate trace and the corresponding  $D_i$  values for a 20-second DASH video. In the y-axis we have converted bit rate into VQM value. As shown in figure 5.16,  $D_i$  is an integer that will accumulate if VQM remains constant or vary within a very small range. For example, for the 10th segment (when  $i$  equals 10), there are 3 consecutive segments before it that have VQM value between  $VQM_{10-\mu}$   $VQM_{10+\mu}$ , therefore

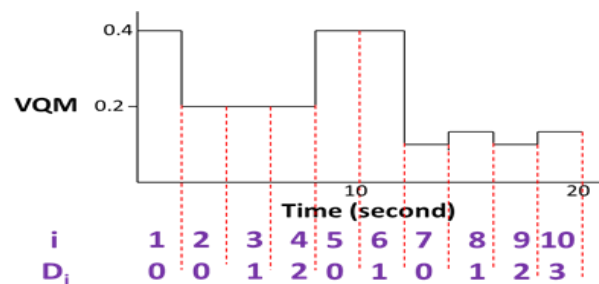


Figure 5.16.  $D_i$  values and VQM values for a 20-second DASH video.

$D_{10}$  equals 3.

We model the first part of impairment (caused by low level itself) as:

$$P_1 = \frac{1}{N} \sum_{i=1}^N VQM_i * e^{k*T*D_i} \quad (5.7).$$

As shown in equation (5.7), the  $P_1$  value (impairment due to low level) is a weighted average of the VQM values of each video segment. The exponential term in equation (5.7),  $e^{k*T*D_i}$ , is used to comply with our observation (e) that the annoyance caused by a low level grows exponentially with the duration that the low level is maintained. We use value  $D_i$  to indicate how long the level of segment  $i$  has been maintained, and multiply  $VQM_i$  with the exponential term to obtain the real annoyance of segment  $i$ . The coefficient  $k$  in equation (5.7) is used to control how fast the annoyance grows with time. The value of  $k$  is determined experimentally and listed in Table 5.6.

The second part of the impairment caused by level fluctuations can be modeled as:

$$P_2 = \frac{1}{N} \sum_{i=1}^{N-1} |VQM_i - VQM_{i+1}|^2 * sign(VQM_{i+1} - VQM_i) \quad (5.8).$$

$$\text{where } sign(x) = \begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.9).$$

The value of  $P_2$  is the average of the square of VQM differences between adjacent segments. According to our observation (f), the impairment caused by increasing switch is much smaller than that caused by decreasing switch. Therefore in equation (5.8) we use sign function (equation (5.9)) to only consider decreasing level switches and exclude increasing level switches.

Finally, the impairment due to level variation denoted as  $I_{LV}$ , is modeled as a weighted sum of  $P_1$  and  $P_2$ :

$$I_{LV} = B_1 * P_1 + B_2 * P_2 \quad (5.10).$$

where  $B_1$  and  $B_2$  are coefficients which need to be derived later. Note that the proposed impairment function  $I_{LV}$  covers all the 3 dimensions of level switch: 1) average level, covered by  $P_1$ ; 2) number of switch, covered by  $P_2$ ; 3) average magnitude of switch, covered by  $P_2$ .

We conducted a two-fold cross validation for the impairment function  $I_{LV}$ . With the 25 test videos for level switch, we randomly choose 15 videos for developing the impairment function  $I_{LV}$ , and use the other 10 videos for validating the derived  $I_{LV}$ . Then we shuffle the 25 test videos, choose another set of 15 videos for developing the impairment function, and use the rest for validation.

Figures 5.17(a) (b) show the results of the two different validations tests, specifically the relation between the subjective impairment values given by viewers with the objective impairment values computed by  $I_{LV}$ . The two validation tests achieve high correlation values of 0.88 and 0.84. We will pick the impairment function derived in the first round of validation as our final selected impairment function  $I_{LV}$ , as the first round validation lead to higher correlation. The corresponding coefficient values,  $B_1$  and  $B_2$  are derived using linear regression technique and are listed in Table 5.6.

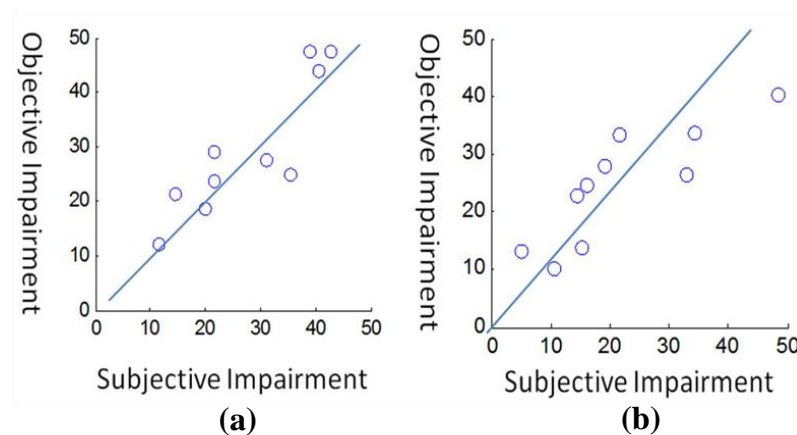


Figure 5.17. Relationship between subjective and objective impairments: (a) first round of validation for  $I_{LV}$ ; (b) second round of validation for  $I_{LV}$ .

## 5.5 Overall User Experience Model

In this section, we develop a DASH User Experience (DASH-UE) model which quantitatively measures the overall user experience, incorporating the impairment functions that we had developed in the previous section. We present results of another round of subjective experiments conducted to derive and validate the DASH-UE model.

We define DASH Mean Opinion Score (DASH-MOS) as a measurement metric for DASH-UE. Since DASH-MOS is determined by initial delay, stall and level variation factors as shown in figure 5.1, we attempt to formulate it using the impairment functions of these factors, similar to the framework of ITU-T E-Model [G107]. ITU-T E-model is developed for audio transmission, where the multiple impairments (such as network delay impairment, audio distortion impairment, etc.) occur simultaneously. E-model offers a way to quantify the combined impact of these audio transmission impairments. Therefore, we borrow the framework of the E-model because our DASH-UE model also needs to quantify the combined effect of multiple impairments.

In ITU-T E-model, the Mean Opinion Score (MOS) is formulated by a transmission rating factor  $R$  [G107]. We duplicate this function for our DASH-MOS formulation:

$$DASH - MOS = 1 + 0.035R + 7 \times 10^{-6} R(R - 60)(100 - R) \quad (5.11).$$

In (5.11), the transmission rating factor  $R$  takes value from range  $[0, 100]$  (the higher  $R$ , the better DASH-UE). DASH-MOS is related with  $R$  through nonlinear mapping, and it is within the range of  $[1, 4.5]$ .

Although the framework of ITU-T E model is helpful for our study, the formula to compute  $R$  factor specified in ITU-T E model is specific to audio transmission and not suitable for DASH video streaming. Therefore, in this paper we propose to formulate the  $R$  factor as:

$$\begin{aligned}
R &= F(I_{ID}, I_{ST}, I_{LV}) \\
&= 100 - I_{ID} - I_{ST} - I_{LV} + \sum_{\substack{i, j \in \{ID, ST, LV\} \\ i \neq j}} f_{ij}(I_i, I_j) \quad (R > 0)
\end{aligned} \tag{5.12}$$

In (5.12),  $R$  is composed of impairment functions due to initial delay, stall, and level variation. These impairment functions have been derived in previous section. Term  $f_{ij}(I_i, I_j)$  indicates the cross-effect between two different impairments and is used to compensate and adjust the  $R$  factor, because when several impairments happen simultaneously, the overall impairment will be different from the sum of each impairment.

In order to derive the formula of function  $f_{ij}(I_i, I_j)$ , and also validate the accuracy of the overall DASH-UE model, we conduct another set of subjective quality assessment experiments with a new group of participants. In the following subsections, we will introduce the new subjective tests, analyze the collected test results, and derive and validate the DASH-UE model.

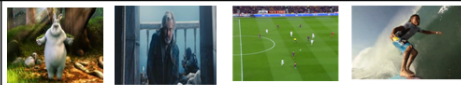
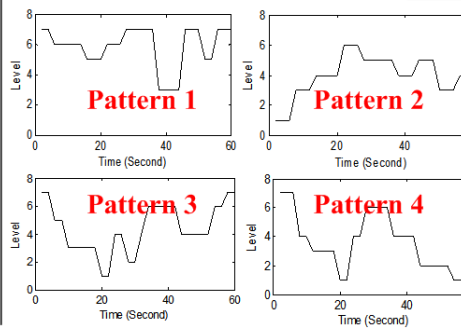
### 5.5.1 Second Round of Subjective Test

Another set of subjective tests has been carried out using a new panel of 47 subjects from UCSD and Qualcomm. The subjective test is still conducted in a controlled lab environment. Unlike the first round of test, this time each viewer is watching videos where the three artifacts (initial delay, stall and level variation) happen simultaneously.

Table 5.9 lists the parameter values we use in this round of subjective tests. Similar to the first round of tests, each test video is about one minute long. We have selected 4 different video contents, including medium motion videos like animation and movie, and high motion videos such as soccer and surfing videos. The initial delay values vary between 2 to 10 seconds, which is a reasonable range considering the whole test video is one minute long. We tried 4 different stall distributions which lead to a wide range of stall impairment values, from



Table 5.9. Parameters for second round of subjective test.

Video Content	
Initial Delay	2s, 6s, 10s
Stall	(No Stall) , (2 stall, 2 sec) (2 stall, 8 sec) , (4 stall, 12 sec)
Level Variation	

no stall to 4 stalls which add up to 12 seconds. Moreover, we include 4 different level variation patterns which exhibit very distinct impairments.

The experiment is divided into two sessions with a 10-minute comfort break between them. This adheres to the ITU-T recommendations that a continuous time period should not exceed half an hour to ensure a subject does not experience fatigue during the test.

The subjects evaluate the overall user experience of video quality (represented by the R-factor shown in equation (5.12)) according to the judging criteria shown in Table 5.5. The R data obtained from the test was scanned for unreliable and inconsistent results. We used the ITU-T [P800] criteria for screening subjective ratings which led to two subjects being rejected and their evaluations are eliminated. The scores from the rest of the subjects were averaged to compute the overall user experience (R value) for each test condition.

## 5.5.2 Model Derivation and Validation

In this subsection, we will first present and analyze the test results. Based on the observations drawn from the results, we will then derive the DASH-UE model. Finally we will present validation results.

Figure 5.18 shows the subjective R values for different stall impairment,  $I_{ST}$ , and different level variation impairment,  $I_{LV}$ , for a fixed initial delay value (here we show result when initial delay equals 2 seconds or 10 seconds). Note the values of  $I_{ST}$  and  $I_{LV}$  used in figure 5.18 are derived from the actual stall and level variations used in each test condition,

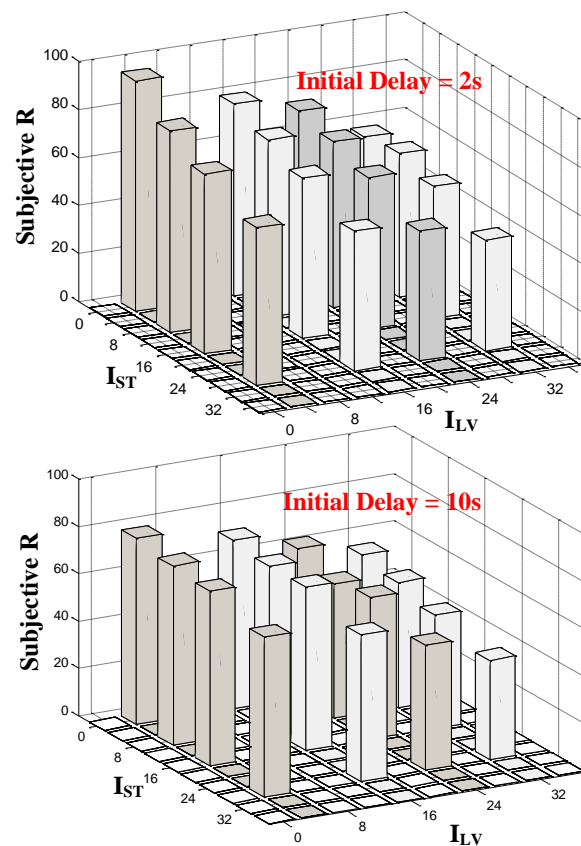


Figure 5.18. Relation between subjective R scores and  $I_{ST}$  and  $I_{LV}$ .

using the stall and level variation impairment functions (equation (5.4) ~ (5.10)). We can see that the overall user experience (R value) will drop as  $I_{ST}$  or  $I_{LV}$  increases. For a fixed level variation pattern (fixed  $I_{LV}$ ), the R value will monotonically decrease when  $I_{ST}$  increases, and the same for  $I_{LV}$ . From figure 5.18, we have the following observation.

**Observation (g):** For a certain initial delay, both stall and level variation will affect the overall UE, no matter how big the initial delay is.

Figure 5.19 shows the results from a different perspective. It shows how subjective R scores vary for different initial delay values under 3 sets of values of stall and level variation: (1) no stall, level variation pattern 1 (the shape of pattern 1 is shown in Table 5.9); (2) no stall, level variation pattern 2; and (3) two stalls which add up to 8 seconds, and level variation pattern 3. We can see that for case (1), when  $I_{ST}$  and  $I_{LV}$  are small, the R value will drop significantly when initial delay increases. For case (3), when  $I_{ST}$  and  $I_{LV}$  are large, there will not be significant difference in R value when initial delay increases from 2 seconds to 10 seconds. We can see that as  $I_{ST}$  and  $I_{LV}$  increase (from case (1) to cases (2) and (3)), initial delay will have less influence on overall user experience. This is due to the fact that people

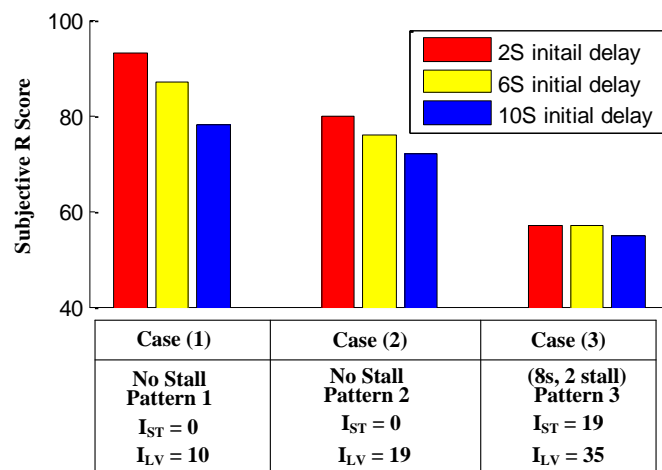


Figure 5.19. Subjective R score under different initial delay, stall and level variation.

actually have higher tolerance for initial delay than stall and level variation. From figure 5.19 we find that:

**Observation (h):** When stall and level variation impairments are prominent, the subject may pay less attention to the impact caused by initial delay. On the other hand, when stall and level variations are marginal, the impact caused by initial delay is more noticeable.

From observations (g) and (h),  $I_{ID}$ ,  $I_{ST}$  and  $I_{LV}$  should not be treated as equally important in the formula  $f_{ij}(I_i, I_j)$  of equation (5.12). The formulas of the compensation terms,  $f_{ij}(I_i, I_j)$ , should be derived such that: when  $I_{ST}$  and  $I_{LV}$  are large, the compensation terms associated with  $I_{ID}$ ,  $(f_{ID, ST}(I_{ID}, I_{ST}) + f_{ID, LV}(I_{ID}, I_{LV}))$ , should approximately cancel out the term  $(-I_{ID})$  in equation (5.12), such that the impact of initial delay on overall user experience is marginal; on the other hand, when  $I_{ST}$  and  $I_{LV}$  are small, the compensation terms associated with  $I_{ID}$  should also be small, such that initial delay will have big impact on overall user experience.

Next, we randomly select 60% of the test results, and use them to train the model for  $R$  (equation (5.12)). During the training, we use different types of functions for  $f_{ij}(I_i, I_j)$ , including  $(I_i + a \times I_j)^n$ ,  $I_i^n \times I_j^m$ , and  $e^{(I_i + a \times I_j)^n}$ , and use non-linear regression to compute the

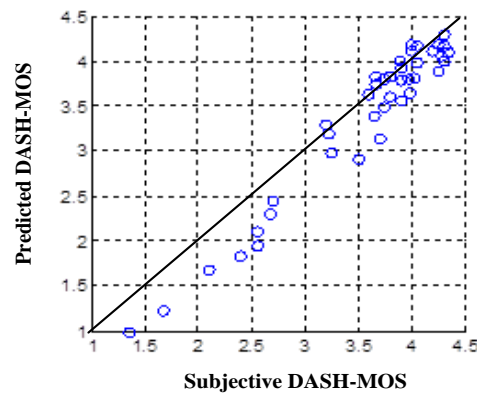


Figure 5.20. Relation between predicted and subjective DASH-MOS.

coefficients for the functions. Then we use the other 40% of test results to validate the proposed R model with all possible  $f_{ij}(I_i, I_j)$  functions. Finally we select the function shown in equation (5.13), since it achieves the highest correlation in the model validation process.

$$R = 100 - I_{ID} - I_{ST} - I_{LV} + C_1 * I_{ID} \sqrt{I_{ST} + I_{LV}} + C_2 * \sqrt{I_{ST} * I_{LV}} \quad (5.13).$$

In equation (5.13), coefficients  $C_1$  equals 0.15,  $C_2$  equals 0.82. Note that in the term  $C_1 * I_{ID} \sqrt{I_{ST} + I_{LV}}$  we have taken  $I_{ID}$  out of the square root to give more compensation for initial delay, which conforms to observation (h) that when  $I_{ST}$  and  $I_{LV}$  are large, people will ignore the impairment caused by  $I_{ID}$ .

Having derived the complete DASH-UE model, we now use the rest of the subjective tests to validate the model. Figure 5.20 shows the correlation between predicted DASH-MOS scores computed by the DASH-UE model (y-axis) and subjective DASH-MOS scores (x-axis) for each of the tests. From figure 5.20 we observe a high correlation of 0.91, and hence conclude that the proposed model can accurately predict user experience of DASH video.

## 5.6 Application of DASH-UE Model to Long Videos

One limitation of the DASH-UE model is that it is derived based on 1-min long test videos, due to the limitation of number of test videos that each subject can watch. In this section, we propose a method to apply the DASH-UE model to long videos without modifying the model, and provide preliminary validation results for the accuracy of this method.

Since our DASH-UE model is derived based on 1-minute test videos, in the new approach, we propose to divide each video into 1-minute intervals, and record the stall and level variation pattern of each minute and calculate the corresponding impairment value. As shown in figure 5.21, we denote the stall impairment and level variation impairment during i-

Table 5.10. Two approaches to compute user experience of long video.

Approach A	$R = \frac{1}{N} \sum_{i=1}^N R_i$ <p>Where: <math>R_i = \begin{cases} F(I_{ID}, I_{ST-1}, I_{LV-1}) &amp; (\text{if } i = 1) \\ F(0, I_{ST-1}, I_{LV-1}) &amp; (\text{if } i \neq 1) \end{cases}</math></p>
Approach B	$R = F(I_{ID}, I_{ST-ave}, I_{LV-ave})$ <p>Where:</p> $I_{ST-ave} = \frac{1}{N} \sum_{i=1}^N I_{ST-i}$ $I_{LV-ave} = \frac{1}{N} \sum_{i=1}^N I_{LV-i}$

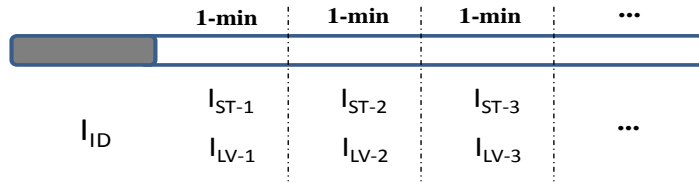


Figure 5.21. Impairment values of each minute of video.

th minute as  $I_{ST-i}$  and  $I_{LV-i}$ . We propose two different approaches to measure the UE for the entire video, as listed in Table 5.10. Approach A computes the entire video’s user experience by taking the average of every minute’s user experience. Alternatively, in approach B, we first calculate the average stall impairment ( $I_{ST-ave}$ ) and level variation impairment ( $I_{LV-ave}$ ) by taking the average of the impairments of each minute. The overall user experience (R) is then computed using the average stall and level variation impairment, together with the initial delay impairment.

Another problem is that the initial delay impairment ( $I_{ID}$ ) is derived based on 1-min short video, and therefore it is not applicable for long form videos. For the same initial delay, the impairment on short video and long video would be different.

Therefore, we have conducted another round of subjective test with long videos using 24 subjects from UCSD. This test consists of two parts. The first part is used for deriving initial delay impairment function ( $I_{ID}$ ) for long video. The second part is used for validating the proposed approaches A and B.

The first part of the test is similar to the test for deriving  $I_{ID}$ . We ask the viewers to watch a 1-min video with different initial delay values including 3, 6, 10, 15, 25 seconds. But this time instead of evaluating the initial delay impairment on the 1-min test video, we ask the viewers how big the impairment would be assuming they are watching a 10-min, 20-min or 60-min video. This evaluation is done immediately after the viewers finish watching the 1-min test video and still clearly remember the amount of annoyance they experienced due to initial delay. The results are shown in figure 5.22.

Based on the results shown in figure 5.22, we apply regression technique and adjust the initial delay impairment function  $I_{ID}$  (originally proposed in equation (5.1)) as:

$$I_{ID} = \min\left\{3.2 * \frac{L_{ID}}{1 + \ln(0.8 + 0.2 * L_{total})}, 100\right\} \quad (5.14).$$

After deriving  $I_{ID}$  for long video, we conduct the second part of subjective test in which we ask each subject to watch 3 videos, with duration of 5 minutes, 10 minutes, and 15 minutes respectively. For these 3 videos, we have distributed all 3 artifacts (initial delay, stall, level variation) simultaneously in every minute of the video. We have also selected different content for the 3 videos, including *Bunny* (for 5 minutes video), *Steel* (for 10 minutes video) and *Soccer* (for 15 minutes video), which are described in Table 5.2. After collecting all the participants' evaluations, we then compare the subjective evaluation of the overall UE (R value), with the predicted R values using approaches A and B.

Figure 5.23 shows the mean value of the subjective R value (given by subjects) and the predicted R values using approach A and approach B. We can see that: 1) approach A will

always lead to a predicted R value closer to the subjects' evaluation compared to approach B; 2) for the 5 and 10 minute videos, the difference between the predicted R value (using approach A) and subjective R value are both below 10, out of a 100 scale, showing high correlation; 3) for the 15 minute video, neither of the approaches A or B can lead to accurate prediction.

The validation results show that for videos up to 10 minutes long, approach A can provide adequate prediction accuracy. According to surveys conducted by comScore, the average length of online videos is about 6.4 minutes long [VLS12]. Therefore we claim that the proposed method (approach A) can be applied to most of the online videos.

However, for videos that are longer (more than 10 minutes long), our approach may not be applied directly. For these videos, instead of providing one UE prediction score for the entire video, we could use the DASH-UE model to provide a minute-by-minute UE score trace. For example, we show in figure 5.24 the predicted UE score trace of the 15-min Soccer test video.

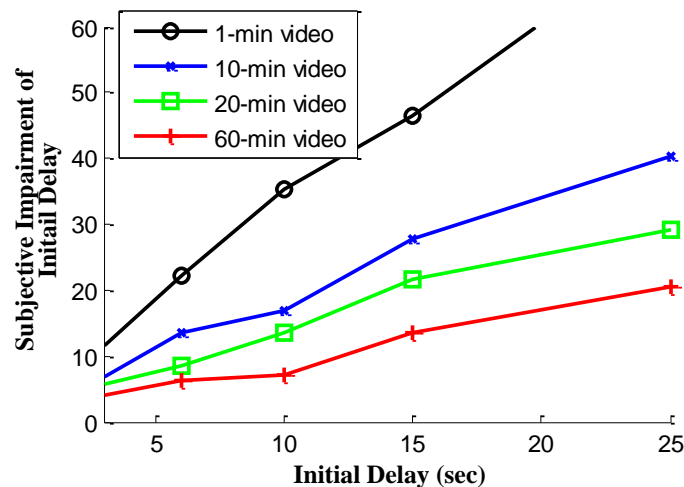


Figure 5.22. Subjective evaluations of initial delay impairment for different video length.



Figure 5.24 also provides us insight into why the prediction accuracy of approaches A and B may be low for longer videos, like the 15-minute Soccer video (shown in figure 5.23). We notice from figure 5.24 that the last few minute intervals have much higher impairment (lower predicted R) than the previous intervals. Since the quality (impairment) experienced in the latter intervals of the video may have higher impact on a viewer's assessment than the quality experienced in the earlier parts, this helps explain the low subjective score 62 seen in figure 5.23. However, our proposed approaches A and B give equal weight to the impairment of each minute interval, and hence predict much higher scores than the subjective score as seen in figure 24. Hence, one way of improving prediction accuracy for long videos may be to use unequal weights for each time interval, with the weights increasing with increasing

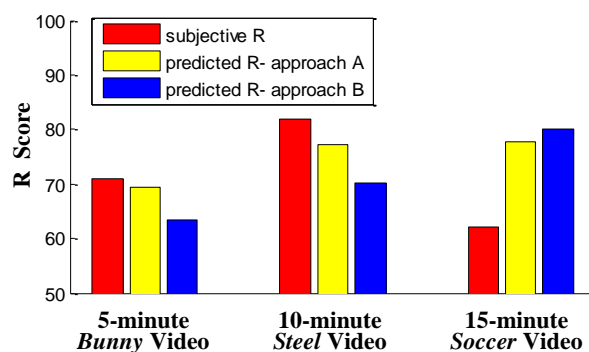


Figure 5.23. Relation between subjective R scores and predicted scores using two approaches for test videos.

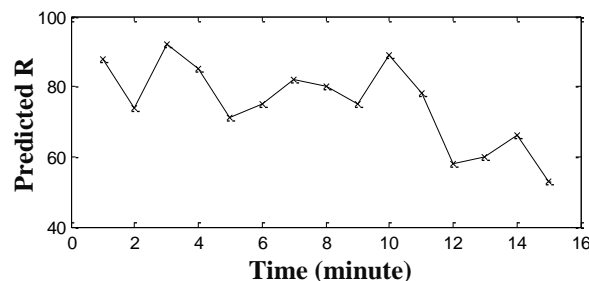


Figure 5.24. Trace of predicted R scores for 15-minute Soccer test video.

intervals. The latter may provide a single UE score for the entire video with sufficient accuracy, as an alternative when a single score is more desired than a minute-by-minute UE scores as suggested earlier for long videos. In general, quality assessment of long videos should be investigated further as part of future work, including the above suggested possible approach.

## 5.7 Conclusion

In this chapter, we have presented a novel user experience model which can quantitatively measure the user experience of DASH video, by taking into account both spatial and temporal artifacts. We first investigate 3 factors which will impact user experience: initial delay, stall and level variation. Secondly, we design and conduct subjective experiments to derive the impairment function for each of the factors. Thirdly, we combine the 3 impairment functions to formulate an overall user experience model by conducting another round of subjective tests in which subjects evaluate video quality when they experience combined artifacts. Finally, we demonstrate applicability of the proposed model to videos up to 10 minutes, longer than the average length of online videos.

The proposed user experience model does not need to access the uncompressed video source and hence can be conveniently incorporated into DASH client software to quantify user experience in real time. Moreover, the proposed model can be used by a DASH service provider to monitor and control the quality of service, as well as optimize the DASH rate adaptation algorithm.

Although the proposed DASH-UE model has considered some video content features such as motion, there are other factors related to the video content and the context of the video viewed, such as the popularity of the video and the type of device the video is watched on,

which may impact user experience. For example, it is possible that a viewer will have different level of tolerance with a video that is interesting to him/her, versus some other less appealing videos. In the future, we plan to study how these other factors will affect user experience and extend our DASH-UE model to consider them. Also, as suggested in the previous section, we plan to study and determine user experience modeling for long videos, including the unequal weight based approach suggested earlier.

Furthermore, another possible extension of this DASH-UE model is to consider the saliency information of video frames such that the important regions of a frame have larger impact on user experience than non-important regions. More specifically, we can first apply the techniques proposed in [LCE98] to detect the salient regions of video frames. Subsequently, based on the saliency information, we can develop a saliency-based video frame quality metric and then replace the VQM metric in  $I_{LV}$  function with this new metric. One can expect the modeling accuracy will be improved by incorporating the saliency information into the DASH-UE model.

## **Acknowledgments**

The text of this chapter, in part or in full, is based on material that has been published in IEEE Transactions on Broadcasting (Y. Liu, S. Dey, M. Ulupinar, M. Luby, Y. Mao, “Deriving and Validating User Experience Model for DASH Video Streaming”), and Proc. of Packet Video Workshop, Dec. 2013 (Y. Liu, S. Dey, D. Gillies, F. Ulupinar, M. Luby, “User Experience Modeling for DASH Video”). The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

# Chapter 6

## Conclusion and Future Direction

This chapter concludes the dissertation with a summary of our principle contributions and some thoughts about the possible future research work.

In Chapter 2, we developed a comprehensive content-aware CMR-UE model which can characterize simultaneous effects of graphic rendering, video encoding and networking factors on the user experience of a CMR application. In Chapter 3, using the proposed CMR-UE model, we develop a Content-Aware Adaptive Rendering (CAAR) algorithm, to dynamically select the optimal rendering setting according to network conditions such that the right tradeoff between video rendering quality and video encoding quality is achieved, and user experience is maximized at any time. Experiments conducted in a real 3.5G network demonstrate the effectiveness of the proposed method.

In Chapter 4, we propose two rendering based techniques which utilize rendering information to enhance video encoding for CMR application to achieve: 1) better video perceptual quality under a fixed bit rate budget; 2) lower encoding complexity and encoding time for every video frame. The rendering-based prioritized encoding technique exploits rendering information including depth map and scene composition information to prioritize

different regions of a game frame, and adapts the QP values for each MB according to their importance and the bit rate budget. Assessment experiments have been carried out to validate the effectiveness of this prioritized encoding technique. The experimental results show that this technique can significantly increase the player's perceptual video quality under a given bit rate budget. The proposed rendering-based encoding acceleration technique uses rendering information to reduce the computational complexity of the encoding process of H.264/AVC technique by: 1) directly calculating the motion vectors and eliminating the need for regular search-based motion estimation; 2) using rendering information to reduce the number of candidate encoding modes for ROD process. Experimental results show that the proposed rendering-based encoding acceleration technique can achieve a significant and consistent encoding time reduction.

In Chapter 5, we presented a novel user experience model which can quantitatively measure the user experience of DASH video, by taking into account both spatial and temporal artifacts. We first investigate 3 factors which will impact user experience: initial delay, stall and level variation. Secondly, we design and conduct subjective experiments to derive the impairment function for each of the factors. Thirdly, we combine the 3 impairment functions to formulate an overall user experience model by conducting another round of subjective tests in which subjects evaluate video quality when they experience combined artifacts. Finally, we demonstrate applicability of the proposed model to videos up to 10 minutes, longer than the average length of online videos. The proposed user experience model does not need to access the uncompressed video source and hence can be conveniently incorporated into DASH client software to quantify user experience in real time. Moreover, the proposed model can be used by a DASH service provider to monitor and control the quality of service, as well as optimize the DASH rate adaptation algorithm.

In the future, we envision two ideas to be the initial steps to extend the UE models and techniques described in previous chapters. First, for cloud mobile rendering application, in addition to using the rendering information explained in Chapter 4, we plan to investigate the use of pixel-domain features of the rendered video frame, such as luminance and chrome values of every pixel, to increase the perceptual video quality as well as accelerate video encoding. Furthermore, these pixel-domain features can be used to detect the object boundaries or to estimate the spatial complexity of a MB, and hence can help in optimizing the mode selection algorithm by eliminating some unlikely intra-modes.

Secondly, in the future we plan to extend the DASH-UE model by including some other video content features and the context of the video viewed, such as the popularity of the video and the type of device the video is watched on, which may impact user experience. For example, it is possible that a viewer will have different level of tolerance with a video that is interesting to him/her, versus some other less appealing videos. In the future, we plan to study how these other factors will affect user experience and extend our DASH-UE model to consider them. Also, as suggested in section 5.6, we plan to study and determine user experience modeling for long videos, including the unequal weight based approach suggested earlier.

# Bibliography

- [JUN11] Juniper Research, Mobile Cloud: Smart Device Strategies for Enterprise & Consumer Markets 2011-2016, Jul. 2011. [Online]. Available: <http://juniperresearch.com/>.
- [HLL12] B.R. Huang, C.H Lin, C.H Lee, "Mobile Augmented Reality based on Cloud Computing", in *Proc. of International Conference on Anti-Counterfeiting, Security and Identification (ASID'12)*, Taipei, Aug. 2012.
- [Dey12] S.Dey, "Cloud Mobile Media: Opportunities, challenges, and directions", in *Proceedings of IEEE International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2012.
- [WD13] S. Wang, S. Dey, "Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications", in *IEEE Transactions on Multimedia*, VOL. 15, NO. 4, Jun. 2013.
- [WD12] S. Wang, S. Dey, "Cloud Mobile Gaming: Modeling and Measuring User Experience in Mobile Wireless Networks", in *ACM SIGMOBILE Mobile Comput. Comm. Rev.(MC2R)*, VOL. 16, ISSUE 1, Jan. 2012, pp. 10-21.
- [AS10] Khan,A. and Lingfen Sun, "Video Quality Prediction Model for H.264 Video over UMTS Network and Their Application in Mobile Video Streaming," in *Proc. of IEEE International Conference on Communications(ICC'10)*, 2010.
- [VC09] Mukundan V. and Maninak C., "Evaluate Quality of Experience for Streaming Video in Real Time," in *Proc. of IEEE Global Communications Conference (GLOBECOM'09)*, 2009.
- [XC12] J. Xue and C. W. Chen, "Mobile JND: environment adapted perceptual model and mobile video quality enhancement," in *Proc. of ACM MMSys '12*, Pages 173-183, February 2012.
- [CHH06] K. Chen, T. Huang, P. Huang, and C. Lei, "Quantifying Skype User Satisfaction", in *Proc. of ACM SIGCOMM*, VOL. 36, Oct 2006.
- [LZKM10] Y. Lu, Y. Zhao, F. A. Kuipers, and P. V. Mieghem, "Measurement Study of Multi-Party Video Conferencing," in *Proc. of IFIP Networking'10*, pages 96-108, 2010.
- [JSSH11] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoffeld, "An Evaluation of

- QoE in Cloud Gaming Based on Subjective Tests,” in *Workshop on Future Internet and Next Generation Networks*, Seoul, Korea, Jun. 2011.
- [BF10] M. Bredel, M. Fidler, “A Measurement Study regarding Quality of Service and its Impact on Multiplayer Online Games”, in *ACM SIGCOMM workshop on Network and System Support for Games, (NetGames)*, 2010.
- [PSG09] D. Priritaz, C. Salzmann and D.Gillet, “Quality of Experience for Adaptation in Augmented Reality”, in *Proc. of 2009 IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, Oct., 2009.
- [Tac04] N.Tack, “3D Graphic Rendering Time Modeling and Control for Mobile Terminals”, in *Proc. of Int. Conf. on 3D Web Technology*, 2004.
- [LWD12] Y. Liu, S. Wang, S. Dey, “Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering”, in *Proc. of IEEE ICNC, Maui*, Jan. 2012.
- [WD09] S. Wang, S. Dey, “Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach,” in *IEEE GLOBECOM*, Nov. 2009.
- [WD10] S. Wang, S. Dey, “Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming,” in *IEEE GLOBECOM*, Dec. 2010.
- [WD12] S. Wang, S. Dey, “Cloud Mobile Gaming: Modeling and Measuring User Experience in Mobile Wireless Networks,” *ACM SIGMOBILE MC2R*, vol. 16, issue 1, Jan. 2012, pp. 10-21.
- [WD13] S. Wang, S. Dey, “Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications,” *IEEE Transactions on Multimedia*, vol. 15, no. 4, Jun. 2013.
- [WDA10] S. Wang, S. Dey, "Addressing Response Time and Video Quality in Remote Server Based Internet Mobile Gaming," in *Proc. of IEEE WCNC*, Sydney, Mar. 2010.
- [ONL] “Onlive”, <http://www.onlive.com>.
- [GAI] “Gaikai”, <http://www.Gaikai.com>.
- [P800] ITU-T Recommendation ITU-T P.800, “Methods for Subjective Determination of Transmission Quality,” 1996.
- [G107] ITU-T Recommendation G.107, “The E-model, a Computational Model for Use in Transmission Planning,” Mar. 2005.
- [G1070] ITU-T Recommendation G.1070, “Opinion Model for Videotelephony,” Apr. 2007.
- [PS] Planeshift, <http://www.planeshift.it/>.
- [BRO] Broadsides, <http://cse125.ucsd.edu/cse125/2012/cse125g1/>
- [SUP1] [http://esdat.ucsd.edu/JETCAS\\_paper.html](http://esdat.ucsd.edu/JETCAS_paper.html)
- [DSA] Dichotomous search algorithm, <http://shmathsoc.org.cn/lu/core%20part/Chap4.pdf>



- [WAN95] B. Wandell, *Foundations of Vision*, MA, USA: Sinauer, 1995.
- [LWD14] Y. Liu, S. Wang, S. Dey, "Content-Aware Modeling and Enhancing User Experience in Cloud Mobile Rendering and Streaming", in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol.4, no.1, pp. 43-56, Mar. 2014.
- [HJS13] M. Hemmati, A. Javadalab and S. Shirmohammadi, "Game as video: bit rate reduction through adaptive object encoding," in *Proc. of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2013, New York, USA.
- [WSBL03] T. Wiegand, G.J.Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560-576, Jul. 2003.
- [SHI13] S. Shirmohammadi, "Adaptive streaming in mobile cloud gaming", in *IEEE COMSOC Multimedia Communications Technical Committee E-Letter*, 2013.
- [LL08] Y. Liu, Z.G. Li, "Region-of-interest based resource allocation for conversational video communication of H.264/AVC," in *IEEE Trans. Circuits Syst. Video Technol.*, VOL. 18, NO. 1, pp. 134–139, Jan. 2008.
- [WFC12] G.-L.Wu, Y.-J. Fu, and S.-Y. Chien, "Region-based perceptual quality regulable bit allocation and rate control for video coding applications," in *Proc. VCIP*, 2012.
- [LGWMZ04] W. Lai, X. Gu, R. Wang, W. Ma, H. Zhang, "A Content-Based Bit Allocation Model for Video Streaming," in *Proc. IEEE international Conference on Multimedia and Expo (ICME)*, 2004.
- [LQI11] Z. Li, S. Qin, and L. Itti, "Visual attention guided bit allocation in video compression," *Image Vis. Comput.*, vol. 29, no. 1, pp. 1–14, Jan. 2011.
- [ATHS14] H. Ahmadi, S. Zad Tootaghaj, M.R. Hashemi, and Shervin Shirmohammadi, "A Game Attention Model for Efficient Bitrate Allocation in Cloud Gaming", *Multimedia Systems, Springer*, Vol. 20, Issue 5, October 2014, pp. 485-501.
- [TMP11] N. Tizon, C. Moreno, and M. Preda, "ROI based video streaming for 3D remote rendering," in 2011 *IEEE 13th International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2011, Hangzhou, China.
- [TAH14] M.R.H.Taher, H.Ahmadi, M.R.Hashemi, "Power-Aware Analysis of H.264/AVC Encoding Parameters for Cloud Gaming," in *Cloud Gaming Systems and Networks in conjunction with IEEE International Conference on Multimedia & Expo*, 2014.
- [FE10] P. Fechteler, P. Eisert, "Accelerated Video Encoding Using Rendering Context Information," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2010.
- [SHJYS14] M. Semsarzadeh, M. Hemmati, A. Javadtalab, A. Yassine, S. Shirmohammadi, "A Video Encoding Speed-up Architecture for Cloud Gaming," in *Cloud Gaming Systems and Networks in conjunction with IEEE International Conference on Multimedia & Expo*, 2014.

- [COS08] G. Cheung, A. Ortega, and T. Sakamoto, "Fast H.264 mode selection using depth information for distributed game viewing," in *Proc. Visual Communications and Image Processing (VCIP)*, 2008.
- [SHC11] S. Shi, C.-H. Hsu and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *Proc. 19th ACM Int. Conf. Multimedia*, Nov. 2011, pp. 103–112.
- [CMG09] B. Ciubotaru, G. Muntean, and G. Ghinea, "Objective Assessment of Region of Interest-Aware Adaptive Multimedia Streaming Quality," in *IEEE Trans. Broadcasting*, vol. 55, no. 2, pp. 202–212, Jun. 2009.
- [DL96] 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.
- [CZ00] T. Chiang and Y. Zhang, "A new rate control scheme using quadratic rate distortion model," in *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 7, no. 2, pp. 246-250, May, 2000.
- [X264] X264 encoder, <http://www.videolan.org/developers/x264.html>
- [CLJ06] I. Choi, J. Lee and B. Jeon, "Fast Coding Mode Selection with Rate-Distortion Optimization for MPEG-4 Part-10 AVC/H.264", in *IEEE Trans. Circuit and Sys. for Video Technology*, vol. 16, no. 12, pp. 1557-1561, Dec, 2006.
- [ZM00] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in *IEEE Trans. Image Processing*, vol. 9, pp.287–290, Feb. 2000.
- [ZLC<sup>+</sup>04] C. Zhu, X. Lin, L. Chau, and L. Po, "Enhanced hexagonal search for fast block motion estimation," in *IEEE. Trans.Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1210–1214, Oct. 2004.
- [SUP2] Supplementary Material, [http://esdat.ucsd.edu/TCSVT\\_paper.html](http://esdat.ucsd.edu/TCSVT_paper.html)
- [LSZ09] Z. Liu, L. Shen, and Z. Zhang, "An efficient intermode decision algorithm based on motion homogeneity for H.264/AVC," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 1, pp. 128–132, Jan. 2009.
- [BJO01] G. Bjontegaard, "Calculation of Average PSNR Differences Between RD Curves", *Doc. VCEG-M33*, Apr. 2001.
- [CIS13] Cisco, San Jose, CA, USA, "Cisco visual networking index: Global mobile data traffic forecast," White Paper, 2013-2018.
- [ISO10] ISO/IEC JTC 1/SC 29/WG 11 (MPEG), "Dynamic adaptive streaming over HTTP," w11578, CD 23001-6, Guangzhou, 2010
- [PW04] M.Pinson and S.Wolf, "A new standardized method for objectively measuring video quality," in *IEEE Trans. On Broadcasting*, vol. 50, no.3, pp.312-322, Sept. 2004.
- [Oze11] Jan Ozer, "Adaptive Streaming in the Field," in *Streaming Media Magazine*, 2011.

- [HR10] S. Hemami and A. Reibman, “No-reference image and video quality estimation: Applications and human-motivated design,” in *Signal Process.:Image Commun.*, vol. 25, no. 7, Aug. 2010.
- [RNR08] M. Ries, O. Nemethova and M. Rupp, “Video quality estimation for mobile H.264/AVC video streaming,” in *Journal of Communications*, Vol. 3, No.1, Jan. 2008, pp.41-50.
- [LVG09] Y.-C. Lin, D. Varodayan, and B. Girod, “Video quality monitoring for mobile multicast peers using distributed source coding,” in *Proc. 5th International Mobile Multimedia Communications Conference*, London, 2009.
- [MC11] R. Mok, E.Chan, “Measuring the Quality of Experience of HTTP Video Streaming”, in *Proceeding of 2011 IEEE International Symposium on Integrated Network Management*, Dublin, 2011.
- [SHR12] K. Singh, Y. Hadjadj-Aoul, and G. Rubino, “Quality of Experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC,” in *Proc. of Consumer Communication and Networking Conference (CCNC 2012)*, Las Vegas, USA, 2012.
- [NEE11] P. Ni, R. Eg, A. Eichhorn, “Flicker Effects in Adaptive Video Streaming to Handheld Devices”, in *Proc. of the ACM International Multimedia Conference (ACM MM)*, 2011.
- [NEE12] P.Ni, R.Eg, A. Eichhorn, “Spatial Flicker Effect in Video Scaling”, in *International Workshop on Quality of Multimedia Experience*, Mechelen, 2012.
- [KS12] A.Khan and L.Sun, “QoE Prediction Model and its Application in Video Quality Adaptation Over UMTS Networks,” in *IEEE Trans. On Multimedia*, vol. 16, no.4, April 2012.
- [CER09] G. W. Cermak, “Subjective video quality as a function of bit rate, frame rate, packet loss rate and codec,” in *Proc. 1st Int. Workshop Quality of Multimedia Experience (QoMEX)*, Jul, 2009, pp. 41–46.
- [MSS10] A. K. Moorthy, K. Seshadrinathan, R. Soundararajan, “Wireless video quality assessment: A study of subjective scores and objective algorithms,” in *IEEE Trans. Circuits Syst. Video Technol.*, vol.20, no. 4, pp. 513–516, Apr. 2010.
- [SSB10] K. Seshadrinathan, R. Soundararajan, A. Bovik, “Study of subjective and objective quality assessment of video,” *IEEE Trans. Image Process.*, vol. 19, no. 6, Jun. 2010.
- [PMJ13] K. Pessemier, K. Moor, W. Joseph, “Quantifying the Influence of Rebuffering Interruptions on the User’s Quality of Experience During Mobile Video Watching,” *IEEE Trans. On Broadcasting*, vol. 59, no. 1, Mar.2013.
- [MCC11] M. Mok, E. Chan, and R. Chang, “Measuring the Quality of Experience of HTTP Video Streaming,” in *Proceedings of IEEE International Symposium on Integrated Network Management*, Dublin, May, 2011.
- [VLS12] Video length statistics by comScore [Online], available: <http://www.tubefilter.com/2012/05/12/average-length-online-video/>

- [LCE98] Itti L, Koch C. and Niebur E., "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254-1259 (1998).
- [FFM] ffmpeg software, <https://www.ffmpeg.org>